Fingerprinting the Datacenter: Automated Classification of Performance Crises

Peter Bodík[1], Moises Goldszmidt[2], Armando Fox[1], Dawn B. Woodard[3], Hans Andersen[4]

[1]University of California, Berkeley
[2]Microsoft Research
[3]Cornell University
[4]Microsoft

Abstract

Contemporary datacenters comprise hundreds or thousands of machines running applications requiring high availability and responsiveness. Although a performance crisis is easily *detected* by monitoring key end-to-end performance indicators (KPIs) such as response latency or request throughput, the variety of conditions that can lead to KPI degradation makes it difficult to select appropriate recovery actions.

We propose and evaluate a methodology for automatic classification and identification of crises, and in particular for detecting whether a given crisis has been seen before, so that a known solution may be immediately applied. Our approach is based on a new and efficient representation of the datacenter's state called a *fingerprint*, constructed by statistical selection and summarization of the hundreds of performance metrics typically collected on such systems. Our evaluation uses 4 months of trouble-ticket data from a production datacenter with hundreds of machines running a 24x7 enterprise-class user-facing application. In experiments in a realistic and rigorous operational setting, our approach provides operators the information necessary to initiate recovery actions with 80% correctness in an average of 10 minutes, which is 50 minutes earlier than the deadline provided to us by the operators. To the best of our knowledge this is the first rigorous evaluation of any such approach on a large-scale production installation.

1. Introduction

A datacenter *performance crisis* occurs when availability or responsiveness goals are compromised by inevitable hardware and software problems [16]. The application operators' highest priority is to stabilize the system and avoid crisis escalation; they typically do this by inspecting collected system metrics (telemetry), logs, and alarms. We aim to provide tools to automate problem *identification*, thereby speeding stabilization. In particular, performance crises may recur because the bug fix for the underlying problem has not yet been deployed, because the fix is based on a misunderstanding of the root cause [3, 10], or because of emergent misbehaviors due to large scale and high utilization[1]. If operators can quickly determine whether an emerging crisis is similar to a previously-seen crisis, a known remedy may avoid escalation and allow root-cause analysis to proceed offline.

Automatic identification of performance crises requires mechanisms to capture these patterns and match them against previous patterns in a database, effectively reducing problem identification to information retrieval. Earlier work [7] showed that while this is possible, crisis identification suffers if either too few or too many metrics are used to distinguish crises, or if the wrong subset of metrics is analyzed; and furthermore that the size and membership of this ideal subset depends on which crises have already been seen. Although the authors' findings were encouraging, the method was evaluated on modest workloads running on few servers and using generous criteria for identification accuracy. In reality, today's applications run on hundreds up to tens of thousands of machines in a datacenter, and since the goal of problem identification is to provide actionable information for initiating recovery, the evaluation criteria should be stringent.

In this paper we present a methodology for automating the identification of performance crises. By identification we mean that if a crisis has been previously seen (using operator-supplied labels as the initial ground truth) it is so labeled; otherwise it is labeled as "new type of crisis." Our main contribution is a methodology for constructing a datacenter *fingerprint*, a digest of the datacenter metrics that

[1] Jeff Dean, Google Fellow, keynote at LADIS 2009 workshop

summarizes datacenter state both across servers in the datacenter and over time. We show that our fingerprints identify and distinguish performance crises with higher accuracy than approaches using all available metrics, approaches using human-selected key performance indicators, and the approach taken by the most closely related work [7] using a different statistical selection method. Our fingerprint representation can be computed efficiently and scales to very large clusters with hundreds of performance metrics per server.

We use a rigorous methodology and stringent accuracy criteria to validate the approach on four months of data from a production datacenter. Our results clearly establish that:

1. When used in a fully-operational setting, our approach achieves identification accuracy of 80% and, on average, identifies the crises ten minutes after they were detected. In contrast, the operators of the datacenter have informed us that automatic identification is still useful up to one hour after a crisis begins, so in 80% of the cases our approach could have reduced the crisis duration by as much as 50 minutes.
2. The subset of metrics automatically selected and summarized by our approach identifies crises better than competing approaches representative of both current industry practice and the most recent literature.
3. The discriminative power of our approach is nearly optimal, as demonstrated by experiments in which we remove the need to update various parameters in an online fashion as each new crisis is seen. These experiments validate that a fingerprint based on collected performance metrics is an effective and compact representation of the datacenter state.
4. Our approach clearly quantifies tradeoffs among false positives, accuracy of identification, and time to identification.

To the best of our knowledge, this is the first time such an approach has been applied to a large-scale production installation with rigorous validation using hand-labeled data.

2. Related Work

In a typical datacenter today, when an application starts experiencing a performance anomaly (e.g. high request latency), an alarm automatically alerts the on-call operator [3]. The operator begins manual investigation using log files, graphs, and other information. It is not unusual for problem identification to take an hour or more, after which the operator can begin corrective action.

We envision that by the time the operator responds to the alarm, she might already have a message in her inbox: "The current crisis is similar to a crisis that occurred two weeks ago. In the former crisis, redirecting traffic to a different datacenter resolved the problem." If the determination of similarity were correct, the operator could avoid tens of minutes of downtime by initiating the same recovery action.

As early as 2003, the authors of [18] proposed the use of compute-intensive modeling techniques to perform such automatic recognition. Since then, researchers have tried to identify operational problems by analyzing performance metrics using machine learning [4, 6, 7, 9, 17, 23, 25], by identifying unusual or noteworthy sequences of events that might be indicators of unexpected behavior [5, 19], and by manually instrumenting the system [2] and creating libraries of possible faults and their consequences [21]. Others have laid out general methodological challenges in using computers to diagnose computer problems [8, 11].

By far the work closest in spirit to our own is the "signatures" approach to identifying and retrieving the essential system state corresponding to previously-seen crises [7]. The authors propose a methodology for constructing "signatures" of server performance problems by first using machine learning techniques to identify the performance metrics most relevant to a particular crisis; second, using the induced models for online identification; and third, relying on similarity search to recognize a previously recorded instance of a particular incident. They showed their approach to be successful in a small transactional system on a handful of performance problems.

We view our methodology as a direct descendant of the "signatures" approach but with several important improvements. First, our fingerprint representation size scales linearly, rather than exponentially, with the number of metrics considered. Second, the representation size is independent of the number of machines and thus can be used with very large deployments. Third, the signatures approach maintains multiple models (one per crisis seen), computes a fitness score to decide which of the existing models are likely to provide the best identification of the current crisis, and then uses these to construct the signature of (and thereby identify) the crisis. In contrast, we only use our models to determine which metrics are relevant for modeling the crises and not for "matching" them. This simplification allows our approach to avoid two related sources of potential error and their corresponding free parameters. First, we need no policies to maintain and ensure the validity of multiple models. Second, since we don't maintain multiple models, we need neither a fitness score to determine which models to apply nor a method to combine the output of multiple models.

