Disk Failure Prediction in Data Centers via Online Learning

Jiang Xiao
SCTS/CGCL
Huazhong University of Science and Technology
Wuhan, China
jiangxiao@hust.edu.cn

Zhuang Xiong
SCTS/CGCL
Huazhong University of Science and Technology
Wuhan, China
xzhuang@hust.edu.cn

Song Wu*
SCTS/CGCL
Huazhong University of Science and Technology
Wuhan, China
wusong@hust.edu.cn

Yusheng Yi
SCTS/CGCL
Huazhong University of Science and Technology
Wuhan, China
hust@hust.edu.cn

Hai Jin
SCTS/CGCL
Huazhong University of Science and Technology
Wuhan, China
hjin@hust.edu.cn

Kan Hu
SCTS/CGCL
Huazhong University of Science and Technology
Wuhan, China
hukan@hust.edu.cn

ABSTRACT

Disk failure has become a major concern with the rapid expansion of storage systems in data centers. Based on SMART (*Self-Monitoring, Analysis and Reporting Technology*) attributes, many researchers derive disk failure prediction models using machine learning techniques. Despite the significant developments, the majority of works rely on offline training and thereby hinder their adaption to the continuous update of forthcoming data, suffering from the '*model aging*' problem. We are therefore motivated to uncover the root cause – the dynamic SMART distribution for '*model aging*', aiming to resolve the performance degradation as to pave a comprehensive study in practice.

In this paper, we introduce a novel disk failure prediction model using *Online Random Forests* (ORFs). Our ORF-based model can automatically evolve with sequential arrival of data on-the-fly and thus is highly adaptive to the variance of SMART distribution over time. Moreover, it has favourable advantage against the offline counterparts in terms of superior prediction performance. Experiments on real-world datasets show that our ORF model converges rapidly to the offline random forests and achieves stable failure detection rates of 93-99% with low false alarm rates. Furthermore, we demonstrate the ability of our approach on maintaining stable prediction performance for the long-term usage in data centers.

KEYWORDS

Failure prediction, online learning, hard disk drive, SMART, storage system reliability

*The corresponding author

ICPP 2018, August 13–16, 2018, Eugene, OR, USA

ACM ISBN 978-1-4503-6510-9/18/08…$15.00
https://doi.org/10.1145/3225058.3225106

ACM Reference Format:
Jiang Xiao, Zhuang Xiong, Song Wu, Yusheng Yi, Hai Jin, and Kan Hu. 2018. Disk Failure Prediction in Data Centers via Online Learning. In *ICPP 2018: 47th International Conference on Parallel Processing, August 13–16, 2018, Eugene, OR, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3225058.3225106

1 INTRODUCTION

The rapid expansion of storage systems in data centers makes the originally accidental component failure become the norm [1]. Disks are among the most frequently failed components. Recent study based on data from Microsoft reports that disk failure dominates 76-95% of all failed components in data centers [2]. Disk failure can induce serious loss of data and subsequent service downtime, which results in enormous losses to the business [3]. Nevertheless, the continuously growing capacity of the disks will raise the probability of sector errors and data corruption [4]. Engineers from Google [5] argue that disks in data center should tolerate higher error rates and the responsibility to cope with those errors should be reassigned to higher storage layers, in order to achieve high performance and low cost. Therefore, disk failure becomes a major concern with the goal of ensuring highly reliable services of data center.

The growing body of research on the improvement of storage system reliability mainly falls into two folds. One is reactive fault tolerance technique that reconstructs data after the occurrence of disk failure, by designing erasure codes and data redundancy mechanism [6]. However, it may affect the disks' read/write operations and degrade the overall system performance. In contrast, proactive fault tolerance technique performs prediction before the failure actually occurs, leveraging the past behavior of a disk. *Self-Monitoring, Analysis and Reporting Technology* (SMART) is one such method, which detects and reports various indicators of drive reliability, for instance, counts of reallocated sectors, load cycles, rates of seek errors, read errors [7]. By adopting statistical and machine learning techniques, a wide variety of SMART-attribute-based proactive disk failure prediction models have been continuously proposed and become increasingly popular in recent years [8–17] .

Depending on the historical data within a fixed period, offline training and evaluation are performed by the above proactive methods. Note that the models can perform extremely well in their early stage of application. However, the prediction effectiveness will be greatly degraded as time goes on. We refer this severe problem associated with current offline models as '*model aging*'. To provide a deeper insight into the fundamental reason for model aging, preliminary experiments are conducted to simulate the practical long-term usage. It comes out to a consistent result of [14] that the sequentially collected data will gradually change the underlying distribution of cumulative SMART attributes, making the prior models lose their validity over time and inadequate for predicting the future SMART data. More specifically, cumulative attributes such as Reallocated Sectors Count, Power-On Hours which record the accumulative occurrences in full life cycle of disks, usually serve as strong indicators for disk failure.

Existing solutions tackle the model aging problem by updating the disk failure prediction model in offline mode. Li *et al.* [14] proposed two offline learning strategies, namely *replacing* and *accumulation*. The former updates the model periodically (*i.e.,* once a week in their case), using only the samples collected within the last cycle, while the latter uses all the samples collected from the beginning to the last cycle, and then the updated model is applied to predict failures during the current cycle. Although the prediction accuracy is well maintained by both strategies, neither of them can circumvent periodically retraining the model. Furthermore, offline learning lacks the ability to adapt to dynamic patterns of arrival data. This motivates us to figure out an alternative mechanism that supports automatic model updating under varying distribution of SMART attributes. *Online learning* [18] exhibits great potential of performance improvement with sequential arrival of data and superiority over offline learning, including real-time predictions and lower memory requirements.

Unfortunately, it is not straightforward but very challenging to apply online learning method for disk failure prediction. The first challenge is how to label SMART samples at online operation. Unlike offline learning that SMART samples can be easily labeled positive (*i.e.*, faulty) or negative (*i.e.*, healthy) with the full knowledge of disk status, online learning can hardly label the incoming samples. This is because offline mode provides static training data that can be used for informing whether a disk in the dataset is faulty or not in advance. In contrast, prediction models in online mode evolve with the serially gathered training data on-the-fly and the actual status of disks remains uncertain. In particular, it is possible for some operating disks to become faulty at an extremely short time, whereas the corresponding samples can not be labeled as positive until the occurrence of disk failure. To this end, we introduce an automatic *online label* method that stores the samples collected within the most recent period of time, *e.g.,* one week, for each disk in the system. These stored samples will remain unlabelled unless disk failure truly occurs or new samples arrive. When a disk fails, the unlabelled samples belonging to the failed disk will be labeled as positive and removed. All the outdated samples will be labeled as negative and replaced by the newly collected samples. Once the samples are labeled, they will be used to update the model.

