Eigenspace-based Anomaly Detection in Computer Systems

Tsuyoshi IDÉ*
Tokyo Research Laboratory
IBM Research
goodidea@jp.ibm.com

Hisashi KASHIMA
Tokyo Research Laboratory
IBM Research
hkashima@jp.ibm.com

ABSTRACT

We report on an automated runtime anomaly detection method at the application layer of multi-node computer systems. Although several network management systems are available in the market, none of them have sufficient capabilities to detect faults in multi-tier Web-based systems with redundancy. We model a Web-based system as a weighted graph, where each node represents a "service" and each edge represents a dependency between services. Since the edge weights vary greatly over time, the problem we address is that of anomaly detection from a time sequence of graphs.

In our method, we first extract a feature vector from the adjacency matrix that represents the activities of all of the services. The heart of our method is to use the principal eigenvector of the eigenclusters of the graph. Then we derive a probability distribution for an anomaly measure defined for a time-series of directional data derived from the graph sequence. Given a critical probability, the threshold value is adaptively updated using a novel online algorithm.

We demonstrate that a fault in a Web application can be automatically detected and the faulty services are identified without using detailed knowledge of the behavior of the system.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Management**]: Database applications - Data Mining; K.6.4 [**Management of Computing and Information Systems**]: System Management

General Terms

Algorithms, Management

*The authors address: 1623-14, Shimotsuruma, Yamato, Kanagawa 242-8502, Japan.

KDD'04, August 22–25, 2004, Seattle, Washington, USA.

Keywords

time sequence of graphs, principal eigenvector, Perron-Frobenius theorem, von Mises-Fisher distribution, singular value decomposition

1. INTRODUCTION

1.1 Anomaly detection from graph sequences

Network systems having various connections and correlations between vertices have attracted much attention in several research fields such as ecology, economics, and solid-state physics. In the data mining community, growing attention is being paid to graphs as a new data structure. Recent studies include: an extension of the a-priori algorithm to graphs [13], clustering graph vertices based on graph spectra [5, 4], and anomaly detection from a graph set based on the maximum description length principle [16]. Reference [22] extensively reviews the state of the art of graph-based data mining. Most of those works today, however, assume that the attributes of graphs are static.

On an abstract level, computer systems are also represented as graphs. What is profound here is that, first, it is possible to define various kinds of network structures. For example, one can consider several structures at each layer of the OSI (Open Systems Interconnection) reference model. Second, interactions between vertices or edge weights are not clearly defined at each layer.

In this paper, we address online anomaly detection for computer systems. We model a Web-based system as a weighted graph, where each node represents a "service" and each edge represents a dependency between services. Since edge weights may vary over time, the problem we address is that of anomaly detection from a time sequence of graphs, namely from a time-dependent adjacency matrix. The dependency matrix will exhibit some change when a monitored system experiences a fault. This change, however, will be difficult to detect by monitoring an individual dependency, i.e., each matrix element. This is especially true in Web-based systems, where the number of service calls fluctuates strongly over time. Even if a detector observes a sudden change in a single service, there is no evidence to conclude whether or not it is due to a fault. It may just be a fluctuation in traffic.

This is a new challenge for graph mining, where one needs to discover an unknown structure hidden deep inside of dependency graphs, and detect faults from their anomalous changes. In this paper, we discuss the dynamics of graphs in

the context of data mining. We first extract a feature vector from an adjacency matrix that represents the activities of the services. The heart of our method is to use the principal eigenvector of the eigenclusters of the graph. To a time-series of directional data derived from the graph sequence, we next apply a pattern extraction technique based on a variational principle. Then we introduce a new online algorithm to update a probability distribution for an anomaly measure based on the von Mises-Fisher theory. Using this technique, anomalies can be detected by comparing with given critical probability that is independent of the details of the system.

1.2 Faults in computer systems

The more the importance of information technology in society increases, the more serious the impact of major faults of computer systems becomes. In recent distributed complex systems, some autonomic system management model [6] is needed rather than the traditional model, where a human administrator constantly monitors the system. It is clear that, as the first step, one needs a tool to sense the whole system in a comprehensive manner and to detect any sign of faults in an automated manner. However, there is no practical method for this because of the intrinsic complexity of phenomena in computer networks.

For instance, network node management systems (NNMSs) today have poor fault detection capabilities although they can gather and visualize information distributed in the system, typically through the Simple Network Management Protocol (SNMP). In fact, because SNMP trap events are thrown too frequently in a default configuration and the individual trap events are not necessarily related to actual faults, some administrators often neglect the trap events. As a result, monitoring a console by a human administrator is practically the only solution to detect the sign of faults.

These problems in existing NNMSs are summarized as follows:

- They are capable of gathering detailed information for each node through the SNMP Management Information Base, but they provide poor means to correlate that information.
- Empirically defined threshold values are directly compared with such observables, and anomalies are defined by simple inequality rules. Deep expert knowledge is needed to find such rules.

These problems are crucial, especially in high volume Web systems. For instance, consider a three-tier system including an HTTP (Hyper Text Transfer Protocol) server, a Web application server (WAS), and a database (DB) server. An appropriate description of the interaction between servers at the *application layer* is essential because the servers are connected at this layer as well as at the Transport layer. However, it is known to be difficult to monitor such higher-level correlations with the use of existing NNMSs.

To be more concrete, consider an anomalous situation in a doubly-redundant system with two HTTP servers and two WASs: The activity of one of the WASs suddenly decreases due to some external load. While this trouble clearly breaks the symmetry between the two WASs, there is no fault at layers below the TCP (transport layer protocol) layer and no change will be detected in response time at relatively low traffic levels. A heartbeat monitoring service may show that all of the processes work well. However, this situation is potentially dangerous because an increase in traffic may cause serious problems far before the traffic volume exceeds the rated capacity of the system.

Faults of this kind are difficult to find by monitoring individual servers using the existing NNMSs. What we will propose in this paper is a novel method to detect such faults. To the best of the authors' knowledge, this is the first report that succeeds in automated detection of faults at the application layer.

The rest of this paper is organized as follows: In the next section, we briefly refer to related work. In Section 3, we define the dependency matrix at the application layer and formally state the problem. In Section 4, we describe a new method of feature extraction and show that graph time sequences are reduced to time-series of directional data. In Section 5, we discuss probabilistic properties of the anomaly measure. In Section 6, we report on experimental results in a benchmark system. In the final section, we summarize the major results in this paper.

2. RELATED WORK

Most recent studies on graph mining address issues that are transplanted from traditional problems in data mining. Frequent pattern search [14, 10] and graph classification [12] are such examples. One of the properties that is inherent in graphs is the presence of graph spectra. References [5, 4] address the problem of graph partitioning. There are also studies that discuss time development of graphs for citation graphs [15] or for the World Wide Web [9], both of which employ the concept of graph spectra. In the field of image processing, incident matrices and their eigenvectors are used to characterize the time development of images [17]. While our approach has a part in common with the eigencluster concept in Ref. [17], we address the dynamics of graphs where edge weights are explicitly time-dependent, rather than gradual changes of incident matrices. To the best of the authors' knowledge, there is little research that focuses on highly dynamic systems such as computer networks in the context of graph mining.

