Identifying Recurrent and Unknown Performance Issues

Meng-Hui Lim[1], Jian-Guang Lou[2], Hongyu Zhang[2], Qiang Fu[2], Andrew Beng Jin Teoh[3], Qingwei Lin[2], Rui Ding[2] and Dongmei Zhang[2]

[1]Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China
[2]Microsoft Research Asia, Beijing, China
[3]School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea
menghuilim@comp.hkbu.edu.hk, {jlou,honzhang,qifu,qlin,juding,dongmeiz}@microsoft.com, bjteoh@yonsei.ac.kr

***Abstract*—For a large-scale software system, especially an online service system, when a performance issue occurs, it is desirable to check whether this issue has occurred before. If there are past similar issues, a known remedy could be applied. Otherwise, a new troubleshooting process may have to be initiated. The symptom of a performance issue can be characterized by a set of metrics. Due to the sophisticated nature of software systems, manual diagnosis of performance issues based on metric data is typically expensive and laborious. In this paper, we propose a Hidden Markov Random Field (HMRF) based approach to automatic identification of recurrent and unknown performance issues. We formulate the problem of issue identification as a HMRF-based clustering problem. Our approach incorporates the learning of metric discretization thresholds and the optimization of issue clustering. Based on the learned thresholds and cluster centroids, we can achieve accurate identification of recurrent issues and unknown issues. Experimental evaluations on an open benchmark and a large-scale industrial production system show that our approach is effective and outperforms the related state-of-the-art approaches.**

Keywords—Issue identification; performance; duplication detection; metrics; automated diagnosis.

I. INTRODUCTION

In recent years, online service systems with 24x7 availability are attracting growing number of users over time and are playing an increasingly important role in our society. The performance of these systems is particularly important for user satisfaction and project success, because users may turn to competing providers if the offered services are unavailable or running slowly. The performance degradations in online service systems can even result in large monetary losses. For instance, an hour-long PayPal outage may have prevented up to $7.2 million in customers transactions[1].

Like traditional systems, online service systems contain bugs too. Many of these bugs cause performance issues in actual operations of the online services. For example, Gill et al. [23] presented a large-scale analysis of network failures in data centers and found that many failures are caused by software problems.

The performance of an online service is often continuously monitored to check whether it violates a predefined Service Level Objective (SLO) or not. The SLO compliance records indicate that the system does not have performance issues during the corresponding time epochs, while the SLO violation records indicate that performance issues have occurred. At the same time, a large amount of telemetry data known as performance metric data (e.g. CPU utilization, Disk I/O, queue length, and memory usage) is also collected by the service monitoring infrastructure for troubleshooting purpose. Such monitoring is typically performed periodically (e.g. 5-minute intervals) and the metric data is recorded. Each record is associated with a SLO state.

When a performance issue occurs, engineers are required to diagnose the issue based on the collected performance metric data. Speedy issue diagnosis is required in order to minimize system downtime and reduce maintenance cost. In current practice, however, such troubleshooting process remains largely manual. In fact, many issues can be very laborious and time-consuming to diagnose, although the service system could often be restored quickly after being diagnosed [6].

In general, diagnosis can be accelerated by recognizing recurring or unknown performance issues. The ability to associate a performance issue with some past similar issues can help engineers rapidly apply a known remedy, thus improving the cost-effectiveness of performance diagnosis. Meanwhile, it is also important to detect unknown issues and prevent them from being misclassified as one of the recurrent issues. Otherwise, efforts could be wasted on applying inappropriate remedy [5][8].

Some approaches have been proposed to achieve rapid diagnosis through recognizing a similar performance issue occurred before [1][4][5][14][20]. These approaches mainly contain two steps: first selecting the issue-related metrics that reflect the symptom of a performance issue (such a process is also known as *metric attribution* [4][5]), and then utilizing the values of these metrics to recognize recurrent issues and detect unknown issues via classification or clustering techniques [1][3][5][7]. In addition, the selected metrics can also be used as indicators of performance issues and can help pinpoint the causes of the issues [5]. However, these two-step approaches are still ineffective and their accuracy can be further improved. For example, for a performance issue, these approaches only select metrics that are discriminative with respect to the SLO compliance data in the first step, but not necessarily discriminative against the other types of performance issues in the second step. Therefore, the selected metrics are often not sufficient to discriminate among different issue types and the accuracy of recurrent/unknown issue identification is adversely affected.

In this paper, we propose a unified approach to improve the effectiveness of recurrent/unknown issue identification and metric attribution simultaneously. In our approach, the metric data is treated as observations and

[1]https://www.datacenterknowledge.com/archives/2009/08/04/paypal-and-the-cost-of-downtime/

the corresponding issue types are treated as a field of hidden variables. We then employ a Hidden Markov Random Field (HMRF) technique that can model the relationships between a field of hidden variables and a set of observations. We build a HMRF-based model that can cluster issues according to the issue types. We also design an EM (Expectation-Maximization) based algorithm to obtain optimal model parameters. When a new record arrives, we determine if it indicates a recurrent issue or an unknown issue by evaluating its distance to the centroid of clusters. In addition, the centroid of clusters can help identify a set of relevant metrics that characterize a performance issue, thus achieving metric attribution through the same optimal model parameters.

We perform experimental evaluations on an open benchmark and a real industrial production system. The results demonstrate that our approach is effective in detecting recurrent and unknown issues – it achieves high Precision and Recall values on the two studied systems and outperforms the baseline and state-of-the-art approaches [1][5]. The experimental results also show that the proposed approach can achieve better metric attribution results than the related approaches. Overall, our contributions can be summarized as follows:

- We propose a HMRF-based clustering model for identifying recurrent and unknown performance issues. We also design an EM-like iterative algorithm to derive optimal parameters of the HMRF-based clustering model. The proposed approach outperforms the related state-of-arts approaches.
- We evaluate the proposed approach on an open benchmark, as well as on a large-scale industrial production system.

The rest of this paper is organized as follows. In Section II, we introduce the background and the related state-of-the-art approaches. We present the problem formulation in Section III and describe our proposed approach to recurrent issue identification in Section IV. In Section V, we present the experimental evaluations to justify the effectiveness of our approach. We discuss the threats to validities in Section VI and related work in Section VII. We conclude the paper in Section VIII.

II. BACKGROUND

A. Performance Issues

Performance is a key concern for large-scale software systems, especially for online service systems. Performance is often measured using some Key Performance Indicators (KPIs), which are defined to reflect the experience of end users. KPIs can be calculated through tracking user requests at the server side or measuring the end-to-end response time at client side. For each KPI, a SLO (Service Level Objective) threshold is defined to check whether the system is healthy or not. When a KPI of a system exceeds its predefined SLO threshold, the system is considered being in a SLO violation state and experiencing performance issue. Otherwise, the system is considered to be healthy (i.e., in the state of SLO compliance).