Another critical challenge lies in the fact that there is a highly unbalanced ratio between the failed disks and normal disks, since disk failure is relatively a rare event and failed disks only account for a tiny part of all disks. According to a real-world dataset [19], the amount of negative samples can be hundreds to thousands of times more than the positive ones. In offline mode, the unbalance situation can be solved by downsampling the negative class of the entire training set to an amount that is close to the size of the positive class [20]. In online mode, however, training samples are gathered over time, which makes sampling impossible. We draw ideas from online bagging method [21] and come up with a feasible solution where the sequential arrival of positive and negative samples are modeled by two Poisson distributions of different rate parameters, respectively. As a result, sequentially arrival negative samples are relatively rarely selected to update the model. Our experiments demonstrate the effectiveness of this method in addressing the sample imbalance issue.

In this paper, we design a novel online learning model based on *Online Random Forests* (ORF) [22] for disk failure prediction. The ORF algorithm inherently has several advantages over previous works, *e.g.*, its overall training and testing procedures of ORF can be easily parallelized as each tree in a forest is built and tested independently from others [22], and models are highly interpretable so they can be used to reveal the real cause of disk failures and help improve the reliability of storage systems [14, 23]. More importantly, ORF generates random trees on-the-fly using the gradually gathered SMART samples. To allow for temporal weighting of knowledge, it also discards outdated trees by leveraging *Out-Of-Bag-Error* (OOBE) [26] and brings forth new trees to fit the current distribution of training data as replacements. Therefore, our ORF-based prediction model possesses automatic adaptivity to the distribution variation of SMART attributes as well as good robustness against label noise, making it appropriate for the practical long-term usage.

Additionally, our online learning method achieves more competitive failure prediction performance than previous works and performs comparable (even better) to the common offline *random forests* (RF). The experiments on dataset collected from 34,535 disks, monitored over three years [19], show that our ORF-based model converges rapidly (within six months) to the performance of the offline RF and achieves *failure detection rates* (FDRs) of 93-99% with reasonable low *false alarm rates* (FARs) after the first six months. Compared with offline RF models using update strategies [14], we show that our ORF-based algorithm can maintain reasonably lower FARs while achieve comparable FDRs with these periodically updated RF models, however, without the need of model retraining after the initial deployment.

In summary, our major contributions presented in this paper are the following:

- To the best of our knowledge, we pioneer the use of online learning method on disk failure prediction to detour the model aging problem of offline counterparts.
- We point out the challenges of applying online learning to disk failure prediction and present our solutions.
- We also propose a specific online learning method for disk failure prediction based on ORF.
- We simulate the long-term use of ORF-based prediction models and demonstrate the effectiveness and adaptivity of our method in real-world data centers.

In the following Section 2, we survey the related work on SMART based disk failure prediction. Section 3 presents our online learning approach for disk failure prediction. We evaluate our models and present the experimental results in Section 4, followed by conclusions in Section 5.

2 RELATED WORK

Different from the traditional reactive fault tolerance technique which troubleshoots a system upon occurrence of disk failures, proactive fault tolerance technique anticipates potential disk failures and provides preventive measurement to prevent failure from occurring. It can also significantly improve the reliability and availability of storage system, however, without affecting the read/write performance of drives. Therefore, many researchers have focused especially on SMART-based proactive disk failure prediction.

SMART is a self-monitoring system supported by most disk manufacturers in their products [7]. The system detects and reports various indicators correlated with impending disk failure. The anomaly detection method used by SMART is simple threshold-based algorithm, which triggers a system warning when any SMART attribute exceeds its predefined threshold. These thresholds are set conservatively by manufacturers to avoid false alarms at the expense of prediction accuracy. Due to the simpleness of threshold algorithm and the conservative settings of thresholds, this technology achieves poor FDRs of 3-10% [10].

To enhance the SMART-based failure prediction performance, machine learning and statistical techniques are proposed to build prediction models. Hamerly and Elkan [8] employed a supervised naive Bayes classifier to predict disk failures and achieved a prediction accuracy of 33% with 0.67% FAR on a small dataset of 1,936 drives (i.e., only 9 of them do fail). A mixture model of naive Bayes submodels trained using *expectation-maximization* was also investigated and achieved a similar performance.

Hughes *et al.* [9] applied a non-parametric statistical method, the *Wilcoxon rank-sum test*, to train prediction models, as they found that most of the critical SMART attributes are non-parametrically distributed. Based on a dataset containing 2-3 months of reliability design test data from 3,744 drives, the highest FDR they achieved was 60% at 0.5% FAR. In their later work [10], several methods including the rank-sum test, *support vector machines* (SVM), *unsupervised clustering*, and *reverse arrangements test* were compared on a dataset containing only 369 drives. When using a certain set of four SMART attributes, the results showed that rank-sum test could outperform SVM with 28.1 percent detection rate. However, SVM provided the best FDR of 50.6% with no measured false alarms when all of 25 attributes were used. Wang at al. [11] then improved the SVM-based prediction model by attaching the change rates of SMART attributes as explanatory variables and reached a FDR of 80% at 0.3% FAR.

Despite the good prediction performance of SVM, its computational efficiency and memory use are too expensive for online monitoring [12]. Wang *et al.* [12] used *Mahalanobis distance* (MD) to aggregate the input variables into one index and detect disk anomaly by setting an appropriate threshold. They found that using only the critical SMART attributes selected by FMMEA (*Failure Modes, Mechanisms and Effects Analysis*) could lead to better performance compared with using all attributes. In their later work [13], they expanded the work and proposed a sliding-window-based generalized likelihood radio test to track the anomaly progression in disks. The method delivered a 68% FDR with zero FAR on the same dataset used by [10].

Back Propagation Artificial Neural Networks (BP ANN) [11] and *Classification and Regression Trees* (CART) [14] were explored by Zhu *et al.* to further improve the FDR. The results on a real-world dataset containing 8 weeks of SMART data from 23,395 drives showed that both models could achieve great FDRs which were about 95% with a reasonable low FAR. Recently, this research group focused on gauging the different health statuses of disk drives. They formulated the disk failure prediction as a multi-level classification problem and predicted the health degree rather than binary status of disks. *Recurrent Neural Networks* (RNN) [15] and *Gradient Boosted Regression Trees* (GBRTs) [16] were used to build the residual life prediction models and both models achieved reasonable accurate health status assessment. On a large real-world dataset, the RNN-based method delivered about 40-60% ACC (*accuracy of residual life level assessment*) on failed samples [17].

Recently, Mahdisoltani *et al.* [25] took a different approach, predicting sector errors rather than disk failures, to improve the reliability of storage system. They explored several techniques including CART, SVM, NN, *Logistic Regression* (LR), and RF, and found that the simplest and easiest to train machine learning models, RF, achieved the highest prediction accuracy. The training of sector error predictors was found to be robust for small training sets or training data coming from a different drive model. In addition, they proposed a number of possible use cases for the predictors and shown that the mean time to detecting errors (and hence the window of vulnerability to data loss) can be greatly reduced by adjusting scrub rates based on error predictions.

