Modeling Probabilistic Measurement Correlations for Problem Determination in Large-Scale Distributed Systems

Jing Gao[†] Guofei Jiang[‡] Haifeng Chen[‡] Jiawei Han[†]
[†]University of Illinois at Urbana-Champaign
[‡]NEC Labs America
[†]{jinggao3,hanj}@illinois.edu, [‡]{gfj,haifeng}@nec-labs.com

Abstract

With the growing complexity in computer systems, it has been a real challenge to detect and diagnose problems in today's large-scale distributed systems. Usually, the correlations between measurements collected across the distributed system contain rich information about the system behaviors, and thus a reasonable model to describe such correlations is crucially important in detecting and locating system problems. In this paper, we propose a transition probability model based on markov properties to characterize pairwise measurement correlations. The proposed method can discover both the spatial (across system measurements) and temporal (across observation time) correlations, and thus such a model can successfully represent the system normal profiles. Problem determination and localization under this framework is fast and convenient. The framework is general enough to discover any types of correlations (e.g. linear or non-linear). Also, model updating, system problem detection and diagnosis can be conducted effectively and efficiently. Experimental results show that, the proposed method can detect the anomalous events and locate the problematic sources by analyzing the real monitoring data collected from three companies' infrastructures.

1. Introduction

Recent years have witnessed the rapid growth of complexity in large-scale information systems. For example, the systems underlying Internet services are integrated with thousands of machines, and thus possess unprecedented capacity to process large volume of transactions. Therefore, large amount of system measurements (metrics) can be collected from software log files, system audit events and network traffic statistics. To provide reliable services, system administrators have to monitor and track the operational status of their infrastructures in real time and fix any problems quickly. Due to the scale and complexity of the system, we have to automate the problem determination process so as to reduce the Mean Time to Recovery (MTTR). It is a challenging task to automatically detect anomalies in a large system because both the normal and anomalous behaviors are heterogeneous and dynamic. In fact, the

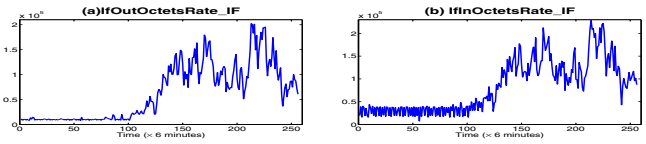

Figure 1. Measurements as Time Series.

widely existing correlations among measurements are very useful for autonomic system management. In this paper, we propose a novel method that can effectively characterize the correlations across different system measurements and observation time. The method captures the complicated and changing normal profiles, and thus can be used to quickly detect and locate system problems.

Each distributed system usually consists of thousands of components, such as operating systems, databases, and application softwares. On each component, we are interested in its usage parameters, such as CPU and memory utilization, free disk space, I/O throughput and so on. Suppose we monitor l measurements for a particular system, and each measurement m^a $(1 \leq a \leq l)$ is uniquely defined by the component (e.g. database) and the metric (e.g. memory usage). Due to the dynamic nature of workloads received by the system, the measurement values usually change with time. Therefore, each measurement m^a can be viewed as a time series. We call the set of time series collected from the system as the monitoring data. Correlations are commonly found among the measurements because some outside factors, such as work loads and number of user requests, may affect them simultaneously. For example, the two measurements shown in Figure 1 are correlated.

For the purpose of problem determination, it is essential to check the correlations among measurements instead of monitoring each measurement individually. A sudden increase in the values of a single measurement may not indicate a problem, as shown by the peaks in Figure 1(a) and Figure 1(b), instead, it could be caused by a flood of user requests. Monitoring multiple measurements simultaneously, we can identify this scenario as normal when we find that many measurements values increase but their correlations remain unchanged. Therefore, profiling

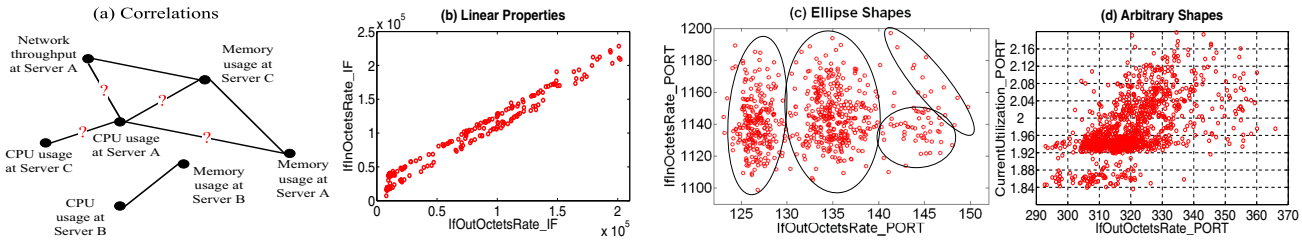

Figure 2. Measurement Correlations: Pair-wise measurement correlations can be shown in two-dimensional space.

measurement correlations can help find the "real" problems and reduce "false positives". We are especially interested in tracking the pair-wise correlations, i.e., the correlations between any two measurements because it can assist quick problem localization. In Figure 2(a), we illustrate pair-wise correlations using a graph where each node represents a measurement and an edge indicates the correlation. At a certain time point, if all the links leading to a measurement m^a have certain problems, the system administrator can directly locate the problem source, i.e., m^a. The pair-wise correlations can be roughly divided into linear and non-linear categories. To observe the correlations more clearly, we extract the values of two measurements m^1 and m^2 at each time point t, and plot (m_t^1, m_t^2) as a point in the two-dimensional space. Figure 2(b)-(c) shows the measurement values extracted from real systems. Clearly, measurements in Figure 2(b) (the rate of traffic goes in and out the same machine), exhibit linear correlations, and Figure 2(c) (in and out traffic rate on two different machines), and Figure 2(d) (PORT throughput and utilization) demonstrate the non-linear relationships. In real monitoring data, we find that nearly half of the measurements have linear relationships with at least one of the other measurements, but the other half only have non-linear ones. Therefore, to model the behavior of the whole system, we need analysis tools that can identify both types of correlations.

