Improving Service Availability of Cloud Systems by Predicting Disk Error

Yong Xu[1], Kaixin Sui[1], Randolph Yao[2], Hongyu Zhang[3], Qingwei Lin[1],
Yingnong Dang[2], Peng Li[4], Keceng Jiang[1], Wenchi Zhang[1], Jian-Guang Lou[1],
Murali Chintalapati[2], Dongmei Zhang[1]
[1]*Microsoft Research, China* [2]*Microsoft Azure, USA*
[3]*The University of Newcastle, Australia* [4]*Nankai University, China*

Abstract

High service availability is crucial for cloud systems. A typical cloud system uses a large number of physical hard disk drives. Disk errors are one of the most important reasons that lead to service unavailability. Disk error (such as sector error and latency error) can be seen as a form of gray failure, which are fairly subtle failures that are hard to be detected, even when applications are afflicted by them. In this paper, we propose to predict disk errors proactively before they cause more severe damage to the cloud system. The ability to predict faulty disks enables the live migration of existing virtual machines and allocation of new virtual machines to the healthy disks, therefore improving service availability. To build an accurate online prediction model, we utilize both disk-level sensor (SMART) data as well as system-level signals. We develop a cost-sensitive ranking-based machine learning model that can learn the characteristics of faulty disks in the past and rank the disks based on their error-proneness in the near future. We evaluate our approach using real-world data collected from a production cloud system. The results confirm that the proposed approach is effective and outperforms related methods. Furthermore, we have successfully applied the proposed approach to improve service availability of Microsoft Azure.

1 Introduction

In recent years, software applications are increasingly deployed as online services on cloud computing platforms, such as Microsoft Azure, Google Cloud, and Amazon AWS. As cloud service could be used by millions of users around the world on a 24/7 basis, high availability has become essential to the cloud-based services. Although many cloud service providers target at a high service availability (such as 99.999%), in reality, service could still fail and cause great user dissatisfaction and revenue loss. For example, according to a study conducted on 63 data center organizations in the U.S, the average cost of downtime has steadily increased from $505,502 in 2010 to $740,357 in 2016 (or a 38 percent net change) [33].

Various software, hardware, or network related problems may occur in a cloud system. Our experience with Microsoft Azure shows that disk problem is the most severe one among hardware issues. A typical cloud system like Azure uses hundreds of millions of hard disk drives. Disk-related problem has become one of the most significant factors that contribute to the service downtime. The importance of disk problem is also observed by researchers in Facebook and Google, who reported that 20-57% of disks experience at least one sector error in 4-6 years [27, 35].

To improve service availability, many proactive disk failure prediction approaches have been proposed [18, 31, 32, 42, 41]. These approaches train a prediction model from historical disk failure data, and use the trained model to predict if a disk will fail (i.e., whether a disk will be operational or not) in near future. Proactive actions, such as replacement of failure-prone disks, can then be taken. The prediction model is mainly built using the SMART [1] data, which is disk-level sensor data provided by firmware embedded in disk drives.

The existing approaches focus on predicting *complete disk failure* (i.e., disk operational/not operational). However, in a cloud environment, before complete disk failure, upper-layer services could already be affected by *disk errors* (such as latency errors, timeout errors, and sector errors). The symptoms include file operation errors, VM not responding to communication requests, etc. Disk errors can be seen as a form of *gray failure* [22], which are fairly subtle failures that can defy quick and definitive detection by a conventional system failure detector, even when applications are afflicted by them. Gunawi et al. also pointed out the impact of fail-slow hardware that is still functional but in a degraded mode [20].

If no actions are taken, more severe problems or even service interruptions may occur. Therefore, we advocate that it is important to predict disk errors so that proactive measures can be taken before more severe damage to the service systems incur. The proactive measures include error-aware VM allocation (allocating VMs to healthier disks), live VM migration (moving a VM from a faulty disk to a health one), etc. In this way, service availability can be improved by predicting disk errors.

In this paper, we develop an online prediction algorithm for predicting disk errors, aiming at improving service availability of a cloud service system. We find that disk errors can be often reflected by system-level signals such as OS events. Our approach, called CDEF (stands for Cloud Disk Error Forecasting), incorporates both SMART data and system-level signals. It utilizes machine learning algorithms to train a prediction model using historical data, and then use the built model to predict the faulty disks. We design the prediction model to have the following abilities:

- Be able to rank all disks according to the degree of error-proneness so that the service systems can allocate a VM to a much healthier one.
- Be able to identify a set of faulty disks from which the hosted VMs should be live migrated out, under the constrains of cost and capacity.

However, it is challenging to develop an accurate disk error prediction model for a production cloud system. We have identified the following challenges:

1. In real-world cloud service systems, the extremely imbalanced data make prediction much more difficult. In average, only about 300 out 1,000,000 disks could become faulty every day. We need to identify the faulty disks and be careful not to predict a healthy disk as faulty. In our work, we propose a cost-sensitive ranking model to address this challenge. We rank the disks according to their error-proneness, and identify the faulty ones by minimizing the total cost. Using the cost-sensitive ranking model, we only focus on identifying the top r most error-prone disks, instead of classifying all faulty disks. In this way, we mitigate the extreme imbalance problem.

2. Some features, especially system-level signals, are time-sensitive (their values keep changing drastically over time) or environment-sensitive (their data distribution would significantly change due to the ever-changing cloud environment). We have found that models built using these unstable features may lead to good results in cross-validation (randomly dividing data into training and testing sets) but perform poorly in real-world online prediction (dividing data into training and testing sets by time). We will elaborate this challenge in Section 2.2. To address this challenge, we perform systematic feature engineering and propose a novel feature selection method for selecting stable and predictive features.

We evaluate our approach using real-world data collected from a production cloud system in Microsoft. The results show that CDEF is effective in predicting disk errors and outperforms the baseline methods. We have also successfully applied CDEF in industrial practice. In average, we successfully reduce around 63k minutes of VM downtime of Microsoft Azure per month.

In summary, we make the following contributions in this paper:

- We propose CDEF, a disk error prediction method. In CDEF, we consider both system-level signals and disk-level SMART attributes. We also design a novel feature selection model for selecting predictive features and a cost-sensitive ranking model for ranking disks according to their error-proneness.
- We have successfully applied CDEF to Azure, a production cloud system in Microsoft. The results prove the effectiveness of CDEF in improving service availability in industrial practice. We also share the lessons learned from our industrial practice.

The rest of this paper is organized as follows: In Section 2, we introduce the background and motivation of our work. Section 3 describes the proposed approach and detailed algorithms. The evaluation of our approach is described in Section 4. We also discuss the results and present the threats to validity. In Section 5, we share our experience obtained from industrial practice. The related work and conclusion are presented in Section 6 and Section 7, respectively.

