Hard Drive Failure Prediction using Decision Trees

Jing Li\textsuperscript{a,b}, Rebecca J. Stones\textsuperscript{b}, Gang Wang\textsuperscript{b}, Xiaoguang Liu\textsuperscript{b}, Zhongwei Li\textsuperscript{b}, Ming Xu\textsuperscript{c}

\textsuperscript{a}College of Computer Science and Technology, Civil Aviation University of China, Tianjin, China
\textsuperscript{b}Nankai-Baidu Joint Lab, College of Computer and Control Engineering, Nankai University, Tianjin, China
\textsuperscript{c}Qihoo 360 Technology Company, Beijing, China

Abstract

This paper proposes two hard drive failure prediction models based on Decision Trees (DTs) and Gradient Boosted Regression Trees (GBRTs) which perform well in prediction performance as well as stability and interpretability. The models are evaluated on a real-world dataset containing 121,698 drives in total. Experimental results show the DT model predicts over 93\% of failures at a false alarm rate under 0.01\%, and the GBRT model can achieve about 90\% failure detection rate without any false alarms. Moreover, the GBRT model evaluates drive health (or fault probability) which provides a quantitative indicator of failure urgency. This enables operators to allocate system resources accordingly for pre-warning migrations while maintaining the quality of user services.

Aiming at practical application of prediction models, we test the models on another real-world dataset with different drive models, on a real-world hybrid dataset with multiple drive models, and on several datasets containing fewer drives. Both prediction models show steady prediction performance, with high failure detection rates (80\% to 96\%) and low false alarm rates (0.006\% to 0.31\%). We also implement a reliability model for RAID-6 systems with proactive fault tolerance and show that the proposed models can significantly improve the reliability and/or reduce construction and maintenance cost of large-scale storage systems.

Keywords:
Hard drive failure prediction; SMART; Decision Tree; Health degree

1. Introduction

Hard drives are one of the most common devices in storage systems, and most of the information in the world is being stored on hard drives [1]. While the failure
of a single hard drive might be rare, a system with thousands of hard drives will often experience failures and even simultaneous failures \cite{2}, resulting in service unavailability, and even permanent data loss. Therefore, reliability is one of the biggest concerns in storage systems.

Given the importance of users’ data, operators insist on high levels of reliability, and typically either store multiple copies of data throughout the system (increasing storage costs), or store data using some kind of erasure code (which consumes resources during reconstruction). On top of this, disk failure prediction is used to further increase reliability via automatic backing up of data on at-risk disks. However, for disk failure prediction to be useful, we require accurate predictions.

In this paper, we explore the ability of decision trees \cite{3} and gradient boosted regression trees \cite{4} to predict hard drive failure based on SMART (Self-Monitoring, Analysis and Reporting Technology) attributes, which continuously reports attributes relating to drive reliability. Besides high prediction accuracy, it has the advantage of giving humanly understandable prediction results (i.e., interpretability), unlike previous approaches. Users can pinpoint the most significant attributes correlated with drive failure by analyzing the output regulations of the tree. Unlike prognostic approaches \cite{5, 6, 7, for example} which focus on assessing the dynamic reliability and failure prognostics for whole systems, we study the degradation state of individual components.

It is possible to directly use SMART attributes for hard drive failure prediction through thresholding, which can achieve FDR (failure detection rate) $\approx 4\%$ and FAR (false alarm rate) $\approx 0.02\%$ \cite{8}, assuming a conservative FAR. There are a range of methods which significantly improve prediction performance beyond this baseline; we survey these below, along with ballpark figures for their FDR and FAR statistics:

- **Learning methods.** Hamerly and Elkan \cite{9} proposed a mixture model of naive Bayes submodels, comparing it to a naive Bayes classifier (both achieving FAR$\approx 0.67\%$; FDR$\approx 33\%$). Murray et al. \cite{10} implemented a support vector machines (SVM) method as a comparison (FAR$\approx 0\%$; FDR$\approx 50.6\%$) along with “preliminary results” in \cite{11} (FAR$\approx 2.5\%$; FDR$\approx 18\%$). Zhao et al. \cite{12} used an SVM as a baseline (FAR$\approx 0\%$; FDR$\approx 43\%$). Agrawal et al. \cite{13} used a maximum likelihood rules learning procedure (FAR$\approx 1.5\%$; FDR$\approx 50\%$). Tan and Gu \cite{14} also used a Bayesian approach (FAR$\approx 0.5\%$; FDR$\approx 65\%$).

- **Statistical tests.** Hughes et al. \cite{8} proposed a rank-sum test (FAR$\approx 0.2\%$;
FDR around 40% to 60%). Murray et al. [11] used a rank-sum test as a baseline (FAR ≃ 0.5%; FDR ≃ 33%, and FAR ≃ 0.7%; FDR ≃ 52.8% in [10]). Wang et al. [15] proposed a Mahalanobis distance method (FAR ≃ 0.56%; FDR ≃ 63%, and FAR ≃ 0%; FDR ≃ 67% in [16]).

- **Hidden (semi-) Markov models.** Salfner and Malek [17] used a hidden semi-Markov model for disk failure prediction (FAR ≃ 1.4%; FDR ≃ 66%). Zhao et al. [12] likewise used a hidden Markov model (FAR ≃ 0.6%; FDR ≃ 55%) and also a hidden semi-Markov model (FAR ≃ 0.6%; FDR ≃ 37%).

Pinheiro et al. [18] identified an inherent limitation to these thresholding methods: they found around 36% of drives fail without meeting any of the thresholds, which limits the best possible FDR ≤ 64%, which matches the above results. Nevertheless, an FDR at this level can “drastically extend the MTTDL of a data storage system” [19].

The authors’ research group has found that network approaches do not have this inherent limitation, and have tested a range of network approaches which have exceeded this limitation: backpropagation neural networks [20] (FAR ≃ 0.48%; FDR ≃ 95%), classification trees [21] (FAR ≤ 0.01%; FDR ≃ 93%), recurrent neural networks [22] (FAR ≃ 0.06%; FDR ≃ 97%), combined Bayesian networks [23] (FAR ≃ 0.08%; FDR ≃ 95%), and (prototype) gradient boosted regression trees [24] (FAR ≃ 0.02%; FDR ≃ 90%). These network approaches set a new standard for failure prediction using SMART attributes in terms of FAR/FDR.

Prior to all this, Sahoo et al. from IBM [25] also used a Bayesian network approach for disk failure prediction, although not for SMART attributes (FDR ≃ 70%). Recently, Chaves et al. [26] has also explored a Bayesian network approach to predicting hard drive failures using SMART attributes, where they report improvements in terms of mean and median quadratic errors.