The aforementioned methods are all designed to train prediction models in batch or offline mode. Though possess great prediction performance and have demonstrated their superiority in a number of circumstances, all these models cannot get rid of the model aging problem, since the underlying distribution of SMART attributes changes over time. Besides that, they all require the access to full training data and wish the dataset sufficient enough to contain adequate information of building high-performance models. However in practice, the full training data may not be given initially but gathered gradually. The process of collecting informative dataset can take a very long time (maybe years), especially for the small-scale data centers, as disk failure is an accidental event while the SMART information of failed disk is what matters. This implies the deployment of models that support online learning is more enlightened, as the online models are capable of integrating future data automatically.

In this paper, we attempt to adopt online learning method for disk failure prediction and present an ORF-based method for practical usage. ORF can evolve with sequential arrival of data on-the-fly and forget old information by controlled discarding outdated trees. As a result, our online learning model can dynamically adapt to new patterns of data and operate without further concern of model aging.

3 THE PROPOSED METHOD

Our goal is to predict whether a disk will fail within a given time interval, based on the SMART data that the disk reported. This problem of predicting future errors can be formulated as a binary classification problem. In the following discussion, we constrain such period into seven days before a faulty event, for the sake of simplicity.

In this section, we start with a brief introduction to ORFs algorithm and then present our ORF-based online learning method in detail. The difficulties of adopting ORFs to build disk failure prediction models are discussed concomitantly.

3.1 Online Random Forests

Online Random Forests (ORFs) [22] are commonly used for classification and regression, as a combination of strengths of *Random Forests* (RFs) [24] and *online machine learning* [18]. Given a training set consisted of two classes, the ensemble method operates by constructing numerous decision trees in the training phase and outputting the class that is the mode of two classes of the individual trees.

ORFs operate in online mode benefiting from online bagging [21] and online tree growth techniques. To achieve the same effect as offline bagging (that is, each training sample is randomly selected by each tree in the forest), ORFs use the online bagging method proposed by Oza *et al.* [21], which applies a *Poisson distribution* function to model the sequential arrival of data. Oza *et al.* proved convergence of this method to offline bagging.

During the growth of a randomized tree in ORF, each decision node creates a set of random tests and chooses the best according to some impurity measurement. For hard drive failure prediction, each random test is in the form of $SMART\_i > \theta$, where $SMART\_i$ represents the SMART attribute of ID number i, and θ is a threshold which decides the left and right partitions of SMART samples in that node. To assess the quality of each test s in the set of N random tests $\mathcal{S} = \{s_1, \ldots, s_N\}$, we use the *Gini Impurity* as the impurity measurement. The Gini Impurity of node $\mathcal{D}$ is calculated as

$$G(\mathcal{D}) = p_0(1 - p_0) + p_1(1 - p_1) \tag{1}$$

where p_0, p_1 are restricted by $p_0 + p_1 = 1$, denoting the label density of the two classes in node $\mathcal{D}$, respectively. Note that $G(\mathcal{D}) \in [0, 0.5]$ and a larger value implies that the dataset is more impure. For a random test $s \in \mathcal{S}$, we assume that node $\mathcal{D}$ is split into left child node $\mathcal{D}_{ls}$ and right child node $\mathcal{D}_{rs}$ according to the test, and the information gain of test s is measured as

$$\Delta G(\mathcal{D}, s) = G(\mathcal{D}) - \frac{|\mathcal{D}_{ls}|}{|\mathcal{D}|} G(\mathcal{D}_{ls}) - \frac{|\mathcal{D}_{rs}|}{|\mathcal{D}|} G(\mathcal{D}_{rs}) \tag{2}$$

where $|\mathcal{D}|$ represents the number of samples contained in node $\mathcal{D}$. Aiming at reducing the impurity of $\mathcal{D}$, we prefer the Gini Impurity of child nodes to be as small as possible. Hence, when a node meets split conditions, the algorithm firstly calculates the information gain of each test created for the node and then chooses the one with highest gain as the splitting function of the node.

Since each decision node gathers statistics of the sequentially collected training samples on-the-fly, two hyper-parameters: *MinGain* and *MinParentSize* [22], are used to control the node splitting. *MinParentSize* (symbolized as α) limits the minimum number of samples a node should see before it splits, ensuring the node has rubust statistics. *MinGain* (symbolized as β) limits the minimum gain a split should achieve, ensuring the split is worthwhile for the predition purpose. Therefore, the essential condition for the node $\mathcal{D}$ to split is $|\mathcal{D}| \geq \alpha$ and $\exists s \in \mathcal{S} : \Delta G(\mathcal{D}, s) \geq \beta$.

3.2 Online Learning for Disk Failure Prediction

Previous studies leverage offline machine learning techniques to design disk failure prediction model, however, their proper functioning relies on the assumption that the entire training data is accessible and sufficient enough to contain the information of building high-performance models. The practical case is that the training data is gradually gathered instead of being given in advance. This implies that the training data collected within the initial period may be insufficient and become the bottleneck of performance. Therefore, the deployment of models that support online learning is more enlightened. More importantly, online learning supports automatic evolution with the sequential arrival of data and is highly adaptive to the dynamic distribution of SMART data, and thus get rid of the model aging problem.

Our online learning technique innovates the prediction model with ORF algorithm, which inherits all the merits of RF and possesses beneficial properties for predicting disk failure. First, the overall training and testing procedures of ORF can be easy parallelized, as each tree in a forest is built and tested independently from others, which makes the time efficiency of ORF much higher than that of gradient boosting methods, such as GBDT (*Gradient Boosted Decision Trees*). Second, ORF models are highly interpretable so they can be used to reveal the real cause of disk failures and help improving the reliability of storage systems [23]. Additionally, ORFs are also more robust against label noise compared to boosting and other ensemble methods [22].