2 Background and Motivation

2.1 Disk Error Prediction

A cloud system such as Microsoft Azure contains hundreds of millions of disks serving various kinds of services and applications. Disks are mainly used in two kinds of clusters, clusters for data storage and clusters for cloud applications. For the former of clusters, redundancy mechanisms such as RAID [30] could tolerate disk failures well. The latter form of clusters hosts a tremendous amount of virtual machines, disk errors could bring undesirable disruptions to the services and applications. In this paper, we focus on the disks used in the cloud application cluster.

For cloud systems such as Microsoft Azure, Amazon AWS, and Google Cloud, service problems can lead to great revenue loss and user dissatisfaction. Hence, in today's practice, the service providers have made every effort to improve service availability. For example, from "four nines" (99.99%) to "five nines" (99.999%), and then to "six niness"(99.9999%). Disks are among the most frequently failing components in a cloud environment and have attracted much attentions from both academia and industry. For example, BackBlaze publishes quarterly reports and the underlying data for users to keep track of reliability of popular hard drives in the market. In their data, disk failure is labelled *0 if the drive is OK, and 1 if this is the last day the drive was operational before failing* [2].

To mitigate cost incurred by disk failures, researchers have proposed to automatically predict the occurrence of disk failure before it actually happens. In this way, proactive measures, such as disk replacement, can be taken. Disk failure prediction has been a hot subject of study. Existing work [9, 18, 31, 32, 41, 42] mostly use the SMART data (Self-Monitoring, Analysis and Reporting Technology, which monitors internal attributes of individual disks) to build a disk failure prediction model.

However, before a disk completely fails, it already started reporting errors. There are various disk errors such as disk partition errors (disk volumes and volume size become abnormal), latency errors (unexpected long delay between a request for data and the return of the data), timeout errors (exceeding the predefined disk timeout value), and sector errors (individual sectors on a drive become unavailable), etc. Disk failures can be detected by a conventional system failure detection mechanisms. These mechanisms often assume an overly simple failure model in which a component is either correct or failed. However, such mechanisms are inadequate to deal with disk errors as they are subtle *gray failures* [22]. In our practice, the disk error data is obtained through root cause analysis of service issues performed by field engineers.

Disk errors are common. For example, a study by Bairavasundaram et al. [8] reports that 5-20% of hard disk drives in Netapps storage systems report sector errors over a period of 24 months. The disk errors can affect the normal operations of upper-layer applications and can be captured by unexpected VM downtime. The symptoms include I/O requests timeout, VM or container not responding to communication requests, etc. If no actions are taken, more severe problems or even service interruptions may occur. Therefore, it is important that disk errors to be captured and predicted before the virtual machines get affected.

2.2 Challenges

In this work, we propose to predict the error-proneness of a disk based on the analysis of historical data. The ability to predict disk error can help improve service availability from the following two aspects:

- VM allocation, which is the process of allocating a VM (virtual machine) to a host. To enable more effective VM allocation, we can proactively allocate VMs to a host with a healthier disk rather than to a host with a faulty disk.
- Live migration, which is the process of moving a running VM among hosts without disconnecting the client or application. To enable more effective live migration, we can proactively migrate VMs from a host with a faulty disk to a host with healthy disks.

To achieve so, we can build a prediction model based on historical disk error data using machine learning techniques, and then use the model to predict the likelihood of a disk having errors in the near future. There are several main technical challenges in designing the disk error prediction model for a large-scale cloud:

Extremely imbalanced data: For a large-scale cloud service system such as Microsoft Azure, each day, at most only 3 disk in ten thousand disks could become faulty. The extreme 3-in-10,000 imbalanced ratio poses difficulties in training a classification model. Fed with such imbalanced data, a naive classification model could attempt to judge all disks to be healthy, because in this way, it has the lowest probability of making a wrong guess. Some approaches apply data re-balancing techniques, such as over sampling and under sampling techniques, to address this challenge. These approaches help raise the recall, but at the same time could introduce a large number of false positives, which dramatically decrease the precision. In our scenario, the cost of false positives is high as the cost of VM migration is inneglectable and the cloud capacity may be affected by the false positives.

Online prediction: Existing work [9, 26] usually deals with prediction problem in a cross-validation manner. However, we found that it is inappropriate for evaluating our disk error prediction model. In cross validation, the dataset is randomly divided into training and testing set. Therefore, it is possible that the training set contains parts of future data, and testing set contains parts of past data. However, when it comes to online prediction (using historical data to train a model and predict future), training and testing data will have no time overlap. Besides, some data, especially system-level signals, are time-sensitive (their values keep changing drastically over time) or environment-sensitive (their data distribution could change due to the ever-changing cloud envi-

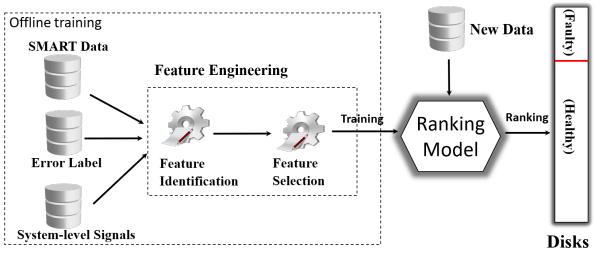


Figure 1: The overview of the proposed approach

ronment). For example, a rack encounters an outage at time t, all disks on it will experience such a change at the same time. Using cross validation, the environment-specific knowledge can spread to both training set and testing set due to random splitting. The knowledge learned from the training set could be applied to the testing set, which causes high accuracy in cross validation but poor result when evaluating new data.

Therefore, to construct an effective prediction model in practice, we should use online prediction instead of cross-validation: the future knowledge should not be known at the time of prediction.

3 Proposed Approach

In this section, we present CDEF (Cloud Disk Error Forecasting), our proposed approach that can improve service availability by predicting disk errors. Figure 2 shows the overview of CDEF. First, we collect historical data about faulty and health disks. The disk label is obtained through root cause analysis of service issues by field engineers. The feature data includes SMART data and system-level signals. We then select for training those features that are stable and predictive. Based on the selected features, we construct a cost-sensitive ranking model, which ranks the disks and identifies the top r ones that minimize the misclassification cost as the predicted faulty disks.

CDEF addresses the challenges described in the previous section by incorporating: 1) a feature engineering method for selecting stable and predictive features 2) a ranking model to increase the accuracy of cost-sensitive online prediction. We describe these two components in this section.