In this paper, we consolidate and build upon this work, focusing exclusively on (a) the decision tree model, referred to as “classification tree” in [21], and (b) the GBRT model, which we show outperform the other models. Here, we evaluate using the traditional metrics: failure detection rate (FDR), false alarm rate (FAR), and time in advance (TIA), with failure probability incorporating varying migration transfer rates (for different storage systems). We make the following additional contributions:

- Instead of the three statistical methods in [21] used to select the critical features (the reverse arrangement test, the rank-sum test, and z-scores), we
use quantile functions, which allows in-depth quantitative measurements for every feature on healthy and drives which fail (see Section 4.2).

- We improve on the regression tree method in [21] by using the promotion model GBRT (see Section 3). Moreover, we develop a better-suited “setter” method for initial target values of training samples (see Section 5.3.2).

- We can identify possible causes of drive failures by analyzing the Decision Tree (see Section 5.4.1).

- Experimental results are obtained using a new, large real-world dataset not used in [21] (see Tables 6 and 7).

- We measure reliability in terms of a more meaningful metric, the expected number of data loss events per usable petabyte per year, instead of mean time to data loss (MTTDL).

The rest of the paper is organized as follows: Sections 2 and 3 introduce the proposed modeling methods for failure prediction. Section 4 gives a description of the datasets and the preprocessing of them for building models, presenting the experimental results in Section 5. In Section 6, we discuss the improvements in reliability, followed by conclusions in Section 7.

2. Decision Tree Model

When we observe a collection of drives over time where some subset of these drives fail at various points in time, then, for brevity, we refer to the drives which fail simply as failed drives. The remaining drives are referred to as good (or healthy) drives. Each SMART attribute has a 6-byte raw value and a 1-byte normalized value ranging from 1 to 253 which is transformed from the raw value [27]. The formats of the raw values are vendor-specific, and are not specified by any standard, so we use normalized values except where specified.

Figure 1 illustrates a simplified example of a decision tree for hard drive failure prediction. The percentages indicate the proportion of samples that satisfy the constraints of ancestral nodes. The two decimals indicate the proportion of failed and good samples at that node, and we shade those with failed proportion at least 0.5. The two child nodes (where relevant) are those that satisfy or do not satisfy the stipulated condition. So e.g. at node 3, we find 2% of samples which have SMART attribute POH (power on hours) at least 90. This bound equates to a drive operating for more than one year. Node 3 splits according to whether or not
RUE (reported uncorrectable errors) is at least 100, and similarly for other nodes. The process continues until there are no more nodes which can split satisfying a minimum split condition (which we describe below), resulting in some nodes (e.g. nodes 4 and 5) not occurring in the decision tree. Each leaf node is labeled with the majority class of the samples at it.

Figure 1: A simplified decision tree for hard drive failure prediction. POH is an acronym for the normalized SMART attribute Power On Hours, RUE = Reported Uncorrectable Errors, TC = Temperature Celsius, SUT = Spin Up Time, and SER = Seek Error Rate.

To find the best split, the decision tree algorithm checks all possible splits, and chooses the one which achieves the greatest gain in information. Assuming node $D$ splits into child nodes $D_1$ and $D_2$ based on condition $X$ (e.g., “POH < 90”), then the information gain for this split is calculated as

$$\text{gain}(D, X) := \text{info}(D) - \text{info}(D, X)$$  \hspace{1cm} (1)

where the information entropy at node $D$ is

$$\text{info}(D) := -p \log_2(p) - (1-p) \log_2(1-p),$$  \hspace{1cm} (2)

where $p$ is the proportion of failed drives in $D$, and

$$\text{info}(D, X) := \frac{|D_1|}{|D|} \text{info}(D_1) + \frac{|D_2|}{|D|} \text{info}(D_2)$$  \hspace{1cm} (3)

is the weighted sum of the information entropies of its two child nodes, where $|D|$ denotes the total number of samples contained at node $D$ (and likewise for $D_1$ and $D_2$).
The decision tree is grown by recursively splitting nodes, until the terminal nodes do not satisfy a minimum split condition or contain only one class. The Minsplit and Minbucket (minimum bucket size) conditions in [28] are used to determine when nodes are split. Minsplit limits the minimum number of samples that must exist at a node before it is considered for splitting. Minbucket limits the minimum number of samples at any leaf node. A complete tree will be built to the maximum depth which satisfies the constraints of Minsplit and Minbucket.

Additionally, even with the Minsplit and Minbucket constraints, it’s possible to overfit the training data which results in poor performance; we alleviate overfitting problems by pruning. In particular, subbranches with low overall information gain will be pruned from the fully-grown tree, simplifying the decision tree. A complexity parameter is used to control the size of the tree and to select an optimal size by controlling the process of pruning. The complexity parameter governs the minimum gain that must be obtained at each split of the decision tree in order to include that split. The detailed decision tree algorithm for training drive failure prediction models, incorporating Minsplit, Minbucket, and pruning using the complexity parameter, is shown in Algorithm 1. The thresholds used in our experiments are described in Section 5.2. The majority vote is the majority class of the samples at a node.

We build decision tree models using SMART attributes and their change rates as input vectors together with the target values representing good or failed drives. To distinguish good and failed drives more effectively, we propose some additional improving strategies: (1) We change the probability distributions of the good and failed samples by adjusting their weights (through replicating failed samples). (2) To reduce false alarms, we add a weighting to the two kinds of errors (false alarms and missed detections), which will affect the choice of variable on which to split the dataset at each node, and affect FAR and FDR (and TIA) differently. To this end, we apply loss weights, user-defined weights which are used to tell the software how important FAR is compared to FDR (e.g. a 1% increase in FAR might be considered as significant as a 10% drop in FDR).

3. Gradient Boosted Regression Trees Model

We use Gradient Boosted Regression Trees (GBRTs) to evaluate drive health degrees. Each test has a quantitative target value describing the drive’s health degree (as opposed to a class label, which only indicates whether it is good or failed). Therefore, operators can schedule the pre-warning handling and allocate system resources accordingly, which can help in achieving a balance between the
Algorithm 1 Training the decision tree model

**Input:** Training data (composed of SMART attributes, attribute change rates, and target values), split conditions (Minsplit, Minbucket), and complexity parameter (CP)

**Output:** Decision tree (pruned) for drive failure prediction, with nodes labeled by their majority votes

1: create root node $T$ which contains all of the drives
2: label $T$ with its majority vote
3: push $T$ onto the stack
4: while the stack is not empty do
5: pop the top element from the stack and store it as $D$
6: if $D$ does not satisfy the Minsplit and Minbucket split conditions then
7: set $D$ as a leaf node
8: else
9: for each possible split $X$ for $D$ do
10: calculate gain($D,X$) using (1), (2), and (3)
11: end for
12: select the split $X^*$ maximizing gain($D,X^*$)
13: split $D$ into child nodes $D_1$ and $D_2$ based on $X^*$
14: label nodes $D_1$, $D_2$ and push them onto the stack
15: end if
16: end while
17: for each node $P$ in the tree do
18: if the gain induced by $P$’s split is less than CP then
19: prune back the entire sub-tree rooted at $P$
20: end if
21: end for

GBRT [4] is a gradient descent boosting technique based on tree averaging, and is an accurate and effective machine learning technique that can be used for both regression and classification problems. To avoid overfitting, the GBRT algorithm trains many tree stumps as weak learners, rather than some full high-variance trees. Thus, instead of split conditions, a tree-depth parameter $d$ is used to control the size of trees.