In a subsequent section, we will show that a sequence of dependency graphs at the application layer can be transformed into a sequence of directional data items (a time series of normalized vectors). For vector series without normalization, unsupervised anomaly detection techniques are extensively discussed in Refs. [24, 23] based on the normal mixture model (Note that traditional rule-based approaches are not appropriate in this case since the signs of faults are hidden deep inside the systems). However, the normal mixtures are not appropriate to handle directional data. Reference [1] employs a mixture of the von Mises-Fisher distribution, and discusses the connection to the cosine measure. We also consider the von Mises-Fisher distribution. Our contribution is to derive a probability distribution of the anomaly measure itself and to give its online update procedure.

Recent studies on signal processing have demonstrated the utility of change-detection techniques to characterize anomalies of network traffic [2, 21]. However, most of those studies address highly aggregated data at the lower layers. We focus on the problem at the application layer in Web-based systems, where traditional autoregressive models with white noise are not appropriate because of the strong fluc-

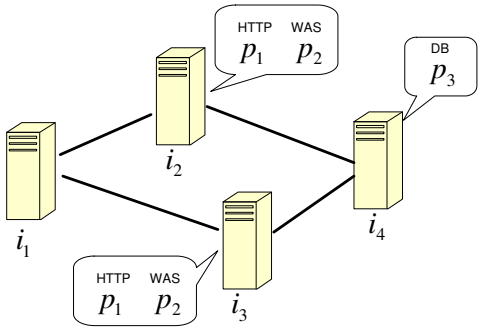

Figure 1: Configuration of benchmark system. IP addresses and port numbers are denoted by i_k ($k = 1, .., 4$) and p_j ($j = 1, 2, 3$), respectively.

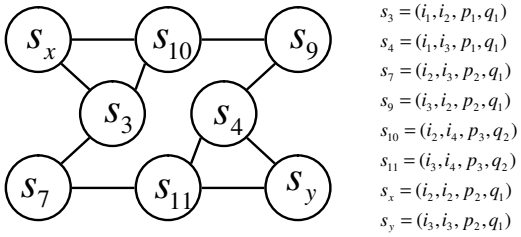

Figure 2: A part of the dependency graph for the system in Fig. 1. Only services which have $Q = q_1$ or q_2 are shown. Graph edges are drawn if $I_s = I_d$ holds between two vertices.

tuation and the heavy tail nature of data. Reference [8] applied the approach of Refs. [24, 23] to a fault detection task in a local-area network. However, it only uses information from a single observation point, and considers only lower layer quantities. Reference [20] proposes an interesting method to correlate multiple observation points. However, it is based on the autoregressive model and its thresholding policy is not probabilistically consistent. Overall, our contribution is to discuss the problem of graph mining in terms of a vector space model, and to give a probabilistically consistent anomaly detection method.

3. DEPENDENCY MATRIX

3.1 Definition

As discussed, we focus on faults occurring in the application layer of Web-based systems. We define a *service* as a quartet of

$$(I_s, I_d, P, Q),$$

where I_s and I_d represent source and destination IP (Internet Protocol) addresses, respectively, and P denotes the port number of the destination application. We also use an attribute called the transaction type Q. Figure 1 illustrates a benchmark system. There are four server boxes in this system, and two server processes with port numbers p_1 and p_2 are installed on each of the boxes at i_2 and i_3. On each of the server processes, multiple applications can be running, and they are distinguished by the value of Q. For example, a service may be defined by (i_1, i_3, p_1, q_1) for i_1=192.168.0.19, i_2=192.168.0.53, p=80, and q_1="Trade". This corresponds to a request to the HTTP server for a transaction type "Trade". Figure 2 shows a subgraph of the dependency graph expected in the system depicted in Fig. 1. We drew links if $I_s = I_d$ holds between two services, and services involving only q_1 and q_2 are shown there. The service type q_1 may be "Trade", and q_2 may be a DB-related service. Generally, the dependency graph of a Web-based system is quite complicated even if the corresponding IP network is simple.

Consider a system with N different services. For the dependencies between services, it is natural to consider the quantity $d_{i,j}$, the number of i's requests for j within a predetermined time interval, as a measure of the dependency. Considering the bursty nature of Web traffic, it is reasonable to use its logarithmically transformed quantity, $\tilde{d}_{i,j} = \ln(1 + d_{i,j})$. Since most transactions in Web-based systems are processed synchronously, a callee returns the control token to its caller after processing a request. Thus, the simplest assumption is that the dependency of a service i on another service j is symmetric:

$$\mathsf{D}_{i,j} = (\tilde{d}_{i,j} + \tilde{d}_{j,i})(1 - \delta_{i,j}) + \alpha_i \delta_{i,j}, \qquad (1)$$

where $\delta_{i,j}$ is Kroneker's delta function and the α_is are constants introduced to stabilize the numerical calculations. By definition, D is a square non-negative matrix. Hereafter, we use a sans serif font to indicate matrices and use bold italic to indicate vectors. The norm of vectors is defined as the L_2-norm.

In principle, the quantity $d_{i,j}$ can be measured through server logs, typically using an API (application program interface) called ARM (application response measurement) [19]. In commercial systems, however, it is more practical to estimate $d_{i,j}$ from some indirect information, since capturing server logs causes additional load. Recently, Gupta *et al.* [7] proposed an algorithm where $d_{i,j}$ is defined as the probability that the duration of a transaction (an instance of a service i) contains the duration of another transaction (an instance of a service j). In such algorithms, $\mathsf{D}_{i,j}$'s are not integer but real numbers including error. In this paper, however, we do not discuss the details of such a method to evaluate the dependency, and concentrate our attention on the online anomaly detection algorithm, given the dependency matrix.

3.2 Problem statement

We consider that the overall behavior of a computer system can be basically characterized by the dependency matrix D. Its dimension N and the definition of each element are assumed to be fixed, but the value of each matrix element strongly varies over time. Considering D as the adjacency matrix of a graph with a fixed structure, the problem we address is described as follows: Given a graph with time-dependent edge weights on a fixed structure, detect anomalies online and identify faulty vertices of the graph without using detailed knowledge on the system behavior.

The main considerations here are that, first, N is so large (on the order of 10^2) that monitoring each matrix element independently is not practical. Second, the behavior of each matrix element strongly fluctuates, and the fluctuation it-

self does not necessarily indicate any anomalies. Anomalous situations are implied by phase transitions in the overall relation between the edge weights. Third, the rule to define the anomalies must be described without ad hoc knowledge specific to the system.