3.1 Feature engineering

3.1.1 Feature Identification

We collect two categories of data, SMART data and system-level signals. SMART (Self-Monitoring, Analysis and Reporting Technology) is a monitoring firmware which allows a disk drive to report data about its internal activity. Table 1 gives some of the SMART features.

Table 1: Examples of SMART features

SMART ID	Description
S2	Start/Stop Count
S12	Power Cycle Count
S193	Load Cycle Count
S187	The number of read errors that could not be recovered using hardware ECC
S5	Count of reallocated sectors. When a read or a write operation on a sector fails, the drive will mark the sector as bad and remap (reallocate) it to a spare sector on disk.
S196	The total count of attempts to transfer data from reallocated sectors to a spare area. Unsuccessful attempts are counted as well as successful.

Table 2: The system-level signals

Signal	Description
PagingError	Windows encounters an error in creating a paging file.
FileSystem-Error	An error occurs when trying to read, write, or open a file.
DeviceReset	Device is forced to reset or shut-down.
TelemetryLoss	Telemetry data cannot be captured over a period.
DataExchange-Disabled	The data exchange integration service cannot be enabled or initialized.
VMFrozen	VM is unresponsible to communication request
Windows Event 129	A Windows event log caused by dropped requests.

More information about SMART can be found in [31].

In cloud systems, there are also various system-level events, which are collected periodically (typically every hour). Many of these system-level events, such as Windows events, file system operation error, unexpected telemetry loss, etc., are early signals of disk errors. Table 2 gives the descriptions of some system-level signals. For example, the *FileSystemError* is an event that is caused by disk related errors, which can be traced back to bad sectors or disk integrity corruption.

Apart from the features that are directly identified from the raw data, we also calculate some statistical features as follows:

Diff Through data analysis, we have found that the changes in a feature value over time could be useful for distinguishing disk errors. We call such a feature $Diff$.

Given a time window w, we define $Diff$ of feature x at time stamp t as follows:

$$Diff(x,t,w) = x(t) - x(t-w) \quad (1)$$

Sigma Sigma calculates the variance of attribute values within a period. Given a time window w, Sigma of attribute x at time stamp t is defined as:

$$Sigma(x,t,w) = E[(X-\mu)^2], \quad (2)$$

where $X = (x_{t-w}, x_{t-w-1}, ..., x_t)$ and $\mu = \frac{\Sigma(X)}{w}$.

Bin Bin calculates the sum of attribute values within a window w as follows:

$$Bin(x,t,w) = \sum_{j=t-w+1}^{t} x(j) \quad (3)$$

In our work, we use three different window sizes 3, 5, 7 in calculating $Diff$, *Bin*, and *Sigma*.

3.1.2 Feature Selection

Through the feature identification process described in the previous section, we have identified 457 features in total from SMART and system-level data. However, we have found that not all of the features can well distinguish between healthy and faulty disks, especially in the context of online prediction.

Feature selection proves very useful in selecting relevant features for constructing machine learning models. Existing feature selection methods fall into two main categories, statistical indicators (Chi-Square, Mutual Information, etc.) and machine-learning based methods like Random Forest [17]. However, in our scenario, the traditional feature selection methods cannot achieve good prediction performance because of the existence of time-sensitive and environment-sensitive features. These features carry information that are highly relevant to the training period, but may not be applicable for predicting samples in the next time period. We call this kind of features non-predictive features, meaning they have no predictive power in online prediction. Our experimental results (to be described in Section 4.3.2) show that the traditional feature selection methods lead to poor performance in our scenario.

Figure 2(b) illustrates an example of a non-predictive feature *SeekTimePerformance*. Line G_train indicates the feature values of healthy disks over time in training set, and Line F_train indicates the feature values of faulty disks in the training set. Clearly, in the training set, the mean feature value of healthy disks is lower than that of faulty disks. We expect the same pattern for the same feature in the testing set (which is collected from the next time period). However, our data shows that it is not the case. In Figure 2(b), Lines G_test and F_test indicate the feature values of healthy and faulty disks over time in the testing set, respectively. Clearly, in the testing set, the mean feature value of healthy disks is higher than that of faulty disks. Therefore, the behavior of this feature is not stable. We consider this feature a non-predictive feature and not suitable for online prediction. As a comparison, Figure 2(a) shows a predictive feature *ReallocatedSectors*, from which we can see that the behavior of this feature is stable - the values of healthy disks are always close to zero and the values of faulty disks keep increasing over time, in both training and testing sets.

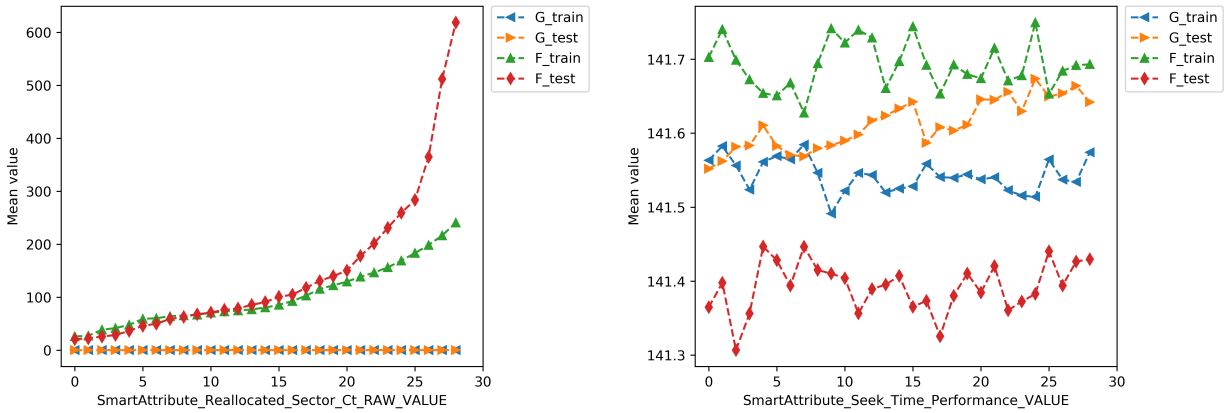
Algorithm 1: Prune non-predictive features

Input : Training data TR with feature set F $(f_1, f_2, , , f_m)$
Output: Reduced feature set F'

```
Split TR by time equivalently into TR1 and TR2
foreach f_i in F do
    // use TR1 to predict TR2, get accuracy result
    r ← train(TR1) and test(TR2)
    // remove data about f_i from TR, then predict
    r_{f_i} ← train(TR1-f_i) and test(TR2-f_i)
    if r_{f_i} > r then
        delete f_i from F
    end
    if number of remaining features <= θ * m
     then
        Break
    end
end
Return F'
```

To select the stable and predictive features, we perform feature selection to prune away the features that will perform poorly in prediction. The idea is to simulate online prediction on the training set. The training set is divided by time into two parts, one for training and the other for validation. If the performance on validation set gets better after deleting one feature, then the feature is deleted until the number of remaining features is less than θ% of the total number of the features. The details are described in Algorithm 1. In our experiment, we set $\theta = 10\%$ by default, which means that the pruning process will stop if the number of remaining features is less than 10%.