In order to diagnose performance issues of an online service system, a large amount of metric data is often collected during system executions. These metric data reflect low-level system events and system resource usage, such as CPU utilization, memory usage, disk queue lengths, I/O operation rate, kernel events, etc.

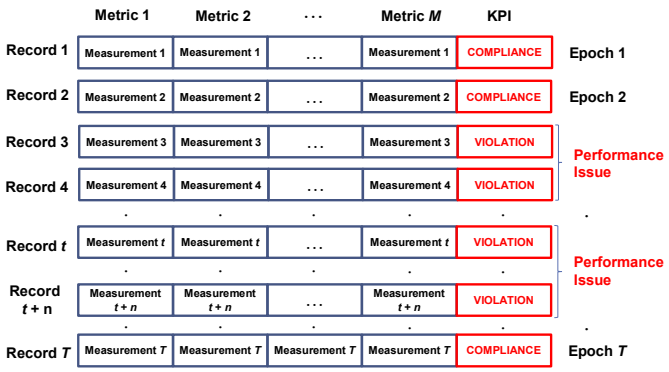


Fig. 1. The performance issue data.

As illustrated in Figure 1, for every time epoch (e.g., every 5 minutes), each metric has a measurement value collected by the monitoring system. Suppose that there are M metrics, we obtain a record $\boldsymbol{x_t} = \{x_{t,1}, x_{t,2}, \ldots, x_{t,M}\}$ at the t^{th} epoch. At the same time, every record also has an associated system-state value that indicates the health state of the system: SLO violation or SLO compliance during the epoch. The compliance/violation of a SLO is determined by the KPIs described above. A performance issue often lasts from several minutes to several hours, and the number of records contained in a single performance issue ranges from several to hundreds.

In today's large-scale 24x7 online service systems, performance issues such as processing bottlenecks can be very common. Over time engineers could observe a large number of performance issues, among them many could be recurrent problems. Once a performance issue is detected, engineers need to resolve the issue and recover the system as quickly as possible. In practice, many recurrent performance issues are still manually detected and fixed, causing long maintenance time. As we have mentioned in Section I, it is desirable to automatically recognize the previously occurred and unknown issues. Such ability can help reduce time wasted on repeating diagnosis efforts and help improve the troubleshooting process.

B. Performance Issue Identification

In recent years, some automated approaches have been proposed to categorize performance issues according to issue types and to identify recurrent issues [1][5][7][13][20]. A general process of these approaches is shown in Figure 2. These approaches are all based on the mining of historical performance records and the associated SLO violation/compliance labels. These approaches find that issue identification is ineffective by simply clustering a large volume of raw metric data. Therefore, these approaches first identify a set of metrics that are relevant to each performance issue. Such a process is termed "metric attribution" by Cohen et al. [4][5]. The values of the attributed metrics are also represented as a vector of *signatures*, which can capture the essentials of the system states. Having constructed a database of signatures, these approaches then utilize clustering/ classification techniques to group SLO violations based on their signatures. For new coming

performance records, the approaches can identify if they are SLO compliance records, or belong to recurrent/unknown performance issues.

Figure 2 shows the commonalities among the performance issue identification approaches. These approaches mainly differ in the process of metric attribution, the construction of signatures, and the clustering/retrieval of signatures. Due to space constraint, we just briefly describe a few state-of-arts approaches here.

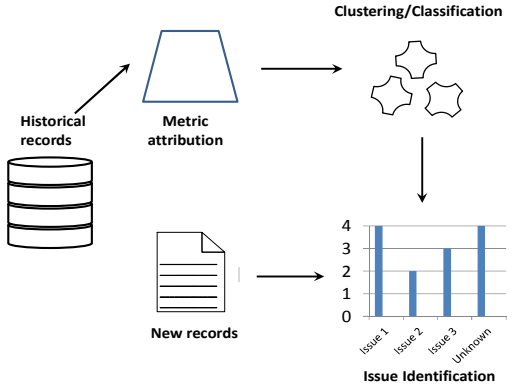


Fig. 2. A general process of performance issue identification approaches.

Signature approach: Cohen et al. [5] proposed to use a *signature* to characterize the performance issues. The signature representation of every record is used for clustering similar issues and identifying recurrent issues. Cohen et al.'s approach first applies a Tree-Augmented Naïve Bayes (TAN) model to select metrics that reflect SLO violation or compliance. A signature vector is then constructed based on the selected metrics. They apply a *k*-means clustering to a database of signatures to find groups of signatures that characterize different performance issues. For a specific signature *S*, it can retrieve from a database all previous instances that are similar to *S*.

Fingerprint approach: Bodik et al. [1] proposed to use a *fingerprint*, a vector representing the performance state of the system, to uniquely identify a performance issue. To generate a fingerprint, they compute quantiles of each performance metric over each epoch. They produce a summary vector that contains three quantile elements per metric for each epoch. The values of these quantile elements are identified either to be hot (abnormally high values), cold (abnormally low values), or normal (normal values) based on some predefined thresholds. The summary vector is then converted into an *epoch fingerprint* by selecting only discriminative metrics that can distinguish normal performance from the SLA violations. For two performance issues, they compare their fingerprints and determine whether they correspond to the same problem.

Although the Signature and Fingerprint approaches have been evaluated on large-scale production systems and are shown to be superior to simpler baseline methods and various alternatives, they still have limitations in associating recurrent issues and detecting unknown issues. These approaches perform a metric selection process for each issue by maximizing the discrimination between SLO compliance and violation records. The output of the metric selection process is concatenated to produce the final set of metrics. They then use the measurement data of the final set of metrics to identify recurrent and unknown issues via clustering. However, for each issue, the measurements of the selected metrics are only discriminative with respect to the SLO compliance data, but not necessarily discriminative against the other types of performance issues. In addition, the limited quantity of selected metrics is often not adequate in differentiating different types of issues. Therefore, it is likely that violation instances from different kinds of performance issues will be erroneously clustered into a single cluster and not differentiable.

Furthermore, in related approaches, the metric discretization thresholds are pre-specified and may not reflect the actual abnormal data ranges in practice. Since the thresholds directly influence the metric selection for a performance issue, the resultant metric attribution may not be accurate. In this paper, we propose a HMRF-based approach, which can address the drawbacks and outperform the related state-of-arts approaches.

III. Problem Formulation

Suppose that there exist v instances of performance issue and each of which belongs to one of the k issue types ($k \leq v$). The v issue instances and the non-issue instances (i.e., compliance instances) can be grouped into $k+1$ clusters (k issue types plus 1 non-issue type). Each cluster consists of performance issues of the same type. An example of issue clustering is shown in Figure 3. There are three issue instances, which consist of records of two issue types (i.e., k_1, k_2). The SLO compliance records can be treated as a special non-issue type. The problem is to automatically group the T records into 3 (2+1) clusters, and assign a new record to a correct cluster. If the new record can be well categorized into a cluster, a reoccurring issue is detected. Otherwise, the issue is an unknown issue. Note that we do not assume prior knowledge about the number/type of the issues.