4. FEATURE EXTRACTION FROM GRAPHS

4.1 Definition of activity vector

Let us assume that the data for the dependency matrix D is sequentially obtained at each time t=1,2,... each a fixed interval, and that the dependency graph has a single connected component. We define the feature vector $\boldsymbol{u}$ of D as

$$\boldsymbol{u}(t) \equiv \arg\max_{\tilde{\boldsymbol{u}}} \left\{ \tilde{\boldsymbol{u}}^T \mathsf{D}(t) \tilde{\boldsymbol{u}} \right\} \tag{2}$$

subject to $\tilde{\boldsymbol{u}}^T \tilde{\boldsymbol{u}} = 1$, where T denotes transpose. Since D is a non-negative matrix, one can see that the maximum value is attained if the weight of $\boldsymbol{u}(t)$ is larger for services where $\mathsf{D}_{ij}(t)$ is larger. If a service i actively calls other services, $\boldsymbol{u}(t)$ has a large weight for the i-th element. Following this interpretation, we call this feature vector an *activity vector*.

By introducing a Lagrange multiplier λ, Eq. (2) can be rewritten as

$$\frac{d}{d\tilde{\boldsymbol{u}}} \left[\tilde{\boldsymbol{u}}^T \mathsf{D}(t) \tilde{\boldsymbol{u}} - \lambda \tilde{\boldsymbol{u}}^T \tilde{\boldsymbol{u}} \right] = 0,$$

so that

$$\mathsf{D}(t) \tilde{\boldsymbol{u}} = \lambda \tilde{\boldsymbol{u}}. \tag{3}$$

While this equation holds for any of the eigenvectors of $\mathsf{D}(t)$, the feature vector corresponding to Eq. (2) is defined as the principal eigenvector (the eigenvector whose eigenvalue is the largest). Since Eq. (3) is homogeneous in $\tilde{\boldsymbol{u}}$, the following property holds for the activity vector:

PROPERTY 1. *The direction of $\boldsymbol{u}$ is invariant with respect to the transformation $\mathsf{D}(t) \to k\mathsf{D}(t)$ for any nonzero real number k.*

Thereby we can exclude overall traffic changes from analysis. It is the eigenvalue that is proportional to the global traffic volume. This is important to abstract a hidden structure from D.

To understand the meaning of $\boldsymbol{u}$ further, one can relate $\boldsymbol{u}$ with a stationary state of a discrete-time linear dynamical system whose equation of motion is given by

$$\boldsymbol{x}(\tau + 1) = \mathsf{D}(t) \boldsymbol{x}(\tau),$$

where τ denotes a virtual time being independent of the actual time t, and $\boldsymbol{x}$ is associated with $\boldsymbol{u}$ by $\boldsymbol{u}$=$\boldsymbol{x}/||\boldsymbol{x}||$. Since $\mathsf{D}(t)$ is symmetric and of full-rank at least for $\alpha > 0$, all eigenvalues are real. Using the eigenvalues, $\boldsymbol{x}(0)$ can be expressed as a linear combination of the eigenvectors, so that

$$\boldsymbol{x}(\infty) = \lim_{n\to\infty} [\mathsf{D}(t)]^n \boldsymbol{x}(0) = \lim_{n\to\infty} \sum_{i=1}^{N} [\lambda_i(t)]^n c_i(t) \boldsymbol{u}_i(t),$$

where the eigenvalues and the normalized eigenvectors are denoted by $\lambda_i(t)$ and $\boldsymbol{u}_i(t)$ for i=1, 2, ..., N, respectively, and $c_i(t)$'s are coefficients of the linear combination. Evidently, the term of the maximum eigenvalue becomes dominant as $n \to \infty$. Thus, we have

$$\boldsymbol{u}(t) = \boldsymbol{x}(\infty)/||\boldsymbol{x}(\infty)||.$$

Namely, the state vector approaches $\boldsymbol{u}$ after an infinite number of transitions. For computer systems, the stationary state can be interpreted as the distribution of the probability amplitude that a service is holding the control token of the system at a virtual time point of τ.

4.2 Activity vectors in disconnected systems

In real computer systems, the dependency graph is often disconnected. For such systems, a permutation matrix P exists such that

$$\mathsf{P}^T \mathsf{D} \mathsf{P} = \begin{bmatrix} \mathsf{D}_1 & & 0 \\ & \mathsf{D}_2 & \\ 0 & & \ddots \end{bmatrix},$$

where D_1, D_2, ... are square submatrices. To be concrete, consider the system shown in Fig. 3. Using

$$\mathsf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

the whole dependency matrix is decomposed into two square submatrices:

$$\mathsf{D}_1 = \begin{bmatrix} 0 & a_{15} & a_{13} & 0 \\ a_{15} & 0 & 0 & a_{56} \\ a_{13} & 0 & 0 & a_{36} \\ 0 & a_{56} & a_{36} & 0 \end{bmatrix}, \; \mathsf{D}_2 = \begin{bmatrix} 0 & a_{24} \\ a_{24} & 0 \end{bmatrix}. \tag{4}$$

Evidently, each submatrix corresponds to a connected subgraph. Since the eigenvalue equation is invariant with respect to orthogonal transformations, the whole eigenvalue equation is written as

$$0 = \det \left| \mathsf{D}_1 - \lambda \mathsf{E}_{(4)} \right| \cdot \det \left| \mathsf{D}_2 - \lambda \mathsf{E}_{(2)} \right|,$$

where $\mathsf{E}_{(n)}$ represents the n-dimensional identity matrix. Consequently, the solution of the whole system can be obtained as the union of the solutions of each connected component. This fact allows us to analyze each subgraph separately.

For each connected component, the Perron-Frobenius theorem [3], which holds for non-negative irreducible matrices, guarantees a useful property of the activity vector:

PROPERTY 2. *In each connected component, the principal eigenvector is positive, where an eigenvector is said to be positive if all the components of $\boldsymbol{u}$ or $-\boldsymbol{u}$ are positive and the corresponding eigenvalue is positive.*

This naturally supports the interpretation of the principal eigenvector as the activity vector, since the magnitude of the activities should be positive. For the principal (i.e. maximum) eigenvalue, the following property holds:

PROPERTY 3. *In each connected component, the principal eigenvalue is real* [1] *and has no degeneracy.*

From this, we understand that the activity vector is free from subtle problems due to level crossings of the eigenstates within a single connected component in the normal state of

[1]In this case, all of eigenvalues are real since Eq. (1) makes D real and symmetric.

the system. If a level crossing easily occurs due to small fluctuations, the transition from one eigenstate to another eigenstate may be recognized as an outlier, resulting in a false alert.