At last, we re-scale the range of all selected features using zero-mean normalization as follows: $x_{zero-mean} = x - mean(X)$.



(a) Predictive features (b) Non-predictive feature

Figure 2: An example of predictive and non-predictive feature

3.2 Cost-sensitive ranking model

Having collected features from historical data, we then construct a prediction model to predict the error-proneness of disks in the coming days. In this step, we formulate the prediction problem as a ranking problem instead of a classification problem. That is, instead of simply telling whether a disk is faulty or not, we rank the disks according to their error-proneness. The ranking approach mitigates the problem of extreme imbalanced fault data because it is insensitive to the class imbalance.

To train a ranking model, we obtain the historical fault data about the disks, and rank the disks according to their relative time to fail (i.e., the number of days between the data is collected and the first error is detected). We adopt the concept of Learning to Rank [24], which automatically learns an optimized ranking model from a large amount of data to minimize a loss function. We adopt the FastTree algorithm [28, 14], which is a form of "Multiple Additive Regression Trees" (MART) gradient boosting algorithm. It builds each regression tree (which is a decision tree with scalar values in its leave) in a step wise fashion. This algorithm is widely used in machine learning and information retrieval research.

To improve service availability, we would like to intelligently allocate VMs to the healthier disks so that these VMs are less likely to suffer from disk errors in near future. To achieve so, we identify the faulty and healthy disks based on their probability of being faulty. As most of the disks are healthy and only a small percentage of them are faulty, we select the top r results returned by the ranking model as the faulty ones. The optimal top r disks are selected in such a way that they minimize the total misclassification cost:

$$cost = Cost1 * FP_r + Cost2 * FN_r,$$

where FP_r and FN_r are the number of false positives and false negatives in the top r predicted results, respectively. Cost1 is the cost of wrongly identifying a healthy disk as faulty, which involves the cost of unnecessary live migration from the "faulty" disk to a healthy disk. Although we have very good technology for live migration, the migration process still incurs an unneglectable cost and decreases the capacity of the cloud system. Cost2 is the cost of failing to identify a faulty disk. The values of Cost1 and Cost2 are empirically determined by experts in product teams. In our current practice, due to the concerns about VM migration cost and cloud capacity, Cost1 is much higher than Cost2 (i.e., we value precision more than recall). The ratio between Cost1 and Cost2 is set to 3:1 by the domain experts based on their experience on disk error recovery. The number of false positives and false negatives are estimated through the false positive and false negative ratios obtained from historical data. The optimum r value is determined by minimizing the total misclassification cost. The top r disks are predicted faulty disks, which are high-risk disks and the VMs hosted on them should be migrated out.

4 Experiments

In this section, we evaluate the effectiveness of our approach. The aim is to answer the following research questions:

RQ1: How effective is the proposed approach in predicting disk errors?

RQ2: How effective is the proposed feature engineering method?

RQ3: How effective is the proposed ranking model?

4.1 Dataset and Setup

Dataset To evaluate the proposed approach, we collect real-world data from a large-scale Microsoft cloud service system. We use one-month data (October 2017) for training, and divide the data of November 2017 into three parts for testing. In each dataset, the ratio between healthy disks and faulty disks is around 10,000 : 3.

Setup We utilize Microsoft COSMOS [3] to store and process data collected from product teams. For ranking algorithm, we use the FastTree algorithm implemented in Microsoft AzureML [4]. We use 200 iterations in FastTree setting. The experimental evaluation is performed on a Windows Server 2012 with (Intel CPU E5-4657L v2 @2.40GHz 2.40 with 1.0 TB Memory).

4.2 Evaluation Metric

Following the existing work [23, 32, 42], we evaluate the accuracy of the proposed approach using the FPR and TPR metrics. We consider faulty disks as positive and healthy ones as negative. True Positive (TP) denotes the faulty disks that are predicted as faulty. False Positive (FP) denotes the healthy disks that are falsely predicted as faulty. True Negative (TN) denotes the healthy disks that are predicted as healthy. False Negative (FN) denotes the faulty disks that are falsely predicted as healthy. False Positive Rate (FPR) denotes the proportion of FP among all healthy disks. $FPR = FP/(FP+TN)$. True Positive Rate (TPR) denotes the proportion of TP among all faulty disks. $TPR = TP/(TP+FN)$.

We also use the ROC curve [5] that plots TPR (True Positive Rate) versus FPR (False Positive Rate), and compute the Area Under Curve (AUC). Following the related work [23, 29], we compute the TPR value when FPR is 0.1%, which indicates how good an algorithm can predict faulty disks under a high precision requirement.

4.3 Results

4.3.1 RQ1: How effective is the proposed approach in predicting disk errors?

We evaluate the effectiveness of the proposed CDEF approach on all three datasets. We also compare CDEF with the Random Forest and SVM based methods proposed in the related work on disk failure prediction [26, 32]. These methods use the Random Forest or SVM classifiers to classify disks based on the SMART data. We treat them as baseline methods in this experiment.

The experimental results are shown in Figure 3. The diagonal lines indicate the accuracy obtained by Random Guess (meaning random prediction with 50% probability). The results show that CDEF outperforms the baseline methods consistently under different FPR/TPR ratios on all datasets. For example, on Dataset 1, the AUC values for our approach is 0.93, while the AUC value for Random Guess, Random Forest, and SVM is 0.5, 0.85, and 0.53, respectively.

We evaluate the effectiveness of the proposed ranking approach in terms of misclassification cost and the TPR value (when FPR is 0.1%). The misclassification cost is obtained as: cost= Cost1*FP+Cost2*FN, where Cost1 and Cost2 are set to 3 and 1 respectively by the product team. Table 3 shows the results. Clearly, CDEF obtains better results than the other two methods. The TPR value is 36.50%, 41.09%, and 29.67% on Dataset 1, 2, and 3, respectively. CDEF is also cost-effective. In average, CDEF achieves around 187.92% cost reduction than Random Forest, and 10.13% cost reduction than SVM. SVM has low cost because SVM is accurate in predicting healthy disks and induces less false positives. But SVM performs worse in predicting faulty disks and induces low TPR.

In summary, the experimental results show that CDEF is effective in predicting disk errors. This is because of two reasons: the proposed feature engineering method and the proposed ranking model. We will show the effectiveness of these two methods in the following RQs.