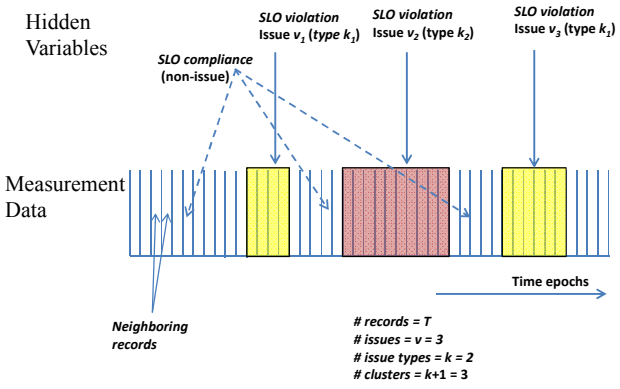


Fig. 3. An example of issue clusters.

In practice, it is customary to expect a strong time-based correlation among the records within each issue [1][7], i.e., two neighboring records often reflect similar performance state. Therefore, we consider the neighboring effect in our work. More specifically, we define a *temporal-neighboring constraint,* which means that every pair of neighboring records tend to indicate the same issue, and are independent of past, non-neighboring records.

The above issue clustering problem can be formulated as a Hidden Markov Random Field (HMRF) problem. A hidden Markov random field is a generalization of a hidden Markov model, which is particularly useful for inferring a hidden field of random variables $L = (l_1, l_2, \ldots, l_i, \ldots, l_n)$ from the corresponding set of measurements $x = (x_1, x_2, \ldots, x_i, \ldots, x_n)$. Each hidden variable l_i is related only to its neighbors within an issue. The hidden variables are mutually related via a neighborhood system. HMRF models have been successfully used in many areas such as image segmentation [21] and bioinformatics [12], because they support probabilistic modeling of information about the dependencies between hidden variables and the associated observations.

In our problem, we try to infer the performance issues through analyzing the observed metric data. This problem is similar to the problem of HMRF. The performance issues can be modeled as hidden variables $L = \{l_t, t \in \mathcal{T}\}$, where $\mathcal{T} = \{1,2, \ldots, T\}$ denotes the time epoch indices. The hidden variable l_t represents the issue associated with a record at the t -th epoch. The neighboring records indicating the same performance issue form a neighborhood, in which every pair of records is interconnected via the *temporal-neighboring constraint.* Our objective is to train a HMRF-based model to categorize records in all the neighborhoods into several issue types (including the non-issue type). Based on the model, we can identify recurrent and unknown issues, and automatically identify relevant metrics for each issue type.

IV. The Proposed Approach to Identifying Recurrent and Unknown Issues

In this section, we describe the proposed HMRF-based clustering technique for identifying recurrent and unknown issues. The overall structure of the proposed approach is shown in Figure 4. Given the historical record data of performance issues, we first discretize the metric data according to a set of initial thresholds. We then build a HMRF-based model that can cluster issues according to the issue types, and derive optimal thresholds and clustering parameters using an EM-based algorithm. The SLO compliance records are treated as the non-issue type. Finally, when a new record arrives, we determine if it indicates a recurrent issue or an unknown issue by evaluating its distance to the centroids of clusters.

A. HMRF-based Clustering

1) Metric Discretization

To identify metrics that are related to a performance issue, each metric x_t is commonly discretized into a binary value $\bar{x}_t$ according to a metric threshold τ. If x_t has a value larger than τ, then $\bar{x}_t = 1$. This indicates that x_t is more likely to be associated with SLO violation and vice versa. Formally:

$$\bar{x}_t = \begin{cases} 0 & \text{if}(x_t - \tau) \leq 0 \\ 1 & \text{if}(x_t - \tau) > 0 \end{cases} \quad (1)$$

To attribute metrics effectively, it is important to identify optimal threshold for each metric. In practice, it is difficult to determine the optimal thresholds because the optimal values of all metrics are not the same. If the optimal thresholds can be identified, the records can be discretized correctly and higher clustering accuracy can be achieved. Therefore, instead of predefining metric thresholds (as what Fingerprint and Signature approaches do), we optimize the thresholds using a learning algorithm, which will be described in Section IV.A(3).

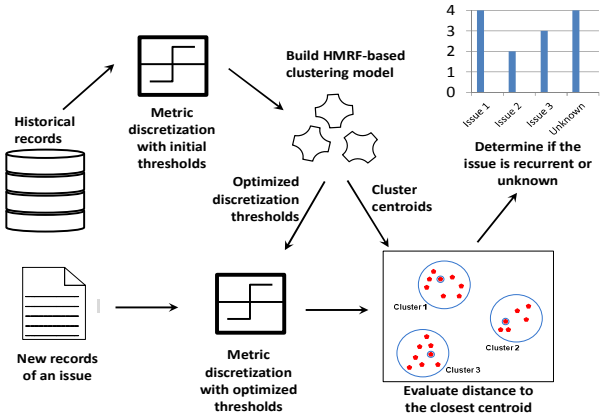


Fig. 4. Overview of the proposed approach.

2) Model Building

In our approach, we use historical data to train a HMRF-based clustering model. Let $\boldsymbol{\ell}$ be a configuration of hidden field L (a particular configuration of cluster labels) and $\boldsymbol{\chi}$ be an observable instance of X (metric vectors), the posterior probability of $\boldsymbol{\ell}$ can be obtained via Bayes's formula as follows:

$$P(\boldsymbol{\ell}|\boldsymbol{\chi}) = \frac{P(\boldsymbol{\chi}|\boldsymbol{\ell})\, P(\boldsymbol{\ell})}{P(\boldsymbol{\chi})} \propto P(\boldsymbol{\chi}|\boldsymbol{\ell})\, P(\boldsymbol{\ell}) \quad (2)$$

where $P(\boldsymbol{\chi})$ is a constant, $P(\boldsymbol{\ell})$ is the prior probability of $\boldsymbol{\ell}$, and $P(\boldsymbol{\chi}|\boldsymbol{\ell})$) is the likelihood of the measurement values given a configuration of issue types. In this section, we describe how we derive $P(\boldsymbol{\ell})$ and $P(\boldsymbol{\chi}|\boldsymbol{\ell})$ based on the nature of HMRF.