We build prediction models using SMART attributes as explanatory variables together with the response variable representing whether the disk will fail within the next seven days. A major challenge of fitting high quality models is the highly unbalanced distribution of SMART training data, since healthy drives account for the vast majority. Note that in the original ORF algorithm, all the training samples are modeled by a same Poisson distribution function, that is to say, both positive and negative classes are treated equally. As a result, the prediction models may exhibit poor performance, as the classification algorithms are typically optimized to maximize the overall accuracy [20] and thus seriously biased towards the negative class. To address this issue, we put forward a strategy developed from online bagging [21] and introduce two hyper-parameters: λ_p and λ_n. We denote the t^{th} tree in the forest as $f_t(x)$ and the entire forest is denoted as $\mathcal{F} = \{f_1, ..., f_T\}$, where T is the number of trees in the forest. When a training sample $\langle \vec{x}, y \rangle$ arrives, each tree $f_t \in \mathcal{F}$ will be updated on the sample for k times. The k for positive and negative sample is generated from $Poisson(\lambda_p)$ and $Poisson(\lambda_n)$, respectively. This strategy can be formulated as

$$k(\langle \vec{x}, y \rangle) = \begin{cases} Poisson(\lambda_p) & \text{y=1} \\ Poisson(\lambda_n) & \text{y=0} \end{cases}, \tag{3}$$

Algorithm 1 ORF for disk failure prediction

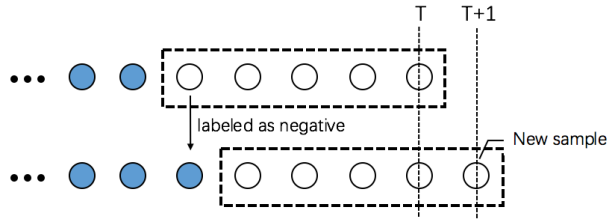
Input: Sequential training sample: $\langle \vec{x}, y \rangle$
Input: Number of trees in forest: T
Input: MinParentSize: α; MinGain: β
Input: Sample balance parameters: λ_p , λ_n
Output: ORF model for disk failure prediction

// For each tree $f_t \in \mathcal{F}$
for $t = 1 \rightarrow T$ **do**
  if $y == 1$ **then**
    $k \leftarrow Poisson(\lambda_p)$
  else
    $k \leftarrow Poisson(\lambda_n)$
  end if
  if $k > 0$ **then**
    // Update the tree f_t for k times
    for $i = 1 \rightarrow k$ **do**
      $\mathcal{D}_j$ = FindLeaf($\vec{x}$)
      UpdateNode($\mathcal{D}_j$, $\langle \vec{x}, y \rangle$)
      if $|\mathcal{D}_j| \geq \alpha$ and $\exists s \in \mathcal{S} : \Delta G(\mathcal{D}_j, s) \geq \beta$ **then**
        // Find the best split from $\mathcal{S}$
        s_j = argmax$_{s \in \mathcal{S}}$ $\Delta G(\mathcal{D}_j, s)$
        CreateLeftChild($\mathcal{D}_{jls}$)
        CreateRightChild($\mathcal{D}_{jrs}$)
      end if
    end for
  else
    // Update the OOBE of tree f_t
    $OOBE_t \leftarrow$ UpdateOOBE($\langle \vec{x}, y \rangle$)
    // Estimate whether f_t is decayed
    if $OOBE_t > \theta_{OOBE}$ and $AGE_t > \theta_{AGE}$ **then**
      // Replace the outdated tree with new tree
      f_t = NewTree()
    end if
  end if
end for

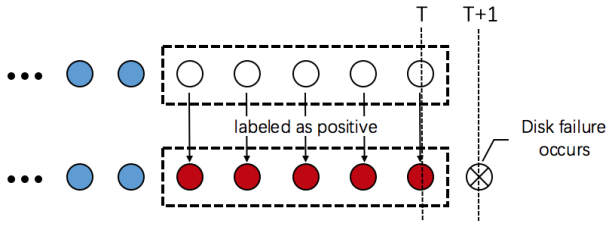
where $k(\langle \vec{x}, y \rangle)$ denotes the update frequency k for sample $\langle \vec{x}, y \rangle$. The hyper-parameters λ_p and λ_n are usually set to constant numbers. In our case, we set λ_p equal to 1 and λ_n to be a decimal much less than 1 (such as 0.01), thus the k for negative sample has a much smaller chance of taking non-zero positive integer value than that for positive sample. As a result, sequentially arrival negative samples are relatively rarely selected by a tree for update.

For compared offline models, we introduce a hyper-parameter: *NegSampleRadio* (symbolized as λ), to balance the training data. Given a training set D, only its subset $D_p + D_{nc}$ is specified as the actual input of the RF model, where D_p represents the set of positive samples and D_{nc} represents a subset of negative samples, which is randomly selected from D_n. The hyper-parameter λ is used to modify the probability distributions of the positive and negative samples, formulized as

$$\lambda = \frac{|D_{nc}|}{|D_p|} \tag{4}$$



(a) If new sample arrives at T+1, the last sample in T's queue will be labeled as negative.



(b) If disk fails at T+1, all samples in T's queue will be labeled as positive.

Figure 1: Automatic online label method. White dots stand for unlabeled samples, blue and red dots stand for negative and positive samples respectively.

and is usually set to a constant number greater than 1. The impact of λ, λ_p ,and λ_n on the prediction performance of offline RF and ORF models is discussed in detail later in section 4.4.

Additionally, our ORF-based prediction model provides a mechanism of unlearning old information, which can be very useful since the underlying distribution of SMART data changes over time. For each tree $f_t \in \mathcal{F}$, the sample $\langle \vec{x}, y \rangle$ can be used to update the tree only if this k takes a non-zero positive integer. When the k takes a value of zero from $Poisson(\lambda)$, the sample $\langle \vec{x}, y \rangle$ then will be used to update the $OOBE$ of tree f_t (denoted as $OOBE_t$). Based on the estimate of $OOBE_t$ and the age of tree f_t (denoted as AGE_t), ORF algorithm discards the outdated trees with significant out-of-bag-error and generates new trees as replacements. As the effect of a single tree in the forest is relatively small, it should be unharmful for the ensemble to discard one tree. However, continuously replacing decayed trees with new fitting trees ensures our prediction models great adaptivity to distribution changes throughout time. The improved ORF algorithm for disk failure prediction is shown in Algorithm 1.

As described in Section 1, sample labeling can be another challenge when prediction models operate in online mode. Unlike offline models learning from static training data, online ones evolve with the sequentially gathered training samples on-the-fly. However, currently collected samples can not be easily labeled because the actual status of the disks are still uncertain. That is to say, some of the disks that are still in operation may be very risky and will fail soon, but their samples can only be labeled as positive after the occurrence of actual failures. We temporarily store the samples collected within the most recent period for each disk and keep them unlabeled unless there is enough confidence to label them. The proposed automatic *online label* method is shown in Figure 1.

Algorithm 2 ORF-based online learning

Input: Gathered SMART sample: $\vec{x}$ (can be NULL if $y = 1$)
Input: Disk identifier: i; Current disk status: y
Output: Prediction for disk: y'

```
// Model update phase
if y == 1 then
    // Disk D_i failed
    while Q_i is not empty do
        x' ← dequeue(Q_i)
        updateORF(⟨x', 1⟩) // call Algorithm 1
    end while
    deleteDisk(D_i)
else
    // Disk D_i is operating
    if isFull(Q_i) then
        x' ← dequeue(Q_i)
        updateORF(⟨x', 0⟩) // call Algorithm 1
    end if
    enqueue(Q_i, x)
    // Prediction phase
    y' ← predictORF(x)
    if y' == 1 then
        // D_i is risky
        // Immediate data migration is recommended
        Trigger an alarm
    end if
end if
```

We denote disk drive with identifier i in the system as D_i. A fixed-length queue, denoted as Q_i, is initialized for D_i to store its last reported SMART samples. These samples stored in Q_i will remain unlabeled until D_i actually fails or new sample arrives. After D_i has failed, all the samples in the queue Q_i will be labeled as positive and then used to update ORF model. While D_i is still in operation, the oldest samples in Q_i will be labeled as negative and replaced continuously with new ones. Meanwhile, ORF model is applied to forecast the health status of D_i with currently gathered SMART sample, and a positive prediction indicates a high risk of D_i. Our detailed ORF-based online learning method is shown in Algorithm 2.