For example, we show all of the eigenvectors in Table. 1 for the system shown in Fig. 3, setting a_{13}, a_{15}, a_{36}, a_{56}, and a_{24} to 4, 10, 3, 3, and 1, respectively. As shown, the principal eigenvectors, $\boldsymbol{u}_1$ and $\boldsymbol{u}_3$, are positive, and the collections of nonzero elements correspond to the connected components, $\{1,3,5,6\}$ and $\{2,4\}$. Following Sarker and Boyer [17], we call the collections *eigenclusters*. An eigencluster is said to be principal if it has the largest principal eigenvalue.

The eigencluster concept provides us with a natural way to cluster services. When the set of all services is unknown, it is practically possible to find the activity vectors by choosing positive vectors from a set of eigenvectors [17]. Specifically, for the eigenvectors of the whole system, we understand the following fact:

PROPERTY 4. *In disconnected systems where the principal eigenvalue is not degenerate, the activity vector of the whole system is that of the principal eigencluster.*

We define the activity vector of the whole system by that of the principal cluster.

4.3 Stability of activity vectors

Although the Perron-Frobenius theorem guarantees that the principal eigenvalue is not degenerate within each single component, it says nothing about inter-component degeneracy. To explore the conditions of no degeneracy stated in Property 4, consider the following situation: A weak perturbation F is applied to a system, and it couples an N_1-dimensional eigencluster D_1 with the other N_2-dimensional eigencluster D_2. Namely, at t and $t+1$, the dependency matrices are given as

$$\mathsf{D}(t) = \begin{bmatrix} \mathsf{D}_1 & \mathsf{0} \\ \mathsf{0} & \mathsf{D}_2 \end{bmatrix},\ \mathsf{D}(t+1) = \begin{bmatrix} \mathsf{D}_1 & \mathsf{F} \\ \mathsf{F}^T & \mathsf{D}_2 \end{bmatrix}. \tag{5}$$

Here F is an $N_1 \times N_2$ nonnegative matrix. Let $\hat{\boldsymbol{s}}_1$ and $\hat{\boldsymbol{s}}_2$ be activity vectors of D_1 and D_1, and let λ_1 and λ_2 be corresponding eigenvalues, respectively. In the unperturbed system, N-dimensional vectors $\boldsymbol{s}_1{=}(\hat{\boldsymbol{s}}_1, \mathbf{0})^T$ and $\boldsymbol{s}_2{=}(\mathbf{0}, \hat{\boldsymbol{s}}_2)^T$ are eigenvectors with eigenvalues of λ_1 and λ_2, respectively, where the $\mathbf{0}$s are the zero vectors with appropriate dimensions. To find the activity vector of the perturbed system, let us limit ourselves to the space spanned by $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. Let $\hat{\boldsymbol{u}}$ be an eigenvector in this space. Since $\hat{\boldsymbol{u}}$ is mapped back onto the original space as $\mathsf{K}\hat{\boldsymbol{u}}$, where

$$\mathsf{K} = [\boldsymbol{s}_1, \boldsymbol{s}_2]$$

is the projection matrix, the eigenequation in the contracted space is expressed as

$$\mathsf{K}^T\mathsf{D}(t+1)\mathsf{K}\hat{\boldsymbol{u}} = \lambda\hat{\boldsymbol{u}}.$$

Note that $\mathsf{K}\mathsf{K}^T = \mathsf{E}_{(2)}$ holds. Explicitly, the perturbed matrix is transformed into

$$\mathsf{K}^T\mathsf{D}(t+1)\mathsf{K} = \begin{bmatrix} \lambda_1 & f \\ f & \lambda_2 \end{bmatrix}, \tag{6}$$

where $f = \hat{\boldsymbol{s}}_1^T\mathsf{F}\hat{\boldsymbol{s}}_2$ is a scalar. If the two eigenvalues are very close to each other, any linear combination between $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ can be an eigenvector. Thus, the activity vector of the whole system may make a transition from, e.g., $\boldsymbol{s}_1$ to $\boldsymbol{s}_2$ due to a small perturbation. The activity vector is unstable in this case.

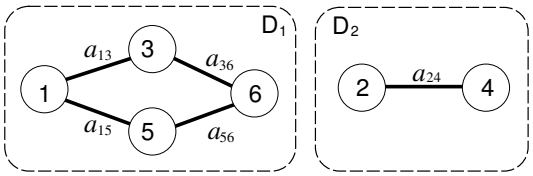

Figure 3: Example of a disconnected graph.

On the other hand, consider the case where $\lambda_1 \gg \lambda_2$ and $\lambda_1 \gg f$ holds. Solving the eigenvalue equation corresponding to Eq. (6), we have

$$\left.\begin{matrix} \lambda_+ \\ \lambda_- \end{matrix}\right\} = \left\{\begin{matrix} \lambda_1 + f^2/\lambda_1 \\ \lambda_2 - f^2/\lambda_1 \end{matrix}\right. , \tag{7}$$

where we neglect higher order terms with respect to f/λ_1 and λ_2/λ_1. The eigenvector corresponding to λ_+ is given by

$$\mathsf{K}\hat{\boldsymbol{u}} \propto \lambda_1\boldsymbol{s}_1 + f\boldsymbol{s}_2. \tag{8}$$

Naturally, this is a positve vector. Equation (7) shows that the perturbation increased the difference of the two levels. The eigenvector of Eq. (8) indicates that $\boldsymbol{s}_1$ is dominant in the whole activity vector as long as f is small. By definition, f is small when the number of edges between the two subsystems is small and/or their edge weights are small. Fortunately, in Web-based systems, important services such as the HTTP server's requests for a WAS are concentrated within the principal eigencluster, and therefore, the condition of $\lambda_1 \gg \lambda_2, f$ is clearly satisfied. Hereafter, we will pay our attention only to the principal eigencluster which is stable against perturbations.

Table 1: Eigenvectors and eigenvalues for the graph shown in Fig. 3. For parameters, see the text.

	$\boldsymbol{u}_1$	$\boldsymbol{u}_2$	$\boldsymbol{u}_3$	$\boldsymbol{u}_4$	$\boldsymbol{u}_5$	$\boldsymbol{u}_6$
1	0.663	0.245	0.	0.	0.245	0.663
2	0.	0.	0.707	−0.707	0.	0.
3	0.295	−0.642	0.	0.	0.642	−0.295
4	0.	0.	0.707	0.707	0.	0.
5	0.642	0.295	0.	0.	−0.295	−0.642
6	0.245	−0.663	0.	0.	−0.663	0.245
λ	11.469	1.570	1.000	−1.000	−1.570	−11.469

4.4 Scalability of our approach

Our feature extraction technique provides a natural way to summarize the information contained in D. The eigencluster decomposition allows us to analyze each single eigencluster separately, and the activity vector extraction technique allows us to further reduce the degrees of freedom. Thus, we expect that the degrees of freedom of each of subproblems are still moderate even when the whole degrees of freedom are very large. In addition, the feature vector has a clear interpretation that is comprehensible to system administrators. Understanding what is happening is as essential as detection itself in practical situations. These are advantages over naive approaches such as defining a feature vector

simply by connecting all of the column vectors, where the scalability cannot be achieved and interpretation of results is often unclear.