Conditional Probability $P(\boldsymbol{\chi}|\boldsymbol{\ell})$

Our approach automatically groups historical records into clusters. It can be assumed that the larger the distance between a data point (a discretized metric vector) $\bar{\boldsymbol{x}}_t$ and the centroid $\bar{\boldsymbol{\mu}}_{l_t}$ of a cluster is, the higher the probability that the data point does not belong to that cluster. Without loss of generality, we use a general form of exponential function to model such monotonic relationship. Hence the conditional probability of $\boldsymbol{\chi}$ given $\boldsymbol{\ell}$ can be expressed as:

$$P(\boldsymbol{\chi}|\boldsymbol{\ell}) = \frac{1}{Z_1} \prod_{t \in \mathcal{T}} e^{-D(\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{\mu}}_{l_t})} \quad (3)$$

where D is a distance function and Z_1 is a normalization constant. In our approach, we define the distance function D as the Hamming distance (because we are dealing with binary data), which measures the minimum number of substitutions required to change from $\bar{\boldsymbol{x}}_t$ to $\bar{\boldsymbol{\mu}}_{l_t}$.

Prior Probability $P(\boldsymbol{\ell})$

In HMRF, the prior probability of $\boldsymbol{\ell}$ can be expressed as a Gibbs distribution [9][10]:

$$P(\boldsymbol{\ell}) = \frac{1}{Z_1} e^{-V(\boldsymbol{\ell})} = \frac{1}{Z_2} e^{-\sum_{\mathcal{N}_p \in \mathcal{N}} V_{\mathcal{N}_p}(\boldsymbol{\ell})} \quad (4)$$

where Z_2 is a normalizing constant, $V(\boldsymbol{\ell})$ denotes the overall potential function, which can be expressed as a sum of potentials $V_{\mathcal{N}_p}(\boldsymbol{\ell})$ over all the neighborhoods $\mathcal{N}$ for the label configuration $\boldsymbol{\ell}$. According to the *temporal-neighboring* constraint, every pair of records within each neighborhood $\mathcal{N}_p$ tends to be grouped into the same issue cluster. With this, $V_{\mathcal{N}_p}(\boldsymbol{\ell})$ can be defined as:

$$V_{\mathcal{N}_p}(\boldsymbol{\ell}) = w_p \cdot \sum_{\forall l_t, l_{t'} \in \mathcal{N}_p, t \neq t'} \mathbb{I}[l_t \neq l_{t'}] \quad (5)$$

where the variable $\mathbb{I}$ denotes an indicator function ($\mathbb{I}[\text{true}] = 1$, $\mathbb{I}[\text{false}] = 0$) and w_p denotes the normalizing weight for the total violations of *temporal-neighboring* constraint in the neighborhood $\mathcal{N}_p$. Equation (5) acts as a penalty function that gives more weights to neighboring records that are not grouped within the same cluster (i.e., not satisfying the temporal-neighboring constraint). It punishes poor clustering configuration and shapes towards an ideal cluster formation where instances from a neighborhood are perfectly enclosed within one of the clusters.

HMRF-kMedoid-EM Algorithm

Input: T records, M metrics
Output: k+1 discretized medoids, M discretized thresholds

1. Initialization: initialize discretization thresholds & discretized medoids.
2. **EM-Step(A)**: Given the discretized measurements, search for the best possible medoids.
3. **EM-Step(B)**: Given the medoids, identify the best discretization threshold of the next metric.
4. Re-iterate EM-Step(A) and EM-Step(B) until maximum iteration number is reached.
5. Output the overall configuration that achieves the lowest J_{obj} as the optimal configuration.

Fig. 5. The algorithm for optimizing thresholds and cluster parameters.

3) Parameter Estimation

To obtain the optimal parameters in a HRMF model from a set of measurement data, we apply the MAP (maximum a posteriori probability) estimation approach and design an Expectation-Maximum (EM) based algorithm.

The Objective Function

We first define an objective function based on the posterior probability of a HMRF model, which is the negative logarithm of $P(\boldsymbol{\ell}|\boldsymbol{\chi})$. By minimizing this function value, we can optimize the M discretization thresholds $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^{M}$ and the $k+1$ cluster centroids $\{\bar{\boldsymbol{\mu}}_l\}_{l=0}^{k}$. In accordance with (2), (3) and (4), the objective function can be expressed as follows:

$$\begin{aligned} J_{obj} &= -\ln\left(P(\boldsymbol{\chi}|\boldsymbol{\ell})\, P(\boldsymbol{\ell})\right) \\ &= -\ln\Big(\frac{1}{Z_1}\prod_{t\in\mathcal{T}} e^{-D\left(\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{\mu}}_{l_t}\right)}\Big) - \ln\left(\frac{1}{Z_2} e^{-\sum_{\mathcal{N}_p\in\mathcal{N}} V_{\mathcal{N}_p}(\boldsymbol{\ell})}\right) \\ &= -\ln\left(\frac{1}{Z}\right) + \sum_{t\in\mathcal{T}} D\left(\bar{\boldsymbol{x}}_t, \bar{\boldsymbol{\mu}}_{l_t}\right) + \sum_{\mathcal{N}_p\in\mathcal{N}} V_{\mathcal{N}_p}(\boldsymbol{\ell}). \end{aligned} \quad (6)$$

Note that $\frac{1}{Z}$ is a constant. The second factor (D component) is concerned with the compactness of the clusters; while the third factor (V component) is concerned with violations of the temporal-neighboring constraints. The minimization of D component encourages binary representation of the discretized metric instances $\bar{\boldsymbol{x}}_t$ to be driven towards the corresponding binary cluster centroids $\bar{\boldsymbol{\mu}}_{l_t}$ via optimizing the discretization thresholds. The minimization of V component ensures that the cluster assignment of the records is optimized via satisfying most of the temporal-neighboring constraints.

Threshold and Clustering Optimization

The minimization of J_{obj} in Equation (6) can be solved using an iterative EM algorithm. As we are dealing with binary data (the discretized records), our algorithm adopts a variant of centroid, called *medoid* [17] as a representative of a cluster. Our algorithm (named *HMRF-kMedoid-EM*) shown in Figure 5 begins with an initialization of metric thresholds and medoids. Given the records being discretized in accordance with the initial threshold configuration, the first stage **EM-Step(A)** searches for a decent clustering configuration that reduces the objective value with a *k*-medoid-like process. In the second stage, **EM-Step(B)** seeks for a good threshold for a particular metric based on the resultant clustering configuration. These two stages of the algorithm iterate until a pre-specified maximum number of iterations is reached. Ultimately, the records will be categorized into $k+1$ clusters (corresponding to 1 compliance type and k issue types). At the same time, the optimal threshold for each metric can also be obtained during the iterative optimization. In this way, our approach finds optimal metric discretization thresholds along with cluster parameters.

More details about the HMRF-kMedoid-EM algorithm and discussions about its convergence, running time complexity and scalability can be found at the webpage: http://www.dropbox.com/s/ymbzoohnyfwawgl/PerformanceIssue.pdf

B. Identifying Recurrent and Unknown Issues

In our approach, once the optimized thresholds and cluster medoids are obtained from the HMRF-kMedoid-EM algorithm, a Hamming similarity threshold τ_{sim} is then specified with respect to each medoid. Clusters are formed based on the values of τ_{sim}. A larger τ_{sim} value leads to a smaller number of clusters, while a smaller τ_{sim} value leads to a larger number of clusters. The specific type of issue that a cluster represents is determined by identifying the issue type that majority records belong to. For instance, a cluster enclosing 60% records of issue A, 20% records of issue B, and 20% records of issue C, is deemed to represent issue A.