Some efforts have been devoted to model the linear measurement correlations in distributed systems (1; 2). Specifically, linear regression models are used to characterize the correlations, such as the one in Figure 2(b). Once the extracted linear relationship is broken, an alarm is flagged. In (3), the authors assume that the two-dimensional data points come from a Gaussian Mixture and use ellipses to model the data "clusters", so the points falling out of the cluster boundaries are considered anomalous events, as shown in Figure 2(c). Despite these efforts, there are many problems that restrict the use of the correlation profiling tools in real systems. First, existing work only focuses on one type of correlations, and thus cannot characterize the whole system precisely. Secondly, the assumption on the form of the data points may not be true (e.g., linear relationships or ellipse-shape clusters). For example, in Figure 2(d), the data points form arbitrary shapes and cannot be modeled by existing methods. Most importantly, how the data evolve is an important part of the system behavior, so besides spatial correlations, correlations across observation time should also be taken into consideration.

In light of these challenges, we propose a grid-based transition probability model to characterize correlations between any two measurements in a distributed system. As shown in Figure 2(d), we partition the space into a number of non-overlapping grid cells and map the data points into corresponding cells. A transition probability matrix is then defined over the two-dimensional grid structure where each entry V_{ij} corresponds to the probability of transitions from grid cell v_i to v_j. We initialize both the grid structure and the transition probability matrix from a snapshot of history monitoring data, e.g., collected from last month, and adapt them online to the distribution changes. We then propose a fitness score to evaluate how well one or all the measurements are described by the correlation models. Once the fitness score drops below a threshold, it indicates that certain system problems may occur. Our contributions are: 1) We propose a novel probability model to characterize both spatial and temporal correlations among measurements from a distributed system. Based on the model, we develop methods to detect and locate system problems. 2) We make no assumptions on the type of correlations and data distributions, therefore, the proposed framework is general and can capture the normal behaviors of the entire distributed system. Also, the model is easy to interpret and can assist later human debugging. 3) We demonstrate the proposed approach's ability of system problem detection and diagnosis by experimenting on one month's real monitoring data collected from three companies' IT infrastructures. We discuss the related work in Section 2, and present the probability model in Section 3. Section 4 and Section 5 introduces how to compute and use the model. In Section 6 and Section 7, we discuss experimental results and conclusions.

2. Related Work

Due to the increase in complexity and scale of the current systems, it becomes important to utilize the measurement correlation information in system logs for autonomic system management. Methods are developed to model correlations

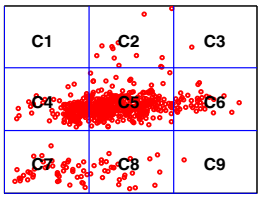

Figure 3. Grid Structure

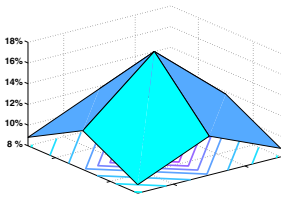

Figure 4. Transitions from c_5

Figure 5. Transition Probability Matrix

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
c_1	21.98%	14.65%	8.79%	14.65%	10.99%	7.33%	8.79%	7.33%	5.49%
c_2	13.16%	19.74%	13.16%	9.87%	13.16%	9.87%	6.58%	7.89%	6.58%
c_3	8.79%	14.65%	21.98%	7.33%	10.99%	14.65%	5.49%	7.33%	8.79%
c_4	13.16%	9.87%	6.58%	19.74%	13.16%	7.89%	13.16%	9.87%	6.58%
c_5	8.82%	11.76%	8.82%	11.76%	17.65%	11.76%	8.82%	11.76%	8.82%
c_6	6.58%	9.87%	13.16%	7.89%	13.16%	19.74%	6.58%	9.87%	13.16%
c_7	8.79%	7.33%	5.49%	14.65%	10.99%	7.33%	21.98%	14.65%	8.79%
c_8	6.58%	7.89%	6.58%	9.87%	13.16%	9.87%	13.16%	19.74%	13.16%
c_9	5.49%	7.33%	8.79%	7.33%	10.99%	14.65%	8.79%	14.65%	21.98%

of request failures (4), or among server response time (5). Correlating monitoring data across complex systems has been studied recently, when algorithms are developed to extract system performance invariants (1; 2) and describe the non-linear correlations (3). Our proposed method distinguishes itself from the above methods by modeling both spatial and temporal correlations among measurements. In markov model based failure prediction methods (6), the temporal information is taken into consideration, but they require the event-driven sources, such as system errors as input. Conversely, our method does not require any knowledge about the system states. The problem of anomaly detection has been extensively studied in several research fields. Particularly, many algorithms have been developed to identify faults or intrusions in Internet (7) or wireless network (8) by examining network traffic data. Different from the above methods, our approach models the data evolution instead of static data points, and thus detects outliers from both spatial and temporal perspectives. In the proposed framework, we partition the two-dimensional data space into grid cells. The idea of space partitioning is motivated by grid-based clustering algorithms (9). The term "grid" refers to the resulting discretized space, thus carries a completely different meaning from that in "grid computing".