HiLighter [4] showed that the use of regularized logistic regression [22] as a classifier results in a metric selection process that is more robust to noise than the naïve Bayes classifier used in the signatures approach. In particular, HiLighter avoids the wrap-around search for the relevant features for each model as was done in the signatures work. However, like signatures, HiLighter must deal with the problems of model management and online selection, and proposes a representation of performance state that can grow exponentially with the number of metrics being recorded.

3. Problem and Approach

A typical datacenter-scale user-facing application runs simultaneously on hundreds or thousands of machines. In order to detect performance problems and perform postmortem analysis after such problems, several *performance metrics* are usually collected on each machine and logged to online or nearline storage. Since large collections of servers execute the same code, under normal load balancing conditions the values of these metrics should come from the same distribution; as we will show, we use this intuition to capture the state of each metric and identify unusual behavior.

Each metric is usually sampled once per aggregation *epoch*—typically a few minutes—and the sampled values may represent a simple aggregate over the aggregation epoch, e.g. the mean. The metrics correspond to hardware, OS, application, or runtime-level measurements, such as the size of the object heap or number of threads waiting in the run queue. Wide variation exists in what is collected and at what granularity; packages such as HP OpenView [1], Ganglia [15], and others provide off-the-shelf starting points.

A small subset of the collected metrics may be *key performance indicators* (KPI's) whose values form part of the definition of a contractual service-level agreement (SLA) for the application. An SLA typically specifies a threshold value for each KPI and the minimum fraction of machines that have to satisfy the requirement over a particular time interval. For example, an SLA might require that the end-to-end interactive response time be below a certain threshold value for 99.9% of all requests in any 15-minute interval.

A performance *crisis* is defined as a prolonged violation of one or more specified SLA's. Recovery from the crisis involves taking the necessary actions to return the system to an SLA-compliant state. If the operators can recognize that the crisis is of a previously-seen type, a known remedy can be applied, reducing overall recovery time. Conversely, if the operators can quickly determine that the crisis does not correspond to any previously seen incident, they can immediately focus on diagnosis and resolution steps, and record the result in case a similar crisis recurs.

Our goal is to automate the *crisis identification* process by capturing and concisely summarizing the subset of the collected metrics that best discriminate among different crises. We next describe our process for doing this, called *fingerprinting* the datacenter, and how we define a similarity metric between two fingerprints to identify recurring problems.

3.1 Fingerprint-based recognition

A fingerprint is a vector representing the performance state of a datacenter application that uniquely identifies a performance crisis. It is based on values of performance metrics and, intuitively, it characterizes which metrics' values have significantly increased or decreased on a large fraction of the application servers. There are four steps to our fingerprint-based recognition and identification technique.

1. We summarize the values of each performance metric in a particular epoch across all the application servers by computing the *quantiles* of the measured values (such as the median of CPU utilization on all servers). Unlike statistics such as the mean, quantiles are more robust to outliers in the distribution of the metric values. As we discuss in Section 3.2, this summarization scales well with the number of servers.
2. Based on the past values of each metric quantile, we characterize its current value as *hot*, *cold*, or *normal*, representing abnormally high, abnormally low, or normal value, respectively. We discuss this step and the choice of hot and cold thresholds in Section 3.2. This gives us a *summary vector* containing one element per quantile per tracked metric, indicating whether the value of that quantile is cold, normal, or hot during that epoch.
3. We identify the *relevant metrics* whose quantile behavior distinguishes normal performance from the performance crises defined by the SLA's. The metric selection process is described in Section 3.3. This subset of the summary vector for a given epoch is the *epoch fingerprint*.
4. Since most crises span multiple epochs, we show how to combine consecutive epoch fingerprints into a *crisis fingerprint*. We define a similarity metric for determining whether two crisis fingerprints correspond to the same underlying problem. These two steps are described in Section 3.4.
5. Finally, we observe that in a real operational setting, crises appear sequentially and identification of a crisis can be based only on information obtained from the previous crises. We hypothesize that crisis identification will be improved through *adaptation* — updating identification parameters each time a correctly-labeled crisis is added to the dataset. The adaptation procedure is described in section 3.5.

3.2 Hot and Cold Metric Quantiles

In the first step, we compactly represent the values of each metric on all servers during a particular epoch. Because servers of the same application typically run the same code, we can view the measured values of the same metric as samples of a random variable whose distribution is unknown. We thus summarize the metric values across all servers using several quantiles of the observed empirical cumulative distribution over an epoch (see an illustration in Figure 1 on using the median of the metrics). In this paper we refer to these quantiles as *metric quantiles*.

We use quantiles instead of other statistics such as mean and variance because quantiles are less susceptible to outliers. This summarization of the state of the metrics does not grow as the number of machines increases, so the size of the fingerprint is only proportional to the number of metrics being collected. In addition, there are well known al-

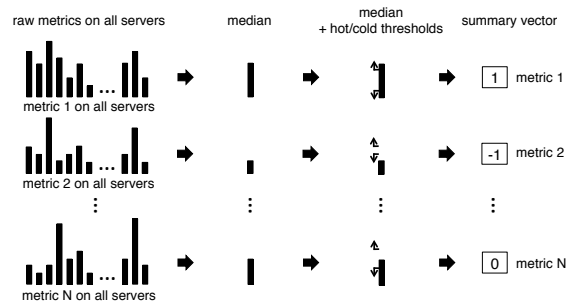


Figure 1. The summary vector of a particular epoch is created in two steps. First, the values of each metric are summarized using one or more quantiles (here we use the median). Second, each metric quantile is discretized into a *hot*, *normal*, or *cold* state based on its hot/cold thresholds (represented by the arrows). Each square in the summary vector represents the state of a particular metric quantile, with $-1, 0, 1$ corresponding to cold, normal, hot respectively.

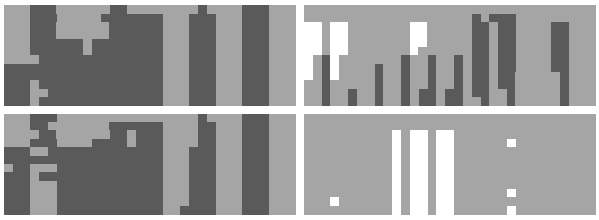


Figure 2. Counterclockwise from top left, fingerprints of crises B, B, D and C from Table 1. Each row is an epoch and each column represents the state of a particular metric quantile, with white, gray, black corresponding to cold, normal, hot $(-1, 0, +1)$ respectively in the fingerprint.

gorithms for estimating quantiles with bounded error using online sampling [12], which guarantee that the entries in the fingerprints can be computed efficiently. In our case study, which involved several hundred machines, we computed the values of the quantiles exactly.

