On Predictability of System Anomalies in Real World

Yongmin Tan, Xiaohui Gu
Department of Computer Science
North Carolina State University
ytan2@ncsu.edu, gu@csc.ncsu.edu

***Abstract*—As computer systems become increasingly complex, system anomalies have become major concerns in system management. In this paper, we present a comprehensive measurement study to quantify the predictability of different system anomalies. Online anomaly prediction allows the system to foresee impending anomalies so as to take proper actions to mitigate anomaly impact. Our anomaly prediction approach combines feature value prediction with statistical classification methods. We conduct extensive measurement study to investigate anomalous behavior of three systems in the real world: PlanetLab, SMART hard drive data, and IBM System S. We observe that real world system anomalies do exhibit predictability, which can be predicted with high accuracy and significant lead time.**

I. Introduction

Modern computer systems (e.g., cloud computing platforms [1], [2], enterprise data centers, massive data analytics [3], and web hosting services) have become increasingly complex as systems grow in both scale and functionality. Unfortunately, such complexity makes systems more vulnerable to various anomalies such as performance bottlenecks, resource hotspots, service level objective (SLO) violations, and various software/hardware failures. System administrators are often overwhelmed by the tasks of correcting system problems under time pressure. Thus, it is imperative to provide automatic system anomaly management to achieve robust computer systems.

Previous system anomaly management work (e.g., [4], [5], [6], [7], [8], [9]) can be classified into two categories: (1) *reactive* approaches that take corrective actions after an anomaly happens, and (2) *proactive* approaches that take preventive actions on all system components beforehand. The reactive approach does not have prevention cost but can have prolonged service downtime, which is often unacceptable for continuously running applications such as data stream processing. Moreover, it is often difficult to reproduce the anomaly-inducing environments to perform offline anomaly diagnosis, which may limit the effectiveness of reactive anomaly correction. In contrast, proactive approach offers better system reliability but can incur prohibitive overhead. To this end, we explore a new *predictive* anomaly management approach that can foresee impending system anomalies through intelligent prediction so that we can take just-in-time corrections to steer the system away from the abnormal state.

To achieve efficient predictive anomaly management, one big challenge is to provide high quality online system anomaly prediction. Although previous work (e.g., [10], [11], [12]) has addressed the anomaly detection problem, anomaly prediction needs to capture pre-anomaly symptoms to raise advance anomaly alert before the anomaly happens. In [13], we presented the initial design of our online anomaly prediction scheme. However, one big question is whether real system anomalies do exhibit certain predictability and whether our anomaly prediction scheme can efficiently capture the predictability. The goal of this work is to provide quantitive answers to the question by conducting a comprehensive measurement study over a range of anomalies in real production systems.

Our anomaly prediction approach aims at achieving *advance* anomaly prediction with a certain *lead time*. The intuition behind our approach is that system anomalies often manifest gradual deviations in some system metrics before the system escalates into the anomaly state. We monitor various system metrics called features (e.g., CPU load, free memory, disk usage, network traffic) to build a feature value prediction model using discrete-time Markov chain schemes. We also induce statistical anomaly classifiers using naive Bayesian or tree augmented naive Bayesian (TAN) learning methods. We then integrate feature prediction and anomaly classifier to forecast the system anomaly state at a future time. In this paper, we conduct an extensive comparative study to quantify the predictability of system anomalies in three production systems: (1) *PlanetLab failure data* that incorporate real host failures over a two-month period on the widely used wide-area computing platform PlanetLab [14]; (2) *SMART disk failure data* [15] that include measurement data of 369 real-world disks, among which 178 disks experience disk failures; and (3) *IBM System S performance anomaly data* that report measurements of SLO violations on the IBM stream processing cluster [3]. Our extensive measurement study reveals the following key observations:

First, real world system anomalies exhibit varied predictability. Our anomaly prediction scheme can achieve i) more than 95% true positive rate and less than 20% false positive rate for the PlanetLab ping failure data under [10,90] seconds lead time requirements; ii) 80-85% true positive rate and less than 10% false positive rate with up to 18 hours lead time for the SMART disk failure data; and iii) 95-85% true positive rate and less than 10% false positive rate with 2 to 18 seconds lead time for the System S performance anomaly data.

Second, using proper discretization methods, the discrete-time Markov chain scheme can achieve high prediction accuracy for most system metrics except those metrics whose values present irregular changing patterns or large variation ranges. However, we observe that low prediction accuracy over a small number of metrics does not significantly affect the accuracy of the integrated anomaly prediction model.

Third, simple statistical classifiers such as naive Bayesian and TAN models can achieve high accuracy given sufficient training data for both normal and abnormal states. The TAN model outperforms the naive Bayesian method in most cases since it relaxes the strong independence assumption made by the naive Bayesian method. However, we also observe one exception in the SMART disk failure dataset where TAN has worse performance than naive Bayesian. The reason is because the TAN model estimates the conditional probability based on not only the class variable but also other metrics. The SMART data set includes some metrics that have large variation range and imbalanced distribution, which results in unreliable estimation of some conditional probabilities.

The rest of the paper is organized as follows. Section II presents the design and algorithms of the anomaly prediction schemes. Section III describes the anomaly data collection and experimental results. We discuss related work in Section IV. Finally, the paper concludes in Section V.

II. System Design

In this section, we present the design details of our system anomaly prediction scheme. We first describe the feature value prediction scheme followed by the statistical anomaly classification methods. We then present the integrated anomaly prediction model.