Each leaf node is weighted by the mean of the health degree (target value) of quality of services and pre-warning migrations. This improves the reliability and availability of storage systems.
its samples, with healthy samples being assigned +1 representing absolute health. For each failed sample, we set its target to a real value in \([-1, 0]\) representing its health degree. When a sample is collected, at the moment the drive fails, its health degree is set to -1, and at the boundary between good and failed, its health degree is set to 0. The health degree of a drive is predicted as the weight of the leaf node it belongs to.

We use regression trees as weak learners. To find the best split, the regression tree algorithm checks all possible split. Determining the best split is achieved using the minimum of squares of nodes (instead of the usual greatest gain in information), namely

\[
sq := \sum_j (y_j - \bar{y})^2, \tag{4}\]

where \(y_j\) is the target variable of the \(j\)-th sample, and \(\bar{y} = \text{ave}_j(y_j)\). The sum (4) is over all samples \(j\) that satisfy the splitting conditions of the ancestor nodes. In this way, we treat drive failure prediction as a regression problem (rather than a classification problem).

For GBRTs, regression trees are introduced at each iteration to adjust for prediction errors (residuals) for each sample vs. the target value from the previous regression trees. The residuals of the \(i\)-th tree (used to determine the \((i+1)\)-th tree) are given by

\[
r^{(i+1)}[j] := r^{(i)}[j] - \alpha T^{(i)}[j] \tag{5}\]

where \(T^{(i)}[j]\) is the prediction for the \(j\)-th sample from the \(i\)-th regression tree, \(\alpha\) is a user-defined learning rate, and \(r^{(1)}[j] = y_j\), and (4) generalizes to

\[
sq = sq(i) := \sum_j \left( r^{(i)}[j] - \overline{r^{(i)}} \right)^2. \tag{6}\]

We build GBRT models using SMART attributes and their rates of change as input vectors together with the target values representing the health degrees of drives. Algorithm 2 gives the details for training the GBRT prediction model. When testing, a drive’s health degree is predicted as the combined predictions by all the regression trees, namely

\[
\text{drive health degree} = \sum_i \alpha T^{(i)}[j].
\]

To improve the performance of the GBRT model on drive health degree prediction, we propose some “setter” methods for initial target values of training
Algorithm 2 Training the GBRT model

**Input:** Training dataset (including actual SMART attributes and health degrees \(y_j\)), learning rate \(\alpha\), number of regression trees \(c\), tree depth \(d\)

**Output:** GBRTs \(T^{(i)}\) used for predicting drive health degree

1: initialize \(r^{(1)}[j] \leftarrow y_j\) for \(j \in \{1, 2, \ldots, n\}\)

2: for regression tree \(i = 1\) to \(c\) do \(\triangleright\) build regression tree \(T^{(i)}\) of depth \(d\)

3: weight root node of \(T^{(i)}\) with \(r^{(i)}\)

4: for \(k = 1\) to \(d\) do

5: for each node \(V\) at depth \(k\) do

6: for each possible split at \(V\) do

7: calculate \(sQ_L + sQ_R\) from (6), where \(L\) and \(R\) are its two proposed child nodes

8: end for

9: split \(V\) to minimize \(sQ_L + sQ_R\)

10: weight \(V\)’s child nodes with \(\text{ave}_s(r^{(i)}[s])\), where the average is over all samples \(s\) which satisfy the splitting conditions of its ancestor nodes

11: end for

12: end for

13: update \(r^{(i+1)}[j] \leftarrow r^{(i)}[j] - \alpha T^{(i)}[j]\) for \(j = 1\) to \(n\)

14: end for

samples, where we set the initial initial values of training samples. A simple function to determine the health degree of the failed sample \(t\) hours before failure is \(h : [0, w] \rightarrow [-1, 0]\) defined by

\[
h(t) = \frac{t}{w} - 1
\]

where \(w\) denotes the size of the *global deterioration window*. That is, the \(w\) hours before failure are a boundary situation between healthy and failed, after which drives deteriorate gradually. With this definition, each failed drive is modeled identically, i.e., (7) does not vary drive to drive. It turns out this function does not perform very well in practice, so we use a modified version.

Motivated by the observation that drives deteriorate gradually at different rates, we instead use another function based on a *personalized deterioration window* to
generate the input for Algorithm 2, namely $h_d : [0, w_d] \rightarrow [-1, 0]$ defined by

$$h_d(t) = \frac{t}{w_d} - 1$$  \hspace{1cm} (8)

where $w_d$ denotes the size of drive $d$’s deterioration window. We set $w_d$ to the time in advance of which $d$ can be predicted to fail by a prediction model. Since the personalized deterioration window distinguishes individual drive’ deterioration processes, this method achieves better prediction performance than the method using the global deterioration window.

A problem with this method is that a very small $w_d$ might result in a sample very close to the actual fault moment to be set a relative high health degree. This means that an urgent warning might be improperly regarded as non-urgent. We instead want to assign a low health degree to a sample close to failure, regardless of the how small $w_d$ is. To this end, we generalize the functions $h_d : [0, w_d] \rightarrow [-1, 0]$ in (8) to

$$h_d(t) = \begin{cases} 
 \frac{t(H + 1)}{U} - 1 & \text{when } 0 \leq t < U, \\
\frac{w_d - t}{w_d - U}H & \text{when } U \leq t \leq w_d,
\end{cases} \hspace{1cm} (9)$$

where we have user-defined parameters $U \in [0, w_d)$ is the urgency window time and $H \in [-1, 0]$ is the degree of urgency. The parameter $U$ defines the boundary between urgent and non-urgent potential failures, which are treated differently; $U$ can be set to a sufficiently long time, say 12 hours, for responding to warnings. Since (9) distinguishes between differing drive deterioration behaviors, this method achieves the best prediction performance. (Note that the functions described by (9) are continuous functions, and are equal to those described by (8) when $U = 0$ and $H = -1$.)

4. Dataset Description And Preprocessing

4.1. Datasets

To evaluate the proposed models, we use real-world datasets collected from two real-world data centers. The statistics are listed in Table 1. Samples were taken from working drives at every hour using smartmontools. Each sample contains all the SMART attribute values for a single drive at an exact time.