We observe that the main factor that differentiates between different types of crises are the different metrics that take extreme values during the crisis. Our objective is to capture this fact in the fingerprint, that is, to encode which metrics increased or decreased significantly on a large number of servers during the crisis. We achieve this by discretizing the value of each metric quantile to one of three states: extremely high (*hot*), extremely low (*cold*), or *normal* relative to its past values. For example, if the median of a metric is *hot*, the most recent value for that quantile is higher than normal.

The computation of hot and cold thresholds is parameterized by hyperparameter p – percentage of past values of a metric quantile that are considered extremely low or high during normal system operation. The cold threshold of a particular metric quantile m (such as median of CPU utilization) is computed as the $p/2^{\text{th}}$ percentile of values of m in the past W days that exclude epochs with SLA violations. The hot threshold of m is computed as $(100 - p/2)^{\text{th}}$ percentile over the same period. For example, for $p = 4\%$ we would use the 2nd and 98th percentiles. In Sections 5 and 6.1 we discuss the choice of quantiles, hyperparameters p and W and examine the sensitivity of our approach to their values.

We can now build a *summary vector* for one epoch: it is a vector of $Q \times M$ elements, where M is the number of metrics and each group of Q elements corresponds to the metric quantiles of individual metrics. An element's value is -1 if the quantile's value is below the *cold* threshold during the epoch, $+1$ if the quantile's value is above the *hot* threshold, and 0 otherwise (see Figure 1).

3.3 Selecting the Relevant Metrics

As we will show in our experiments (Section 5.1), achieving robust discrimination and high identification accuracy requires selecting a subset of the metrics, namely the *relevant* metrics for building the fingerprints. We determine which metrics are relevant in two steps. We first select metrics that correlate well with the occurrence of each individual crisis by borrowing techniques from machine learning, specifically *feature selection and classification* on data surrounding each crisis. Second, we use metrics most frequently selected in the previous step as the relevant metrics used for building all fingerprints. The summary vector is converted into an *epoch fingerprint* by selecting only the relevant metrics.

Feature selection and classification is a technique from statistical machine learning that first induces a function between a set of features (the metrics in our case) and a class (crisis or no crisis) and tries to find a small subset of the available features that yields an accurate function. Let $X_{m,t}$ be the vector of metrics collected on machine m at time t and $Y_{m,t}$ be 1 if m violated an SLA at time t, or 0 otherwise. A classifier is a function that predicts the performance state of a machine, Y, given the collected metrics X as input. The feature selection component picks a subset of X that still renders this prediction accurate. In our approach we use logistic regression with L1 regularization [22] as the statistical machine learning method. The idea behind regularized logistic regression is to augment the model fitting to minimize both the prediction error and the sum of the model coefficients. This in turn forces irrelevant parameters to go to zero, effectively performing feature selection. It has been (empirically) shown in various settings that this method is effective even in cases where the number of samples is comparable to the number of parameters in the original model [13], as is the case in our scenario in which the number of possible features (over 100 per server for several hundred servers) exceeds the number of classification samples. Note that the crises do not need to be labeled when performing the metric selection, so there is no burden on the operator; this step is completely automated.

3.4 Matching Similar Crises

In the final step we summarize epoch fingerprints during a single crisis into a *crisis fingerprint* and compare crisis fingerprints using a distance metric. First, because crises usually last for more than one epoch, we create a crisis fingerprint by averaging the corresponding epoch fingerprints, thus summarizing them across time. For example, Figure 2 shows epoch fingerprints of four crises. Each row represents an epoch, each column represents a metric quantile, and white, gray, and black represent the values -1, 0, and 1, respectively of the cold, normal, and hot state respectively. The three left-most columns of the top-right crisis in the figure would be summarized as $\{\frac{-7}{12}, \frac{-4}{12}, \frac{6}{12}\}$; there are 12 epochs in the crisis and the column sums are -7, -4, and 6. Notice that the quantiles often don't move in the same direction—for example, see the left-most three columns in the top-right crisis—which is important for identification.

If the Euclidean distance between the fingerprints of a pair of crises is less than the *identification threshold* T, the crises are considered identical, otherwise they are different. Intuitively, if T is too low, some identical crises would be classified as different (false negative), while if T is too high, different crises would be classified as identical (false positive/false alarm). We define the false alarm rate α as the number of pairs of different crises that are incorrectly classified as identical divided by the number of pairs of crises that are different. We ask the operator to specify an acceptable bound on α, and we set T to the maximum identification threshold that respects that bound. A ROC (receiver operating characteristic) curve, such as the one in Figure 5, is particularly useful for visualizing this. In our experiments, we set α close to zero, essentially guaranteeing no false positives in practice. We illustrate the effects of increasing α on the identification process in Figure 6.

In the methodology above we assume that the operator is able to correctly label a crisis sometime after resolution, as the threshold T may then need to be adjusted to maintain the desired rate of false positives. If a significant number of past crises cannot be reliably labeled, we instead pose crisis matching as an unsupervised online clustering problem. Such an approach requires more sophisticated probabilistic models and computational statistical inference; we report some early results on this aspect of the problem in Section 6.2.

3.5 Adaptation

Once a new crisis is resolved and an operator correctly labels it, we update the fingerprinting parameters: hot and cold thresholds, set of relevant metrics, and identification threshold T. First, we update the hot and cold thresholds based on values of metric quantiles in the past W days as described in Section 3.2. Second, we select the most frequent metrics from the most recent C crises as described in Section 3.3. Third, we update the identification threshold T based on all the past labeled crises to achieve expected false alarm rate of α. Finally, fingerprints of the past crises are recalculated based on the new fingerprinting parameters.

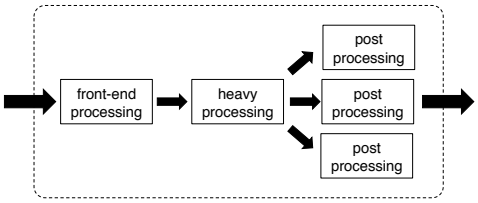


Figure 3. Processing on machines in the datacenter under study.

4. Evaluation

Using the ground truth labels provided by human operators of a production system described in Section 4.1, we evaluate our approach and compare it to three alternatives: a) one that relies only on the operator-identified Key Performance Indicators (KPI's) for crisis identification, b) one that uses all available metrics for identification, and c) one that models crises using the *signatures* approach described in [7]. Our evaluation consists of three parts. First, we evaluate the *discriminative power* of our approach and compare it to the other approaches, using the entire available dataset. That is, we quantify how accurately each approach classifies two crises as identical or not. This part of the evaluation, described in Section 4.2, establishes an upper bound on identification accuracy for all approaches.