A. Feature Evolving Pattern Model

We use finite discrete-time Markov chains (DTMC) to model the evolving patterns of various system features such as CPU consumption, memory usage, and input/output data rate. For example, Figure 1 shows a Markov model for a metric ranging from 0 to 30 with three discretized states. To build a Markov chain model for a metric x with M distinct states, we learn the transition probability matrix P_x: an $M \times M$ matrix where the element p_{ij} at row i and column j denotes the conditional probability of making a transition from state i to state j. We derive P_x from a training data set by counting the number of different state transitions observed.

Assuming the Markov chain is homogeneous, we can derive the feature value distribution of x for any time in the future by applying the Chapman-Kolmogorov equation: after t time units, the probability distribution for metric x is $\pi_t = \pi_{t-1} P_x = \pi_{t-2} P_x^2 = ... = \pi_0 P_x^t$, where π_t and π_0 denote the probability distribution at time t and the initial probability distribution for the metric x, respectively. Given a current state i, if we want to predict the state at a future time t, we only need to check those elements of matrix $\pi_t = \pi_0 P_x^t$ at row i to decide the most probable state (i.e., the state with the largest transition probability) that the current state i will evolve to after time t.

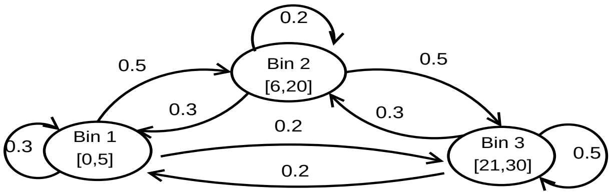


Fig. 1. Feature evolving pattern prediction. The metric value ranges from 0 to 30. We partition the values of the metric into three states. The arcs are labeled with state transition probability.

To use DTMC, we need to perform discretization to transform continuous feature values into discrete states. Common discretization techniques include equal-width and equal-depth approaches. The equal-width approach divides the range of a feature value into M equal-width bins while the equal-depth approach puts the same number of samples into each bin. However, we find that the above two approaches can incur high prediction error during our experimental study.

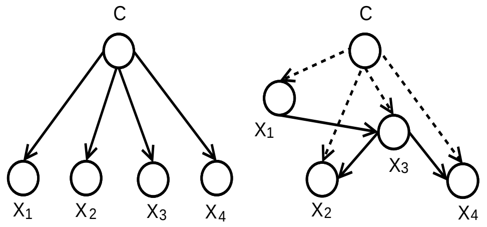
To address the problem, we propose a hybrid discretization approach. We first use the equal-width approach to create M bins. We check the number of data samples fallen into each bin. If there is no bin with extremely small number of data samples (e.g. less than 10% of the second smallest bin), the discretization is accepted. Otherwise, we apply the equal-width approach again but use more bins (e.g. $2M$). Then we recursively merge some bins with their neighbors. In each iteration, we merge the bin containing the smallest number of data samples. We proceed until the total number of bins is reduced to the target number M. The merit of this hybrid approach is two-folded: it preserves the original continuous attribute distribution; and it eliminates the negative effect of some outliers by balancing the number of data samples allocated to different bins. In the experimental section, we will show the impact of M and different discretization approaches on the accuracy of feature value prediction.

It is also possible to apply other prediction methods such as Kalman filter to predict feature values at a future time. We choose DTMC in this work since DTMC can provide the probabilities of all possible values a feature can have at a future time. Thus, we can easily integrate the feature value prediction results with the statistical anomaly classifiers to compute the anomaly probability at a future time. The details about the integrated anomaly prediction model will be described in Section II-C.

B. Statistical Anomaly Classification

The goal of the anomaly classifier is to decide whether the system is currently running in a normal or abnormal state. Let $\vec{\mathbf{x}_t}$ denote a measurement sample, which is a vector of system metric values $[x_0, \ldots, x_n]$ at time t. Let C_t denote the system state at time t[1], which can take one of the two states $\{abnormal(1), normal(0)\}$. The input to the classifier is a training data set that contains a time series of records $\langle \vec{\mathbf{x}_t}, C_t \rangle$.

[1] we will omit the subindex t when the context is clear.



(a) naive Bayesian classifier

(b) TAN classifier

Fig. 2. Statistical anomaly classifier for one class variable C and four feature variables x_1, x_2, x_3, x_4.

Note that our classifier training process is supervised since we depend on an anomaly detector [16] (e.g., application-specific anomaly predicates [10]) to provide a proper class label C_t for each training sample $\vec{\mathbf{x}}_t$.

Ideally, the classifier should be able to produce posterior probabilities, i.e. $P(C = 1|\vec{\mathbf{x}})$ and $P(C = 0|\vec{\mathbf{x}})$ for a given measurement $\vec{\mathbf{x}}$. We then compare the posterior probabilities for $abnormal$ and $normal$ classes to decide the classification result. That is, the system is classified as abnormal if the following inequality holds:

$$\log P(\text{“}abnormal\text{”}|\vec{\mathbf{x}}) - \log P(\text{“}normal\text{”}|\vec{\mathbf{x}}) > \delta \qquad (1)$$

Otherwise, the system is considered to be in the normal state. Larger δ means stronger classification confidence since the likeliness of one class is overwhelmingly greater than that of the other class. A typical value of δ is either zero or the prior difference of the likelihood derived from the training data.

However, computing the posterior probability can be challenging: we need to evaluate $P(C = c|\vec{\mathbf{x}})$ for every possible $\vec{\mathbf{x}}$ in the multi-dimensional feature space. If the dimensionality is high, the computation will be very costly. We leverage Bayes' rule to transform the posterior probability $P(C = c|\vec{\mathbf{x}})$ into the conditional probability $P(\vec{\mathbf{x}}|C = c)$. We apply naive Bayesian classifier [17] and tree-augmented naive Bayesian (TAN) network [18] in this work.