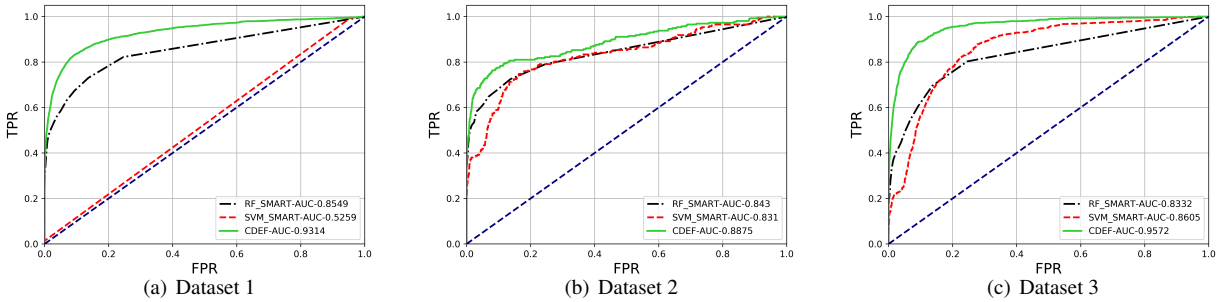
Table 3: Experimental results of CDEF on three datasets

	CDEF		RandomForest		SVM	
	Cost	TPR	Cost	TPR	Cost	TPR
Dataset 1	2508	36.50%	3157	30.51%	2907	15.51%
Dataset 2	234	41.09%	1211	34.11%	258	21.71%
Dataset 3	760	29.67%	1675	18.81%	792	7.20%

4.3.2 RQ2: How effective is the proposed feature engineering method?

In our work, we propose to use system-level signals in disk error prediction. We also propose a feature selection method to select the predictive features for model training. In this RQ, we evaluate the effectiveness of our proposed feature engineering method. We experiment with three feature engineering methods: S (traditional SMART-based features), S+A (SMART and system-level signals), and S+A+F (SMART and system-level signals with feature selection, which is used in CDEF). All other experimental settings remain the same.

The results are shown in Figure 4. We can see that the results achieved by incorporating system-level signals outperform those achieved by SMART alone on all the three datasets. Furthermore, by incorporating SMART and system-level signals with feature selection, we can obtain the best results on all the three datasets. In average, the TPR value (when FPR is 0.1%) is 27.6%, 30.3%, and 35.8%, for S, S+A, and S+A+F, respectively. These results confirm the effectiveness of the proposed feature engineering methods.



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 3: ROC of comparative methods

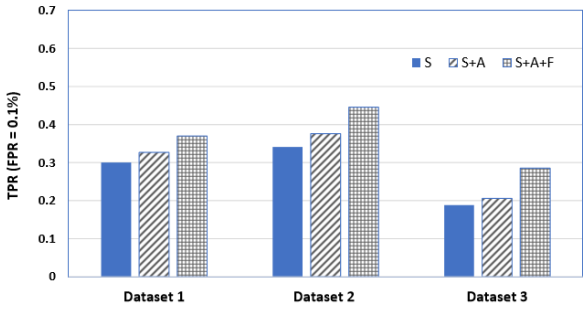


Figure 4: Evaluation results with different features
S: traditional SMART-based features; S+A: SMART and system-level signals; S+A+F: SMART and system-level signals with feature selection.

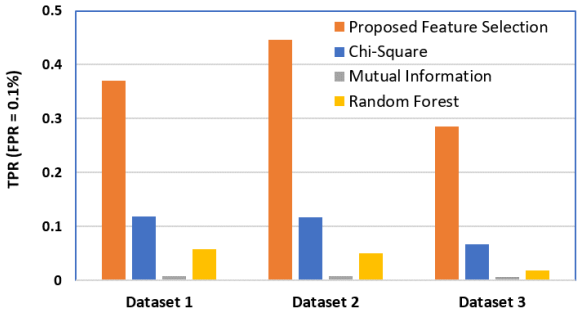


Figure 5: The comparison between the proposed feature selection method and existing methods

We also evaluate the effectiveness of CDEF using the features selected by the proposed feature selection method and the features selected by conventional feature selection methods Chi-Square, Mutual Information, and Random Forest [17, 21]. The results are given in Figure 5, which shows that the proposed feature selection method outperforms the conventional feature selection methods on all datasets.

4.3.3 RQ3: How effective is the proposed ranking model?

In our work, we propose to use a cost-sensitive ranking method to rank the disks and then select the top r disks as faulty ones by minimizing the total misclassification cost. In this RQ, we evaluate the effectiveness of the proposed ranking approach.

To perform classification for imbalanced data, one common approach is to apply the over-sampling technique SMOTE [10] to balance the training data for model construction. The other approach is weighted classification, which is essentially cost-sensitive learning [12] that learns from extremely imbalanced data and assigns a larger weight to minority class. The weight is usually set inversely to the sample portion. In our experiment, we compare the proposed cost-sensitive ranking method with these two approaches. To better evaluate the accuracy of the proposed method, we also compare with the random guess method.

We evaluate the effectiveness of the proposed ranking approach in terms of misclassification cost. The proposed cost-sensitive ranking model achieves the minimum cost among all comparative methods on all datasets. For example, on Dataset 2, the misclassification cost obtained by our model is 234, while cost obtained by Random Guess, weighted classification, and classification with SMOTE are much higher (1146662, 717, and 7812, respectively).

We also evaluate the effectiveness of the proposed ranking approach in terms of TPR and FPR values. Figure 6 shows the ROC curves achieved by the comparative methods. Table 4 shows the TPR values when FPR is 0.1%, achieved by different methods on all the datasets. Clearly, our cost-sensitive ranking method achieves the best accuracy values. For example, on Dataset 2, the TPR value (when FPR is 0.1%) achieved by our model is 41.09%, while the values achieved by Random Guess, weighted classification, and classification with SMOTE are much lower (0.1%, 27.91%, and 27.94%, respec-

tively). The AUC value achieved by our model is 88.75%, while the values achieved by Random Guess, weighted classification, and classification with SMOTE are 0.5%, 84.22%, and 83.56%, respectively.

In summary, the experimental results confirm the effectiveness of the proposed cost-sensitive ranking model.

4.4 Discussions of the Results

In our work, we do not use cross-validation to build and evaluate the proposed approach. Instead, we do online prediction - using the data before a certain date to train the model and use the data after the date to test the model. Existing work on failure prediction such as [9, 26] uses cross-validation to evaluate their machine-learning based models. In our scenario, cross-validation can lead to much better results than online prediction, as shown in Figure 7. For example, on Dataset 1, using cross-validation we can obtain TPR value of 91.64% (when FPR is 0.1%), while using online prediction the TPR value is only 36.50%. However, our experiences show that cross-validation may not always reflect the actual effectiveness of a prediction model. Online prediction should be used in practice.