4 EXPERIMENTAL RESULTS

This section aims to evaluate the prediction performance of the ORF model in practice. It is necessary to simulate the long-term use of such online learning mechanism and compare it with the manual update strategies for offline models.

4.1 A Look at the Field Data

To validate our online learning method, we use the public dataset provided by Backblaze [19] which took a daily snapshot of each operational disk drive in their data centers since 2013. The snapshot includes all SMART values reported by a drive along with some basic information, *e.g.,* time stamp, the drive's serial number, and model number. There are over 100, 000 disks recorded in the dataset, covering more than 30 different disk models. Note that the SMART attributes are manufacturer-specific as their encoding and meaning are distinct across disk models. Prior works [14, 20] pointed out that separate training is in demand for different disk models, by which the entire dataset is divided. Since some disk models have very small population and few available SMART attributes, we select two models with the highest data volume to conduct our experiments. Table I presents the statistics of chosen disk models. Compared with the dataset used by Wang *et al.* [14], our datasets cover a much longer duration of sample acquisition and contain more failed drives, which allow us for better simulation and assessment.

Table 1: Overview of dataset

	STA	***STB***
DiskModel	ST4000DM000	ST3000DM001
Capacity(TB)	4	3
#GoodDisks	34,535	2,898
#FailedDisks	1,996	1,357
Duration	39 months	20 months

4.2 Feature Selection

Before in-depth analysis, a prerequisite called 'feature selection' is executed to remove redundant and irrelevant features. This pre-processing can not only reduce the time of model training and prediction, but enhance the prediction performance [10].

For both datasets *STA* and *STB*, each disk drive reports 24 SMART attributes. Note that each of these SMART attributes contains two values: a 6-byte raw value (denoted as *Raw*) and a 1-byte normalized value (denoted as *Norm*) [7]. The *Norm* of a SMART attribute is usually calculated from the *Raw* by a vendor-specific formula. Since some *Norm*s may lose accuracy while their corresponding *Raw*s are more sensitive to health status of disks [14]. We treat both *Norm* and *Raw* of each SMART attribute as candidates, such that there are 48 features in total to explore.

We first operate Wilcoxon rank sum tests on each feature to indicate if it can differentiate between the positive and negative samples, such that 20 features are filtered out by this test since they fail to make a distinction. We then continue to study how the remaining 28 features can contribute disk failure detection separately, by comparing the FDRs of RF models built on different combinations of these features. As a result, nine of them are found to be redundant and thus abandoned. Table II lists the selected 19 features, including 9 *Norm*s and 10 *Raw*s.

The range of values spanned by different features varies widely. To avoid bias towards features with larger values, we apply feature scaling for data normalization, referring to the following formula:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{5}$$

where x is the original value of a feature, x_{max} and x_{min} are the maximum value and the minimum value of this feature for the set of data with the same disk model, respectively.

4.3 Metrics

We use *failure detection rate* (FDR) and *false alarm rate* (FAR) for evaluation. A failed disk is correctly detected only when at lease

Table 2: Selected SMART Features

ID#	Attribute Name	Norm	Raw	Rank*
1	Read Error Rate	√		13
5	Reallocated Sectors Count	√	√	3
7	Seek Error Rate	√		7
9	Power-On Hours		√	5
12	Power Cycle Count		√	11
183	Runtime Bad Block		√	8
184	End-to-End Error	√	√	4
187	Reported Uncorrectable Errors	√	√	1
189	High Fly Writes	√		10
193	Load Cycle Count	√	√	6
197	Current Pending Sector Count	√	√	2
198	Uncorrectable Sector Count	√	√	9
199	UltraDMA CRC Error Count		√	12

*The Rank represents the sort of SMART attribute's contribution.

one of the samples collected within the last week before failure is predicted positive. FDR is defined as the fraction of failed disks that are correctly predicted to be failed:

$$\text{FDR} = \frac{\#\text{true positives}}{\#\text{true positives} + \#\text{false negatives}}$$

A good disk is mis-classified if any of the samples collected outside the latest week is predicted to be positive. FAR means the fraction of good disks that are mis-classified as failed:

$$\text{FAR} = \frac{\#\text{false positives}}{\#\text{true negatives} + \#\text{false positives}}$$

Note that there is usually a trade-off between FDR and FAR, and the FARs should be kept the same when we make comparison with other prediction models.

4.4 Evaluation Results

Experimental Setup. To evaluate our ORF model, disks in each dataset are randomly divided into training set and test set, where the training set contains 70% of all the good and failed disks and the remaining 30% are in the test set.

As mentioned before, our goal is to predict whether a disk will fail within the next seven days. To better simulate the real-world scenario, for each failed disk in the training set, only samples collected within the last week before failure are labeled as positive ($y = 1$). The rest samples of this disk are labeled as negative ($y = 0$), as the disk did not break down within a week after these samples were collected. For each good disk in the training set, however, the samples collected in the latest week can not be labeled, as there is no guarantee that the disk will not fail in the next few days, and the samples collected outside the latest week can be labeled as negative samples. It should be noted that there may be some samples of the failed disks showing obvious fault characteristics but labeled as negative. However, these samples only account for a very small part of all negative samples and have a rare chance of being selected to train or update prediction models. Moreover, both ORF model is highly robust against label noise, thus the impact of these samples on prediction performance is negligible.

Table 3: Impact of λ on Offline RF

λ	*STA*		*STB*	
	FDR(%)	**FAR(%)**	**FDR(%)**	**FAR(%)**
1	98.22 ± 0.25	11.88 ± 2.62	92.26 ± 0.27	6.46 ± 0.54
2	99.02 ± 0.31	2.33 ± 0.95	88.77 ± 0.38	1.35 ± 0.18
3	**98.16±0.74**	**0.76±0.17**	**85.45±0.54**	**0.73±0.06**
4	94.58 ± 0.64	0.05 ± 0.04	82.61 ± 0.65	0.67 ± 0.04
5	92.00 ± 0.14	0.00	80.00 ± 0.55	0.66 ± 0.07
Max	35.14 ± 0.18	0.00	29.45 ± 0.76	0.00

Hyper-parameter Impact. A major impedance for prediction model is the highly unbalanced distribution of training data, since good disks account for the vast majority and only a small fraction of samples reported by failed disks are labeled positive. When trained on intrinsic imbalanced datasets, the outcome of prediction models may be no longer adequate since typical classifiers are optimized to maximize the overall accuracy [20]. To overcome this limitation, we introduce hyper-parameters λ_p, λ_n for the ORF model and λ for the offline RF model to strike a balance of the training data. These hyper-parameters have great impact on the prediction performance. To identify the optimal parameters, we train the models by varying the settings and then apply the final models on the test set.