Naive Bayesian Classifier. The assumption of a naive Bayesian classifier is that each metric is independent given the class label, illustrated by Figure 2(a). To compute $P(C = c|\vec{\mathbf{x}})$, we apply the Bayes' rule to transform the posterior probability into the conditional probability:

$$P(C = c|\vec{\mathbf{x}}) = \frac{P(\vec{\mathbf{x}}|C = c)P(C = c)}{P(\vec{\mathbf{x}})} \qquad (2)$$

We neglect the denominator $P(\vec{\mathbf{x}})$ which does not depend on C, and only focus on the numerator of the fraction. We further apply the naive independence assumption so that we transform $P(\vec{\mathbf{x}}|C = c)$ into $\prod_{i=1}^{n} P(x_i|C = c)$.

Tree-Augmented Naive Bayesian (TAN) Network. The TAN model extends the naive Bayesian model by considering dependencies among metrics with a constraint that each metric has at most one parent in the network other than the class variable. The structure of the TAN model is a tree rooted at the class variable C and contains conditional probabilities for each tree node, illustrated by Figure 2(b). The posterior probability $P(C|\vec{\mathbf{x}})$ is not exactly equal to, but still proportional to the multiplication of the conditional probabilities of each individual metric. However, different from the naive Bayesian classifier, not all metrics are independent now. We applied an existing scheme[19] to learn the TAN model. For example, we can derive the following proportional relations for the TAN model shown in Figure 2(b): $P(C = c|\vec{\mathbf{x}}) \propto P(x_1|C = c)P(x_2|C = c, x_3)P(x_3|C = c, x_1)P(x_4|C = c, x_3)$.

The posterior probability $P(C = c|\vec{\mathbf{x}})$ is proportional to the probability that C is assigned with “abnormal” or “normal”. We use an odds ratio [20], denoted by $\Omega(\vec{\mathbf{x}})$, to assign class labels to a sample vector $\vec{\mathbf{x}}$, i.e., the system is classified as abnormal if the following inequality holds:

$$\Omega(\vec{\mathbf{x}}) = \frac{P(C = 1|\vec{\mathbf{x}})(1 - P(C = 0|\vec{\mathbf{x}}))}{P(C = 0|\vec{\mathbf{x}})(1 - P(C = 1|\vec{\mathbf{x}}))} > \alpha \qquad (3)$$

The threshold α is a tunable parameter that can be used to control the classification confidence. A typical value of α is one.

C. Integrated Anomaly Prediction

To achieve advance anomaly prediction, our scheme integrates feature value prediction and statistical anomaly classification. Through the feature evolving pattern model, we can predict the values of each metric at a future time. The anomaly classifier is then used to perform classification over future predicted metric values. In other words, during the computation of conditional probabilities in naive Bayesian or TAN classifier, we replace the metric values in Equation 2 and Equation 3 with the predicted metric values in the form of prediction probabilities from the Markov model.

For the naive Bayesian classifier, we need to replace the deterministic discrete value of x_i with all possible discrete values that x_i can take. We denote $P(x_i[s, t])$ as the probability that x_i takes value s at a future time t, given the current value of x_i. Therefore, $P(x_i|C = c)$ becomes $\sum_s P(x_i[s, t]) \cdot P(x_i[s, t]|C = c)$. Since we build a Markov prediction model separately for each collected metric, we aggregate all metrics in $\vec{\mathbf{x}}$ to get the predicted posterior probabilities $\hat{P}(C = c|\vec{\mathbf{x}})$ and use Equation 1 to get the anomaly prediction result.

For the TAN classifier, we compute the predicted posterior probability in the similar way as the naive Bayesian model except for those metrics whose conditional probabilities depend on other metrics, such as x_2, x_3, x_4 in Figure 2(b). For example, the metric x_2 depends on the metric x_3. If we want to evaluate $P(x_2|C = c, x_3)$ at a future time, we need to sum all possibilities of both metric x_2 and metric x_3. To be specific, if we use $P(x_2[s_2, t])$ and $P(x_3[s_3, t])$ to denote the probability that x_2 and x_3 take values s_2 and s_3 at a future time t given the current value of x_2 and x_3, respectively, we can compute $P(x_2|C = c, x_3)$ as $\sum_{s_2} P(x_2[s_2, t]) \sum_{s_3} p(x_3[s_3, t]) \cdot P(x_2[s_2, t]|C = c, x_3[s_3, t])$. We aggregate all metrics in $\vec{\mathbf{x}}$ to get the predicted posterior probability $\hat{P}(C = c|\vec{\mathbf{x}})$ and use odds ratio in Equation 3 to get the anomaly prediction result.

III. System Evaluation

In this section, we evaluate our online anomaly prediction scheme. We perform a comprehensive measurement study over three real-world systems. We first describe our evaluation methodology. Next, we present and analyze our experimental results.

A. Evaluation Methodology

We have implemented the anomaly prediction system and deployed it on several real world computing infrastructures such as PlanetLab [14], NCSU virtual computing lab (VCL) [2], and IBM System S stream processing cluster [3]. One big challenge for this measurement study is to collect real system anomaly data from deployed production systems. Although previous research projects have collected various failure data [21], most of them lack fine-grained continuous measurement data that are required by the anomaly prediction system. One exception is the SMART disk failure data [15], which have been included in our measurement study. To collect more fine-grained real world system anomaly data, we have developed a scalable continuous monitoring system [22] and deployed it on the PlanetLab, VCL, and IBM System S. We have collected a set of real system anomaly data by monitoring those systems for extended period of time. All collected metrics are used to train both the feature value predictor and the anomaly classifier. We now describe the anomaly trace data used in this paper.