Next, we simulate the operational setting in which our approach is designed to be used, in which crises appear sequentially and identification of a crisis is based only on information obtained from the previous crises. In these experiments we use adaptation described in Section 3.5. Section 4.3 describes how we evaluate the accuracy and time-to-identification of our technique and quantify the loss of accuracy resulting from the use of only partial information.

Finally, in Section 4.4 we compare our approach to the others, again in an operational setting. However, since each approach uses different techniques for adaptation, to make the comparison meaningful we remove the need for adaptation by providing access to the entire dataset for training. We refer to this as operational setting with an *oracle*.

4.1 System Under Study

We evaluate our approach on data from a commercial datacenter running a 24×7 enterprise-class user-facing application. It is one of four datacenters worldwide running this application, each containing hundreds of machines, serving several thousand enterprise customers, and processing a few billion transactions per day.[2] Most machines execute the same application, as depicted in Figure 3. The incoming workload is processed on the machine in three stages:

[2] The exact numbers are considered confidential by the company that operates the datacenter.

ID	# of instances	label
A	2	overloaded front-end
B	9	overloaded back-end
C	1	database configuration error
D	1	configuration error 1
E	1	configuration error 2
F	1	performance issue
G	1	middle-tier issue
H	1	request routing error
I	1	whole DC turned off and on
J	1	workload spike

Table 1. List of identified performance crises. Names are indicative of the root cause. We stress that almost all these crises manifested themselves through multiple metrics, and that there is overlap between the metrics of the different crises.

light processing in the front-end, core of the execution in the second stage, followed by some back-end processing. The requests are then distributed to the clients or to another datacenter for archival and further processing. We have no visibility to the clients or to machines in the other datacenters.

For each server, we sample about 100 metrics each averaged over a period of 15 minutes. The 15-minute averaging window is established practice in this datacenter, and we had no choice on this matter; similarly, we have no access to any other performance counters or to information allowing us to reconstruct the actual path of each job through the server. The metrics include counts of alerts set up by the operators, queue lengths, latencies on intermediate processing steps, summaries of CPU utilization, and various application-specific metrics.

The operators of the site designate three *key performance indicators* (KPI's) corresponding to the average processing time in the front end, the second stage, and one of the post-processing stages. Each KPI has an associated service-level agreement (SLA) threshold determined as a matter of business policy. A performance crisis is declared when 10% of the machines violate *any* KPI SLA's. This definition is set by the operators and we did not tamper with it.

We use four months of production data from January to April 2008. During this period, the datacenter operators manually diagnosed and labeled 19 crises, ranging from configuration problems to unexpected workloads to backlogs caused by a connection to another datacenter. These crises were labeled by the operators according to the determined underlying cause. Descriptive labels of the crises, and the number of times each type occurred, appear in Table 1. We also had access to collected metrics and 20 unlabeled crises that occurred between September and December 2007. We use this additional data when simulating online deployment, but we don't use it to evaluate identification accuracy.

4.2 Evaluating Discrimination

Discrimination measures how accurately a particular crisis representation classifies two crises as the same or distinct. We compare our method to three other approaches, in each case using the entire dataset so as to give each method the maximum information possible. This establishes the baseline ability to capture the differences between different crises for each method. As is standard, we compare the different approaches using ROC curves [14] that represent the tradeoff between the false alarm rate (incorrectly classifying two different crises as identical) and recall (correctly classifying two identical crises) over the whole range of the identification threshold T. It is standard practice to represent this comparison numerically by computing the area under the curve (AUC). The optimal approach will have an AUC of 1, indicating that there is no tradeoff between detection and false positives. By using an ROC curve for comparison, we take into account all possible cost-based scenarios in terms of the tradeoff between missing identical crises versus considering different crises to be identical.

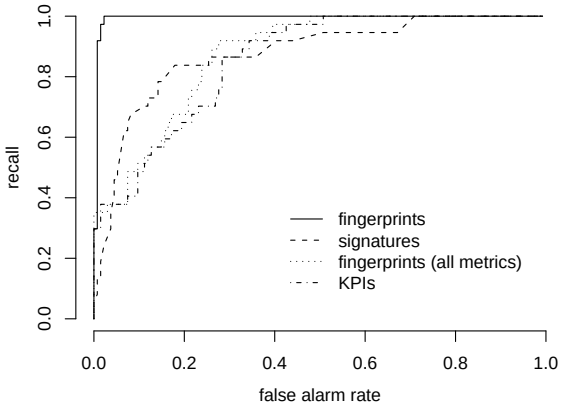
4.3 Evaluating Identification Accuracy and Stability

Identification accuracy measures how accurately our approach labels the crises. A human operator has hand-labeled each crisis in our dataset, but in an operational setting the crises are observed sequentially. Each crisis is labeled *unknown* if the distance between its fingerprint and all fingerprints of past crises is greater than the identification threshold T; otherwise it is labeled as being identical to the closest crisis. Since many crises (indeed, all those in our dataset) last longer than a single 15-minute epoch, we must also define *identification stability*—the likelihood that once our approach has labeled a crisis as known, it will not change the label later while the crisis is still in progress. In each epoch, the identification algorithm emits either the label of a known crisis, or the label `x` for *unknown*. A sequence of K identifications is *stable* if it consists of $n \geq 0$ consecutive `x`'s followed $K - n$ consecutive *identical* labels. Since the operators of this application informed us that identification information is useful up to one hour into a crisis, we use $K = 5$. For example, if A and B are labels of known crises, the sequences `xxAAA`, `BBBBB`, and `xxxxx` are all stable, whereas `xxAxA`. `xxAAB`, `AAAAB` are all unstable. Given this stability criterion, a sequence is *accurate* if it is stable *and* the labeling is correct; that is, either all labels are `x`'s and the crisis is indeed new, or the unique non-`x` label matches that of a previously-seen crisis that is identical to the current crisis. Further, for a previously-seen crisis we can define *time to identification* as the first epoch after crisis onset during which the (correct) non-`x` label is emitted.

We emphasize that from the point of view of Recovery-Oriented Computing [16], stability is essential because the system operator's goal is to initiate appropriate recovery actions as soon as possible once the crisis has been identified.