In cross validation, the dataset is randomly divided into training and testing sets. Therefore, it is possible that the training set contains parts of future data, and the testing set contains parts of past data. However, in real-world online prediction, training and testing sets are divided by time. The past data is used to train the model and the future data is used to test the model.

The gap is magnified when there are time-sensitive features and environment-sensitive features. In disk error prediction, some features have temporal nature and their values vary drastically over time. Some features may be easily affected by environmental changes to the cloud system. For example, the disks on the same rack or the same motherboard encounter similar attribute changes caused by unstable voltage. However, such changes may not happen before the time of prediction. Using cross-validation we may utilize the knowledge that should not be known at the time of prediction, thus obtaining better evaluation results. Therefore, cross-validation is not suitable for evaluating our model in practice. The problem of cross-validation in evaluating an online prediction model is also observed by others [36].

4.5 Threats to Validity

We have identified the following threats to validities:

Subject systems: In our experiments, we only collect data from one cloud service system of one company. Therefore, our results might not be generalizable to other systems. However, the system we studied is a typical, large-scale cloud service system, from which sufficient data can be collected. Furthermore, we have applied our approach in the maintenance of the cloud system. In future, we will reduce this threat by evaluating CDEF on more subject systems and report the evaluation results.

Data noise: After a disk is identified to be faulty, it could be sent to repair. After that, some disks could be returned and used again. Therefore, a small degree of noise may exist in the labeling of a disk.

Evaluation metrics: We used the FPR/TPR metrics to evaluate the prediction performance. These metrics have been widely used to evaluate the effectiveness of a disk fault prediction mode [32]. Prior work [38] points out that a broader selection of metrics should be used in order to maximize external validity. In our future work, we will reduce this threat by experimenting with more evaluation measures such as Recall/Precision.

5 Lessons Learned from Practice

We have successfully applied CDEF to the maintenance of Microsoft Azure, which is a large-scale cloud service system that allows IT professionals to build, deploy, and manage applications. The cloud service achieves global scale on a worldwide network of data centers across many regions. Due to the unreliable nature of the underlying commodity hardware, various issues occur in Azure every day. Without proper handling of these issues, Azure service availability could be seriously affected. We found disk error is the most severe one among all hardware issues.

CDEF is currently used by Azure to preferentially select healthier disks for VM allocation and live migration. After deploying CDEF, in average, we successfully saved around 63k minutes of VM downtime per month. Note that 99.999% service availability means that only 26 seconds per month of VM downtime is allowed. Therefore, CDEF has significantly improved service availability of Microsoft Azure.

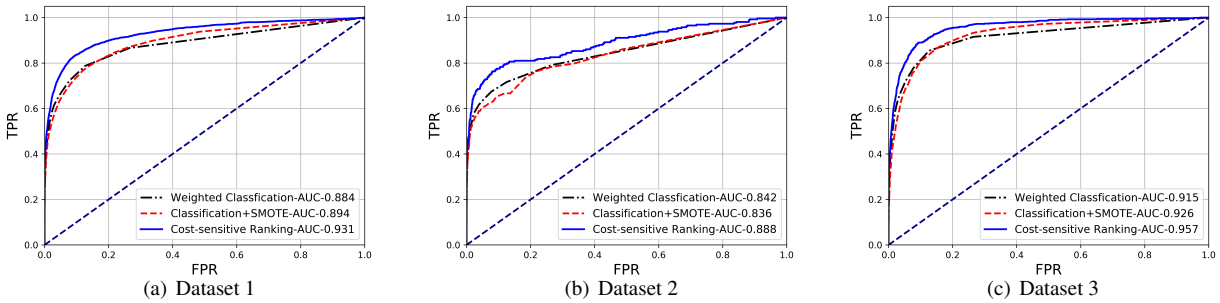
Currently the training is performed daily over the past 90-day data, and keeps a moving window of 90 days. The cutting point r in the ranking model is set along with the training process. When a disk is predicted as faulty, we mark the host node unallocable and trigger live migration process. We also run disk stress test on the predicted disks before they are taken out for replacement.

We have learned the following lessons from our industrial practice:

- **Continuous training**. Many factors could affect the distribution of disk error data, such as bugs in OS driver/firmware, workload on clusters, platform maintenance, etc. A model trained in the past will not always work in the future. Therefore, we build a continuous training pipeline. For every predicted disk error, we also let the disk go through a disk

Table 4: The cost and TPR values (when FPR is 0.1%) achieved by the proposed cost-sensitive ranking model

	Random Guess		Cost-sensitive ranking		Weighted Classification		Classification+SMOTE	
	Cost	TPR	Cost	TPR	Cost	TPR	Cost	TPR
Dataset 1	1447986	0.1%	2508	36.50%	2910	26.52%	9442	24.63%
Dataset 2	1146662	0.1%	234	41.09%	717	27.91%	7812	27.94%
Dataset 3	1446929	0.1%	760	29.67%	1234	17.42%	8239	17.68%



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 6: ROC of cost-sensitive ranking and classification

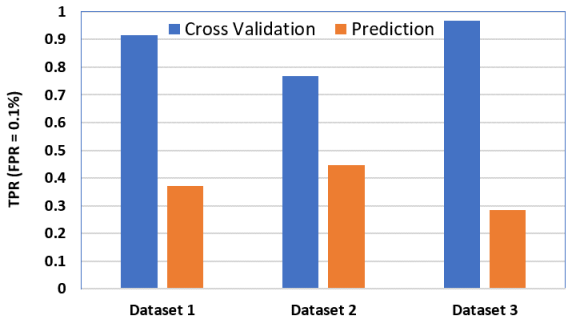


Figure 7: Evaluation results - cross validation vs. online prediction

stress test to check if it is really faulty. This forms a continuous feedback loop between disk error prediction and disk stress test.

- **Cost-effectiveness**. Prediction alone may not make much impact if the cost of recovery operation is really high (because the cost of leaving the host node as it is might be cheaper than the cost of taking the recovery operation). Furthermore, the cost to recover a node with one VM on top is much cheaper than the cost of recovery with 10 VMs in terms of VM availability. Thus, the cost of recovery could vary depending on the state of the host node, the recovery operation, etc. The prediction could be even more useful if we can better estimate the cost.