3. Transition Probability Model

At time t, the values of two system measurements m^1 and m^2 can be regarded as a two-dimensional feature vector $\mathbf{x}_t = (m_t^1, m_t^2)$. Then the task is to build a model M based on the incoming data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots$ to describe the correlations. Suppose $\mathbf{x}$ is drawn from $\mathcal{S} = \mathcal{A}^1 \times \mathcal{A}^2$, a 2-dimensional bounded numerical space. We partition the space $\mathcal{S}$ into a grid consisting of non-overlapping rectangular *cells*. We first partition each of the two dimensions into intervals. A *cell* is the intersection of intervals from

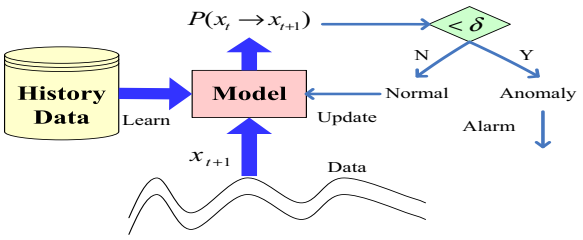

Figure 6. The Framework of Correlation Modeling

the two dimensions, having a form $c = (v^1, v^2)$, where $v^a = [l^a, u^a)$ is one interval of $\mathcal{A}^a$ ($a \in \{1, 2\}$). A data point $\mathbf{x} = (m^1, m^2)$ is contained in the cell c if $l^a \leq m^a < u^a$ for $a = 1$ and $a = 2$. If $\mathcal{A}^1$ and $\mathcal{A}^2$ are partitioned into s^1 and s^2 intervals, there are altogether $s = s^1 \times s^2$ cells. The collection of all the non-overlapping rectangular cells is called *grid structure*: $\mathcal{G} = \{c_1, c_2, \ldots, c_s\}$.

We define the probability of having a new observation $\mathbf{x}_{t+1}$ based on $\mathcal{G}$. To simplify the problem, we assume that the future observation is only dependent on current value and not on any past ones (markov property), i.e., $P(\mathbf{x}_{t+1}|\mathbf{x}_t, \ldots, \mathbf{x}_1) = P(\mathbf{x}_{t+1}|\mathbf{x}_t)$. The experimental results in Section 6 show that this assumption works well in practice. Suppose $\mathbf{x}_{t+1} \in c_j$ and $\mathbf{x}_t \in c_i$, we then approximate $P(\mathbf{x}_{t+1}|\mathbf{x}_t)$ using $P(\mathbf{x}_{t+1} \in c_j|\mathbf{x}_t \in c_i)$, which is the probability of $\mathbf{x}_{t+1}$ falling into cell c_j when $\mathbf{x}_t$ belongs to cell c_i ($c_i, c_j \in \mathcal{G}$). To facilitate later discussions, we use $P(\mathbf{x}_t \rightarrow \mathbf{x}_{t+1})$ to denote $P(\mathbf{x}_{t+1}|\mathbf{x}_t)$, and use $P(c_i \rightarrow c_j)$ to denote $P(\mathbf{x}_{t+1} \in c_j|\mathbf{x}_t \in c_i)$. Since c_i and c_j are drawn from the collection of grid cells $\mathcal{G} = \{c_1, c_2, \ldots, c_s\}$, we can define a s by s matrix V where $V_{ij} = P(c_i \rightarrow c_j)$. Row i ($1 \leq i \leq s$) of the matrix V defines a discrete probability distribution $P(c_i \rightarrow c_j)$ ($\sum_{j=1}^{s} P(c_i \rightarrow c_j) = 1$) for the transitions from c_i to any cell in the grid ($c_j \in \mathcal{G}$). A snapshot of monitoring data from two measurements is plotted in Figure 3. The feature space is partitioned into nine grid cells: $c_1, c_2, \ldots, c_9$, and in Figure 5, we show an example probability matrix $V_{9\times9}$. Suppose $\mathbf{x}_t$ is contained in cell c_5, the discrete probability distribution of $\mathbf{x}_{t+1}$ given $\mathbf{x}_t$ is then characterized by $V_{51}, V_{52}, \ldots, V_{59}$. As shown in Figure 4, higher probability on c_j indicates that $\mathbf{x}_{t+1}$ is more likely to jump to c_j when its original location is c_5.

Therefore, the model to characterize the pair-wise correlations consists of the grid structure and the probability matrix: $M = (\mathcal{G}, V)$. In Section 4, we discuss the methods to initialize and update the model. In section 5, we describe how to use the model to determine system problems.

4. Model Computation

The framework of learning and updating the correlation probability model for problem determination is depicted graphically in Figure 6. We first initialize the model from a set of history data. The model is then put into use on

the continuously flowing monitoring data. Based on the observed $\mathbf{x}_{t+1}$ and $\mathbf{x}_t$, the model outputs $P(\mathbf{x}_t \rightarrow \mathbf{x}_{t+1})$ and if it is below a certain threshold δ, an alarm is flagged. We update the model to incorporate the actual transition made by $\mathbf{x}_{t+1}$ if it is normal. Since the model is comprised of grid structure $\mathcal{G}$ and probability matrix V, we present the learning algorithms for both of them as follows.