The data from the first data center, denoted “W”, was used in our previous work [20] and contains a total of 23,395 drives from an enterprise-class model,
Table 1: Dataset statistics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Class</th>
<th>No. drives</th>
<th>Period</th>
<th>No. samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>“W”</td>
<td>Good</td>
<td>22,962</td>
<td>7 days</td>
<td>3,837,568</td>
</tr>
<tr>
<td></td>
<td>Failed</td>
<td>433</td>
<td>20 days</td>
<td>158,150</td>
</tr>
<tr>
<td>“Q_all”</td>
<td>Good</td>
<td>98,060</td>
<td>7 days</td>
<td>16,428,076</td>
</tr>
<tr>
<td></td>
<td>Failed</td>
<td>243</td>
<td>60 days</td>
<td>306,023</td>
</tr>
<tr>
<td>“Q_s”</td>
<td>Good</td>
<td>38,985</td>
<td>7 days</td>
<td>6,543,468</td>
</tr>
<tr>
<td></td>
<td>Failed</td>
<td>106</td>
<td>60 days</td>
<td>133,497</td>
</tr>
</tbody>
</table>

labeled good or failed\(^1\). For good drives, the samples within a 7 day period are present in the dataset, so good drives will ordinarily have 168 samples (except some which might be missing due to e.g. sampling or storing errors). For failed drives, samples in a period of 20 days before actual failure were recorded. Some failed drives might have fewer samples if they had not survived 20 days of operation since we began to collect data.

The data from multiple rooms of the second data center, denoted “Q_all”, is a hybrid dataset, containing 28 drive models from 5 manufacturers. The total number of drives is 98,303, with only 243 failed drives and 98,060 good drives. For good drives, the samples in a week are recorded. For failed drives, samples in a period of 60 days before actual failure were recorded. Since some failed drives were not able to survive 60 days of operation since we began to collect data, they had less than 1,440 samples. Almost half of the drives in “Q_all” are from a single Seagate model (not the same with that in “W”), so we also use the single-model dataset, denoted “Q_s”, which contains only these drives. The “Q_s” dataset has 39,117 drives in total, with 39,011 good drives and 106 failed drives.

For every drive in the “W” dataset, we have 23 meaningful attributes from its SMART record. However, some attributes are useless for failure prediction because their values are the same for good and failed drives and do not change during operation. So we filter them out and use only ten attributes to build the prediction models. Since some normalized values lose accuracy, their corresponding raw values are more predictive of the health condition of drives, so we select two raw

\(^1\)The dataset is now available at http://pan.baidu.com/share/link?shareid=189977&uk=4278294944.
Table 2: Basic features (preliminary selected SMART attributes) for the “W” and “Q.s” datasets. All but two attributes (ID numbers 11 and 12) are normalized values.

<table>
<thead>
<tr>
<th>ID #</th>
<th>Attribute Name</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Raw Read Error Rate</td>
<td>“W”, “Q.s”</td>
</tr>
<tr>
<td>2</td>
<td>Spin Up Time</td>
<td>“W”, “Q.s”</td>
</tr>
<tr>
<td>3</td>
<td>Reallocated Sectors Count</td>
<td>“W”, “Q.s”</td>
</tr>
<tr>
<td>4</td>
<td>Seek Error Rate</td>
<td>“W”, “Q.s”</td>
</tr>
<tr>
<td>5</td>
<td>Power On Hours</td>
<td>“W”, “Q.s”</td>
</tr>
<tr>
<td>6</td>
<td>Reported Uncorrectable Errors</td>
<td>“W”, “Q.s”</td>
</tr>
<tr>
<td>7</td>
<td>High Fly Writes</td>
<td>“W”, “Q.s”</td>
</tr>
<tr>
<td>8</td>
<td>Temperature Celsius</td>
<td>“W”, “Q.s”</td>
</tr>
<tr>
<td>9</td>
<td>Hardware ECC Recovered</td>
<td>“W”</td>
</tr>
<tr>
<td>10</td>
<td>Current Pending Sector Count</td>
<td>“W”, “Q.s”</td>
</tr>
<tr>
<td>11</td>
<td>Reallocated Sectors Count (raw value)</td>
<td>“W”</td>
</tr>
<tr>
<td>12</td>
<td>Current Pending Sector Count (raw value)</td>
<td>“W”</td>
</tr>
<tr>
<td>13</td>
<td>End-to-End Error</td>
<td>“Q.s”</td>
</tr>
<tr>
<td>14</td>
<td>Command Timeout</td>
<td>“Q.s”</td>
</tr>
<tr>
<td>15</td>
<td>Airflow Temperature Celsius</td>
<td>“Q.s”</td>
</tr>
<tr>
<td>16</td>
<td>Offline Uncorrectable</td>
<td>“Q.s”</td>
</tr>
</tbody>
</table>

values in addition to the ten normalized values to build models. This results in 12 basic features for the “W” dataset.

For every drive in the “Q.s” dataset, we record the normalized values of 20 SMART attributes in its samples. We exclude 7 useless attributes whose values are the same for good and failed drives, leaving 13 basic features to build the prediction models. Table 2 lists the basic features for the “W” and “Q.s” datasets.

4.2. Feature Selection Using Statistical Methods

Since some features are not strongly correlated with future drive failure and including them may have a negative impact on predictor performance [10], we use some statistical methods to select only the critical features. Following the method in [20], for each sample in “W” and “Q.s”, we calculate the differences between the current values of the basic features and their corresponding values six hour prior as new features, i.e., change features.
For each basic and change feature for the SMART attributes in Table 2, we plot the empirical inverse cumulative distribution function, or quantile function (some examples are shown in Figure 2). For each drive (healthy or failed), we take the last sample to compute the quantile functions. Separate plots are generated for failed and healthy drives, thereby enabling us to visually inspect (i.e., “eyeball”) these plots and hand pick a selection of discriminatory features. To further justify our manual selections, we use three non-parametric statistical methods—reverse arrangement test, rank-sum test, and z-scores [10] (see Section 5.2).

Due to space limitations, we only show the quantile functions of six basic features for “W” (the top 4 and least 2 effective) in Figure 2. Some observations are evident from Figure 2:

1. We see that 90% of good drives do not have any Reallocated Sector Errors, whereas 60% failed drives have some. This makes it a good discriminator between good and failed drives.
2. For all the good drives in the dataset “W”, the value of Power on Hours (POH) is $\geq 90$, while for 60% failed drives, the value is $\leq 90$. (Note that based on how POH is normalized, a lower normalized value indicates a longer operation time of the drive.)
3. Two other good discriminators are: (a) Seek Error Rate, where for 60% of good drives, the value is $\leq 76$, while for 90% of failed drives, the value is $\geq 76$, and (b) Raw Read Error Rate, where for 70% of good drives, the value is $\geq 78$, while for 50% failed drives, the value is $\leq 78$.
4. Both the normalized and raw values of Current Pending Sector Count have a similar distributions for both good and failed drives, making it a poor predictor of drive failure.