Given a newly arrived performance record, it is desirable if past similar issues can be identified accurately so that past diagnoses can be leveraged and appropriate repairs can be identified. Furthermore, a good diagnostic approach should be capable of detecting unknown violation issues when the new issue and all previous issues are sufficiently dissimilar.

The identification of recurrent issues and unknown issues can be performed as follows: First, the new records of a performance issue are discretized using the optimized threshold configuration. The records are then associated with the nearest medoid and annotated with the

corresponding cluster index. Second, a new record is deemed belong to the issue type represented by the medoid if majority of the discretized records are within τ_{sim} distance of the medoid. Otherwise, the new record is tagged as unknown type. The identification of recurrent and unknown issues is illustrated in Figure 6.

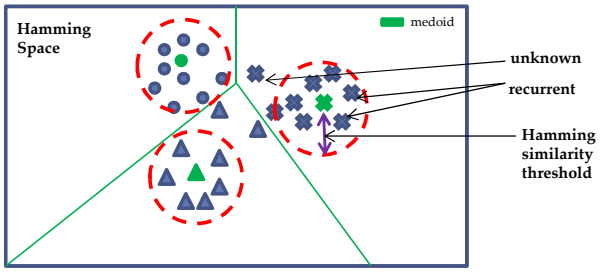


Fig. 6. The identification of recurrent and unknown issues.

One advantage of our approach is that it can improve the effectiveness of recurrent/unknown issue identification and metric attribution simultaneously. Having identified the optimal threshold and medoid values, we can identify the metrics that are relevant to a performance issue. Intuitively, the binary '1's in each medoid indicate that the metrics are relevant to the corresponding issue (i.e., the "attributed metrics"). These metrics are relevant metrics that can distinguish normal performance from the performance violations. Typically there are a large number of metrics involved in performance data, the attributed metrics can facilitate the diagnosis of a performance problem [1][4][5].

V. The Proposed Approach to Identifying Recurrent and Unknown Issues

A. Systems and Datasets

To evaluate the effectiveness of our approach, we perform several experiments on the following systems:

TPC-W: We select the Transaction Processing Performance Council BenchmarkTM W (TPC-W) [18] for synthetic generation of performance issues. TPC-W is a transactional eBusiness benchmark, simulating the activities of a business oriented transactional web server. In our experiment, TPC-W is implemented in Java and deployed on a Windows Server 2003. A client running on another machine is used to simulate synthetic-requests sent from multiple end-users to the TPC-W server according to a pre-configured workload schema. In order to simulate performance issues, a standalone program is employed to purposely exhaust specific system resources such as CPU and disk I/O. The three types of performance issues are (A1) extreme CPU utilization, (A2) excessive disk I/O consumption and (A3) mixed CPU and disk I/O exhaustion. We generate 2000 records consisting of 41 performance issues that last for 3 to 24 epochs. The issue types A1, A2, and A3 have 250, 208, and 220 associated records, respectively. Each record contains 88 performance metrics.

System X: We also collect data from a real production system called System X (the name of the system is not provided due to confidentiality reasons), which is an online service system serving millions of users. Similar to other online services, System X is expected to provide high-quality service on 7x24 basis. In this study, 20 performance issues that have been manually diagnosed by industrial engineers within a three-week period are adopted. These performance issues can be categorized into four types: (B1) Heavy workload, (B2) Latch contention, (B3) Heavy disk writing, and (B4) Heavy CPU usage. Each performance issue lasts for 5 to 20 epochs. We collect 1000 records for these issues together with their neighboring compliance records. Each record contains 689 system-level metrics, including performance counters, Windows NT® kernel events, and application-specific measurements. The issue types B1, B2, B3, and B4 have 103, 22, 26 and 22 associated records, respectively.

The issue types are summarized in Table I.

TABLE I. The Performance Issue Types

System	Issue Type	Description	Records
TPC-W	A1: Extreme CPU utilization	Excessive processing resources /computing tasks handled by CPU.	250
	A2: Excessive disk I/O consumption	Excessive disk input/output activities.	208
	A3: Mixed CPU and disk I/O exhaustion	Excessive CPU usage and disk input/output activities.	220
System X	B1: Heavy workload	Heavy job-processing load.	103
	B2: Latch contention	Heavy SQL server usage as a result of repetitive requests for latch by a server program.	22
	B3: Heavy disk writing	Heavy data writing on a disk in SQL server.	26
	B4: Heavy CPU usage	Heavy usage of CPU processing resources.	22

B. Research Questions

We have identified the following two research questions:

RQ1: Can the proposed approach identify recurrent issues?

This RQ examines the effectiveness of our approach in associating new records with past similar issues. To answer RQ1, four-fold cross-validation was carried out in the experiments. The data in three folds is used for model training and the rest fold is used for testing. The final testing results are averaged. In this part of the experiment, instances from all types of issues are considered.

RQ2: Can the proposed approach detect unknown issues while recognizing recurrent issues?

This RQ evaluates the ability of our approach in detecting unknown performance violation issues while recognizing past similar issues at the same time. In this part of the experiment, each issue type is treated as an unknown issue at a time and the associated records are removed from the training set and are added to the testing set. We then evaluate the effectiveness of the proposed approach in detecting both unknown and recurrent issues. This experiment is repeated for each issue type in the dataset and the average evaluation result is taken.

C. Experimental Settings

We use the Precision-Recall curve [22] to measure the effectiveness of the proposed approach. For each issue type k (i.e., each cluster) Recall measures the fraction of the relevant records whose issue types are successfully identified. Precision measures the fraction of identified records that are indeed relevant. The Precision and Recall values for each issue type k are defined as follows:

$$Precision_k = \frac{TP}{TP + FP} \quad (7)$$

$$Recall_k = \frac{TP}{TP + FN} \quad (8)$$

where TP (True Positive) is the number of correctly identified records (i.e., the testing records that are assigned to the correct issue type). FP (False Positive) is the number of identified records that are irrelevant (i.e., the irrelevant testing records that are assigned to the target issue type). FN (False Negative) is the number of relevant records that are failed to be identified (i.e., the relevant testing records that are not assigned to the target issue type).