PlanetLab anomaly data. We collected measurement data on the widely used planetary-scale open computing platform PlanetLab. We monitored about 400 PlanetLab nodes distributed all over the world. Our system collects 66 host metrics such as CPU load, virtual memory states, disk usage, and network traffic. The detailed description about those metrics can be found on either PlanetLab monitoring site [23] or our InfoScope monitoring site [24]. The metric sampling period is 10 seconds. We started the data collection since January 2009. The dataset used in this set of experiments was collected from Nov. 14th to Nov. 24th, 2009.

Our monitoring infrastructure is capable of capturing three types of node anomalies: 1) ping failure: a host is not responsive to successive five ping trials initiated by the management node; 2) SSH failure: a node can not be accessed through “ssh” command; and 3) monitoring sensor failure: the monitoring sensor program running on that node crashed and cannot be restarted. Each detected failure is recorded in a failure log with the node name and timestamp. The system stores monitoring metric data received from different nodes in separate log files. We correlate the failure log with monitoring metric logs using node name and failure start time. We label 100 records right before each failure occurrence as “abnormal” and other records as “normal”.

SMART disk failure data. Our second anomaly dataset is SMART (self monitoring and reporting technology) data for a collection of real-world disks [15]. The dataset contains time series of SMART attributes collected by SMART incorporated in most modern hard disk drives. In this dataset, there are totally 369 distinct disks, 178 of which are labeled as *good* while the remaining 191 ones are labeled as *failed*. Each disk has a time-series of samples of SMART attribute values. Those samples are collected at a nearly-regular interval of two hours. Most good disks have approximately 300 samples. The number of samples for those failed disks range from 10 to 300. This is because for those failed disks, collected data may get corrupted or even lost before failure occurrences.

In the SMART dataset, one sample originally consists of 59 attributes. We remove those attributes that are obviously not useful for prediction, such as serial number, frame, and hours.

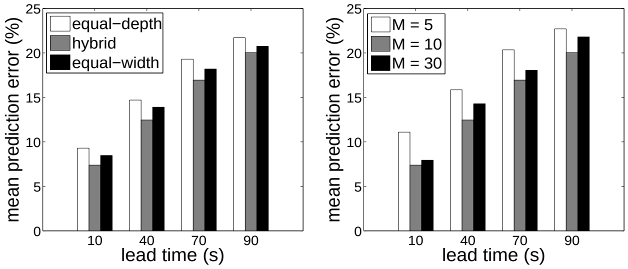
IBM System S anomaly data. We collected our third anomaly trace on the IBM System S [3], a large-scale data stream processing system running on a commercial cluster consisting of about 250 blade servers. We run a complicated multi-modal stream analysis reference application [25]. We collected 21 system metrics with the sampling interval of two seconds. The system anomalies include bottleneck anomaly, throughput anomaly, and processing time anomaly. Those anomalies are caused by various reasons such as memory leak, CPU starvation, and buffer management error.

Evaluation metrics. We evaluate our anomaly prediction algorithms in three aspects. First, we evaluate the feature value prediction accuracy. We use the *mean prediction error* (MPE) to measure the deviation of true values from predicted values. For a collected metric x_i at time t, we know its current discrete value S_t. We derive its future discrete value S'_{t+T} with a lead time T using the algorithm described in section 2.1. We then transform the predicted discrete value S'_{t+T} into a predicted metric value $x'_i(t+T)$ using the average value of all data samples inside the bin representing the discrete value S'_{t+T} . We calculate the mean prediction error for a metric x_i for the whole testing data set D as follows:

$$MPE_i(T) = AVG_D\left(\frac{|x_i(t+T) - x'_i(t+T)|}{x_i(t+T)}\right) \qquad (4)$$

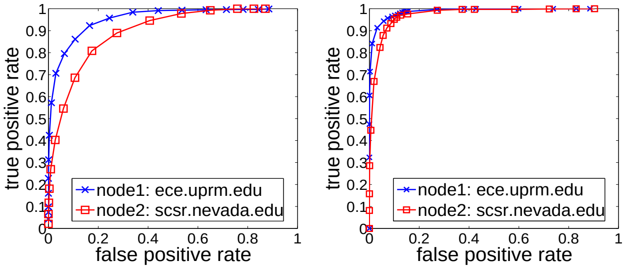
Second, we evaluate the performance of the classifier using the *receiver operating characteristic* (ROC) curve. The ROC curve is often used to show the tradeoff between true positive rate and false positive rate of a classification algorithm. To draw a ROC curve, we change the value of threshold δ in Equation 1 for the naive Bayesian classifier and the value of threshold α in Equation 3 for the TAN classifier to generate a series of (true positive rate, false positive rate) pairs.

Third, we evaluate the performance of the integrated anomaly prediction algorithm using the standard true positive rate A_{tp} and false positive rate A_{fp} metrics. Given a lead time T, the anomaly prediction model infers a class label $\hat{c}(t+T)$ at time t for a future record $\vec{\mathbf{x}}_{t+T}$. On the other hand, $\vec{\mathbf{x}}_{t+T}$ has been annotated with its true label $c(t+T)$. By comparing the predicted label $\hat{c}(t+T)$ with the true label $c(t+T)$, we calculate true positives N_{tp} which corresponds to the number of abnormal samples correctly predicted; true negatives N_{tn} which corresponds to the number of normal samples correctly predicted; false positives N_{fp} which corresponds to the number of normal samples mistakenly predicted as abnormal; false negatives N_{fn} which corresponds to the number of abnormal samples mistakenly predicted as normal. Thus, we can calculate the true positive rate A_{tp} and false



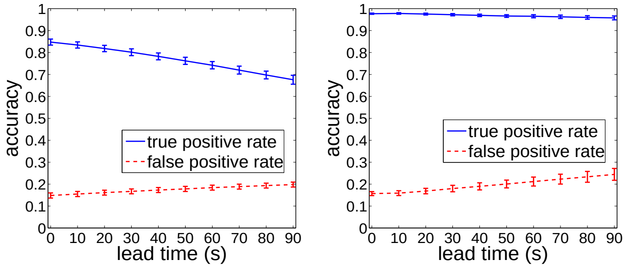
(a) quantization scheme (b) quantization granularity

Fig. 3. Markov prediction error for PlanetLab data.