To be concrete, the number of tests (N) is set to 5,000 and the number of trees (T) in the forest is set to 30. We run experiments with more trees, but no significant improvement is observed. Other important parameters for ORF are set as: $MinParentSize\ \alpha = 200$, $MinGain\ \beta = 0.1$. These parameter settings remain unchanged in the following discussion unless otherwise specified. We repeat each experiment five times and calculate the average and standard deviation of the FDR and FAR.

As described in section 2, the λ is used to downsample the negative class to an amount that is close to the size of the positive class. The impact of λ on the prediction performance of offline RF model is presented in Table III. We can observe that, if no action is taken to balance the training data ($\lambda = Max$), the RF models will be seriously biased towards the good disks and lead to poor FDRs. With the decrease of λ, the FDR of the RF model gradually increases along with the FAR. To ensure an acceptable FAR, we set the $\lambda = 3$ for the RF model as well for other offline models.

λ_p and λ_n are used to adjust the ratio of sequentially arrival positive and negative samples being selected to update the ORF model. A larger value of λ_n means that the negative samples have a greater chance of being selected. We set $\lambda_p = 1$ and the impact of λ_n on the ORF model is presented in Table IV. As expected, adjusting the λ_n provides a trade-off between FDR and FAR. When the λ_n is set to 0.02, our ORF models achieve competitive prediction performance with a 98% FDR for *STA* and a 85% FDR for *STB* at acceptable FARs. In the following experiments, we set $\lambda_p = 1, \lambda_n = 0.02$ for ORF models.

Algorithms Compared. In addition to RF algorithm, we also compare our ORF algorithm with two common classification algorithms: SVM and *decision trees* (DT). For SVM, we use the LIBSVM [28] library. Parameters for SVM are set as follows: *svm_type* = C-SVC,

Table 4: Impact of λ_n on ORF

λ_n	*STA*		*STB*	
	FDR(%)	FAR(%)	FDR(%)	FAR(%)
0.01	98.50 ± 0.19	24.88 ± 3.33	90.91 ± 0.23	5.25 ± 0.58
0.02	**98.08 ± 0.37**	**0.66 ± 0.35**	**85.64 ± 0.37**	**0.85 ± 0.14**
0.03	95.86 ± 0.75	0.10 ± 0.11	74.37 ± 1.75	0.58 ± 0.04
0.05	84.44 ± 0.65	0.01 ± 0.01	59.58 ± 0.50	0.30 ± 0.05
0.10	65.67 ± 3.11	0.00	47.81 ± 0.93	0.12 ± 0.03
1.00	23.58 ± 0.00	0.00	28.23 ± 1.55	0.09 ± 0.03

kernel_type = *radial basis function* (RBF). The *gamma* in RBF kernel and *cost* are adjusted to trade off between FDR and FAR. We perform a grid search to find the parameter combination that produces the highest FDR with a FAR less than 1%. For DT, we use the $fitctree$ [29] function provided by Matlab to build the classification tree model. Some critical parameters are set as follows: $SplitCriterion$ = gdi (Gini's diversity index), $MaxNumSplits$ = 100. Different $Weights$ for positive and negative classes can be used to adjust prediction performance. Other parameters are set to default values.

Evaluation of ORF Model. We simulate the sequential arrival of training data according to the timestamp of labeled samples, and our ORF model evolves with the data over time. During the evolution procedure, we evaluate the prediction performance of the ORF model on the test set monthly. To ensure a fair comparison, each month we build offline models with all the training data collected so far and investigate their performance on the same test set.

Figure 2 and Figure 3 depict the FDRs of ORF and the three offline models on dataset *STA* and *STB*, respectively. All the points on both figures are measured under the constraint that the FAR is around 1.0%. At the beginning, all models exhibit poor prediction accuracy and the FAR of them even cannot be adjusted below 2% due to the fact of lacking valid samples. Thus, we do not plot all the figures starting from the first month. In addition, the curves after the 21^{st} month are omitted since all models tend to stabilize their performance after the 15^{th} month. As we expected, the offline RF model shows better prediction performance than SVM and DT models. Furthermore, the experiments on dataset *STA* show that our ORF model can converge rapidly (within six months) to the performance of the offline RF and achieves stable FDRs of 93-99% after the first six months. Note that in the 6^{th} month, there are only 299 positive samples (45 failed disks) in the training set, which account for only 4% of all positive samples. However, the ORF model still achieves desirable prediction performance. This observation can demonstrate the applicability of our ORF model in small-scale data centers, where the valid data is very limited. The results in Figure 3 also indicate that our ORF model can perform comparable performance against offline RF and outperforms the other two offline algorithms. In summary, our ORF model demonstrates its superiority over the offline counterparts for accurately predicting disk failure even with highly unbalanced datasets.

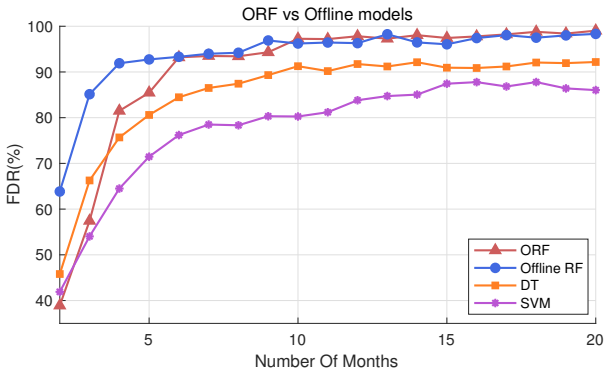


Figure 2: Failure detection rates of ORF and offline models on dataset *STA*. All points on each curve ensure FARs around 1.0%.

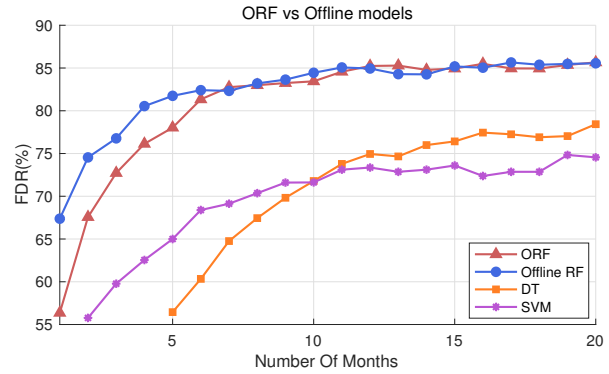


Figure 3: Failure detection rates of ORF and offline models on dataset *STB*. All points on each curve ensure FARs around 1.0%.