For the numerical calculations, an extremely fast and simple algorithm called the power method [18] is known to find the principal eigenvector. While the activity vector must be calculated online whenever D is updated in the given time interval Δt, typically on the order of a few tens of seconds, our experience shows that the time to convergence is far less than Δt even for N on the order of 10^3.

5. ANOMALY DETECTION

5.1 Extraction of typical activity pattern

Now we consider how to detect anomalous changes from the sequence of activity vectors $\{\boldsymbol{u}^{(t)}\}$ for $t = 1, 2, ...$ Since $\boldsymbol{u}^{(t)}$ is normalized, this is a time sequence of *directional data*. The basic procedure is to extract a typical pattern from the past activity vectors, and to calculate the dissimilarity of the present activity vector from this typical one.

We define a matrix $\mathsf{U}(t)$ by

$$\mathsf{U}(t) = [\boldsymbol{u}(t), \boldsymbol{u}(t-1), \cdots, \boldsymbol{u}(t-W+1)],$$

where W is a window size. Clearly, $\mathsf{U}(t)$ is an $N \times W$ matrix. It is natural to suppose that the typical pattern is a linear combination of the column vectors:

$$\boldsymbol{r}(t) = c\sum_{i=1}^{W} v_i \boldsymbol{u}(t-i+1), \tag{9}$$

where c is the normalization constant to satisfy $\boldsymbol{r}^T\boldsymbol{r} = 1$. If the v_is are independent of i, $\boldsymbol{r}(t)$ is parallel to the mean vector. In order to further optimize the coefficients $\boldsymbol{v}^T = (v_1, v_2, ..., v_W)$, we again consider the following extremum principle:

$$\boldsymbol{v}(t) \equiv \arg\max_{\tilde{\boldsymbol{v}}} \left|\left|\sum_{i=1}^{W} \tilde{v}_i \boldsymbol{u}(t-i+1)\right|\right|^2 \tag{10}$$

subject to $\tilde{\boldsymbol{v}}^T\tilde{\boldsymbol{v}} = 1$. This equation implies that the optimal v_is are those that produce the strongest constructive interference between $\boldsymbol{u}$s. Notice that Eq. (9) can be expressed as

$$\boldsymbol{r}(t) = c\mathsf{U}(t)\boldsymbol{v}(t). \tag{11}$$

Then the extremum equation reads

$$\frac{d}{d\tilde{\boldsymbol{v}}}\left[\tilde{\boldsymbol{v}}^T\mathsf{U}(t)^T\mathsf{U}(t)\tilde{\boldsymbol{v}} - \mu\tilde{\boldsymbol{v}}^T\tilde{\boldsymbol{v}}\right] = 0,$$

where we set the Lagrange multiplier to be $c^2\mu$. Hence, we have an eigenequation

$$\left[\mathsf{U}(t)^T\mathsf{U}(t)\right]\tilde{\boldsymbol{v}} = \mu\tilde{\boldsymbol{v}}. \tag{12}$$

While each eigenvector of $\mathsf{U}(t)^T\mathsf{U}(t)$ satisfies this equation, we define the typical pattern as the principal eigenvector. Using Eq. (12) and the normalization conditions, it is easy to show that

$$c = 1/\sqrt{\mu}. \tag{13}$$

Equations (11), (12), and (13) suggest that $\boldsymbol{r}(t)$ *is the principal left singular vector of* $\mathsf{U}(t)$, where a singular vector is said to be principal if it corresponds to the largest singular value. Again, the power method [18] is a good way to perform the singular value decomposition (SVD).

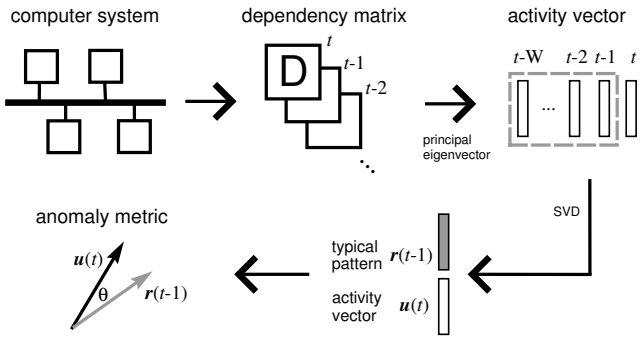

Figure 4: Summary of our anomaly detection procedure.

5.2 Anomaly metric and its probability distribution

To evaluate the dissimilarity, we introduce the quantity $z(t)$, defined as

$$z(t) \equiv 1 - \boldsymbol{r}(t-1)^T\boldsymbol{u}(t). \tag{14}$$

The value of $z(t)$ is unity if the present activity vector is orthogonal to the typical pattern at $t-1$, and zero if the present activity vector is identical to the typical pattern. In the present context, if $z(t)$ is greater than a given threshold, we infer that an anomalous situation is occurring in the system. We summarize our anomaly detection procedure in Fig. 4.

For directional data, the major distribution is the von Mises-Fisher (vMF) distribution [1],

$$p(\boldsymbol{u}) \propto \exp\left[\frac{\cos\theta}{\Sigma}\right], \tag{15}$$

where $||\boldsymbol{u}|| = 1$. The angular variable $\theta \in [0, \pi]$ is defined as the arccosine of the inner product between $\boldsymbol{u}$ and a mean direction. Here, we take $\boldsymbol{r}(t-1)$ as the mean direction at t:

$$\cos\theta = \boldsymbol{r}(t-1)^T\boldsymbol{u}.$$

The value $\Sigma > 0$ is a constant parameter called the angular variance. Intuitively, the vMF distribution describes fluctuations of $\boldsymbol{u}$ around the mean direction. The vMF distribution is the most natural distribution for directional data in that it can be derived using the maximum entropy principle under a general condition. While Ref. [1] discusses the utility of a mixture model of the vMF distribution, the single vMF model is sufficient in the present application.