(a) NaiveBayes (b) TAN

Fig. 4. Classification ROC curves for PlanetLab data.



(a) NaiveBayes + Markov (b) TAN + Markov

Fig. 5. Advance anomaly prediction accuracy for PlanetLab data.

positive rate A_{fp} in a standard way as follows,

$$A_{tp} = \frac{N_{tp}}{N_{tp} + N_{fn}}, A_{fp} = \frac{N_{fp}}{N_{fp} + N_{tn}} \qquad (5)$$

B. Results and Analysis

We now present our anomaly prediction results. For each examined system, we first show the mean prediction error of the Markov predictor under different parameters. We then evaluate the naive Bayesian classifier and the TAN classifier using ROC curves. Finally, we show the integrated anomaly prediction accuracy under different lead time. We also report the overhead of the anomaly prediction system.

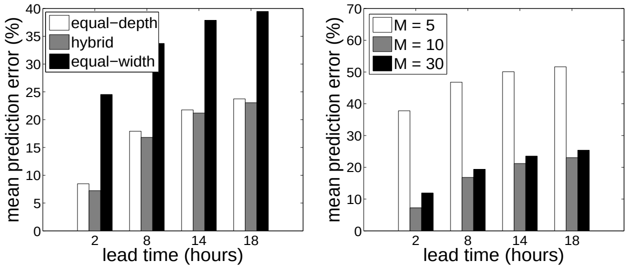
1) PlanetLab Anomaly Prediction: We first examine failure data collected on the PlanetLab. We focus on the host ping failures since we find that SSH failures and sensor program startup failures are rare. Host ping failures occur frequently on the PlanetLab but with various frequencies and durations on different hosts. We observe ping failures on nearly 400 PlanetLab nodes. The average number of occurrences for one node is 15. The average duration of ping failures is 6 hours. For each host, we use first half of the data as training and second half as testing.

Figure 3(a) shows the MPE achieved by the feature value prediction model using different discretization approaches. Generally, MPE increases as the lead time becomes larger. We observe that our Markov predictor achieves reasonable prediction accuracy in all cases. The results show that our hybrid discretization approach consistently achieves lower prediction error than both the equal-width and equal-depth approaches. Figure 3(b) shows the MPE of the feature value prediction model using the hybrid discretization approach under different number of bins. Among the three M values, we observe that the predictor achieves the best prediction accuracy when M is 10. With a small number of bins (e.g., $M = 5$), the discretization scheme tends to group a large range of data samples into one bin. As a result, the "representative" value of one discrete bin may no longer be representative. Even though the prediction in terms of the bin identifier is correct, the difference between the metric's true value and the representative value of that bin in Equation 4 will be large. On the other hand, with a large number of bins (e.g., $M = 30$), each bin is assigned with less training data. If the training dataset is not large enough, the Markov chain will get insufficiently trained. Thus, the predictor may make more mistakes when predicting bin identifier, which will also incur larger feature value prediction error.

We then evaluate the performance of the naive Bayesian classifier and the TAN classifier. When we use the classifier in isolation, it means that we classify whether the current system state exhibits abnormal behavior. Figure 4(a) and Figure 4(b) show the ROC curves for two PlanetLab hosts as examples. Optimal performance should be at the top left of each figure with high true positive rate and low false positive rate. We observe that our classifiers have good performances. Furthermore, the TAN model performs slightly better than the naive Bayesian model.

We now evaluate the performance of our advance anomaly prediction scheme shown by Figure 5(a) and Figure 5(b). We average the results among five PlanetLab hosts and show standard error bars for both true and false positive rates. We have several observations: 1) our system can still achieve reasonably good prediction accuracy for future system state; 2) prediction accuracy drops as the lead time becomes larger, which indicates that predicting anomalies in more distant future is more challenging; 3) TAN classifier has more predictive power than the naive Bayesian classifier, and is more robust under increasing lead time; 4) both algorithms are stable with small standard error bars, which implies the anomaly prediction algorithms are robust for different node failures.

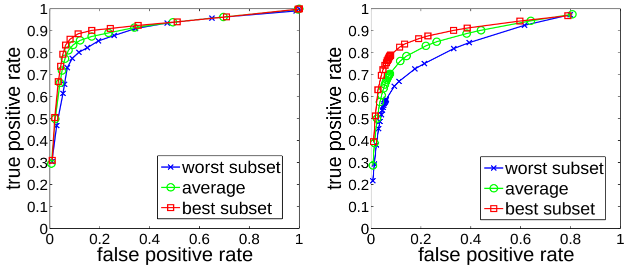
2) SMART Anomaly Prediction: We now present the anomaly prediction results for the SMART dataset. We split the original SMART dataset into six subsets, each of which contains failed and normal disks. Therefore, we conduct six-fold cross-validation for all related experiments. First, we show the impact of different configurations on the prediction accuracy of the Markov predictor in Figure 6(a) and Figure 6(b). Again, we observe that our hybrid discretization approach



(a) quantization scheme

(b) quantization granularity

Fig. 6. Markov prediction error for SMART data.