Based on the above observations, it is reasonable to believe that distinctions in the first four features are correlated with the distinction between good and failed drives, and the last two features do not have effect on the distinction. We likewise consider the remaining SMART attributes.

For “W”, ten basic features and three change features are selected for model building. We retain basic features 1 through 9 and 11 from Table 2, and change features for 1, 9, and 11. For “Q_s”, ten basic features, 1, 3 through 8, 10, 15, and 16, and four change features, 3, 6, 10, and 16, are selected.

In the hybrid dataset “Q_all”, drives from different models do not have the same SMART attributes. To standardize the data, we use the 14 critical parameters (i.e., those selected using quantile functions) of “Q_s” as the critical parameters.
for “Q_all”, with missing values set to the average of the previous and subsequent values (with respect to time).

5. Experimental Results

5.1. Experimental Setup

To evaluate the models, we divide the datasets into training and test sets with respect to time. (This deviates from the usual method of randomly dividing the dataset into training and test sets, which would be unrealistic.) For each healthy drive, we take the earlier 70% of the samples as training data, and the later 30% as test data. Since the time of failure for failed drives was not recorded, we divide the failed drives in “W” and “Q_all” randomly into training and test sets in a 7 to 3 ratio, and the drives in “Q_s” in a 1 to 1 ratio (due to there being fewer failed drives in “Q_s”).

We further process the datasets in the following way: (a) We randomly choose 3 samples for “W” and 1 sample for “Q_s” and “Q_all” per good drive in the training set as good samples to train models. In this way, we can reduce the great disparity between good and failed samples while providing enough information to describe the health condition of the drives. (b) Like [20], for each failed sample, we use only the samples within a time window to train models, that is, the last $n$
hours before failure actually occurs, with the underlying motivation that the last $n$ samples will be indicative of impending failure. We test time window sizes $n$ in \{12, 24, 48, 96, 168, 240\}.

The Receiver Operating Characteristic (ROC) curve is used for presenting the prediction performance of models. For drive failure prediction problem, the ROC curve indicates the trade-off between the failure detection rate (FDR) and the false alarm rate (FAR). Here, FDR is the fraction of failed drives that are correctly classified as failed, and FAR is the fraction of good drives that are incorrectly classified as failed. We can adjust the trade-off between them by tuning the algorithm parameters. Another important metric of drive failure prediction is the time in advance (TIA) which describes how long in advance we can detect impending failures.

5.2. Feature Selection

The experimental results in [20], show the advantage of the backpropagation artificial neural network (BP ANN) model in prediction performance over the other previous models. So, in this subsection, we apply BP ANN and the decision tree (DT) models to verify the effectiveness of our selected features.

For the “W” dataset, we use three feature sets respectively composed of the 12 basic features detailed in Table 2, the 19 features hand picked in [20], and the 13 critical features selected by statistical methods. In this experiment, we set the time window to 12 hours (as did [20]). That is, the samples collected within last 12 hours before failure are used as failed samples. For the BP ANN models based on 12, 19, and 13 features, the input layers respectively have 12, 19, and 13 nodes, the hidden layers respectively contain 20, 30, and 13 nodes, and all the three output layers have 1 node. The maximum number of iterations is set to 400 and the learning rate is set to 0.1. Some important decision tree parameters are set as follows: Minsplit = 20, Minbucket = 7, CP = 0.001.

For the “Q_s” dataset, we use two feature sets respectively composed of 13 basic features detailed in Table 2, and the 14 critical features selected by statistical methods. We also set the time window to 12 hours. For the BP ANN models based on 13 and 14 features, the input layers respectively have 13 and 14 nodes, both the hidden layers have 10 nodes, and both the output layers have 1 node. The maximum number of iterations is set to 400 and the learning rate is set to 0.1. For the decision tree models, the parameters are set to the same value as those with “W” dataset.

When we test a drive, we check its samples in chronological order, and predict that the drive is going to fail if any sample is classified as failed. Otherwise, the
Table 3: Effectiveness of the critical features on the “W” dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>FAR (%)</th>
<th>FDR (%)</th>
<th>TIA (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP ANN</td>
<td>12 features</td>
<td>0.44</td>
<td>89.5</td>
<td>348</td>
</tr>
<tr>
<td></td>
<td>19 features</td>
<td>0.25</td>
<td>90.2</td>
<td>345</td>
</tr>
<tr>
<td></td>
<td>13 features</td>
<td><strong>0.22</strong></td>
<td><strong>91.7</strong></td>
<td><strong>359</strong></td>
</tr>
<tr>
<td>Decision Tree</td>
<td>12 features</td>
<td>0.57</td>
<td>95.5</td>
<td>352</td>
</tr>
<tr>
<td></td>
<td>19 features</td>
<td>0.63</td>
<td>94.7</td>
<td>351</td>
</tr>
<tr>
<td></td>
<td>13 features</td>
<td><strong>0.56</strong></td>
<td><strong>95.5</strong></td>
<td><strong>351</strong></td>
</tr>
</tbody>
</table>

Table 4: Effectiveness of the two different feature sets on the “Q_s” dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>FAR (%)</th>
<th>FDR (%)</th>
<th>TIA (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP ANN</td>
<td>13 features</td>
<td>0.18</td>
<td>92.0</td>
<td>1165</td>
</tr>
<tr>
<td></td>
<td>14 features</td>
<td>0.25</td>
<td>98.0</td>
<td>998</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>13 features</td>
<td>0.42</td>
<td>92.0</td>
<td>1146</td>
</tr>
<tr>
<td></td>
<td>14 features</td>
<td>0.33</td>
<td>96.0</td>
<td>1060</td>
</tr>
</tbody>
</table>

drive is classified as good. Table 3 and Table 4 show the results for the “W” and “Q_s” datasets, respectively.

Both of the two feature sets, 13 features for the “W” and 14 features for the “Q_s” dataset selected by our methods, outperform other feature sets in prediction performance with both prediction models. Therefore, all of the following analyses and discussions about the “W” dataset are based on the 13 features, and those about the “Q_s” and “Q_all” datasets are based on the 14 features.

Compared with BP ANN, the DT model does not perform well on the “Q_s” dataset; we attribute this to the simplicity of the tree. When we adopt some optimization strategies on the DT model, it can achieve a better prediction performance, as shown in Section 5.4.1. Additionally, since we collected the samples in a period of 60 days before actual failure for the failed drives in “Q_s”, the TIAs of BP ANN and DT on “Q_s” are about 1,000 hours.