Problem identification workflow
When a crisis is detected based on KPIs:
    update hot/cold thresholds (Section 3.2)
    update past crises' fingerprint entries (3.1)
During first K epochs of crisis (we use $K = 5$, see Sec. 4.3):
    update crisis fingerprint with new data (3.1)
    find most similar past crisis P (3.4)
    if similarity within identification threshold (3.5)
      emit label P, else emit label X
When crisis is over:
    operators verify label of crisis
    update set of relevant metrics (3.3)
    update identification threshold T (3.5)

Figure 4. Problem identification workflow. Each line refers to a section that describes that step in detail.



type of fingerprint	AUC
fingerprints	0.994
fingerprints with all metrics	0.873
KPIs	0.854
signatures	0.876

Figure 5. ROC curves for crisis discrimination and the area under the curves (AUC) for the fingerprinting and alternative approaches.

Unstable identification could lead the operator to initiate one set of actions only to have the identification procedure "change its mind" later and apply a different label to the crisis, which would have implied different recovery operations. There is an inherent tradeoff between time to identification and stability of identification; we quantify this tradeoff in Section 5 and show how a system operator can control the tradeoff by setting a single parameter in our algorithm.

4.4 Comparing to Other Approaches

When comparing our identification accuracy to that of other approaches, to make the comparison meaningful we *eliminate* the adaptation described in Section 3.5 from both our approach and those we compare against. With adaptation in place, any comparison would also have to compare the relative loss of accuracy of each method when only partial information is used to make decisions. Instead of adaptation, we use an *oracle* to set the best parameters for each method, allowing us to show each approach at its best.

To remove adaptation from the fingerprinting approach, we compute the identification threshold T based on an ROC curve over all labeled data, we select a single set of relevant metrics using models induced on the labelled crises, and we compute hot and cold thresholds based on the whole dataset. We use this set of parameters throughout the experiment.

To explain how we remove adaptation from the signatures approach in [7], we first briefly review the adaptation it usually performs. The signatures approach builds a classifier for each crisis that tries to predict whether the current system state will result in an SLA violation on the KPIs. The SLA state serves as ground truth for the classifier, and the *signature* captures the subset of metrics that form the features used by the classifier that achieves the highest accuracy. Crisis recognition consists of first selecting a subset of the models with highest prediction accuracy on the current crisis, and then building a signature based on the most relevant metrics. This entire procedure requires setting many parameters, including the number of epochs on which the model is evaluated and the number of models being selected (or a threshold on the Brier score for selection). Adaptation consists of periodically merging similar models and deleting inactive/obsolete models [25]; these processes depend on additional free parameters.

To remove adaptation from the signatures approach, we allow it to always select the optimal model for each crisis. This gives the signatures approach a model management technique that is omniscient, clairvoyant, and optimal. In addition, since our system consists of hundreds of servers rather than the handful of servers used for evaluation in [7], rather than assigning a model to each server we assign a model to the datacenter and summarize the metrics using quantiles. Finally, in place of the naive Bayes models used in [7], we use logistic regression with L1 regularization for feature selection; since the logistic regression models were more accurate in our setting than those using naive Bayes, this gives the signatures approach another advantage.

We point out that our criteria for accuracy are much more stringent and more realistic than those used in [7]. In that work, an identification was considered successful as long as the actual crisis was *among* the k most similar crises selected by the algorithm, according to a distance metric. In contrast, our identification is successful if the single correct label is produced, and in case the crisis is a previously-unseen crisis, identification is successful only if it reports "unknown" and does *not* assign the label of any known crisis. Furthermore, the stability criterion, which is a prerequisite to accuracy in our approach, has no analogue in [7].

4.5 Experimental Setup and Procedure

As the approach is intended to be used in an "online" operational setting in which crises occur sequentially and they must be identified as soon as possible, we simulate this setting in our experiments. That is to say, at the point the crisis is detected through an SLA violation, the system executes the operations indicated in Figure 4 in order to decide whether the crisis has occurred before. After each crisis, various parameters are automatically updated as described in previous sections. We note that the final verification of the crisis label is performed offline and may require several iterations and different tier level operators, all of which are outside the scope of this paper.

5. Results

The main results reported in this section used the following settings. The fingerprints where built using three quantiles for each metrics; in addition to the median, we added the 25^{th} and 95^{th} percentiles to capture the variance. In the selection of the relevant metrics we used classifier models containing ten metrics (the balanced accuracy was high enough and the standard deviation in the cross-validation was low), and we used the most frequent 30 metrics over the past 20 crises as the relevant metrics for the fingerprints. Finally we used a moving window of 240 days to set the hot/cold thresholds using $p = 4\%$. Section 6.1 describes the sensitivity analysis and how to set these parameters in a realistic setting.

5.1 Discriminative Power

As discussed in Section 4.2, we start our evaluation of the fingerprint approach by examining its basic capabilities in classifying two crises as the same or distinct, and comparing this ability to alternative approaches. The graph in Figure 5 shows the ROC curves and AUC for each approach. The fingerprint approach exhibits an AUC of 0.994, which means that in terms of discrimination, this approach is able to maximize the detection rate with extremely few false positives. Comparing to the alternative approaches, using the KPI's alone gets an AUC of 0.854 and the approach using all the metrics gets 0.874. Furthermore looking at the shape of the curves it is clear that neither is able to discriminate at the same level as the fingerprints. Using the KPIs simply does not provide enough power to discriminate between the different types, and using all the metrics simply obfuscates the discriminative signal by introducing noise.[3] Finally, the signatures approach performs better than both these two approaches but still well bellow the fingerprinting approach proposed in this paper.

[3] Metrics that are not correlated with the crises may take extreme values during both crises and "normal" intervals.

operational setting	known acc.	unknown acc.
oracle	98%	93%
adaptation, bootstrap w/ 10	77%	82%
adaptation, bootstrap w/ 5	76%	83%
adaptation, bootstrap w/ 2	78%	74%

Table 2. Summary of the results for different settings.

5.2 Fully operational setting

In the following experiments we simulate the conditions under which the approach would be used for identification of crises in an operational setting where crises arrive one at a time and we update fingerprinting parameters as we observe them. To remove dependencies on a particular order we perform experiments using 20 random permutations of the crises, one of which is the actual chronological order, and report the average accuracy across all runs.

At the onset of a new crisis, we perform the following: a) update relevant metrics, hot/cold thresholds, and identification threshold T as described in Section 3.5, b) update fingerprints of past crises, c) perform identification using the distances between crises and the identification threshold.

Recall from Section 4.3 that a new crisis C is said to be identified accurately if the identification is *stable* over five epochs and if the label is correct. If C is previously known (i.e. if the set of crises that occurred in the past contains some crisis identical to C), the correct label would be that of the previously seen crisis. If the past set of crises contains no crisis identical to C, the correct label would be *unknown*. We compute the *known accuracy* (fraction of correct identifications for previously seen crises), the *unknown accuracy* (fraction of correct identifications for previously unseen crises), and also the *time to identification* (the average time between crisis detection and its correct identification).