To derive the marginalized distribution with respect to θ from Eq. (15), we perform a transformation of the variables from $\boldsymbol{u}$ to angular variables $\{\theta, \theta_2, ..., \theta_{N-1}\}$ of the N-dimensional spherical coordinates. By using

$$d^{N-1}\Omega = d\theta d\theta_2 \cdots d\theta_{N-1} \sin^{N-2}\theta \sin^{N-3}\theta_2 \cdots \sin\theta_{N-2},$$

where $d^{N-1}\Omega$ is the area element on the unit sphere in an N-dimensional Euclidean space, the marginalized distribution for θ is written as

$$p(\theta) = \int p(\boldsymbol{u}) \sin^{N-2}\theta \sin^{N-3}\theta_2 \cdots \sin\theta_{N-2} \prod_{i=2}^{N-1} d\theta_i.$$

Since

$$z(t) \simeq \frac{\theta^2}{2},\ \cos\theta \simeq 1 - \frac{\theta^2}{2},\ \sin\theta \simeq \theta$$

hold for $|\theta| \ll 1$, we see that the distribution for z is given by

$$q(z) \propto \exp\left[-\frac{z}{2\Sigma}\right] z^{\frac{N-1}{2}-1},$$

where we used $\theta d\theta = dz$ and reset $\Sigma/2$ to be Σ. We see that the distribution of $z \in [0,1]$ can be approximated by the χ^2-distribution with $N-1$ degrees of freedom. Since the probability density function (pdf) of the χ^2-distribution rapidly decreases as $\chi^2 \to \infty$ for a moderate value of the degrees of freedom, the normalization constant can be evaluated by integrating over $[0,\infty)$. Hence,

$$q(z) = \frac{1}{(2\Sigma)^{(n-1)/2}\Gamma((n-1)/2)} \exp\left[-\frac{z}{2\Sigma}\right] z^{\frac{n-1}{2}-1}, \quad (16)$$

where Γ represents the gamma function. In the above equation, we replaced N with a real number n. As discussed in the next section, the actual degrees of freedom is considerably smaller than N. One reason is that some of services are inactive and have small activities in the activity vector, being almost independent of t. In the derivation of the vMF distribution, an implicit assumption is that all of the degrees of freedom are equally active. We regard n as a measure of the effective size of each eigencluster. We call n the effective dimension.

5.3 Online calculation of threshold value

Using Eq. (16), the first and the second moments are calculated as

$$\langle z\rangle = (n-1)\Sigma,\ \ \langle z^2\rangle = 2(n^2-1)\Sigma^2.$$

These relations provide a way to evaluate n and Σ experimentally:

$$n - 1 = \frac{2\langle z\rangle^2}{\langle z^2\rangle - \langle z\rangle^2},\ \ \Sigma = \frac{\langle z^2\rangle - \langle z\rangle^2}{2\langle z\rangle}. \quad (17)$$

The moments are easily calculated online. Using an identity

$$\frac{1}{t}\sum_{i=1}^{t} z(t) = \left(1 - \frac{1}{t}\right)\frac{1}{t-1}\sum_{i=1}^{t-1} z(i) + \frac{1}{t}z(t).$$

and setting $1/t$ as β, we have

$$\langle z\rangle^{(t)} = (1-\beta)\langle z\rangle^{(t-1)} + \beta z(t). \quad (18)$$

Similarly for the second moment, we have

$$\langle z^2\rangle^{(t)} = (1-\beta)\langle z^2\rangle^{(t-1)} + \beta z(t)^2. \quad (19)$$

Naturally, β satisfies $0 < \beta < 1$ and is called the discounting factor. Note that Eqs. (17)-(19) give an online version of maximum likelihood estimation algorithm for the vMF distribution in an approximated manner.

Since $1/\beta$ can be associated with the number of data points, a rough estimate of β may be $\beta \sim \Delta t/L$, where L denotes the time scale we are interested in. Similarly, W can be estimated as $W \sim L/\Delta t$. In our benchmark system, we empirically take L on the order of 10 minutes.

Based on the above discussion, we have an online algorithm to calculate a threshold value to judge whether it is anomalous or not:

1. Give a critical boundary $0 < p_c < 1$.
2. Calculate $\langle z\rangle$ and $\langle z^2\rangle$ at t using Eqs. (18) and (19).
3. Calculate $n-1$ and Σ using Eq. (17).
4. Find $z_{\rm th}$ numerically such that $\int_0^{z_{\rm th}} dz q(z) = p_c$.
5. Emit an alert if $z(t) > z_{\rm th}$.

The above algorithm includes three parameters, p_c, β, and W. Since β and W can be easily estimated with L and Δt, the only parameter we must specify is substantially p_c, which is totally independent of the details of the system.

Table 2: Services appearing in the principal eigencluster

Index	I_s	I_d	P	Q
0	0.0.0.0	0.0.0.0	0	(none)
1	192.168.0.19	192.168.0.53	80	Plants
2	192.168.0.19	192.168.0.54	80	Plants
3	192.168.0.19	192.168.0.53	80	Trade
4	192.168.0.19	192.168.0.54	80	Trade
5	192.168.0.54	192.168.0.53	5558	JMS
6	192.168.0.53	192.168.0.54	9081	Plants
7	192.168.0.53	192.168.0.54	9081	Trade
8	192.168.0.54	192.168.0.53	9081	Plants
9	192.168.0.54	192.168.0.53	9081	Trade
10	192.168.0.53	192.168.0.52	50000	DB2
11	192.168.0.54	192.168.0.52	50000	DB2

6. EXPERIMENT

6.1 Experimental settings

The configuration of our benchmark system is illustrated in Fig. 1. As shown, the HTTP servers and WASs are doubly redundant. On the WASs, two applications, "Trade" and "Plants", are running. Trade is a standard benchmark application called Trade 3 [11], and Plants is a sample application bundled with IBM WebSphere Application Server V5.0 and simulates an online store dealing with plants and gardening tools. For both, the number of clients was fixed to be 16 and the think time was randomly chosen from 0 to 4 seconds.

We generated a matrix D every 20 seconds using a method that evaluates $d_{i,j}$ from captured IP packets. Loopback packets were ignored in the experiments, so that the services s_x and s_y in Fig. 2 are not observed for i_1 = 192.168.0.53 and i_2 = 192.168.0.54. The principal eigencluster is defined in Table 2, and small perturbations affecting it were ignored. In Table 2, the zeroth service was introduced to describe the situation where an optimal pair between callee and caller could not be identified. For example, services triggered by those outside the intranet will be associated with the zeroth service.

Apart from these, there are other service types, "DB2" and "JMS", in Table 2. DB2 denotes a request for the DB server, and JMS is for communications related to the Java Messaging Service.

6.2 Statistical properties in the normal state

We calculated $\boldsymbol{u}$ and z online over a period when the system exhibited no failures. The dependency matrix was generated over 52.7 minutes, so we had 158 matrices. The α_i

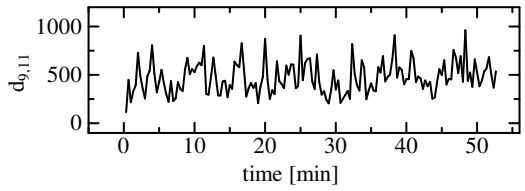

Figure 5: Time dependence of $d_{9,11}$.