5.3. Model Evaluation

When we evaluate the decision tree (DT) and gradient boosted regression tree (GBRT) models, we use only the “W” dataset.
Table 5: Impact of the time window on the decision tree model.

<table>
<thead>
<tr>
<th>Time window</th>
<th>FAR (%)</th>
<th>FDR (%)</th>
<th>TIA (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 hours</td>
<td>0.31</td>
<td>94.0</td>
<td>354</td>
</tr>
<tr>
<td>24 hours</td>
<td>0.33</td>
<td>94.0</td>
<td>355</td>
</tr>
<tr>
<td>48 hours</td>
<td>0.39</td>
<td>95.5</td>
<td>351</td>
</tr>
<tr>
<td>96 hours</td>
<td>0.21</td>
<td>96.2</td>
<td>352</td>
</tr>
<tr>
<td>168 hours</td>
<td><strong>0.09</strong></td>
<td><strong>95.5</strong></td>
<td><strong>355</strong></td>
</tr>
<tr>
<td>240 hours</td>
<td>0.11</td>
<td>93.2</td>
<td>361</td>
</tr>
</tbody>
</table>

5.3.1. Evaluating the Decision Tree Model

To distinguish good and failed drives more effectively, we modify the probability distributions of the good and failed samples in the training set to affect node splitting when building trees as follows. We boost the failed sample set by giving it a higher weight, which results in the failed sample set to amounting to 20% of the total drives and the good sample set to occupy 80%. In addition, due to good drives being the overwhelming majority, a high FAR implies too many false alarms and results in heavy processing cost. So to lower FAR, the loss weight specified for FAR is 10 times higher than that for FDR, which will affect the choice of variable on which to split the dataset at each node.

We test the impact of varying the time window on the prediction performance of the DT model. Six different time windows are used to train the DT models; the good training samples remain the same. The results are shown in Table 5. As expected, adjusting the time window provides a coarse way to trade off FDR with FAR. When the time window is set to 168 hours (i.e., 7 days), the DT model obtains the best performance with a FDR of 95.5% at the FAR of 0.09%.

Since an individual sample does not give a reliable prediction of an impending drive failure due to measurement noise, it is not appropriate to predict that a drive is going to fail if only one sample is classified as failed by the model. Motivated by this, we apply the plain voting-based detection algorithm [20] to the DT model. When making a prediction for a drive, we check the last \( N \) consecutive samples, which we call voters, before a time point, and predict that the drive is going to fail if more than \( N/2 \) samples are classified as failed; otherwise the next time point is tested. If all time points pass, the drive is classified as a good drive. As \( N \) increases, the FAR of the DT model drops quickly while its FDR decreases
slowly. With 27 voters, the DT model predicts over 93% failures at a FAR of 0.009%. The details can be seen in Figure 6. We use the voting-based detection algorithm with the conservative setting of \( N = 7 \) in the following experiments of DT model.

5.3.2. Evaluating the Gradient Boosted Regression Trees Model

To evaluate the health degree model based on gradient boosted regression trees, we first train a DT model using the training set, and then use it to determine the TIA for each failed drive in the training set. If a failed drive is missed by the DT model, its deterioration window is set to 24 hours. The target value of each failed sample is then set by (9), where \( U \) is set to 12 and \( H \) is set to \(-0.95\). We take samples in different time periods before failure to train the GBRT model; specifically, we choose 12 samples spread evenly within the deterioration window for each failed drive. To evaluate the effectiveness of the health degree model, for comparison we train another GBRT model with the failed time window set to 12 hours where the target values are set to \(+1\) and \(-1\) respectively for good and failed samples.

When we build the GBRT models, we set the learning rate \( \alpha = 0.1 \), number of iterations \( c = 500 \), and tree-depth \( d = 5 \). We use a new average voting-based detection algorithm for the GBRT models: For each test drive, if the average output of the last Num samples is lower than some threshold, the drive is predicted to fail, and the next time point is tested otherwise. Figure 3 plots the ROC curves of the health degree model and the control group as the threshold varies; in Figure 3, Num is set to 7. The classifier model achieves e.g. a FDR above 90% with zero FAR, which is better than the DT model’s performance.

For the health degree model, since the samples for failed drives are clustered together in time, it does not predict as well as the classifier model, but it nevertheless results in a better description of the deterioration of drives. Figure 4 shows the predicted health degree of a single failed drive by the health degree model, i.e., the GBRT model with target value if each failed sample set by (9), and the classifier model, i.e., the GBRT model with the failure time window set to 12 hours and \( \pm 1 \) target values for good/failed samples. As time proceeds, the health degree predicted by health degree model trends downward, i.e., the drive has worse health closer to failure, whereas the health degree of the same drive predicted by the classifier model hovers around \(-1\). This pattern was similarly verified for 10 randomly chosen failed drives. We conclude that the health degree model can help us better predict drive failures.

In the following experiments for the GBRT model, we use the same method
Figure 3: ROC curves for the GBRT models. The points on curve of health degree model are obtained by setting threshold equal to $-0.25, -0.2, \ldots, 0$ from left to right. The points on curve of classifier are obtained by setting threshold equal to $-0.45, -0.4, \ldots, 0$ from left to right.

Figure 4: The outputs of one failed drive predicted by health degree and classifier model.

of setting the initial target values of the training samples as for the health degree model, and the same parameter settings to train the GBRT models, and the average voting-based detection algorithm with $\text{Num} = 7$ to test drives.

The health degree model has another advantage over binary classifiers, such as the DT model. Since it outputs real values, we can achieve a fine-grained trade off between the FDR and the FAR by simply applying the model with different detection thresholds, whereas the DT model can only make the trade-off coarsely by tuning some training and detection parameters. In other words, the health degree model provides a way to estimate the fault probability more finely, as well as additional flexibility in adjusting performance.

Besides prediction accuracy, providing sufficient time (measured by TIA) to users for backing up or migrating data, is also important. For correct predictions, Figure 5 shows the distribution of hours in advance, respectively, where DT has
93.2% FDR and 0.009% FAR, and GBRT has 87.2% FDR and no false alarms. In both models, all but around 2.5% of correct predictions are predicted 24 hours before failure, and a 24-hour window is a goal of hard drive manufacturers [29], and moreover, both models have an average TIA of over two weeks.

A long TIA implies that when a drive is predicted to fail, it does not immediately break down but will incur a process of gradual deterioration. This can be exploited e.g. by dynamically changing the storage method for at-risk data [30] (which reduces storage costs) or continuing to use drives which are predicted to fail far in advance and adjusting migration rates [24] (which improves availability and reduces migration costs).

All the experiments for the DT and GBRT models are performed on a standard PC desktop, since none of these methods need significant computational resources. The training of DT models can be completed within several seconds, and the training of GBRT models can be completed within several minutes. Moreover, failure prediction is even faster than training. The time cost of the proposed methods is suitable for on-line real-time monitoring of large-scale storage systems.