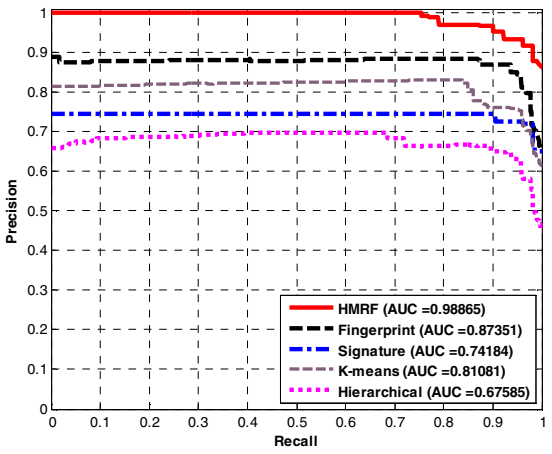
For all issue types (i.e., all clusters), the overall Precision and Recall values are measured as the average of the $Precision_k$ and $Recall_k$ values for each issue type k. The overall Precision and Recall values are between 0 and 1. The higher the better. The maximum Precision and Recall values are achieved when all identified testing records are relevant and all relevant records are identified, respectively.

A Precision-Recall curve is generated based on the overall Precision and Recall values, which are resulted from the variation of the similarity threshold τ_{sim} (the Hamming similarity threshold used in clustering, as described in Section IV.B) over a range of possible values. Given the same Recall value, the larger the Precision value, the better the performance. Similarly, given the same Precision value, the larger the Recall value, the better the performance. We also use the AUC metric (the area under the precision-recall curve) for model comparison.

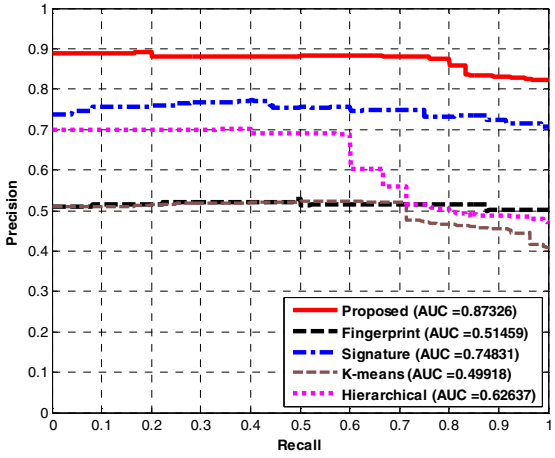
To identify unknown issues while recognizing recurrent issues, $Precision_k$ and $Recall_k$ values are computed for each issue type k (including the unknown type) based on the following definitions of TP (True Positive). When a recurrent issue type is under evaluation, testing records whose distances is smaller than τ_{sim} from the cluster medoid of the same issue type is considered as a TP. When the unknown issue type is under evaluation, testing record whose distance is larger than τ_{sim} from the closest cluster medoid of any issue type is considered as a TP. Similarly, FP and FN can be defined. In this way, the $Precision_k$ and $Recall_k$ for each issue type k (including the unknown type) are computed. The overall Precision and Recall values are the average of $Precision_k$ and $Recall_k$ values for all issue types (including the unknown type).

We compare the proposed approach with two baseline clustering techniques: k-means and hierarchical clustering. We apply these techniques directly on the raw metric data. k-means clustering derives k clusters (each representing an issue type) from the training records and assigns each testing record to the cluster with the nearest mean. Hierarchical clustering constructs a hierarchy of clusters. We apply the agglomerative hierarchical clustering technique, which is a bottom-up clustering approach. At the beginning each record belongs to its own cluster. Then, the closest pair of clusters is selected and merged. Using k-means and hierarchical clustering, we can group performance records by their issue type. Our proposed approach should outperform the simple baseline approaches.

We also compare the effectiveness of the proposed approach with that of Signature and Fingerprint approaches. Note that for Signature approach, although its ability to detect unknown violation issues is not incorporated in the original paper [5], it can be added by simply applying a similarity threshold with reference to the cluster representatives. For the Fingerprint approach, following the settings used in the original paper [1], the number of relevant metrics for the performance issues is fixed to 30 and 100 for TPC-W and System X respectively. The hot and cold thresholds for the selected metrics are computed as 2nd and 98th percentile of the observed (compliance) metric values. The fingerprint and the signature representations of each record are then used for the identification of recurrent/unknown issues.



a) TPC-W



(b) System X

Fig. 7. Precision-Recall curves for recognizing recurrent performance issues.

D. Experimental Results

RQ1: Can the proposed approach identify recurrent issues?

Figure 7 shows the performance of the proposed approach in detecting recurrent issues of the two studied systems, measured using Precision-Recall curves. For the TPC-W system, the Precision value is around 97% when Recall value is 80%, and the AUC value is 98.87%. For System X, the Precision value is around 87% when Recall value is 80%, and the AUC value is 87.33%. These results are considered satisfactory.

Figure 7 also shows that the proposed approach significantly outperforms the related approaches. In terms of AUC, for the TPC-W system, the proposed approach outperforms the Fingerprint, the Signature, the *k*-means, and the Hierarchical approaches by 11.51%, 24.68%, 17.78%, and 31.28% respectively. For the System X, the proposed approach outperforms the Fingerprint, the Signature, the *k*-means, and the Hierarchical approaches by 35.87%, 12.49%, 37.41%, and 24.69% respectively. The overall better performance of the proposed approach indicates that the proposed clustering technique and the parameter optimization algorithm are beneficial.

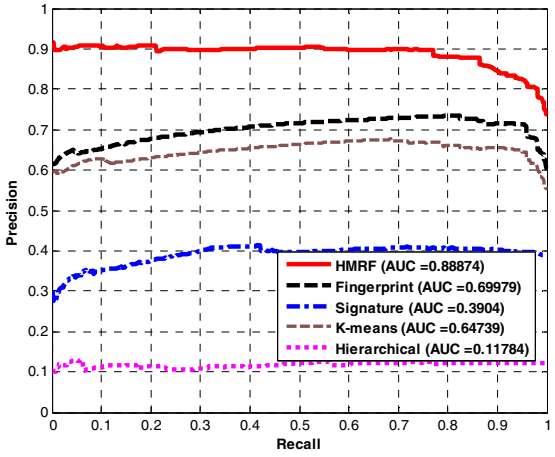
Table II shows the Precision values of the proposed approach in identifying recurrent issues of each issue type when the Recall values are 0.5 and 0.8. For example, when Recall is 0.8, for identifying recurrent A1 issues of TPC-W system, the proposed approach achieves a Precision value of 0.8880. When Recall is 0.8, for identifying recurrent B1 issues of System X, the proposed approach achieves a Precision value of 0.9570. Similar results are obtained for other issue types, except for B2 issues. We found that in two of the folds, the proposed approach could not identify B2 issues as effective as others, causing part of the B1 issues being misclassified as B2 issues. Despite such slight deficiency, the overall results show that the proposed approach can still effectively identify recurrent issues for most issue types.