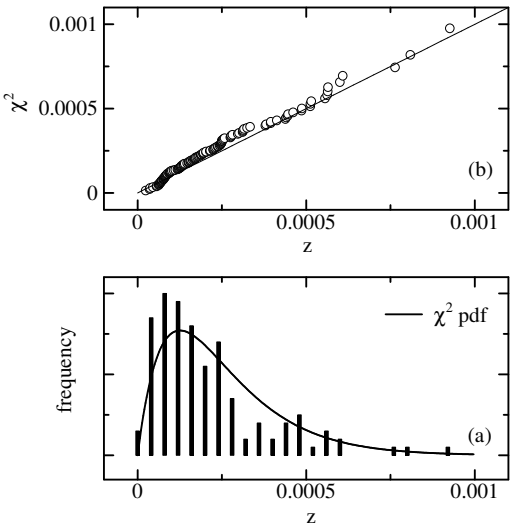

Figure 6: Statistics of z in the normal state. (a) Comparison of the experimental frequency and the χ^2 pdf. (b) The quantile-quantile plot.

values were taken as small random numbers on the order of 0.01. To see the fluctuation in D, we show in Fig. 5 the time dependence of $d_{9,11}$ as an example. We see that there are approximately 500 calls within 20 seconds under these experimental conditions and that the amplitude of fluctuation of $d_{9,11}$ is almost of the same order as the average. Hence, it makes little sense to place a threshold value on an isolated $d_{i,j}$.

To experimentally validate the pdf of z, we plotted the frequency distribution of z in Fig. 6 (a), where the χ^2 pdf is also shown. The parameters of the χ^2 pdf were calculated using all of the 158 data points with no discounting. The result was $n = 4.62$ and $\Sigma = 6.79 \times 10^{-5}$. In spite of the limitation of the number of data points, the frequency distribution is a good fit to the χ^2 pdf. We also drew a quantile-quantile plot in Fig. 6 (b). As shown, the experimental data is well placed on the 45 degree line. These results clearly support our formulation in Sec. 5.2.[2]

6.3 Response to traffic changes

To demonstrate the anomaly detection capability, we induced an artificial change to the normal state data under the rule

$$d_{i,11} \rightarrow \gamma d_{i,11}$$

[2]We rounded the $n-1$ value to be 4 to fit the χ^2 pdf because of the limitation of the numerical library we used. This is the main reason the deviation of the χ^2 pdf from the experimental frequency.

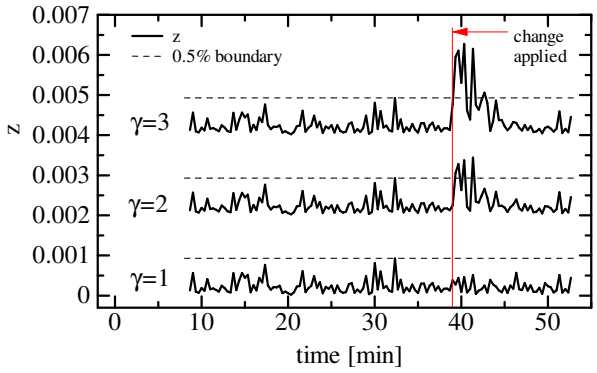

Figure 7: Detection of an artificially applied change in D.

for all i when $t > 39.0$ [min]. The system has two paths to call the DB server. This rule indicates that one of them is artificially enhanced. The results for z are shown in Fig. 7 for $\gamma = 1, 2$, and 3, where the $p_c = 0.5\%$ lines calculated using the n and Σ values above are also shown for reference. The corresponding z_{th} value is 0.00093. To make them visible, the curves for $\gamma = 2$ and 3 are shifted vertically. Since the fluctuation of the $d_{i,j}$s is extremely large, the values of $\gamma = 2$ and 3 are still reasonable values (Note that we did not directly modify D or $\tilde{d}_{i,j}$, but only $d_{i,j}$). Despite this fact, the figure shows that our method clearly detected the change.

6.4 Detection of an application fault

Next, we performed a more realistic experiment: A bug in one of the applications ("Plants") only on 192.168.0.54 causes a malfunction of the service of 11 at a time point. The server process itself continues running, so the network communication is normal at the IP layer or below. Since two Web servers are working on the system, a client may feel no change in response time as long as the overall traffic is sufficiently small. Although this defect occurs within a single service, it can cause a massive change in D. In fact, the dependencies of the services directly related with the service 11 will be considerably changed. What we would like to detect is a transition of this kind.

Figure 8 shows the generated time-series of the activity vector. We see that a sudden change in activities is observed at t_A and t_B, which correspond to the malfunction of the service 11 and its recovery. From the figure, the activities of the services 2, 6, and 11 are clearly decreased during this period. This result demonstrates that the service activity vector actually expresses the activity of services, and suggests a way to visualize the whole system.

To detect this fault automatically, we calculated z and its threshold value, following the algorithm explained in Subsection 5.3. In Fig. 9, we depicted the z values with vertical bars and the threshold values with thick gray curves for W =5, 25, and 50. The discounting factor and the critical boundary were taken as $\beta = 0.005$ and $p_c = 0.5\%$, respectively. While the result is considerably affected by the choice of W, we observe clear features at t =35.0 and 45.7 minutes, which correspond to $t_A < t_B$ in Fig. 8, respectively. These time points are highlighted with dashed vertical lines in Fig. 9. Note that the feature at t_B (recov-

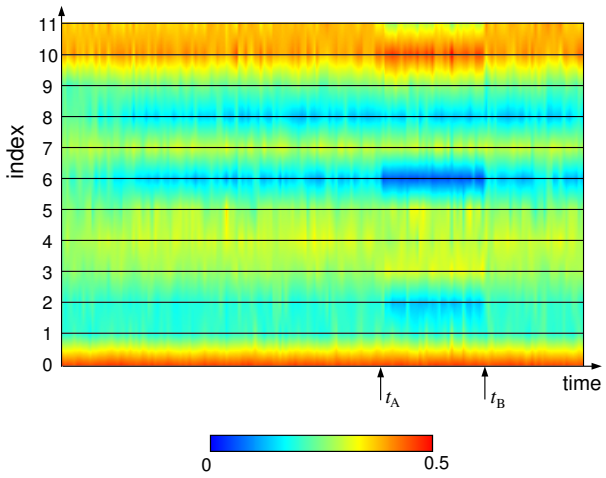

Figure 8: Time dependence of the activity vector. The failure duration starts at t_A and ends at t_B, as shown by arrows. The definition of the service indices are shown in Table 2

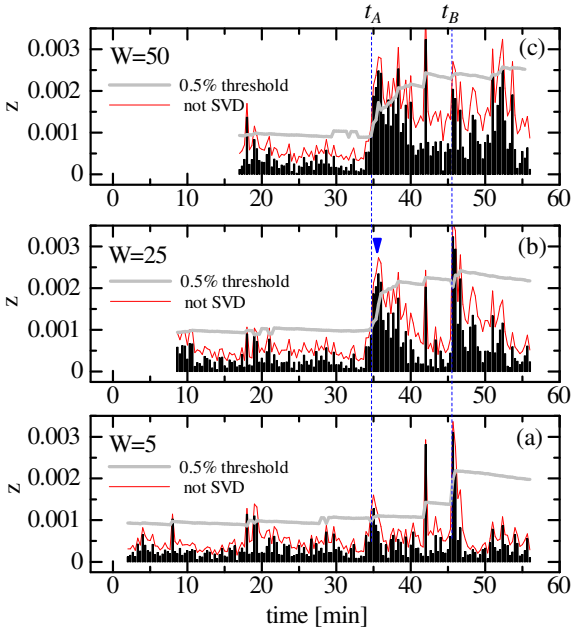

Figure 9: The dependence of z for W = (a) 5, (b) 25, and (c) 50. The 0.5% threshold is denoted by gray curves.