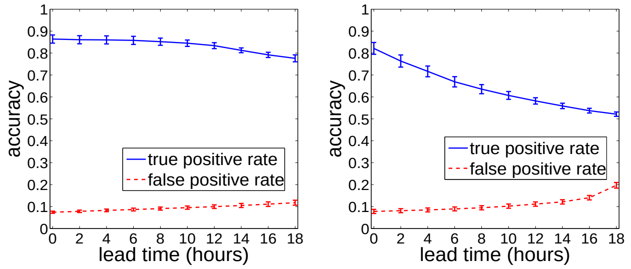
(a) NaiveBayes

(b) TAN

Fig. 7. Classification ROC curves for SMART data.

consistently performs better than the other two approaches. Particularly, the equal-width discretization yields much higher prediction error than the other two approaches in the SMART dataset. The reason is because some SMART metrics have a wide range of values. The equal-width discretization tends to make the range of each bin very big. Therefore, one specific data sample may be numerically far away from the representative value of its bin, especially when the bin contains a small number of training samples. Similarly, the prediction accuracy is the highest when M is neither too small nor too big. For the similar reason as that of the equal-width discretization scheme, the prediction accuracy with bin number equalling to five is much worse than the other two cases.

Figure 7(a) and Figure 7(b) show the ROC curves of the naive Bayesian classifier and the TAN classifier. We show the best and the worst data subsets as well as the average result. For the SMART dataset, low false positive rate is favored since it is costly to replace a good hard drive that is incorrectly predicted to crash soon. We observe that both naive Bayesian classifier and TAN classifier can achieve reasonable detection rate while keeping the false alarm rate very low. ROC curves also implies that we can adjust the threshold to trade-off true and false positive rates. However, different from the previous set of experiments, we observe that the TAN classifier performs worse than the naive Bayesian classifier this time. The reason is that the conditional probabilities of some metrics in the TAN model depends on not only the class label but also other metrics. In the SMART dataset, some metrics have large value range and the distribution of different bins can be very imbalanced. Thus, the estimation of some conditional probabilities become unreliable when some of the metric bins contain very few training samples. This problem is not so acute in the case of the naive Bayesian classifier since it assesses the conditional probability only based on the class variable, and the values of the class variables are adequately represented in the SMART dataset.



(a) NaiveBayes + Markov

(b) TAN + Markov

Fig. 8. Advance anomaly prediction accuracy for SMART data.

Figure 8(a) and Figure 8(b) show the performance of the integrated anomaly prediction algorithms using the naive Bayesian classifier and TAN classifier, respectively. Since the TAN model has worse classification accuracy than the naive Bayesian model, the prediction accuracy also exhibits the same trend. The prediction model using the naive Bayesian classifier can achieve 85-80% true positive rate and 2-9% false positive rate for a lead time of up to 18 hours.

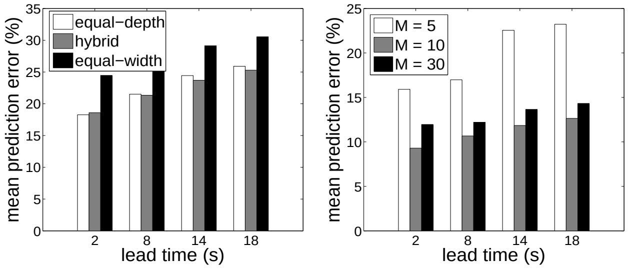
3) System S Anomaly Prediction: We now present the advance prediction results for the System S dataset. Different from previous two systems, this set of experiments focus on performance anomalies (e.g., prolonged processing time, low throughput) caused by various faults such as insufficient resources or program bugs. We collected data from six hosts so that we use six-fold cross-validation in the experiments.

Figure 9(a) and 9(b) show the feature value prediction accuracy under different discretization schemes and quantization granularity, respectively. Again, we observe that the hybrid discretization scheme using 10 discrete bins achieve the best prediction accuracy.

In Figure 10(a) and Figure 10(b), we show the accuracy of the naive Bayesian classifier and the TAN classifier for detecting a performance anomaly caused by a memory leak bug fault. We set the service level objectives (SLOs) in advance and label collected metric vectors with SLO violation or compliance. We observe that both classifiers achieve nearly perfect performance.

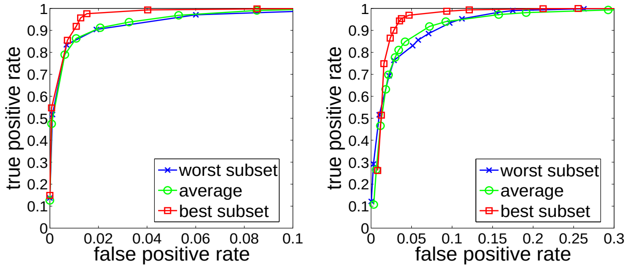
Figures 11(a) and 11(b) show the accuracy of the two integrated anomaly prediction models for predicting System S performance anomalies. We observe that both prediction models can achieve very good prediction accuracy for the System S performance anomaly dataset.

We now evaluate the overhead of our online anomaly prediction model. Table I shows the average training time and prediction time of the two prediction models. The training time includes the time of building the Markov model and inducing the anomaly symptom classifier. The prediction time includes the time to retrieve state transition probabilities, calculate posterior probabilities, and synthesize the classification result for a singe data record. These statistics are collected over 100 different experiment runs. We observe that the total training time



(a) quantization scheme (b) quantization granularity

Fig. 9. Markov prediction error for System S data.



