Eigenspace-based anomaly detection in computer systems

IBM Research, Tokyo Research Laboratory

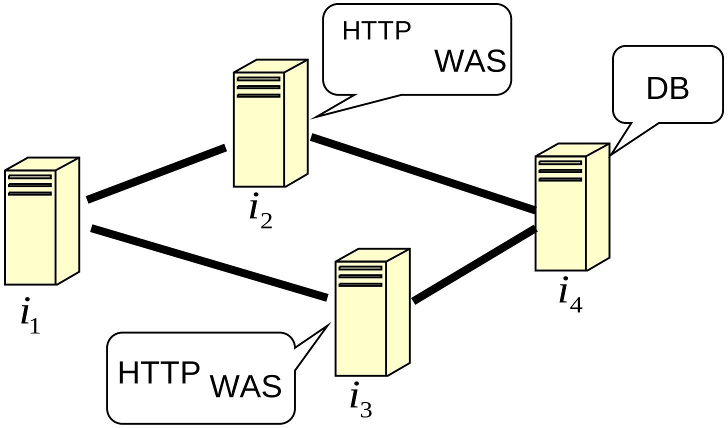
Tsuyoshi Ide and Hisashi Kashima

Outline

- Motivation
- Modeling Web-based systems
- Problem statement
- Feature extraction
- Anomaly detection
- Experiment
- Summary

Motivation:
Fault detection in computer systems at the application layer

- **Faults at the *application layer* are hard to detect using existing technologies**
  - This is especially true for Web-based systems with redundancy

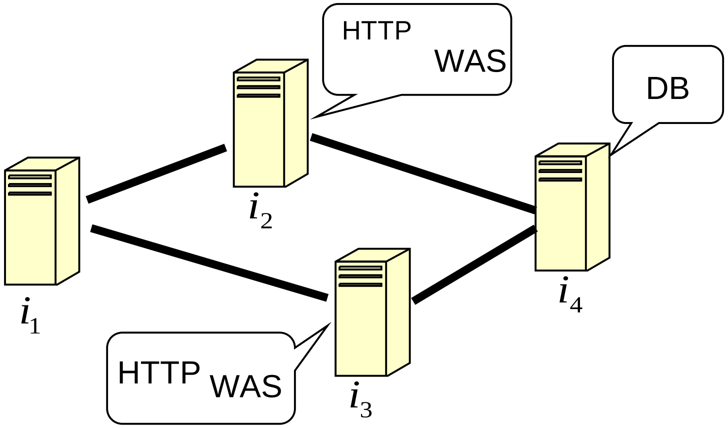


- **Why?**
  - The ***dependencies*** between servers make everything complicated
  - They are highly ***dynamic***: Observed metrics greatly vary overtime.

Computer system = strongly-correlated dynamic system

Motivation:
Knowledge discovery from correlated (or structured) dynamic systems

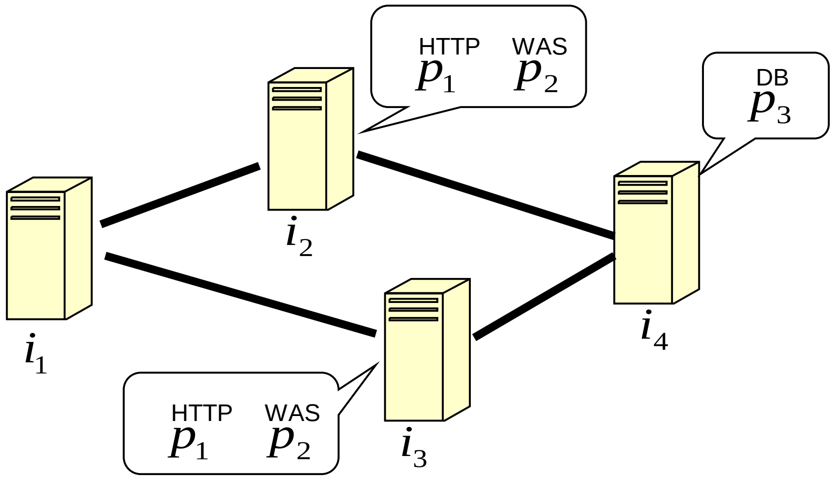
- **Data mining from structured data has recently attracted attention**
- **However, most of the graph mining studies focus mainly on static data**
  - We address the dynamic correlated systems, and
  - We develop a tool suitable for analyzing these systems.