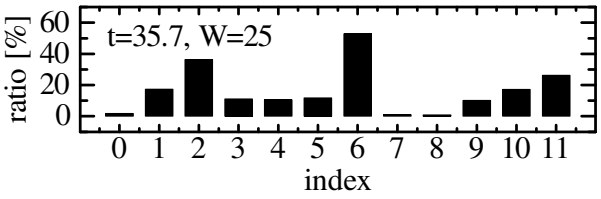

Figure 10: Change ratio of the activity vector for W =25 at a time point marked with a small triangle in Fig. 9 (b).

ery from the malfunction) demonstrates the learnability for gradual changes of the environment. The dependence on W is an inevitable consequence of the choice of the applications. Since the benchmark applications simulate human behavior, they must have a characteristic time scale. Comparing Fig. 8 with Fig. 9, we conclude that an appropriate value of W is about 25 (8.3 minutes). We see that this value of W allows us to pinpoint the time points t_A and t_B.

The curves plotted with thin lines ("not SVD") in Fig. 9 represent the result using another pattern extraction method, where the v_is are set to be constant in Eq. (9). Specifically, we simply took the mean vector instead of $\boldsymbol{r}$. The trend of z is similar to that of the SVD-based method, but is blurred out by the noise. This result demonstrates the effectiveness of the SVD-based pattern extraction technique.

To identify faulty services, we calculated the change rate of activities as compared to $\boldsymbol{r}$ for $W = 25$ at a time point with the peak of the feature around t_A. The result is shown in Fig. 10. We recognize that the high z value is ascribed to the changes of services 6, 2, and 11. This result is understood as follows. Because of the sudden decrease of service 11, the activity of its caller 6 also decreases. Since loopback packets are not observable, the change of the service 11 should affect the activity of service 2, which is a direct caller of the Plants service from 192.168.0.54 to 192.168.0.54 at $P = 9081$. Considering the consistency between IP addresses, it is easy to perform an inference like this. The result in Figs. 8, 9, and 10 demonstrate the utility of our anomaly detection method.

For the limitations of our approach, first, the probability of false alarms will be finite even if W is set to be the optimal value. As understood from Fig. 9, there is small finite probability of having outliers beyond a threshold value. Second, since the basic assumption of our approach is the stability of the direction of the activity vector, our approach is not appropriate for anomaly detection of rarely invoked services. Finally, there is much room for improvement in the calculations of the threshold values since the numerical library we used handles only integer degrees of freedom in the χ^2 pdf.

7. SUMMARY

We have proposed a new approach to anomaly detection in computer systems. First, we discussed why the principal eigenvector of the dependency matrix is a good feature vector which has a clear interpretation related to the activities of services. Second, we described a method to extract a typical pattern from a time-sequence of the feature vectors, based on an extremum principle. We showed that the

optimal choice of the typical pattern is the principal left singular vector of a matrix which contains activity vectors as column vectors. Third, we defined an anomaly measure z, and derived its probability distribution as an approximation of the von Mises-Fisher distribution. Our theoretical analysis showed that z obeys the χ^2 distribution with $n-1$ degrees of freedom, where n is the effective size of the principal eigencluster. Based on this result, we derived an online algorithm to calculate threshold values of z. Only a value of the critical probability p_c is needed to determine the threshold. Finally, we demonstrated that our method is capable of detecting a class of faults in a benchmark system.

8. ACKNOWLEDGMENTS

We are grateful to T. Fukuda, J. Sakuma, A. Inokuchi, and H. Takeuchi for stimulating discussions. We also thank H. Etoh for assisting with the experiment and K. Yoda for help in implementing the anomaly detector. T.I. acknowledges fruitful discussions with K. Inoue.

9. REFERENCES

[1] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra. Generative model-based clustering of directional data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 19–28, 2003.

[2] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proceedings of the Second ACM SIGCOMM Workshop on Internet Measurment*, pages 71–82, 2002.

[3] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*, volume 9 of *Classics in applied mathematics*. SIAM, 1994.

[4] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[5] C. H. Q. Ding, X. He, and H. Zha. A spectral method to separate disconnected and nearly-disconnected web graph components. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 275–280, 2001.

[6] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003.

[7] M. Gupta, A. Neogi, M. K. Agarwal, and G. Kar. Discovering dynamic dependencies in enterprise environments for problem determination. In *Proceedings of 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, pages 221–233, 2003.

[8] H. Hajji. Baselining network traffic and online faults detection. In *Proceedings of IEEE International Conference on Communications*, volume 1, pages 301–308, 2003.

[9] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Natural communities in large linked networks. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 541–546, 2003.

[10] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 549–552, 2003.

[11] IBM. Trade3; `http://www-306.ibm.com/software/webservers/appserv/benchmark3.html`.

[12] A. Inokuchi and H. Kashima. Mining significant pairs of patterns from graph structures with class labels. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 83–90, 2003.

[13] A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50:321–354, 2003.

[14] M. Kuramochi and G. Karypis. Discovering frequent geometric subgraphs. In *Proceedings of the Second IEEE International Conference on Data Mining*, pages 258–265, 2002.

[15] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 903–910, 2001.

[16] C. Noble and D. Cook. Graph-based anomaly detection. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 631–636, 2003.

[17] S. Sarkar and K. Boyer. Quantitative measures for change based on feature organization: Eigenvalues and eigenvectors. *Computer Vision and Image Understanding*, 71:110–136, 1998.

[18] G. Strang. *Linear Algebra and its Applications*. Academic Press, 1976.

[19] The Open Group. Application response measurement — ARM; `http://www.opengroup.org/tech/management/arm/`.

[20] M. Thottan and C. Ji. Anomaly detection in IP networks. *IEEE Transactions on Signal Processing*, 51(8):2191– 2204, 2003.

[21] H. Wang, D. Zhang, and K. G.Shin. Detecting SYN flooding attacks. In *Proceedings IEEE INFOCOM 2002*, pages 1530 –1539, 2002.

[22] T. Washio and H. Motoda. State of the art of graph-based data mining. In *SIGKDD Explorations Special Issue on Multi-Relational Data Mining*, volume 5, pages 59–68, 2003.

[23] K. Yamanishi and J. Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 676–681, 2002.

[24] K. Yamanishi, J. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 320–324, 2000.