4.1. Grid Structure

Initialization. Based on a set of history data $\{\mathbf{x}_t\}_{t=1}^n$, we seek to design a grid structure $\mathcal{G}$, defined by a set of grid cells $\{c_1, c_2, \ldots, c_s\}$. Each cell is represented by a rectangle in the two-dimensional space. We compute the grid cells by setting their boundaries on the two dimensions separately. Formally, each cell c is defined as the intersection of any interval from each of the two dimensions, and the grid structure is thus represented by $\{(v_i^1, v_j^2)\}_{i=1,j=1}^{s^1,s^2}$, where v_i^1 and v_j^2 are intervals of A^1 and A^2 respectively.

Now the problem is: For data mapped onto one dimension a: $X^a = \{x_1^a, x_2^a, \ldots, x_n^a\}$, we wish to discretize $\mathcal{A}^a$ into s^a intervals to hold all the data points. We would compute transition probabilities based on the grid structure, so it should reflect the data distribution. Also, the computation needs to be efficient since multiple pairs of measurements may be watched. Therefore, we propose an efficient approach to partition each dimension into intervals adaptive to the data distribution based on MAFIA (9), a clustering method. We first get the upper and lower bound l^a and u^a from X^a and divide $[l^a, u^a)$ into small equal-sized units with unit length z^a. Note that z^a is much smaller than the actual interval size of the grid structure. We count the number of points falling into each unit. Adjacent units are then merged to form an interval if their counts are similar with respect to a threshold, or are both below a density threshold. The basic idea behind this is to represent the dense areas using more cells, and regions with similar probability densities can be represented using one cell because they may have similar transition patterns. If the data are equal-distributed, we ignore the above procedure and simply divide the dimension into equal-sized intervals. We run the above procedure for each dimension and obtain all the cells by intersecting intervals of the two dimensions. Figure 7 shows an example of the history data and the grid structure the algorithm generates for the data.

Update. During the online process, most of the time, a new observation $\mathbf{x}_{t+1}$ falls into one of the cells defined by the grid structure $\mathcal{G}$. However, it is likely that $\mathbf{x}_{t+1}$ is out of the boundary defined by $\mathcal{G}$. Then either $\mathbf{x}_{t+1}$ is an outlier, or the underlying distribution has changed. We wish to ignore the outliers, but only adapt the grid structure according to the distribution evolution. However, it is challenging to distinguish between the two cases in a real-time manner. We observe that real data usually evolve gradually, thus

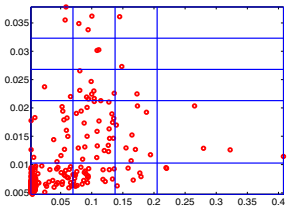

Figure 7. Initial Grid

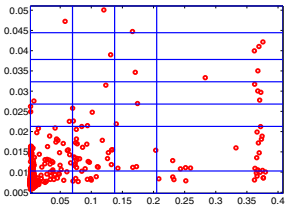

Figure 8. Updated Grid

we assume that the boundary of the grid structure is also changing gradually. Therefore, when $\mathbf{x}_{t+1}$ is not contained in any cells of $\mathcal{G}$, we only update $\mathcal{G}$ if $\mathbf{x}_{t+1}$ is *close enough* to the grid boundary. For each dimension A^a, we compute the average interval size r_{avg}^a offline during initialization and suppose the upper bound of $\mathcal{G}$ on dimension A^a is u^a. When $x_{t+1}^a > u^a$ for $a = 1$ or 2, we first judge if $x_{t+1}^a \leq u^a + \lambda^a \cdot r_{avg}^a$, where λ^a is a parameter indicating the maximum number of intervals to be added. If it holds true, we take it as a signal of potential distribution evolution and add intervals to the dimension until $\mathbf{x}_{t+1}$ is contained within the boundary. New cells are incorporated into $\mathcal{G}$ as the intersections of the added intervals and the intervals from the other dimension. Note that we do not delete cells having sparse densities to maintain the rectangular shape of the grid structure for fast computation. Figure 8 shows the online data and the accordingly updated grid structure, whereas the offline structure is illustrated in Figure 7. It can be seen that the data evolve along the vertical axis, and thus two more intervals are added to accommodate such changes.

4.2. Transition Probability Matrix

We seek to compute $P(c_i \rightarrow c_j)$ for any c_i and c_j in $\mathcal{G}$, i.e., the transition probability between any pair of cells. One natural solution is to compute the empirical distribution based on the set of monitoring data D. Specifically, let $P(c_i \rightarrow c_j)$ be the percentage of examples jumping to c_j when it originally stays at c_i. Although the empirical probability can capture most transitions, it may not be accurate on the transitions which are under represented or even unseen in past records. We therefore need to adjust the empirical distribution to make it smooth over the space so that an unseen transition may still have chances to occur in the future. Therefore we introduce a prior into the distribution using the following bayesian analysis technique (10): $P(c_i \rightarrow c_j|D) = \frac{P(D|c_i \rightarrow c_j)P(c_i \rightarrow c_j)}{P(D)}$ where $c_i \rightarrow c_j$ indicates the existence of a transition from cell c_i to c_j and D is monitoring data set. The transitions are assumed to be independent of each other. Also, our aim is to infer $c_i \rightarrow c_j$, so the term $P(D)$ is not relevant and can be omitted:

$$P(c_i \rightarrow c_j|D) \propto P(c_i \rightarrow c_j) \prod_{t=1}^{n-1} P(\mathbf{x}_t \rightarrow \mathbf{x}_{t+1}|c_i \rightarrow c_j) \quad (1)$$

where n is the size of D. The two steps under this bayesian framework include: 1) define a prior distribution for $P(c_i \rightarrow c_j)$ for any c_i and c_j, and 2) update the distribution based on each observed transition from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. After all data points in D are seen, we can obtain the posterior probability $P(c_i \rightarrow c_j|D)$. We explain the two steps as follows.