5.3.3. Model comparison

We compare the proposed DT and GBRT models with two state-of-the-art prediction models, the BP ANN model and the PLATE model [31], for their prediction performance on the “W” dataset. The results are shown in Figure 6. The PLATE model is a simple threshold based predictor. For a sample, if its count of reallocated sectors exceeds the preset threshold, it is classified as failed by the PLATE model. We set the threshold to 200. For all the four models, the voting-
Failure detection rate (%)  

Figure 6: Prediction results of DT, GBRT, and BP ANN models on the “W” dataset using the voting-based detection method. The points on each curve are achieved by setting the number of voters $N$ as 1, 3, ..., 11, 15, 17 and 27 from right to left. Omitted are the results for the PLATE model; as $N$ increases, its FDR decreases slowly from 57.1% to 55.6% while its FAR is always measured at 0.74%.

based failure detection algorithm is used. The detection threshold of GBRT is set to 0.2.

We make two observations: First, as $N$ increases, the FARs of the DT, GBRT, and BP ANN models drop quickly while their FDRs remain at high level. Meanwhile, the FAR of the PLATE model remains unchanged while its FDR drops slowly. Second, compared to the BP ANN and PLATE models, the DT and GBRT models achieve both a higher FDR and lower FAR, and can reach a very low false alarm rate while maintaining a high detection rate. In addition, for all four models, the average time in advance is about 335 to 350 hours.

5.4. Simulating Practical Use

We evaluate the DT and GBRT models by simulating their application in real-world data centers—being used with different drive families, being used with multiple drive models, and being used in small-scale data centers.

5.4.1. Different drive models

Different models of drives have different characteristics which may impact their reliability, even if they are made by the same manufacturers. Consequently, effectiveness with varying drive models is an important factor in prediction models. However, previous work has paid little attention to this, partly because of the unavailability of appropriate datasets. We test the proposed DT and GBRT models
Table 6: Effectiveness of the DT and GBRT models on the “Q_s” dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>FAR (%)</th>
<th>FDR (%)</th>
<th>TIA (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>0.12</td>
<td>96.0</td>
<td>1016</td>
</tr>
<tr>
<td>GBRT</td>
<td>0.02</td>
<td>86.0</td>
<td>1277</td>
</tr>
</tbody>
</table>

on the “Q_s” dataset, which is composed of different drive models from that of the “W” dataset.

For the DT model, we use a 12-hour time window and the same parameter settings as in Section 5.3.1 except for the loss weight specifications for FAR and FDR. In this experiment, the loss weight specified for FAR is 20 times higher than that for FDR. For the GBRT model, the detection threshold is set to 0. The prediction results are shown in Table 6. On the “Q_s” dataset, both the DT and GBRT models maintain good performance, as on “W”, which demonstrates the effectiveness of the proposed models with varying drive models.

The experimental results also illustrate the advantage of DT model in interpretability. For example, by analyzing the trees, we can find the most effective rules at predicting “W” drive failures are:

\[(TC < 23.5 \& SUT < 98.5),\]  
\[(TC \geq 24.5 \& RUE < 99.5 \& SUT < 99 \& POH < 95.5 \& SER < 79.5),\]  
\[(24.5 > TC \geq 23.5 \& SUT < 98.5 \& RSCr \geq 1.5), \text{ and}\]  
\[(TC \geq 24.5 \& RUE \geq 99.5 \& SER < 78.5 \& POH < 94.5 \& SUT < 90).\]

And the most effective rules at predicting “Q_s” drive failures are:

\[(POH \geq 88.5 \& CPSC < 99.5), \text{ and}\]  
\[(POH \geq 95.5 \& CPSC \geq 99.5 \& SUT < 92.5 \& SER \geq 71.5).\]

5.4.2. Multiple drive models

It is not unusual for there to be multiple drive models in a storage system in a real-world setting. Building a prediction model for every drive model would

---

2The acronyms are: SUT = Spin Up Time, TC = Temperature Celsius, RUE = Reported Uncorrectable Errors, RSCr = Reallocated Sectors Count (raw value), POH = Power on Hours, SER = Seek Error Rate, CPSC = Current Pending Sector Count.
be impractical, so training prediction models using samples from different drive models becomes inevitable in such data centers. To evaluate the effectiveness of the proposed models in such an environment, we test them on the “Q_all” dataset.

The “Q_all” dataset is a hybrid dataset. The drives are collected from multiple rooms of a data center, and contain 28 drive models of 5 manufacturers. We use the same time window and parameter settings as in Section 5.4.1 to train the DT model. The detection threshold of GBRT is set to 0. The prediction performance of the DT and GBRT models on the “Q_all” dataset is shown in Table 7.

Both the DT and GBRT models perform worse (in terms of FDR) than in the single-drive dataset (Table 6), which is because the multiple drive models can influence the statistical behavior of failures. Nevertheless, the prediction performance is still acceptable for practical use.

5.4.3. Number of drives

The datasets used in above experiments, “W”, “Q_s”, and “Q_all”, were collected from two large data centers. In the real world, however, prediction models will most likely be used in small and medium-sized data centers. To evaluate the effectiveness of prediction models applying to small and medium-size data centers, we test them with synthesized datasets containing fewer drives. We create four small datasets (named $W_1$, $W_2$, $W_3$, and $W_4$) by randomly choosing 10%, 25%, 50%, and 75% of all the good and failed drives respectively from the “W” dataset. So the smallest dataset $W_1$ contains only 2,790 good drives and 43 failed drives.

On all four datasets, we use the same time window and parameter settings as in Section 5.3.1 to train the DT model, and the same targets setting and parameter settings as in Section 5.3.2 to train the GBRT model. The detection threshold of GBRT is set to $0.6$, $0$, $0.4$, and $0$ for the datasets $W_1$, $W_2$, $W_3$, and $W_4$, respectively.

Table 8 shows the prediction performance of the DT and GBRT models with these datasets. As we would expect, both the DT and GBRT models decrease in performance for the smaller datasets. However, even with the dataset that is one
order of magnitude smaller than the original dataset, both models obtain acceptable FDR and FAR. Moreover, both models maintain an average TIA about two weeks.