To evaluate the performance of our method with adaptation, we run three sets of experiments, each time starting with a different number of labeled crises. When starting with two labeled crises, we achieve known and unknown accuracy of 78% and 74%[4]. Starting with five and ten crises yields accuracies of approximately 76% and 83% (see Table 2 and Figure 6).

Besides accuracy, the time at which the method makes a decision regarding the identity of the crisis is important to the operator. The dependency among these three evaluation metrics is made clear in Figure 6. We note that our method is able to make the identification with 80% accuracy within ten minutes of crisis detection, even in a fully operational setting where the relevant metrics and identification threshold are adapted in an online fashion. Operators of this web application mentioned that correct identification is useful even one hour after the crisis was detected.

[4] We report accuracies for $\alpha = 0.001$ – conservative value that guarantees almost no false alarms

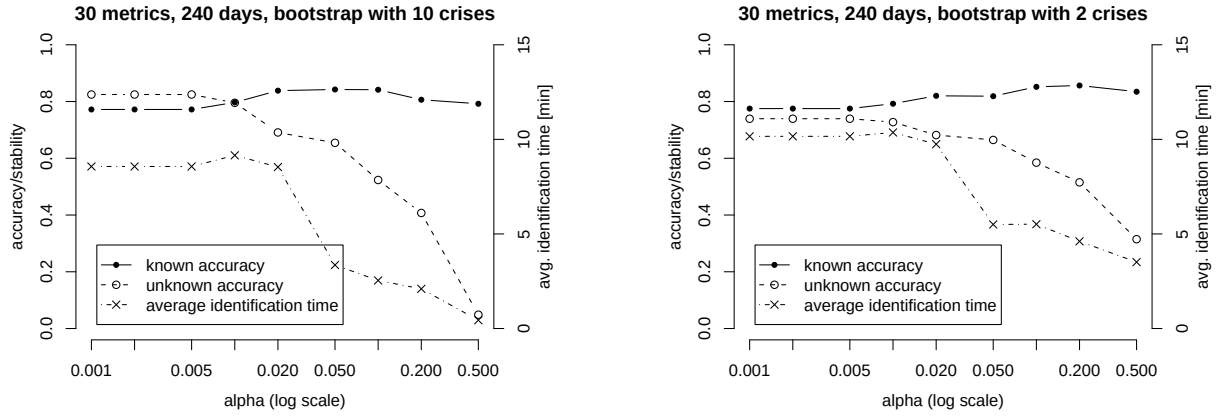


Figure 6. Known accuracy, unknown accuracy, and time of identification results in **full operational setting** when using 30 metrics in fingerprints, 240 days of moving window, and bootstrapping with ten and two labeled crises (top). X-axes represents different values of the false alarm rate parameter specified by the operator.

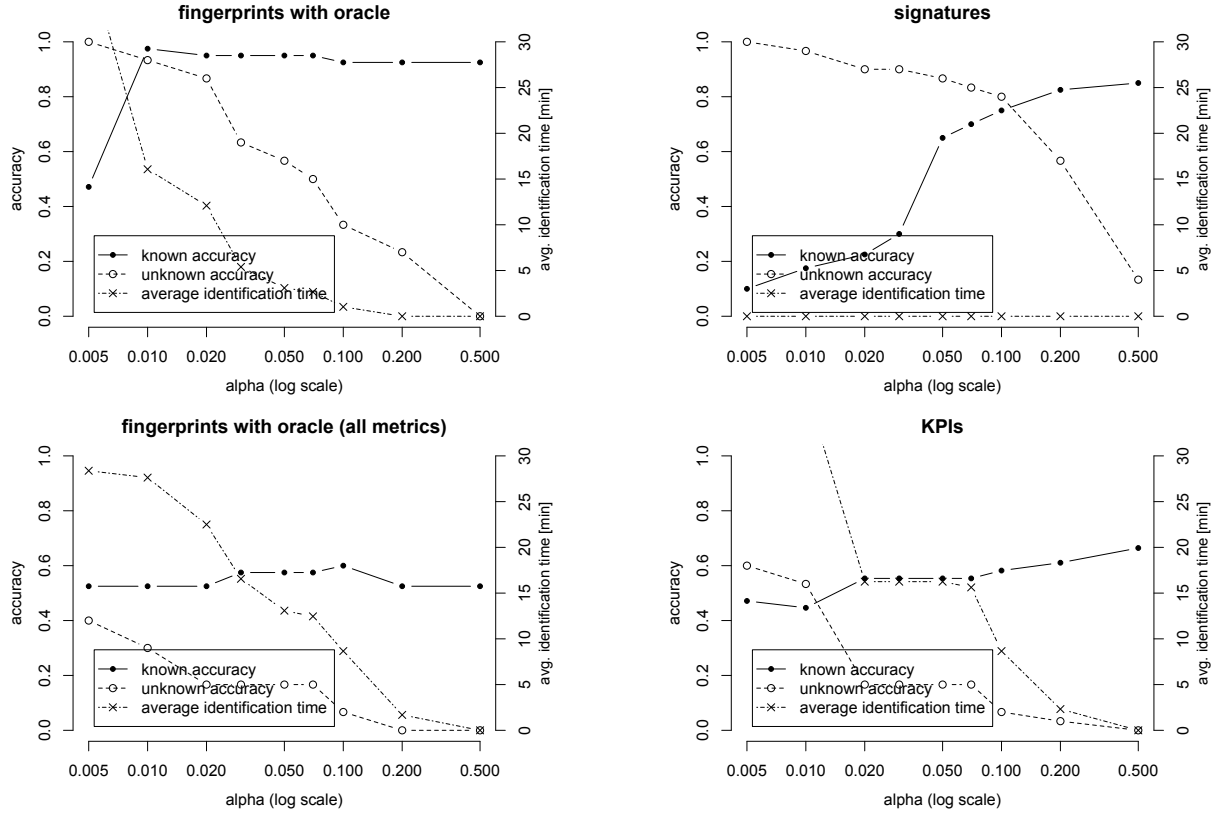


Figure 7. Known accuracy, unknown accuracy, and time of identification for different crisis signatures when using an **oracle**: fingerprints (top left), signatures [7] (top right), fingerprints using all the available metrics (bottom left), and KPIs (bottom right). X-axes represents different values of the false alarm rate parameter specified by the operator.

5.3 Operational setting with an oracle

To compare our approach to the three alternative approaches, we eliminate the adaptation as described in Section 4.4. Instead, each method uses the best settings of its parameters based on the whole dataset as if provided by an oracle. These parameters are not updated as new crises arrive.

For each of the approaches we executed five runs with different initial set of crises and performed identification on the remaining 14 crises. The initial set of crises always contained two crises of type “B”, one of type “A”, and two other crises that varied between runs. We report the average of all the evaluation metrics across the five runs in Figure 7.