Prior Distribution. Bayesian methods view the transition from c_i to c_j as a random variable having a prior distribution. Observation of the monitoring data converts this to a posterior distribution. When a transition is seldom or never seen in the data, the prior will play an important role. Therefore, the prior should reflect our knowledge of the possible transitions. The question is, given $\mathbf{x}_t \in c_i$, which cell is the most probable of containing $\mathbf{x}_{t+1}$? With respect to our assumption that the monitoring data evolve gradually, the transition would have the "*spatial closeness tendency*", i.e., the transitions between nearby cells are more probable than those between cells far away. To support this claim, we check the number of transitions with respect to the cell distance in two days' measurement values. We find that the total number of transitions is 701, among which 412 occurs inside the cells, i.e., the data points would simply stay inside a certain cell. There are 280 transitions between a cell and its closest neighbor. As the cell distance increases, it becomes less likely that points move among these cells. Therefore, the "spatial closeness tendency" assumption is valid. Based on this finding, we define the prior distribution as $P(c_i \rightarrow c_j) \propto \frac{P(c_i \rightarrow c_i)}{w^{d(c_i,c_j)}}$ where $d(c_i, c_j)$ is the distance between c_i and c_j, and w is the rate of probability decrease. If we observe that $\mathbf{x}_t$ belongs to c_i, it is most likely that $\mathbf{x}_{t+1}$ stays at c_i as well. We set $P(c_i \rightarrow c_i)$ to be the highest and as c_j departs further away from c_i, $P(c_i \rightarrow c_j)$ decreases exponentially. From the definition and the constraints that $\sum_{j=1}^{s} P(c_i \rightarrow c_j) = 1$, the prior probability of having transitions from c_i to any cell can be computed. An example prior distribution of transiting from cell c_{12} to other cells is shown in Figure 9. It can be seen that the transition probability at c_{12} is the highest, followed by the probability of transitions to its closest neighbors.

Distribution Updates. According to Eq. (1), to update the prior distribution, we need to multiply it by $P(\mathbf{x}_t \rightarrow \mathbf{x}_{t+1}|c_i \rightarrow c_j)$. If x_{t+1} in fact falls into c_h, we should set $P(x_t \rightarrow x_{t+1}|c_i \rightarrow c_h)$ to be the highest among all the pairs of cells. Also, due to the "spatial closeness tendency", it is likely that a future transition can occur from c_i to c_h's neighbors. Again, we assume an exponential decrease in the transition probability with respect to the cell distance and use the following update rule:

$$P(\mathbf{x}_t \rightarrow \mathbf{x}_{t+1}|c_i \rightarrow c_j) \propto \frac{P(\mathbf{x}_t \rightarrow \mathbf{x}_{t+1}|c_i \rightarrow c_h)}{w^{d(c_h,c_j)}} \quad \text{if } \mathbf{x}_{t+1} \in c_h \text{ and } \mathbf{x}_t \in c_i \tag{2}$$

On Eq. (1), we take log over all the probabilities, and the updates can be performed using additive operations.

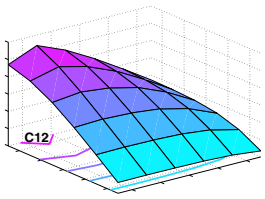

Figure 9. Initial Transitions

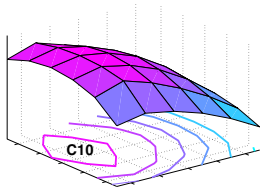

Figure 10. Updated Transitions

Note that we update the transition probability only on normal points, but not on outliers with zero probability. The updating equation is applied on the i-th row of the transition probability matrix where c_i is the cell $\mathbf{x}_t$ belongs to. Note that the updating procedure starts from x_1 where $P(c_i \rightarrow c_j|x_1)$ is assumed to be the prior: $P(c_i \rightarrow c_j)$, and is repeatedly executed for $i = 2, \ldots, n-1$. The prior distribution shown in Figure 9 is updated using six days' monitoring data and the posterior probability distribution on cell c_{12} is depicted in Figure 10. The prior probability of going from c_{12} to c_{12} is the highest, but it turns out that many transitions from c_{12} to c_{10} are observed, so the probability at c_{10} is the highest in the posterior.

5. Problem Determination and Localization

In this section, we discuss how to determine problems in a distributed system with l measurements available. Since we build pair-wise correlation models for any two measurements, we have $l(l-1)/2$ models to characterize all the correlations within the whole system. We propose a *fitness score* as an indicator for the probability of having system problems, which is defined at the following three levels and measures how well the models fit the monitoring data.