TABLE II. THE PRECISION VALUES OF RECURRENT ISSUE IDENTIFICATION FOR EACH ISSUE TYPE

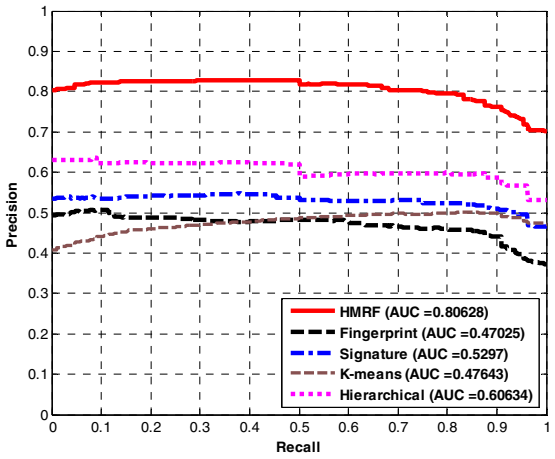
System	Issue Type	Recall = 0.5	Recall = 0.8
TPC-W	A1: Extreme CPU utilization	1	0.8880
	A2: Excessive disk I/O consumption	1	0.9946
	A3: Mixed CPU and disk I/O exhaustion	1	1
	SLO Compliance	1	1
System X	B1: Heavy workload	1	0.9570
	B2: Latch contention	0.4071	0.4174
	B3: Heavy disk writing	1	1
	B4: Heavy CPU usage	1	1
	SLO Compliance	0.9986	0.9986

RQ2: Can the proposed approach detect unknown issues while recognizing recurrent issues?

Figure 8 depicts the evaluation results of the proposed approach in detecting unknown performance issues while recognizing recurrent issues, measured using Precision-Recall curves. For the TPC-W system, the Precision value is around 90% when the Recall value is 70%, and the AUC value is 88.87%. For System X, the Precision value is around 80% when the Recall value is 80%, and the AUC value is 80.63%. When compared to the related approaches, it is evident that the proposed approach outperforms the others. In terms of AUC, for the TPC-W system, the proposed approach outperforms the Fingerprint, the Signature, the *k*-means, and the Hierarchical approaches by 18.90%, 49.83%, 24.14%, and 77.09%, respectively. For System X, the outperforming margins are 33.61%, 27.66%, 32.99%, and 20.00%, respectively.



a) TPC-W



(b) System X

Fig. 8. Precision-Recall curves for recognizing recurrent performance issues and identifying unknown performance issues.

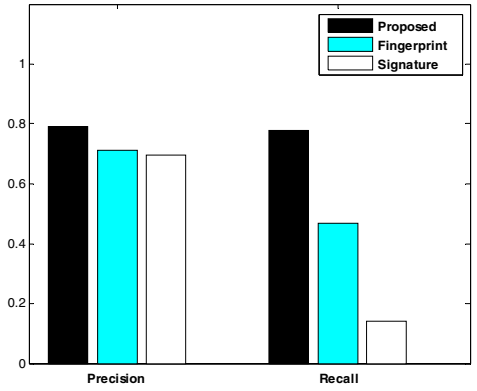
These results indicate that the records of unknown issues can be better separated from the known-issue records by the proposed approach. The results also imply that the optimized thresholds and clustering configurations do not overfit the training data, and they have predictive power in detecting unknown issues during testing. These are what make the proposed approach a better approach in detecting unknown issues than the baseline (*k*-means and hierarchical clustering) and state-of-arts (Fingerprint and Signature) approaches.

E. Discussion on Metric Attribution

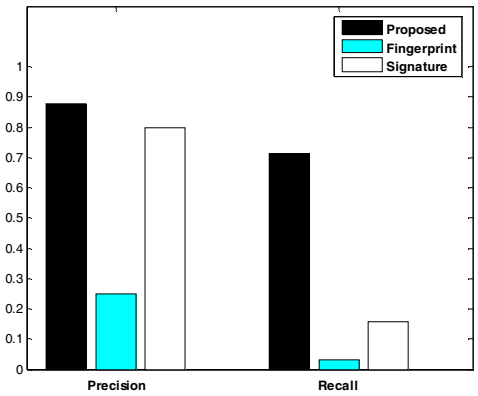
Diagnosis of performance issues typically involves selecting a subset of "attributed metrics" (i.e., relevant metrics) that reflect the symptom of a performance issue. As described in Section IV.B, our approach can achieve

recurrent/unknown issue identification and metric attribution simultaneously. After clustering, the binary '1's in each cluster's medoid indicate that the metrics are relevant to the corresponding issue.

We explore the effectiveness of our approach in metric attribution using the same subject systems. The results are shown in Figure 9. The metrics attributed for each type of performance issue are extracted from the medoids of our approach and from the cluster representatives of Fingerprint and Signature approaches. The ground truths for the metrics attributed for each issue are manually annotated.



(a) TPC-W



(b) System X

Fig. 9. Accuracy results of metric attribution.

The metric attribution results on TPC-W are shown in Figure 9(a). It is observed that the Signature approach achieves the lowest Precision and Recall values. The results on System X are shown in Figure 9(b). The Fingerprint approach could not identify most of the relevant metrics of the evaluated issues. The Signature approach performs slightly better. For both systems, the proposed approach achieves better overall results. This is because the Signature and the Fingerprint approaches discretize records according to a predefined threshold and perform metric selection based on the discretized data. Due to the incorrectly attributed metrics, these approaches only extracted a small set of metrics and some of these metrics are not directly related to the underlying performance issue, thus resulting in low Recall values. The proposed approach uses learned thresholds in discretization and therefore achieves a much higher Precision and Recall values than the Fingerprint and Signature approaches.

In summary, the results on both TPC-W and System X demonstrate the effectiveness of our approach in metric attribution.

VI. THREATS TO VALIDITY

We have identified the following threats to validities:

- Limited subject systems: in our experiments we only use data collected from two subject systems. Although these two systems include one large-scale industrial system and one open benchmark, the generality of the proposed approach should be further evaluated. In the future, we will explore whether our approach can be applied to a variety of systems.
- Data accuracy: the performance issues used in our evaluation are manually labeled. For the TPC-W system, the data is labeled by us. For System X, the data is labeled by engineers in product teams. We assume that these data is of high quality. However, it is possible that a small amount of data might be incorrect due to occasional human errors.
- Large amount of data: the approach we proposed requires a large amount of metric data collected by the monitoring infrastructure over a long period. For a small or short-living system, the amount of metric data is often small, thus making the statistical analysis inappropriate.

VII. RELATED WORK

Research on large-scale industrial software systems shows that many problems observed in the field are performance-related problems [19]. Various approaches have also been proposed to automate the identification of software performance issues by mining performance metric data. These approaches can be classified into two broad categories: (1) issue detection and (2) issue classification. The key difference between these two categories is that the former is a binary identification problem (discrimination of issues from non-issues); while the latter is a multi-class identification problem (identification of issue types).