4.5 Simulating Practical Long-term Use

Having understood the performance improvement of ORF over the offline counterparts, we further simulate practical long-term use in data centers. Since the underlying distribution of SMART attributes changes over time, the long-term use of offline trained model without update may cause seriously decline in performance. ORF-based online learning method is claimed to evolve with the sequential arrival of data and possess adaptivity to new patterns of training data. To assess the effectiveness of the automatic evolution mechanism, we compare the ORF-based method with two update strategies for offline models: the *accumulation* update strategy and the 1-month *replacing* strategy [14]. Recall that the accumulation strategy updates the model periodically, *e.g.*, once a month, using all the data collected from the beginning, while the 1-month replacing strategy only uses the data collected within the current month.

To evaluate the performance of these update methods, we firstly label the samples the same way we did in Section 4.4, and then we divide all the labeled samples in a dataset according to their timestamp and each subset contains the samples collected in the same month. For convenience, we denote the i^{th} month's subset of

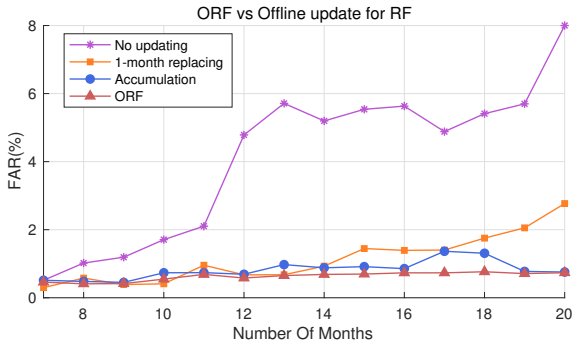


Figure 4: FARs of ORF and monthly updated RFs on dataset *STA*.

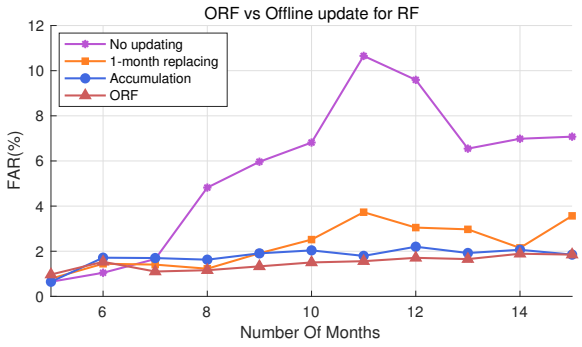


Figure 5: FARs of ORF and monthly updated RFs on dataset *STB*.

STA as STA_i, where $i \in \{1, \ldots, 39\}$. Without any update strategy, we train the offline model using samples collected within the first few months (*i.e.*, first six months for *STA* and first four months for *STB*) and then apply it to test the samples collected in the following months. The accumulation update strategy uses all the samples in $\{STA_1, \ldots, STA_{i-1}\}$ to train the offline model, while the 1-month replacing strategy uses only the samples in STA_{i-1}, and then the updated models are applied to test STA_i, where $i = 7, \ldots, 39$. As for our ORF-based model, we directly apply the model trained in $(i-1)^{th}$ month to test STA_i and no offline retraining is needed. Note that the way we evaluate the models in the i^{th} month is different (mainly on the test set) from what we did in the previous subsection. We previously focused on comparing the ability of different models when they were used for disk failure prediction, and here we want to evaluate the actual performance of the models in practical use.

We test the ORF model and different offline update strategies for RF model on both dataset *STA* and *STB*. Figure 4 and Figure 5 show the FARs of the two models when they are in long-term use. Figure 6 and Figure 7 shows the FDRs of the models. As can be seen, if the RF model is used without updates, its FAR will gradually increase along with the descent of FDR. This illustrates that offline models may gradually lose their effectiveness for disk failure prediction as time goes on. Note that a FAR greater than 5% is unacceptable because it implies too many false alarms and results in heavy processing cost. We can conclude that it is essential for offline trained prediction models to be updated periodically, which is also confirmed by work [14].

The accumulation update strategy works, since retrained models can well fit the current training data and achieve good prediction performance in the near months. Figure 6 shows the FDR of these monthly updated RF models varies obviously from 93%-100%. This is because there are significant differences between the number of failed disks and the number of unpredictable failures[1] in each month. By contrast, the 1-month replacing strategy performs a little worse, especially in stability. One possible reason is that the newly updated models are trained with only the samples collected in one month and thus have a less robust estimate of the statistics.

[1]The SMART attributes of these disks do not show obvious fault characteristic. These disk failures are mainly caused by mechanical or electronic component problems.

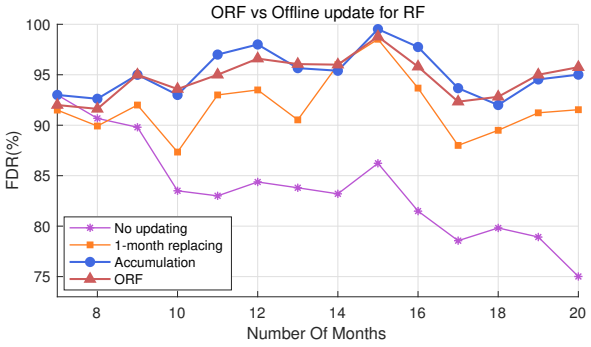


Figure 6: FDRs of ORF and monthly updated RFs on dataset *STA*.

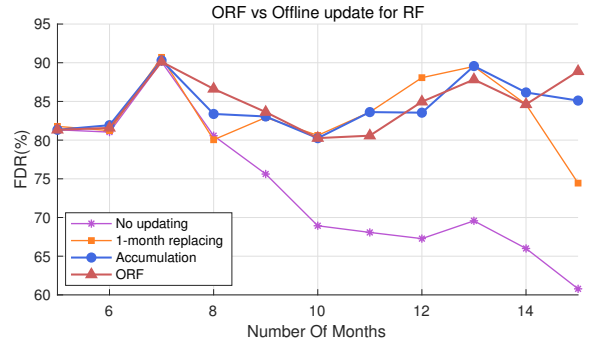


Figure 7: FDRs of ORF and monthly updated RFs on dataset *STB*.

Compared with these update strategies for RF model, it is shown that our ORF-based algorithm can maintain reasonably lower FARs while achieve comparable FDRs with these periodically updated RF models. Moreover, no model retraining is required after the initial deployment.