(a) NaiveBayes (b) TAN

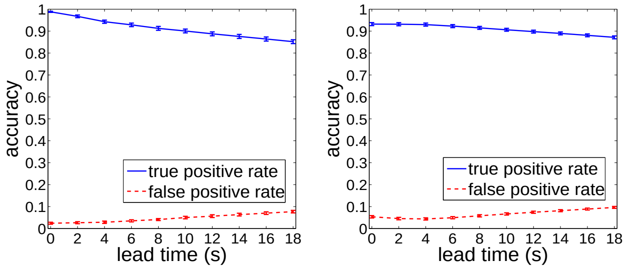
Fig. 10. Classification ROC curves for System S data.

is within several hundreds of milliseconds and the prediction requires less than 150 microseconds. We notice that the naive Bayesian classifier is faster than the TAN classifier in both training and prediction. The above overhead measurements show that our approach is practical for performing online prediction of system anomalies.

There are several other factors that may affect the performance of our anomaly prediction scheme. First, the prediction accuracy depends on the amount and the quality of training data (i.e., prediction model bootstrapping). Ideally, the training dataset should be large enough to cover all pre-anomaly symptoms. Second, the prediction accuracy can be improved by providing adaptability to dynamic execution environments [26]. We plan to further refine our anomaly prediction model along those directions in future work.

IV. Related Work

System anomaly prediction has recently received much research attention. Previous work can be classified into three categories: data-driven, symptom-driven and event-driven approaches. Data-driven methods [27] learn and classify recurring failure patterns from historical data. Symptom-based approaches [28], [29] evaluate periodic measurements of system parameters such as memory consumption, input workload, number of processes. Event-based methods [30], [31], [32], [33] directly analyze time series of error events. Different from previous work, our research focuses on black-box online failure prediction by combining feature value prediction with statistical anomaly classification. Our work is closely related to the Tiresias system [34] which also proposed a block-box failure prediction solution for distributed systems. However, one major difference is that Tiresias first applies anomaly detection on individual metrics to generate a vector of feature anomalies, and then use clustering method to predict failures. Thus, the accuracy of failure prediction depends on accurate anomaly detectors to generate correct anomaly vectors. In contrast, our approach does not require feature anomaly detectors but combines feature value prediction with whole system classification using user-defined anomaly predicates.



(a) NaiveBayes + Markov (b) TAN + Markov

Fig. 11. Advance anomaly prediction accuracy for System S data.

time / scheme	NaiveBayes + Markov	TAN + Markov
training time	412 ±1.5 ms	510 ±2 ms
prediction time	105 ±0.01 us	125 ±0.02 us

TABLE I
Anomaly prediction system cost.

Recently, statistical learning methods have been shown to be promising for autonomic failure management. Cohen et al. proposed to apply the TAN model to correlate system-level metrics to system states [6], and capture the essential characteristic called signature of a system state [16]. Chen et al. explored a decision tree learning approach to diagnose failures [35]. Several machine learning techniques have been used to correlate disk failures with SMART parameters [36], [15]. In comparison, we focus on exploring online learning techniques for classifying future system state.

Considerable research efforts have been conducted on system log analysis. Lin and Siewiorek studied a 22-month log for identifying transient and intermittent error processes [30]. Sahoo et al. evaluated the rule-based classification and Bayesian networks for failure prediction on an IBM cluster [37]. Liang et al. collected RAS event logs from BlueGene/L and proposed three prediction schemes based on the correlations between failure occurrences [38]. Xu et al. proposed a console log mining algorithm to detect runtime problems in Hadoop file system [39]. In comparison, our work focuses on online characterization of black box system anomalies using performance and resource metrics.

Mickens et al. presented several statistical analysis methods for predicting host availability [40]. Power et al. investigated the predictive power of different statistical schemes and learning approaches [41]. Schroeder et al. studied failure statistics in a large-scale high-performance computing system [42]. Pinheiro et al. conducted a comprehensive statistical study on failure trends in a large disk drive population [43]. Javadi et al. discovered statistical models of host availability in a large-scale distributed system SETI@home [44]. Different from the above work, this work focuses on quantifying the predictability of real-world system anomalies.

V. Conclusions

In this paper, we have presented a comprehensive measurement study to quantify the predictability of various system anomalies: PlanetLab host ping failures, SMART disk failures, and IBM System S performance anomalies. We have developed an integrated anomaly prediction scheme that combines efficient feature value prediction with statistical classification methods. To the best of our knowledge, our work makes the first attempt to quantify the tradeoff between prediction accuracy and prediction lead time using real world system anomalies. Our experimental results show that real world system anomalies do exhibit predictability and our anomaly predictor can achieve high prediction accuracy with significant lead time.

Acknowledgment

This work was sponsored in part by NSF CNS0915567 grant, NSF CNS0915861 grant, U.S. Army Research Office (ARO) under grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI), IBM Exploratory Stream Analytics Award, and IBM Faculty Award. Any opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF, ARO, or U.S. Government. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] "Amazon Elastic Compute Cloud," http://aws.amazon.com/ec2/.

[2] "Virtual Computing Lab," http://vcl.ncsu.edu/.

[3] B. Gedik, H. Andrade, K.-L. Wu, P. S. Yu, and M. Doo, "SPADE: the system s declarative stream processing engine," in *Proc. of SIGMOD*, 2008.

[4] E. Kiciman and A. Fox, "Detecting Application-Level Failures in Component-based Internet Services," *IEEE Transactions on Neural Networks*, 2005.

[5] J. M. Agosta, C. Diuk, J. Chandrashekar, and C. Livadas, "An Adaptive Anomaly Detector For Worm Detection," in *Proc. of sysML*, 2007.

[6] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. S. Chase, "Correlating Instrumentation Data to System States: A Building Block for Automated Diagnosis and Control," in *Proc. of OSDI*, Dec. 2004.

[7] J.Tucek, S. Lu, C. Huang, S. Xanthos, and Y. Zhou, "Triage: Diagnosing production run failures at the user's site," in *Proc. of SOSP*, 2007.