- **Faulty disks will get even worse.** Our experience shows that before a disk completely fails, it may already start emitting errors that affect upper-layer applications and services. That is why incorporating the system-level signals is better than using SMART alone. We found that disk errors, in average, occur 15.8 days earlier than complete disk failure. Our experience also shows that, before completely fails, the status of a disk will actually get worse over time. For example, for faulty disks, the value of the feature *ReallocatedSectors* increases by 3 times during the last week of its operation. The value of system-level signal *DeviceReset* even increases by 10 times during the same period. This finding confirms our intention to detect disk error earlier before it makes severe impact on application usage.

6 Related Work

6.1 Disk Failure Prediction

There are a large amount of related work on predicting disk failures. For example, BackBlaze publishes quarterly report [6] for users to keep track of reliability of popular hard drives in the market. Most of the modern hard drives support Self-Monitoring, Analysis and Reporting Technology (SMART), which can monitor internal attributes of individual drives. SMART is used by some manufacturers to predict impending drive failure by simple threshold-based method [31, 34].

As the prediction performance of the thresholding algorithm is disappointing, researchers have proposed various machine learning models for predicting disk failures. For example, Zhu et al. [42] predicted disk failure based on raw SMART attributes and their change rates, and neural network and SVM model are applied. Ganguly et al. [16] utilized SMART and hardware-level features

such as node performance counter to predict disk failure. Ma et al. [25] investigate the impact of disk failures on RAID storage systems and designed RAIDShield to predict RAID-level disk failures.

Tan et al. [37] proposed an online anomaly prediction method to foresee impending system anomalies. They applied discrete-time Markov chains (DTMC) to model the evolving patterns of system features, then used tree-augmented naive Bayesian to train anomaly classifier. Dean et al. [11] proposed an Unsupervised Behavior Learning (UBL) system, which leverages an unsupervised method Self Organizing Map to predict performance anomalies. Wang et al. [41] also proposed an unsupervised method to predict drive anomaly based on Mahalanobis distance. There are also other work [19, 40] in online machine learning [7], which aims to update the best predictor at each step for steaming data (as opposed to batch learning techniques). While our "online prediction" focuses on the prediction workflow: always using a batch of historical data to predict the future (as opposed to cross-validation). Furthermore, unlike [37], we deal with the evolving features by proactively selecting the consistently predictive features. Unlike [11, 41] that can be used even when label data is difficult to get, we adopt a supervised method as we have quality labeled data. We will compare our method with unsupervised-learning based methods in our future work.

For feature selection, Botezatu et al. [9] selected the most relevant features based on statistical measures. Gaber et al. [15] used machine learning algorithms to extract features representing the behavior of the drives and predict the failure of the drives. However, these feature selection methods are not able to prune non-predictive features in online prediction scenario.

All these related work focus on disk failure prediction based on SMART and other hardware-level attributes. While our work focuses on predicting disk errors that affect the availability of virtual machines, before complete disk failure happens. We incorporate both SMART and system-level signals, which proves better than using SMART data alone. Also, most of the related work evaluate their prediction model in a cross validation manner, which is not appropriate in real-world practice. In our work, we perform online prediction and propose a novel algorithm to select stable and predictive features.

6.2 Failures in Cloud Service Systems

Although tremendous effort has been made to maintain high service availability, in reality, there are still many unexpected system problems caused by software or platform failures (such as software crashes, network outage, misconfigurations, memory leak, hardware breakdown, etc.). There have been some previous studies in the literature on failures of a data center. For example, Ford et al. studied [13] the data availability of Google distributed storage systems, and characterized the sources of faults contributing to unavailability. Their results indicate that cluster-wide failure events should be paid more attention during the design of system components, such as replication and recovery policies. Vishwanath and Nagappan [39] classified server failures in a data center and found that 8% of all servers had at least one hardware incident in a given year. Their studies could be helpful to reduce the hardware faults, especially the networking faults. Huang et al. [22] also found that the major availability breakdowns and performance anomalies we see in cloud environments tend to be caused by subtle underlying faults, i.e., gray failure rather than fail-stop failure. The above-mentioned related work shows that failures in cloud systems can be triggered by many software or hardware issues. In our work, we only focus on disk error prediction. In particular, disk errors can be also seen as a form of *gray failures*: the system's failure detectors may not notice them even when applications are afflicted by them.

7 Conclusion

Disk error is one of the most important reasons that cause service unavailability. In this paper, we propose CDEF, an online disk error prediction approach that can predict disk errors proactively before they cause more severe damage to the cloud system. We collect both SMART and system-level signals, perform feature engineering, and develop a cost-sensitive ranking model. We evaluate our approach using real-world data collected from a cloud system. The results confirm that the proposed approach is effective and outperforms related methods. The ability to predict faulty disks enables the live migration of existing virtual machines and allocation of new virtual machines to the healthy disks, thus improving service availability. We have also successfully applied the proposed approach to Microsoft Azure.

There are many viable ways of extending this work. We have applied our approach to hard disk drives in production. In the future, we will apply it to other disk types such as Solid State Drive. We will also explore the synergy between disk error prediction and other cloud failure detection techniques such as [22], and propose an integrated solution to service availability improvement.

8 Acknowledgements

Our special thanks go to Girish Bablani, Gil Shafriri, Aaron Colling, Zheng Mu, Brijesh Ramachandran, John Kim, Sandeep Bhadsavle, Ashish Munjal, and Nisarg Sheth for making this research impactful for Azure service. We especially thank our shepherd Xiaohui Gu for the constructive comments.

References

[1] http://www.distributed-generation.com.

[2] https://www.backblaze.com/b2/hard-drive-test-data.html.

[3] https://azure.microsoft.com/en-au/services/cosmos-db/.

[4] https://studio.azureml.net/.

[5] https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[6] https://www.backblaze.com/blog/hard-drive-failure-rates-q3-2017/.

[7] http://www.distributed-generation.com.

[8] BAIRAVASUNDARAM, L. N., GOODSON, G. R., PASUPATHY, S., AND SCHINDLER, J. 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* (New York, NY, USA, 2007), SIGMETRICS '07, ACM, pp. 289–300.

[9] BOTEZATU, M. M., GIURGIU, I., BOGOJESKA, J., AND WIESMANN, D. Predicting disk replacement towards reliable data centers. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), ACM, pp. 39–48.

[10] CHAWLA, N. V., BOWYER, K. W., HALL, L. O., AND KEGELMEYER, W. P. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res. 16*, 1 (June 2002), 321–357.

[11] DEAN, D. J., NGUYEN, H., AND GU, X. UBL: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems. In *Proceedings of the 9th international conference on Autonomic computing* (2012), ACM, pp. 191–200.

[12] DOMINGOS, P. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (1999), ACM, pp. 155–164.