5 CONCLUSION

In this paper, we present a novel way for proactive fault tolerance using the online learning method. There exist two major challenges for training disk failure prediction models in online mode: 1) how to label the sequentially gathered samples on-the-fly? 2) how to overcome the highly imbalance distribution of healthy and failed disks? For the former, we introduce an automatic *online label* method which temporarily keep the samples collected in the latest days to be unlabeled until a disk failure occurs or new samples arrive. For the latter, two Poisson distributions are proposed to model the sequential arrival of positive and negative samples, and the negative samples have a smaller chance of being selected by the model for update. Consequently, negative samples for training can be comparable with the size of positive samples.

Additionally, an ORF-based prediction model is built to support our method. Compared to the offline models proposed by previous works, our ORF model has lower memory requirements, good robustness against label noise, and better prediction performance. The experiments on real-world datasets show that our ORF models can achieve FDRs of 93-99% with reasonable low FARs. In addition, our ORF model possesses special superiority for practical use, as it can perform automatic update with sequential arrival of data in real-time and thus is highly adaptive to the dynamic distribution of training data. As a result, our online learning method can get rid of the model aging problem and no offline update is needed. To evaluate the effectiveness of our method, we simulate the practical long-term use of ORF models and compare it with update strategies for offline RF models. The results show that our ORF-based algorithm can maintain reasonably lower FARs while achieve comparable FDRs with these periodically updated RF models. Thus, we demonstrate the ability of our online learning method on maintaining stable prediction performance for the practical long-term use.

Finally, although our method is built and evaluated on two disk models from Seagate, it can be easily applied to other disk models and manufacturers as long as SMART is supported. Moreover, our method should work for a wide range of detection applications where the training data becomes available sequentially or it is necessary for the algorithm to automatically adapt to new patterns of data.

ACKNOWLEDGMENTS

This work is supported by National Key Research and Development Program under Grant 2018YFB1003600, National Science Foundation of China under Grant No.61702203, Science and Technology Planning Project of Guangdong Province in China under Grant 2016B030305002, and Pre-research Project of Beifang under Grant FFZ-1601.

REFERENCES

[1] B. Schroeder and G. A. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proceedings of 5th USENIX Conference on File and Storage Technologies (FAST'07)*, pp. 1–16, Feb. 2007.

[2] I. Manousakis, S. Sankar, G. McKnight, Thu D. Nguyen, and R. Bianchini. Environmental Conditions and Disk Reliability in Free-Cooled Datacenters. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST'16)*, pp. 53–65, Feb. 2016.

[3] Data center downtime costs. *http://www.emerson.com/en-us/News/Pages /Net-Power-Study-Data-Center.aspx*.

[4] L. N. Bairavasundaram, G. R. Goodson, B. Schroeder, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. An analysis of data corruption in the storage stack. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08)*, Feb. 2008.

[5] E. Brewer, L. Ying, L. Greenfield, R. Cypher, and Theodore. Disks for Data Centers. Keynote talk for *FAST'16*, Technical report, Google, Feb. 2016.

[6] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-performance, reliable secondary storage. In *ACM Computing Surveys (CSUR)*, vol 26, no. 2, pp. 145–185, June, 1994.

[7] S.M.A.R.T. *https://en.wikipedia.org/wiki/S.M.A.R.T.*

[8] G. Hamerly and C. Elkan. Bayesian approaches to failure prediction for disk drives. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pp. 202–209, Jun. 2001.

[9] G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan. Improved disk-drive failure warnings. In *IEEE Transactions on Reliability*, vol. 51, no. 3, pp. 350–357, Sept. 2002.

[10] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple instance application. In *Journal of Machine Learning Research*, vol. 6, pp. 783–816, May. 2005.

[11] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma. Proactive drive failure prediction for large scale storage systems. In Proceedings of *29th IEEE Conference on Massive Storage Systems and Technologies (MSST)*, pp. 1–5, May. 2013.

[12] Y. Wang, Q. Miao, E. W. Ma, K.-L. Tsui, and M. G. Pecht. Online anomaly detection for hard disk drives based on mahalanobis distance. In *IEEE Transactions on Reliability*, vol. 62, no. 1, pp. 136–145, Mar. 2013.

[13] Y. Wang, W.M. Ma, W. S. Chow, and K.-L. Tsui. A Two-Step Parametric Method for Failure Prediction in Hard Disk Drives. In *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 419–430, Feb. 2014.

[14] B. Zhu, G. Wang, X. Ji, J. Li, and Y. Jia. Hard Drive Failure Prediction Using Classification and Regression Trees. In Proceedings of *44th IEEE Conference on Dependable Systems and Networks (DSN)*, pp. 383–394, Dec. 2014.

[15] C. Xu, G. Wang, X. Liu, D. Guo, and T. Liu. Health Status Assessment and Failure Prediction for Hard Drives with Recurrent Neural Networks. In *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3502-3508, Nov. 2016.

[16] J. Li, R. J. Stones, G. Wang, X. Liu, Z. Li, and M. Xu. Hard drive failure prediction using Decision Trees. In *Reliability Engineering and System Safety*, vol. 164, pp. 55–65, Mar. 2017.

[17] J. Li, R. J. Stones, G. Wang, Z. Li, X. Liu, and X. Kang. Being accurate is not enough: New metrics for disk failure prediction. In *Proceedings Symposium on Reliable Distributed Systems (SRDS)*, pp. 71–80, Dec. 2016.

[18] O. Fontenla-Romero, B. Guijarro-Berdinas, D. Martinez-Rego, B. Perez-Sanchez, and D. Peteiro-Barral. Online machine learning. In *Efficiency and Scalability Methods for Computational Intellect*, pp. 27–54, 2013.

[19] The backblaze hard drive data and stats. *https://www.backblaze.com/b2/hard-drive-test-data.html*.

[20] M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann. Predicting Disk Replacement towards Reliable Data Centers. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 39–48, Aug. 2016.

[21] N. C. Oza and S. J. Russell. Online bagging and boosting. In *Artificial Intelligence and Statistics*, pp. 105–112, 2001.

[22] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line Random Forests. In Proceedings of *12th International Conference on Computer Vision Workshops (ICCV)*, pp. 1393–1400, 2009.

[23] V. Agrawal, C. Bhattacharyya, T. Niranjan, and S. Susarla. Discovering rules from disk events for predicting hard drive failures. In Proceedings of *International Conference on Machine Learning and Applications*, pp. 782–786, Dec. 2009.

[24] L. Breiman. Random forests. *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.

[25] F. Mahdisoltani, I. Stefanovici, and B. Schroeder. Proactive error prediction to improve storage system reliability. In *Proceedings of the 2017 USENIX Annual Technical Conference (USENIX ATC'17)*, pp. 391–402, Jul. 2017.

[26] L. Breiman. Out-of-bag Estimation. *Citeseer*, 1996.

[27] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *Proceedings of 5th USENIX Conference on File and Storage Technologies (FAST'07)*, pp. 17–29, Feb. 2007.

[28] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. In *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, Apr. 2011.

[29] Fit binary classification tree for multiclass classification. *https://cn.mathworks.com/help/stats/fitctree.html*.