[8] G. Candea, S. Kawamoto, Y. Fujiki, G. Friedman, and A. Fox, "Microreboot- A Technique for Cheap Recovery," in *Proc. of OSDI*, Dec. 2004.

[9] K. Vaidyanathan, R. E. Harper, S. W. Hunter, and K. S. Trivedi, "Analysis and Implementation of Software Rejuvenation in Cluster Systems," in *Proc. of SIGMETRICS*, 2004.

[10] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen, "Detecting Past and Present Intrusions Through Vulnerability-Specific Predicates," in *Proc. of SOSP*, Oct. 2005.

[11] L. Cherkasova, K. M. Ozonat, N. Mi, J. Symons, and E. Smirni, "Anomaly? application change? or workload change? towards automated detection of application performance anomaly and change," in *Proc. of DSN*, 2008, pp. 452–461.

[12] J. Breese and R. Blake, "Automating computer bottleneck detection with belief nets," in *Proc. of UAI*. San Francisco, CA: Morgan Kaufmann, 1995, pp. 36–45.

[13] X. Gu and H. Wang, "Online Anomaly Prediction for Robust Cluster Systems," in *Proc. of ICDE*, Apr. 2009.

[14] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," in *Proc. of HotNets-I*, Princeton, New Jersey, October 2002.

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

[16] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox, "Capturing, indexing, clustering, and retrieving system history," in *Proc. of SOSP*, 2005, pp. 105–118.

[17] P. Langley, W. Iba, and K. Thompson, "An analysis of bayesian classifiers," in *Proc. of AAAI*, 1992, pp. 223–228.

[18] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[19] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Trans. Information Theory*, pp. 462–467, Nov. 1968.

[20] A. W. F. Edwards, "The measure of association in a 2x2 table," *Journal of the Royal Statistical Society. Series A (General)*, pp. 109–114, 1963.

[21] "Computer Failure Data Repository," http://cfdr.usenix.org/.

[22] Y. Zhao, Y. Tan, Z. Gong, X. Gu, and M. Wamboldt, "Self-Correlating Predictive Information Tracking for Large-Scale Production Systems," in *Proc. of ICAC*, Jun. 2009.

[23] "CoMon," http://comon.cs.princeton.edu/.

[24] "InfoScope Distributed Monitoring System," http://dance.csc.ncsu.edu/projects/infoscope/index.html.

[25] K.-L. Wu and et al., "Challenges and experience in prototyping a multimodal stream analytic and monitoring application on system S," in *Proc. of VLDB*, 2007, pp. 1185–1196.

[26] Y. Tan, X. Gu, and H. Wang, "Adaptive System Anomaly Prediction for Large-Scale Hosting Infrastructures," in *Proc. of PODC*, 2010.

[27] R. Vilalta, C. V. Apte, J. L. Hellerstein, S. Ma, and S. M. Weiss, "Predictive algorithms in the management of computer systems," *IBM Systems Journal*, 2002.

[28] R. Singer, K. G. andJ.P. Herzog, R. King, and S. Wegerich, "Model-based nuclear power plant monitoring and fault detection: Theoretical foundations," in *Proc. of ISAP*, jul 1997, pp. 60–65.

[29] S. Garg, A. V. Moorsel, K. Vaidyanathan, and K. Trivedi, "A methodology for detection and estimation of software aging," *Software Reliability Engineering, International Symposium on*, pp. 282–292, 1998.

[30] T.-T. Lin and D. Siewiorek, "Error log analysis: statistical modeling and heuristic trend analysis," *Reliability, IEEE Transactions on*, vol. 39, no. 4, pp. 419 –432, oct 1990.

[31] F. Salfner, M. Schieschke, and M. Malek, "Predicting failures of computer systems: a case study for a telecommunication system," in *Proc. of IPDPS*, 2006.

[32] B. Eckart, X. Chen, X. He, and S. L. Scott, "Failure prediction models for proactive fault tolerance within storage environments," in *Proc. of MASCOTS*, 2008, pp. 181–188.

[33] F. Salfner and M. Malek, "Using hidden semi-markov models for effective online failure prediction," in *Proc. of SRDS*, 2007, pp. 161–174.

[34] A. W. Williams, S. M. Pertet, and P. Narasimhan, "Tiresias: Black-box failure prediction in distributed systems," in *Proc. of IPDPS*, 2007.

[35] M. Y. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. A. Brewer, "Failure diagnosis using decision trees," in *Proc. of ICAC*, 2004.

[36] G. Hamerly and C. Elkan, "Bayesian approaches to failure prediction for disk drives," in *Proc. of 18th ICML*, 2001, pp. 202–209.

[37] R. K. Sahoo and et al., "Critical event prediction for proactive management in large-scale computer clusters," in *Proc. of SIGKDD*, 2003.

[38] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, and R. Sahoo, "Bluegene/l failure analysis and prediction models," in *Proc. of DSN*, 2006.

[39] W. Xu, L. Huang, A. Fox, D. A. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proc. of SOSP*, 2009, pp. 117–132.

[40] J. W. Mickens and B. D. Noble, "Exploiting availability prediction in distributed systems," in *Proc. of NSDI*, 2006.

[41] R. Powers, M. Goldszmidt, and I. Cohen, "Short term performance forecasting in enterprise systems," in *Proc. of KDD*, 2005, pp. 801–807.

[42] B. Schroeder and G. A. Gibson, "A large-scale study of failures in high-performance computing systems," in *Proc. of DSN*, 2006, pp. 249–258.

[43] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *Proc. of FAST '07*, Feb. 2007.

[44] B. Javadi, D. Kondo, J.-M. Vincent, and D. Anderson, "Mining for statistical models of availability in large-scale distributed systems: An empirical study of seti@home," in *Proc. of MASCOTS*, sept. 2009.