1) ***Each pair of measurements at a given time***: For a pair of measurements m^a and m^b at time $t+1$, suppose the most updated model derived from the monitoring data from time 1 to t is $M_{t+1}^{a,b}$. $\mathbf{x}$ represents the two dimensional feature vector consisting of measurement values from m^a and m^b. Suppose $\mathbf{x}_t$ falls into cell c_i. At time $t+1$, the model $M_{t+1}^{a,b}$ outputs the transition probability from c_i to any cell c_j in the grid $(1 \leq j \leq s)$. We define a ranking function $\pi(c_j) : \pi(c_j) < \pi(c_k)$ if $P(c_i \rightarrow c_j) > P(c_i \rightarrow c_k)$. In other words, c_j would be ranked higher if the probability of going from c_i to c_j is higher. We then define the fitness score as $Q_{t+1}^{a,b} = 1 - \frac{\pi_{M_{t+1}^{a,b}}(c_h)-1}{s_{M_{t+1}^{a,b}}}$ where c_h is the cell $\mathbf{x}_{t+1}$ actually belongs to, and $s_{M_{t+1}^{a,b}}$ is the number of grid cells in model $M_{t+1}^{a,b}$. Outliers that lie outside the grid have transition probability 0, and thus their fitness scores are 0 as well. Figure 11 illustrates the fitness score computation through an example. Suppose $\mathbf{x}_t$ is contained in cell c_4 and the transition probability from c_4 to other cells is shown in the left part of the figure. If $\mathbf{x}_{t+1}$ is in cell c_5, we first sort

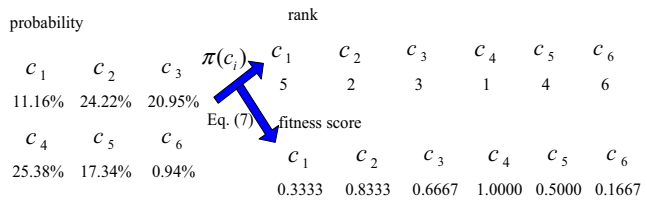

Figure 11. Fitness Score Computation

the cells according to the transition probability and c_5 is ranked at the 4-th place. Then to compute the fitness score, we have $\pi_{M_{t+1}^{a,b}} = 4$ and $s_{M_{t+1}^{a,b}} = 6$, so the result is 0.5. To examine the effect of fitness scores, we repeat the above procedure for the other cells and the results are shown in Figure 11. As can be seen, the fitness score Q measures the fitness of model $M_{t+1}^{a,b}$ on the observed monitoring data. 2) ***Each measurement at a given time***: For a measurement $m^a (1 \leq a \leq l)$, we can derive $l-1$ different models, each of which characterizes the correlations between m^a and m^b $(b = 1, \ldots, a-1, a+1, \ldots, l)$. At time $t+1$, the fitness score for m^a is computed as: $Q_{t+1}^a = \frac{\sum_{b \neq a} Q_{t+1}^{a,b}}{l-1}$ where $Q_{t+1}^{a,b}$ is the fitness score for the model built upon m^a and another measurement m^b. The fitness score of a single measurement is determined by the fitness of correlation models constructed for its links to all the other measurements. 3) ***At a given time***: We aggregate the scores from l measurements into one score Q_{t+1}, which can be used to judge if there are any problems in the entire system at time $t+1$. Again, this can be achieved by averaging the fitness scores of all the measurements.

At the finest level, $Q_{t+1}^{a,b}$ only evaluates the correlation model between two measurements (e.g., one link in Figure 2(a), such as the link between "CPU Usage at Server A" and "Memory Usage at Server C"). Q_{t+1}^a is the aggregation of $Q_{t+1}^{a,b}$, i.e., examining the $l-1$ links leading to one node. For example, the fitness score for measurement "CPU Usage at Server A" is computed based on all its links. Q_{t+1} works for the entire system by aggregating all the fitness scores (e.g., all the links in Figure 2(a)). In general, this evaluation framework can provide different granularity in the data analysis for system management. For less important system components, we may merge their fitness scores so that the system administrators can monitor a single score for system-wide problems. If the average score deviates from the normal state, the administrators can drill down to Q_{t+1}^a or even $Q_{t+1}^{a,b}$ to locate the specific components where system errors occur. We can expect a high fitness score when the monitoring data can be well explained by the model, whereas anomalies in system performance lead to a low score.

6. Experiments

We demonstrate the effectiveness and efficiency of the proposed method through experiments on a large collection of real monitoring data from three companies' infrasturcture. Due to privacy issue, we cannot reveal their names and will denote them as *A*, *B* and *C* in the following discussions. Each company provides a certain Internet service and has over a hundred servers to support user requests every day. On each server, a wide range of system metrics are monitored that are of interests to system administrators, for example, free memory amount, CPU utilization, I/O throughput, etc. A metric obtained from a machine represents a unique measurement. For example, CPU utilization on machine with IP "x.x.x.x" is one measurement. We expect that correlations exist among measurements from the same machine, as well as across different machines, because the whole system is usually affected by the number of user requests.

For each group, there are roughly 3000 measurements. We select 100 from each group and conduct the experiments on the $3 \times \binom{100}{2}$ pairs of measurements. To test on the difficult cases, we enforce the following selection criteria: 1) The sampling rate should be reasonably high, at least every 6 minutes; 2) The measurements do not have any linear relationships with other measurements; and 3) The measurement should have high variance during the monitoring period. We wish to find out the proposed transition probability model's ability in profiling the system normal behaviors. To achieve this, we sample a training set to simulate history data, and a test set, which can be regarded as online data, from the one month's monitoring data (May 29 to June 27, 2008). We compute a model from the training set and evaluate it on the test set. To examine how the sizes of the training and test set affect the model performance, we construct the following training and test sets and conduct experiments on all the combinations for each of the three groups. ***Training sets***: 1) 1 day (May 29), 2) 8 days (May 29-June 5), and 3) 15 days (May 29-June 12). ***Test sets***: 1) 1 day (June 13), 2) 5 days (June 13-June 17), 3) 9 days (June 13-June 21), and 4) 13 days (June 13-June 25).