Fingerprinting achieves very high known and unknown accuracies of 97.5% and 93.3%. The approaches based on using all the metrics and the KPIs achieve accuracies of 50% and 55% respectively. The signatures approach [7] performs better than the baselines and achieves accuracies of 75% and 80%, but still worse than fingerprinting.

5.4 Summary of empirical results

From the discrimination and identification experiments, we conclude that:

1. Maintaining only the *relevant* metrics for distinguishing crises from normal operation allows the fingerprinting approach to attain much higher identification accuracy.
2. The KPI's alone provide insufficient information to distinguish or identify different types of crises.
3. The fingerprinting approach, based on keeping concise representations of the datacenter state, performs significantly better than the signatures approach of [7].
4. In the realistic, fully operational setting, we correctly identify 80% of the crises with an average time between crisis detection and its identification of ten minutes.

6. Fingerprinting in Practice

As with any approach based on collecting and analyzing data, applying the approach in an operational setting requires setting some parameters and understanding the sensitivity of the technique to these parameters. Also, since crisis identification is not 100% accurate, we need to establish an acceptable level of uncertainty in production use based on both technical and business reasons. In this section we address each of these operational considerations.

6.1 Setting algorithm parameters

As described in Figure 4 and Section 3.5, we automatically update fingerprinting parameters when a new crisis is detected and after it is over. These updates are based on a set of hyperparameters that may not require any changes, but should be reviewed periodically or when a major change occurs in the datacenter. These hyperparameters include the number of metrics to be part of the fingerprint, number of days W and percentage p of metric values considered extreme when computing hot and cold thresholds, and α used when computing identification threshold T.

The effects of the hyperparameters on the identification accuracy and time to detection could be estimated offline by running identification experiments using past labeled crises. The operators would use ROC curves and graphs such as the ones in Figure 6, to select a suitable operating point of datacenter fingerprinting. Producing these graphs took on the order of minutes for the 19 crises we used in our experiments. After selecting the hyperparameters, the adaptation of the fingerprinting parameters and the identification algorithm proceed automatically with little intervention by the operators. To illustrate the approach, we report on some of the experiments we performed in order to set the hyperparameters and study their effect on the results.

In our reported results we use $p = 4\%$ as the percentage of metric values considered extreme when computing the hot and cold thresholds. When experimenting with values of 2%, 10% and 20% we observe that the area under the ROC curve (as in Section 5.1) decreased from 0.99 to 0.96 – a small change and still far better than the competing approaches.

Instead of using all three quantiles when summarizing metrics, we also tried using just the median. This reduced the identification accuracy for known crises by 5 points in the oracle setting, and by 2–3 points in the fully-operational setting; still better than competing approaches as reported in Section 3.4 and illustrated in Figure 2. Our intuition is that some pairs of crises are distinguished by three quantiles that don't all move in the same direction, and observing the movement of only a single quantile would necessarily fail to capture such differences.

In the experiments in Section 5 we used $M = 30$ metrics for the fingerprint; in additional experiments we observed a decrease in the accuracy of the identification as we tried fingerprints of 20, 10, and 5 metrics, with the moving window size of $W = 240$ days. As we changed W to 120, 30, and 7, we observed that reducing the number of metrics in the fingerprint actually compensates. This is not surprising: as the size of the fingerprint decreases, the fingerprint adjusts more nimbly to rapid changes in values, which comes as a consequence of reducing W. But as W increases and a greater variety of crises is seen, additional information is needed in the fingerprint to capture the differences among them.

Also, as mentioned in Section 3.4, we update T to avoid false positives (α set to 0.1%). In Figure 6 we show the effects of increasing the false positive rate (which in turn will affect T). As expected, the known accuracy will increase while the unknown accuracy will decrease. Although in our datacenter the increase is marginal and the false positive rate will start affecting the result when it is larger than 2%.

Finally, when comparing two crises, we first compute the crisis fingerprints by averaging the corresponding epoch fingerprints. In all the experiments in Section 5, we average across epochs -30 minutes, ..., 60 minutes, relative to the start of the crisis (the limit of 60 minutes was set by the datacenter operators). Figure 8 shows that ranges that start at least 30 minutes before the beginning of the crisis quickly achieve high levels of discrimination.

6.2 Crisis Labeling, Bootstrapping and Uncertainty

As described in Section 3.4 the automatic adaptation of the threshold T depends on the selection of the actual false positive rate, which in turn needs the accurate (forensic) labeling of a significant set of crises. In addition, the highest accuracy results in the full operational setting (Section 5.2) were obtained when the identification process was bootstrapped with 5 labeled crises. We remark that the labeling does not

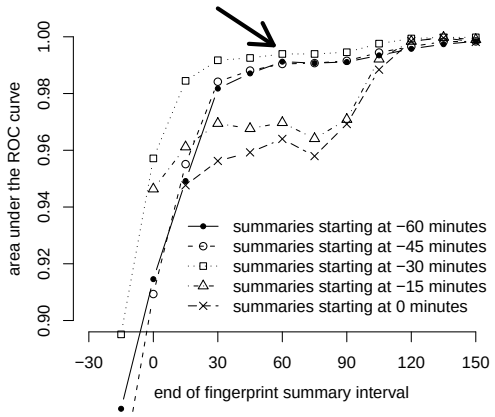


Figure 8. Area under the ROC curve (discriminative power) of fingerprints when summarized over different ranges. Each line on the graph represents ranges that start at the same epoch, while the x-axis represents the end of the range. The arrow points to AUC corresponding to the range $\langle -30\text{minutes}, +60\text{minutes}\rangle$ used in all our experiments.

have to necessarily point to the root cause, but merely group similar crises together, and it does not need to occur in real time. Nevertheless, a natural question is what to do when a new application is first deployed (or significantly modified) and no prior crisis data is available for bootstrapping, or there is significant cost of the forensic analysis of the crises. To this end we have recently proposed and evaluated an approach based on the same fingerprinting representation but a different identification and pattern matching process that is based on performing online clustering of the crises. Notice that this is non-trivial as we need to first decide on the number of clusters, plus decide whether a new crisis merits a new cluster. The approach we are pursuing is based on first modeling the crises as a time series of fingerprints (instead of collapsing them into a crisis fingerprint as in Section 3.4) and then imposing a Dirichlet process mixture (DPM) for grouping the crises online and deciding whether a new cluster is needed. DPMs are well known constructs that have been used successfully in online document clustering [24]; due to space limitations we omit details on the mathematics of the model and the computational procedure, which are described fully in [20].[5] By virtue of being consistently defined in terms of a probability distribution, the approach obviates the need for a distance metric and a threshold for clustering. In addition, in our initial experiments simulating the same full operational environment, the approach achieves accuracies compared to those reported in Section 5.2, without the need for bootstrapping or forensic crises labeling. Moreover, this approach reports a full posterior probability on the clustering, enabling optimal decision making by providing a real uncertainty measure on the identification of the crisis.