The issue detection approaches include Markov monitoring [15], Aniketos [16], logistic-regression-based threshold setting [1], and abnormality-enhanced classification [3]. Smyth [15] proposed a Markov monitoring approach to allow detection of anomalous states that were unknown a priori. Aniketos [16] was introduced to construct a tight convex hull to group compliance instances and to consider exterior instances as issues. Breitgand et al. [2] modeled the SLO violations using logistic regression and used the model to detect SLO violations by adapting metric thresholds to the changing workload. To improve detection accuracy, Bronevetsky et al. [3] analyzed the normal and faulty runs, and proposed a novel technique that combines classification algorithms with information on the abnormality of application behavior. Malik et al. [13] compared the ability of a PCA-based approach in uncovering the performance deviation to that of a random sampling approach, a clustering approach, and a

supervised approach (WRAPPER). All four approaches use performance metric data obtained from a load test. Their study shows that the supervised approach is more effective than the unsupervised approaches but incurs higher overhead.

Issue classification approaches are generally more useful for debugging performance issues than the issue detection approaches. These techniques analyze performance metric data, and apply machine learning techniques to classify issues. For example, Falcon [7] utilizes both labeled and unlabeled data to improve the overall accuracy of diagnosis. It is an active-learning technique, which facilitates manual labeling effort via maximizing the benefits gained from newly-diagnosed unknown instances. Natu et al. [14] focused on automated debugging of SLO violations by initiating a two-stage process of feature selection using a Classification and Regression Trees (CART) method followed by a statistical change-point detection algorithm. Cohen et al. [4] employed a Tree Augmented Bayesian Network (TAN) model to automatically identify a group of metrics that correlate with SLO states. Zhang et al. [20] extended Cohen et al.'s work [4] to changing workloads and external disturbances adaptation via maintaining an ensemble of TAN models, such that a new model is added whenever existing ones do not accurately capture the current system behaviors. Cohen et al. [5] extended their previous approach [4] by proposing a Signature approach. Bodik et al. [1] proposed the Fingerprint approach, which outperforms not only simpler baseline methods, but also the Signature approach proposed in [5]. By far the work closest to ours is the Signature approach and the Fingerprint approach. The experimental comparison of our approach to the Signature [5] and Fingerprint [1] approaches is presented in Section V. The results show that the proposed approach outperforms the related state-of-arts approaches.

VIII. CONCLUSIONS

This paper addresses the problem of automated identification of recurrent and unknown performance issues, which is an important step of performance diagnostics for large-scale software-intensive systems, especially online service systems. Our approach formulates the performance issue identification problem as a HMRF-based clustering problem. We perform experimental evaluations on an open benchmark (TPC-W system) and a real industrial production system (System X). The results show that our approach leads to more effective recurrent issues identification, unknown issue detection, and metric attribution results than the state-of-arts approaches. The results also justify the efficacy of our approach as a useful performance diagnostic tool for real-world online services. In future, we will evaluate the proposed approach on a variety of systems, including systems of other organizations. Having identified performance issue types, we will also explore techniques for effective localization of the problematic components.

Our experimental data and prototype tool are available at:

http://www.dropbox.com/s/pj1miqu00ryoj9a/HMRF-kMedoid-EM.zip

REFERENCES

[1] P. Bodik, M. Goldszmidt, A. Fox, D. Woodard, and H. Andersen, Fingerprinting the datacenter: automated classification of performance crises, in Proc. EuroSys '10, 111-124, 2010.

[2] D. Breitgand, E. Henis, E., and O. Shehory, Automated and adaptive threshold setting: Enabling technology for autonomy and self-management, in Proc. ICAC'05, 204-215, 2005.

[3] G. Bronevetsky, I. Laguna, B. de Supinski, and S. Bagchi, Automatic fault characterization via abnormality-enhanced classification, in Proc. DSN'11, 1-12, 2011.

[4] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. Chase, Correlating instrumentation data to system states: a building block for automated diagnosis and control, in Proc. OSDI'04, 231-244. 2004.

[5] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox, Capturing, indexing, clustering, and retrieving system history, in Proc. SOSP'05, 105-118, 2005.

[6] R. Ding, Q. Fu, J.-G. Lou, Q. Lin, D. Zhang, J. Shen, T. Xie. Healing Online Service Systems via Mining Historical Issue Repositories, in Proc. ASE 2012, 318-321, 2012.

[7] S. Duan and S. Babu, Guided problem diagnosis through active learning, in Proc. ICAC'08, 45-54. 2008.

[8] Q. Fu, J.-G. Lou, Q. Lin, R. Ding, D. Zhang, Z. Ye, T. Xie, Performance issue diagnosis for online service systems, in Proc. SRDS'12, 273-278, 2012.

[9] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE TPAMI, 6(6), 721-724. 1984.

[10] Y. Gong, and W. Xu, Markov Random Fields and Gibbs Sampling, Machine Learning for Multimedia Content Analysis, 115-147, 2007.

[11] E. Kıcıman and A. Fox, Detecting application-level failures in component-based Internet services. IEEE Trans. on Neural Networks. 16, vol 5, 1027-1041. 2005.

[12] H. Li, W. Zhi, and J. Maris, A Hidden Markov Random Field Model for Genome-wide Association Studies, Biostatistics, 2009, 11(1):139-150.

[13] H. Malik, H. Hemmati, and A. E. Hassan, Automatic detection of performance deviations in the load testing of large scale systems, in Proc. ICSE'13, 1012-1021, 2013.

[14] M. Natu, S. Patil, V. Sadaphal, and H. Vin, Automated debugging of SLO violations in enterprise systems, in Proc. COMSNETS'10, 1-10. 2010.

[15] P. Smith, Markov Monitoring with Unknown States, IEEE Journal on Selected Areas in Communications, 12(9), 1600-1612, 1994.

[16] E. Stehle, K. Lynch, M. Shevertalov, C. Rorres, and S. Mancoridis, On the use of computational geometry to detect software faults at runtime, in Proc. ICAC'10, 109-118, 2010.

[17] S. Theodoridis and K. Koutroumbas, Pattern Recognition, 2006.

[18] TPC-W Benchmark: http://www.tpc.org/tpcw/

[19] E. J. Weyuker, F. I. Vokolos, Experience with performance testing of software systems: Issues, an approach, and case study, IEEE Trans. Softw. Eng., 26(12), 1147 - 1156, 2000.

[20] S. Zhang, I. Cohen, M. Goldszmidt, and J. Symons, Ensembles of models for automated diagnosis of system performance problems, in Proc. DSN'05, 644-653. 2005.

[21] Y. Zhang, S. Smith, and M. Brady, Hidden Markov Random Field Model and Segmentation of Brain MR Images, IEEE Trans Med Imaging. 2001 Jan; 20(1):45-57.

[22] X. Zhou. Statistical Methods in Diagnostic Medicine. Wiley & Sons. 2002.

[23] P. Gill, N. Jain, N. Nagappan, Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications, in Proc. SIGCOMM'11: 350-361, 2011.