Problem Determination. In this part, we assess the performance of the proposed method in system problem determination. The distributed systems in use are usually stable and do not have any critical failures. Therefore, we test our methods on three pairs of system measurements where potential problems occur as identified by the system administrators. Based on these events, we can get some general ideas about the proposed method's effectiveness in problem determination. Figure 12 depicts the fitness scores for three pairs of measurements where the ground-truth problems are found. The test set is one day's monitoring data and the problems are found in the morning (Group A), or in the afternoon (Group B and C). The two measurements are CurrentUtilization_PORT and ifOutOctetsRate_PORT (Group A), ifOutOctetsRate_PORT and ifInOctetsRate_PORT (Group B), and CurrentUtilization_IF and ifOutOctetsRate_IF (Group C). It clearly shows

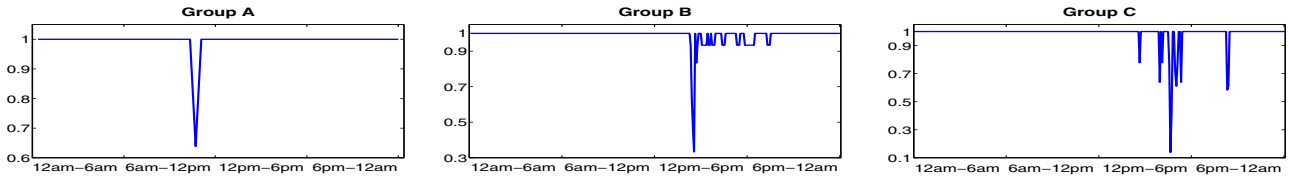

Figure 12. Fitness Scores When System Problems occur

that the anomalies identified by the proposed transition probability method are consistent with the ground-truth in all the three cases. During the period when a problem occurs, we can observe a deep downward spike in the plot of fitness score, which means that this problematic time stamp receives a much lower fitness score compared with normal periods. To provide some intuitive ideas about how the method detects these anomalies, we show the normal and anomalous transitions for the experiments on Group B. From 12am up to 2pm, the values of the two measurements stay within the normal ranges [47.321,22588] & [88.83,34372], however, an anomalous jump to the grid cell [22588,45128] & [102940,137220] is observed, which leads to the downward spike in the fitness score. After that, the measurements fall into either the above normal ranges or [22588,67670] & [34372,51510], which gives a little disturbance to the fitness scores until 8pm. Finally, the measurements go back to their normal values and thus the fitness score stabilizes at 1. Note that we omit the transition probability here, but only give the normal and anomalous transitions to illustrate the basic idea. So the proposed model can help detect the system problems as well as investigate the problem causes.

We also try to identify the specific machine where the problem locates within the whole distributed system. To do so, we compute the average fitness score among measurements collected from the same machine and plot the score distribution across each information system in Figure 14. The locations with low fitness scores are the potential problem sources. Because the monitoring data from the three information systems have different characteristics and distributions, the scales of fitness scores on the three groups are different. We can see that most of the fitness scores are above a certain threshold within each group, which implies that most of the servers are stable and have few problems. There are only a few servers with low average scores, where the system administrators need to check carefully. For example, in Figure 14, there is only 1 machine scoring at below 0.9 in group A, much lower than the scores of other machines. We should pay more attention to this server in future monitoring and analysis.

Offline versus Adaptive. In the following experiments, we show the method's performances on all pairs of measurements by analyzing the fitness scores. As discussed, the real distributed system exhibit normal behaviors most of the time, therefore, a good model should predict the

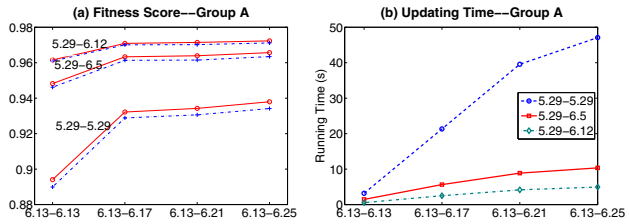

Figure 13. Average Fitness Score and Updating Time

system behaviors well and generate a high average fitness score. First, we compare the following two methods: ***Offline*** methods where the model is derived from the training set offline, and ***Adaptive*** methods where the model is initialized from the training set but updated based on online test set. It would be interesting to see if online model updating can provide additional benefits to the offline model. In Section 5, we show that, at each sampling point, a fitness score Q_{t+1} is computed to reflect the effectiveness of the current model. Therefore, we can evaluate the performances of offline and adaptive methods by averaging the fitness scores computed according to their generated model at each time stamp. When the model is continuously good, the average fitness score would be high. Due to the space limit, we only show the plots for experiments on group A. The experiments on the other two groups have similar patterns. The results are shown in Figure 13(a), where solid and dotted lines represent adaptive and offline methods respectively. It can be seen that the adaptive method usually improves the fitness score over the offline method, especially when the training set is small. When history data are limited, online updating of the model is necessary. But when we have sufficient history data, the models from offline analysis can predict reasonably well on the test set. When the size of the test set increases, we can observe an increase in the fitness scores, which can be explained by the fact that large sample size usually reduces the estimates' variance. Typically, the average fitness score is between 0.8 and 0.98, indicating that the proposed model captures the transitions in monitoring data and is capable of predicting the future.