[5] A full account will be provided in the final version.

However, the computational process is more complicated (in particular requiring a taxing offline component) and identification takes on average twice as long as the approach we describe here (20 minutes vs. the 10 minutes we report in Section 5.2). This observation suggests a possible hybrid approach: we can start the process with the more sophisticated model based on DPMs [20], and once we have a sufficient number of labeled crises, switch to our simpler matching approach (Section 3.4) to minimize time to identification. Another hybrid approach might use the simpler matching for very fast identification, while using the DPM approach to calculate an uncertainty on the identification. We are working on the details of these hybrid approaches as ongoing research.

7. Conclusions

We described a methodology for constructing datacenter "fingerprints" using statistical techniques. The goal of these fingerprints is to provide the basis for automatic classification and identification of performance crises in a datacenter. Different from root-cause diagnosis, identification facilitates rapid online repair of service disruptions, allowing potentially time-consuming diagnosis to occur offline later. The goal of rapid recovery is consistent with statements by leading Web application operators that today's 24x7 Web and cloud computing services require total downtime to be limited to 50–56 minutes per year[6].

Under realistic conditions and using real data and very stringent accuracy criteria, our approach provides operators the information necessary to initiate recovery actions with 80% correctness in an average of 10 minutes, which is 50 minutes earlier than the deadline provided to us by the operators. Indeed, our criteria may be more stringent that required in practice, since operators may just want to see a list of candidate crises most similar to the current one.

Furthermore, the visualizations of the fingerprints themselves are readily interpretable by human operators: when we showed a few of these fingerprints to the application operators, they very quickly recognized most of the corresponding crises, even though they are not experts in machine learning. Interpretability and gaining operator trust are important for any machine learning technique that will be used in an advisory mode in a production installation; we are now working with the operators on such a live deployment.

References

[1] HP OpenView, welcome.hp.com/country/us/en/ prodserv/software.html.

[2] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using magpie for request extraction and workload modelling. In *OSDI'04: Proceedings of the 6th conference on Symposium*

[6] Marvin Theimer, senior principal engineer, Amazon Web Services; keynote at LADIS 2009 workshop.

on Opearting Systems Design & Implementation, pages 18–18, Berkeley, CA, USA, 2004. USENIX Association.

[3] P. Bodík, A. Fox, M. I. Jordan, D. Patterson, A. Banerjee, R. Jagannathan, T. Su, S. Tenginakai, B. Turner, and J. Ingalls. Advanced tools for operators at Amazon.com. In *Hot Topics in Autonomic Computing (HotAC)*, 2006.

[4] P. Bodík, M. Goldszmidt, and A. Fox. Hilighter: Automatically building robust signatures of performance behavior for small- and large-scale systems. In A. Fox and S. Basu, editors, *SysML*. USENIX Association, 2008.

[5] M. Y. Chen, E. Kıcıman, A. Accardi, E. A. Brewer, D. Patterson, and A. Fox. Path-based failure and evolution management. In *Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI'04)*, San Francisco, CA, March 2004.

[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. 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2004)*, San Francisco, CA, Dec 2004.

[7] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox. Capturing, indexing, clustering, and retrieving system history. In A. Herbert and K. P. Birman, editors, *SOSP*, pages 105–118. ACM, 2005.

[8] B. Cook, S. Babu, G. Candea, and S. Duan. Toward Self-Healing Multitier Services. 2007.

[9] S. Duan and S. Babu. Guided problem diagnosis through active learning. In *ICAC '08: Proceedings of the 2008 International Conference on Autonomic Computing*, pages 45–54, Washington, DC, USA, 2008. IEEE Computer Society.

[10] K. Glerum, K. Kinshumann, S. Greenberg, G. Aul, V. Orgovan, G. Nichols, D. Grant, G. Loihle, , and G. Hunt. Debugging in the (very) large: Ten years of implementation and experience. In *22nd ACM Symposium on Operating Systems Principles (SOSP 2009)*, Big Sky, Montana, Oct 2009.

[11] M. Goldszmidt, I. Cohen, S. Zhang, and A. Fox. Three research challenges at the intersection of machine learning, statistical inference, and systems. In *Proc. Tenth Workshop on Hot Topics in Operating Systems (HotOS-X)*, Santa Fe, NM, June 2005.

[12] S. Guha and A. McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal on Computing*, 38(5):2044–2059, 2009.

[13] K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale L1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

[14] N. Lachiche and P. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using roc curves. In *20th International Conference on Machine Learning (ICML03)*, 2003.

[15] M. Massie. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, July 2004.

[16] D. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupamn, and N. Treuhaft. Recovery oriented computing (roc): Motivation, definition, techniques, and case studies. Technical report, UC Berkeley, March 2002.

[17] S. Pertet, R. Gandhi, and P. Narasimhan. Fingerpointing correlated failures in replicated systems. In *SYSML'07: Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques*, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.

[18] J. A. Redstone, M. M. Swift, , and B. N. Bershad. Using computers to diagnose computer problems. In *9th Workshop on Hot Topics in Operating Systems (HotOS-IX)*, Elmau, Germany, 2003.

[19] P. Reynolds, J. L. Wiener, J. C. Mogul, M. A. Shah, C. Killian, and A. Vahdat. Experiences with pip: finding unexpected behavior in distributed systems. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 1–2, New York, NY, USA, 2005. ACM.

[20] D. Woodard and M. Goldszmidt. Model-based clustering for online crisis identification in distributed computing. Technical report, Microsoft Research, 2009.

[21] S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High speed and robust event correlation. *Communications Magazine, IEEE*, 34(5):82–90, 1996.

[22] M. Young and P. T. Hastie. L1 regularization path algorithm for generalized linear models, 2006.

[23] C. Yuan, N. L. J.-R. Wen, J. Li, Z. Zhang, Y.-M. Wang, and W.-Y. Ma. Automated known problem diagnosis with event traces. In *EuroSys 2006*, Leuven, Belgium, April 2006.

[24] J. Zhang, Z. Ghahramani, and Y. Yang. A probabilistic model for online document clustering with application to novelty detection. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1617–1624. MIT Press, Cambridge, MA, 2005.

[25] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, and A. Fox. Ensembles of models for automated diagnosis of system performance problems. In *2005 Intl. Conf. on Dependable Systems and Networks (DSN 2005)*, Yokohama, Japan, June 2005.