[13] FORD, D., LABELLE, F., POPOVICI, F., STOKELY, M., TRUONG, V.-A., BARROSO, L., GRIMES, C., AND QUINLAN, S. Availability in globally distributed storage systems. In *Proc. OSDI* (2010).

[14] FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29* (2000), 1189–1232.

[15] GABER, S., BEN-HARUSH, O., AND SAVIR, A. Predicting hdd failures from compound smart attributes. In *Proceedings of the 10th ACM International Systems and Storage Conference* (2017), SYSTOR '17, ACM, pp. 31:1–31:1.

[16] GANGULY, S., CONSUL, A., KHAN, A., BUSSONE, B., RICHARDS, J., AND MIGUEL, A. A practical approach to hard disk failure prediction in cloud platforms: Big data model for failure management in datacenters. In *Big Data Computing Service and Applications (BigDataService), 2016 IEEE Second International Conference on* (2016), IEEE, pp. 105–116.

[17] GENUER, R., POGGI, J.-M., AND TULEAU-MALOT, C. Variable selection using random forests. *Pattern Recognition Letters 31*, 14 (2010), 2225 – 2236.

[18] GOLDSZMIDT, M. Finding soon-to-fail disks in a haystack. In *Proceedings of the 4th USENIX Conference on Hot Topics in Storage and File Systems* (Berkeley, CA, USA, 2012), HotStorage'12, USENIX Association, pp. 8–8.

[19] GU, X., AND WANG, H. Online anomaly prediction for robust cluster systems. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on* (2009), IEEE, pp. 1000–1011.

[20] GUNAWI, H. S., SUMINTO, R. O., SEARS, R., GOLLIHER, C., SUNDARARAMAN, S., LIN, X., EMAMI, T., SHENG, W., BIDOKHTI, N., MCCAFFREY, C., ET AL. Fail-slow at scale: Evidence of hardware performance faults in large production systems.

[21] GUYON, I., AND ELISSEEFF, A. An introduction to variable and feature selection. *J. Mach. Learn. Res. 3* (Mar. 2003), 1157–1182.

[22] HUANG, P., GUO, C., ZHOU, L., LORCH, J. R., DANG, Y., CHINTALAPATI, M., AND YAO, R. Gray failure: The achilles' heel of cloud-scale systems. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems* (New York, NY, USA, 2017), HotOS '17, ACM, pp. 150–155.

[23] LI, J., JI, X., JIA, Y., ZHU, B., WANG, G., LI, Z., AND LIU, X. Hard drive failure prediction using classification and regression trees. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (June 2014), pp. 383–394.

[24] LIU, T.-Y. Learning to rank for information retrieval. *Found. Trends Inf. Retr. 3*, 3 (Mar. 2009), 225–331.

[25] MA, A., TRAYLOR, R., DOUGLIS, F., CHAMNESS, M., LU, G., SAWYER, D., CHANDRA, S., AND HSU, W. Raidshield: Characterizing, monitoring, and proactively protecting against disk failures. *ACM Trans. Storage 11*, 4 (Nov. 2015), 17:1–17:28.

[26] MAHDISOLTANI, F., STEFANOVICI, I., AND SCHROEDER, B. Proactive error prediction to improve storage system reliability. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)* (Santa Clara, CA, 2017), USENIX Association, pp. 391–402.

[27] MEZA, J., WU, Q., KUMAR, S., AND MUTLU, O. A large-scale study of flash memory failures in the field. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems* (New York, NY, USA, 2015), SIGMETRICS '15, ACM, pp. 177–190.

[28] MICROSOFT. Machine learning fast tree, https://docs.microsoft.com/en-us/machine-learning-server/python-reference/microsoftml/rx-fast-trees, 2017.

[29] MURRAY, J. F., HUGHES, G. F., AND KREUTZ-DELGADO, K. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research 6*, May (2005), 783–816.

[30] PATTERSON, D. A., GIBSON, G., AND KATZ, R. H. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1988), SIGMOD '88, ACM, pp. 109–116.

[31] PINHEIRO, E., WEBER, W.-D., AND BARROSO, L. A. Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2007), FAST '07, USENIX Association, pp. 2–2.

[32] PITAKRAT, T., VAN HOORN, A., AND GRUNSKE, L. A comparison of machine learning algorithms for proactive hard disk drive failure detection. In *Proceedings of the 4th International ACM Sigsoft Symposium on Architecting Critical Systems* (New York, NY, USA, 2013), ISARCS '13, ACM, pp. 1–10.

[33] PONEMONINSTITUTE. Cost of data center outages, 2016. https://planetaklimata.com.ua/instr/Liebert_Hiross/Cost_of_Data_Center_Outages_2016_Eng.pdf.

[34] SCHROEDER, B., AND GIBSON, G. A. Disk failures in the real world: What does an MTTF of 1, 000, 000 hours mean to you? In *FAST* (2007), pp. 1–16.

[35] SCHROEDER, B., LAGISETTY, R., AND MERCHANT, A. Flash reliability in production: The expected and the unexpected. In *14th USENIX Conference on File and Storage Technologies (FAST 16)* (Santa Clara, CA, 2016), USENIX Association, pp. 67–80.

[36] TAN, M., TAN, L., DARA, S., AND MAYEUX, C. Online defect prediction for imbalanced data. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2* (Piscataway, NJ, USA, 2015), ICSE '15, IEEE Press, pp. 99–108.

[37] TAN, Y., AND GU, X. On predictability of system anomalies in real world. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on* (2010), IEEE, pp. 133–140.

[38] TANTITHAMTHAVORN, C., MCINTOSH, S., HASSAN, A. E., AND MATSUMOTO, K. Comments on 'researcher bias: The use of machine learning in software defect prediction'. *IEEE Transactions on Software Engineering 42*, 11 (Nov 2016), 1092–1094.

[39] VISHWANATH, K. V., AND NAGAPPAN, N. Characterizing cloud computing hardware reliability. In *SOCC* (New York, NY, USA, 2010), ACM, pp. 193–204.

[40] WANG, C., TALWAR, V., SCHWAN, K., AND RANGANATHAN, P. Online detection of utility cloud anomalies using metric distributions. In *Network Operations and Management Symposium (NOMS), 2010 IEEE* (2010), IEEE, pp. 96–103.

[41] WANG, Y., MIAO, Q., MA, E. W. M., TSUI, K. L., AND PECHT, M. G. Online anomaly detection for hard disk drives based on mahalanobis distance. *IEEE Transactions on Reliability 62*, 1 (March 2013), 136–145.

[42] ZHU, B., WANG, G., LIU, X., HU, D., LIN, S., AND MA, J. Proactive drive failure prediction for large scale storage systems. In *2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST)* (May 2013), pp. 1–5.