Updating Time. In this part, we evaluate the adaptive method's efficiency. First, once we have a new observation, we simply determine the grid cell it falls into and look up

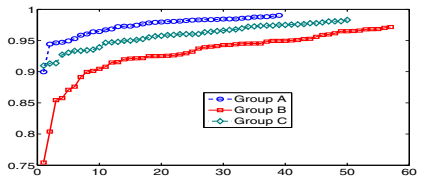

Figure 14. Q Scores w.r.t Locations

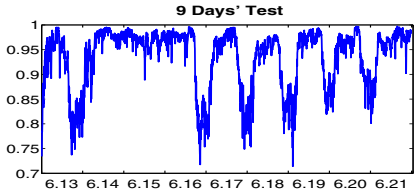

Figure 15. Q Scores for Nine Days

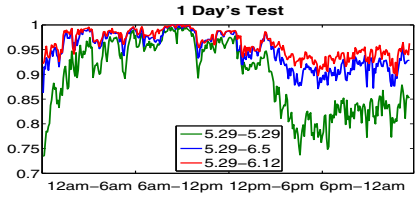

Figure 16. Q Scores for One Day

the transition probability matrix to get the prediction, so the time of applying the model to make predictions is negligible. On the other hand, we have relatively more time to spend for offline analysis. Therefore, the time of updating the model online is the most important part in efficiency analysis. Figure 13(b) shows the online updating time of the adaptive method. When the training samples are sufficient (9 days or 15 days), it costs below 10 seconds to process more than 4,000 monitoring data points, i.e., less than 2.5 milliseconds per sample, much smaller than the sampling frequency (6 minutes). If the period of the training set drops to one day, the updating time increases greatly. Because the history data set do not contain enough examples to initialize the model accurately, the model has to be updated frequently online. However, even in the worst case, the updating time is less than 23 milliseconds per sample. So the proposed method is efficient and can be embedded in online monitoring tools.

Periodic Patterns. The volume of user requests usually affects the system behaviors. Heavier work loads can make the system less predictable. Therefore, when we examine the fitness score at each time stamp Q_{t+1} over a period of 9 days, we find some interesting periodic patterns in Figure 15. We initialize the model using one day's monitoring data, then update and evaluate it on the data from June 13 to June 21. It is obvious that higher fitness scores are obtained during the time when the system is less active including the weekends. At peak hours, the model has lower fitness scores because the system is heavily affected by the large volume of user requests and would be difficult to predict correctly. When more history data are employed in building the initial model, the fitness scores can be improved greatly. To illustrate this, we vary the size of the training set and plot the fitness scores on one day's monitoring data (June 13), shown in Figure 16. When only one day's data are used as training set, the fitness score drops when heavy workloads increase the prediction complexity. But the model initialized from 15 days' history data greatly improves the stability, with a fitness score above 0.9 during both peak and non-peak hours. The results suggest that it is important to incorporate more training samples that share similar properties with the online data to learn the initial model.

7. Conclusions

In this paper, we develop a novel statistical approach to characterize the pair-wise interactions among different components in distributed systems. We discretize the feature space of monitoring data into grid cells and compute the transition probabilities among the cells adaptively according to the monitoring data. Compared with previous system monitoring techniques, the advantages of our approach include: 1) It detects the system problems considering both spatial and temporal information; 2) The model can output the problematic measurement ranges, which are useful for human debugging; and 3) The method is fast and can describe both linear and non-linear correlations. Experiments on monitoring data collected from three real distributed systems involving 100 measurements from around 50 machines, show the effectiveness of the proposed method.

References

[1] G. Jiang, H. Chen, and K. Yoshihira, "Discovering likely invariants of distributed transaction systems for autonomic system management," *Cluster Computing*, vol. 9, no. 4, pp. 385–399, 2006.

[2] M. A. Munawar, M. Jiang, and P. A. S. Ward, "Monitoring multi-tier clustered systems with invariant metric relationships," in *Proc. of SEAMS*, 2008, pp. 73–80.

[3] Z. Guo, G. Jiang, H. Chen, and K. Yoshihira, "Tracking probabilistic correlation of monitoring data for fault detection in complex systems," in *Proc. of DSN*, 2006, pp. 259–268.

[4] M. Chen, E. Kiciman, E. Fratkin., A. Fox, and E. Brewer, "Pinpoint: problem determination in large, dynamic internet services," in *Proc. of DSN*, 2002, pp. 595–604.

[5] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 13–24, 2007.

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

[7] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, "Detection and identification of network anomalies using sketch subspaces," in *Proc. of IMC*, 2006, pp. 147–152.

[8] P. Chhabra, C. Scott, E. Kolaczyk, and M. Crovella, "Distributed spatial anomaly detection," in *Proc. of INFOCOM*, 2008, pp. 1705–1713.

[9] S. Goil, H. Nagesh, and A. Choudhary, "Mafia: Efficient and scalable subspace clustering for very large data sets," in *Technical Report*, Department of Electrical and Computer Engineering, Northwestern University, 1999.

[10] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis* (2nd ed.). Chapman and Hall, 2004.