Table 8: Prediction performance on small-sized datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>FAR (%)</th>
<th>FDR (%)</th>
<th>TIA (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>W₁</td>
<td>0.09</td>
<td>82.3</td>
<td>329</td>
</tr>
<tr>
<td></td>
<td>W₂</td>
<td>0.20</td>
<td>96.9</td>
<td>336</td>
</tr>
<tr>
<td></td>
<td>W₃</td>
<td>0.21</td>
<td>87.5</td>
<td>334</td>
</tr>
<tr>
<td></td>
<td>W₄</td>
<td>0.10</td>
<td>90.0</td>
<td>336</td>
</tr>
<tr>
<td>GBRT</td>
<td>W₁</td>
<td>0.31</td>
<td>82.3</td>
<td>302</td>
</tr>
<tr>
<td></td>
<td>W₂</td>
<td>0.05</td>
<td>93.7</td>
<td>339</td>
</tr>
<tr>
<td></td>
<td>W₃</td>
<td>0.20</td>
<td>90.6</td>
<td>343</td>
</tr>
<tr>
<td></td>
<td>W₄</td>
<td>0.006</td>
<td>90.0</td>
<td>345</td>
</tr>
</tbody>
</table>

6. Reliability Analysis

We use several Markov models to evaluate the benefits of the decision tree models on reliability. We assume independent and exponentially distributed probabilities for drive failures, failure warnings, and failure repairs (when a failure is predicted or occurred).

Eckart et al. [19] deduced a formula for approximating the Mean Time To Data Loss (MTTDL) of a single hard drive with failure prediction:

\[
\text{MTTDL} \simeq \frac{\text{MTTF}}{1 - \frac{\mu \text{FDR}}{\mu + \gamma}}
\]  

where MTTF is the Mean Time To Failure of a single drive and \( \gamma = 1/\text{TIA} \) and \( \mu = 1/\text{MTTR} \), where MTTR is the Mean Time To Repair. This formula describes a direct relationship between the MTTDL of a single drive and the failure prediction accuracy. For a single drive, we transform its MTTDL in (10) to the expected data loss probability within one year, by which we can better understand the reliability differences between drives with different prediction models.

Table 9 shows the probability of a single drive incurring data loss over a one year time period with different prediction models. This is calculated via the cu-
Table 9: Impact of failure prediction on the reduction of the data loss probability within one year. MTTF = 1,390,000 hours, MTTR = 8 hours. For the GBRT model, FDR = 0.8722 and γ = 1/344 hours⁻¹, and for the DT model, FDR = 0.9549 and γ = 1/355 hours⁻¹.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data loss prob. (%)</th>
<th>% reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>No prediction</td>
<td>0.630</td>
<td></td>
</tr>
<tr>
<td>GBRT</td>
<td>0.093</td>
<td>85.2</td>
</tr>
<tr>
<td>DT</td>
<td>0.042</td>
<td>93.4</td>
</tr>
</tbody>
</table>

Cumulative distribution function (cdf) of the exponential distribution:

\[
cdf(t) = 1 - \exp\left(-t/MTTDL\right) \\
= 1 - (1 - t/MTTDL + \cdots) \quad \text{using the Taylor series for } e^x \text{ about } 0 \\
\simeq t/MTTDL.
\]

Thus, when \( t = 1 \) year, \( t \) will be much smaller than MTTDL (which we find to be around thousands of years) where the above approximation holds.

Both the two prediction models can improve the reliability significantly, even for the simplest single hard drive configuration. Moreover, although the failure detection rate advantage of DT model over the GBRT model is small, it halves the probability of single-drive data loss. This implies that even a small improvement in prediction accuracy is worthwhile. Note that the FAR of the GBRT model was measured as low as 0, which would reduce consequent backup costs for an operator.

To evaluate the benefits of drive failure prediction on large-scale storage systems, we build a Markov model, as shown in Figure 7, which we use for measuring the reliability of RAID-6 systems with proactive fault tolerance. For any system configuration with \( N \) drives, there are \( 3N + 1 \) states: \( N + 1 \) prediction states \( P_i \) in which \( i \) drives are predicted to fail; \( N \) single-erasure states, \( SP_i \), where one drive has already failed and \( i \) other drives are predicted to fail; \( N - 1 \) double-erasure states \( DP_i \); and the absorbing state \( F \) where data loss occurs.

Commercial companies often prefer consumer SATA hard drives which are much cheaper but less reliable than enterprise-class SAS drives in data centers. Therefore, we compare four RAID systems: two RAID-6 systems respectively composed of SAS and SATA drives, another SATA RAID-6 system, and a SATA RAID-5 system. The latter two systems employ the DT model, and the first two
are traditional reactive storage systems without failure prediction. Figure 8 plots the systems’ reliability as their size increases. We calculate the first two systems’ MTTDL using

$$MTTDL_{RAID-6} \simeq \frac{MTTF^3}{N(N-1)(N-2)MTTR^2}$$

as in [32], and the latter two systems’ MTTDL by: (a) the Markov models shown in Figure 7, and (b) using the method in [19].

We transform the MTTDL to a more useful measure: the expected number of data loss events per usable petabyte within one year (given by (11)), by which one can better understand the reliability differences between different systems.

Figure 8 plots the reliability in RAID systems. We can see that although the mean time to failure (MTTF) of SATA drives is 30% lower than that of SAS drives, the SATA RAID-6 system with the DT model results in several orders of magnitude fewer data loss events than that of the RAID-6 system composed of SAS drives but without drive failure prediction. Thus, with the help of the DT model, we can significantly improve the reliability of storage systems constructed with inexpensive drives. The curves of the other three systems are similar, especially when the systems are large. By employing the proposed prediction model, one can maintain similar or better reliability, while using less expensive hard drives.

7. Conclusions

In this paper, we study hard drive failure prediction methods to improve the reliability and availability of storage systems. We first select critical features using
Figure 8: The expected number of data lost events per usable petabyte within one year in RAID systems. We use a DT model where FDR = 0.9549 and $\gamma = 1/355$ hours$^{-1}$. The MTTF of the SAS drive is set to 1,990,000 hours, and the MTTF of the SATA drive is set to 1,390,000 hours. The capacity of every drive is 1 TB.

an easy and intuitive method. Then, we propose a new hard drive failure prediction model based on the decision trees (DTs) and a health degree evaluation model based on the gradient boosted regression trees (GBRTs). Compared with previous binary classifiers, including the DT model, the GBRT model can give each drive a continuous (i.e., non-binary) value representing its health degree. Consequently, users can prioritize handling at-risk hard drives, which improves both reliability and availability. Moreover, the GBRT model provides an easy way to tune the detection rate and the false alarm rate by adjusting a failure detection threshold.

Compared with the state-of-the-art models, the PLATE and BP ANN models, the proposed models perform better in prediction accuracy as well as stability and interpretability. We test the prediction models on real-world datasets, the drives of which are collected from multiple rooms of two data centers, and from several dozens of drive models from 5 manufacturers. Both the DT and GBRT models show steady and good prediction performance. These experimental results indicate that the new models are suitable for practical use in real-world data centers.

To evaluate the benefits of the models on system reliability, we present a Markov model describing RAID-6 systems with failure prediction. The simulation results show that the proposed models can significantly improve reliability and/or reduce cost.
References