$\boxed{\text{knowledge discovery from dynamic systems}}$

Modeling Web-based systems: definition

- **Service**
  - $s \equiv (I_{\text{source}}, I_{\text{dst}}, \text{port\#}, \text{trans.type})$
  - contains *two* IP addresses

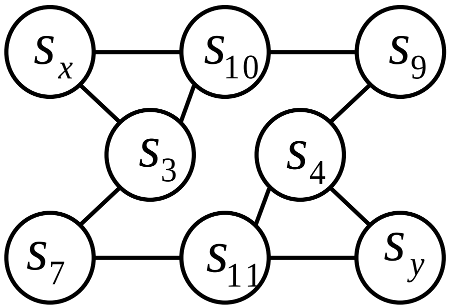


- **Service dependency**
  - # of a service's request for another service
  - log transform and symmetrize

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

- **Service dependency graph**
  - nodes: services
  - edge weights: service dependencies
  - defined as an undirected graph

$s_3 = (i_1, i_2, p_1, q_1)$
$s_4 = (i_1, i_3, p_1, q_1)$
$s_7 = (i_2, i_3, p_2, q_1)$
$s_9 = (i_3, i_2, p_2, q_1)$
$s_{10} = (i_2, i_4, p_3, q_2)$
$s_{11} = (i_3, i_4, p_3, q_2)$
$s_x = (i_2, i_2, p_2, q_1)$
$s_y = (i_3, i_3, p_2, q_1)$



Modeling Web-based systems: How to find D

Modeling Web-based systems: considerations

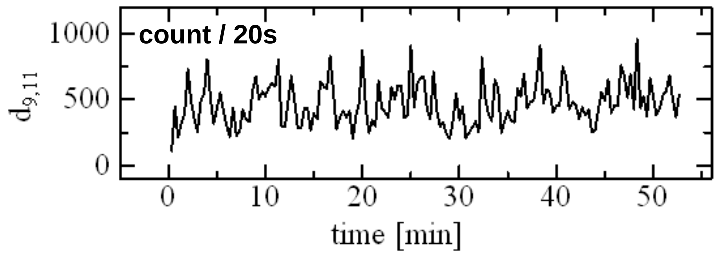
- **# of edges is relatively large**
  - e.g. more than 1000 edges for 50 services
  - the dependency (or adjacency) matrix might be sparse, but generally we do not know how sparse it is

Services in a benchmark system

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

- **Edge weights greatly vary over time**
  - autoregressive models are inappropriate in a time scale of several minutes

Service dependency between 9 &11



Problem statement:
Online anomaly detection from a time series of graphs

- Given a time-dependent graph with a fixed structure,
- detect anomalies ***online*** in an ***unsupervised*** manner.

Practical requirements

- Use a simpler feature rather than the graph itself.
- Establish a simple thresholding policy.

Why challenging ?

- Edge weights are highly dynamic
- A change in individual edge weights does not necessarily indicates a fault

We wish to detect a “phase transition” of the graph

Feature extraction:
The principal eigenvector is the summary of the activity of services

- **Definition of the "service activity vector (SAV)"**

$$\boldsymbol{u}(t) \equiv \arg\max_{\tilde{\boldsymbol{u}}} \left\{ \tilde{\boldsymbol{u}}^T \underline{\mathsf{D}(t)} \tilde{\boldsymbol{u}} \right\} \qquad \text{subject to } \tilde{\boldsymbol{u}}^T \tilde{\boldsymbol{u}} = 1$$

dependency matrix at *t*

<u>Why "activity"</u>?

- If D_{12} is large, then u_1 and u_2 should be large because of argmax (note: D is a positive matrix).
- So, if s_1 actively calls other services, then the weight in s_1 should be large.

Mathematically, this equation is reduced to the eigenvalue equation:

$$\mathsf{D}(t)\tilde{\boldsymbol{u}} = \lambda \tilde{\boldsymbol{u}}. \qquad \text{subject to } \tilde{\boldsymbol{u}}^T \tilde{\boldsymbol{u}} = 1$$

Feature extraction:
Also interpreted as the "stationary state" of the system

- **If we regard D as the time evolution operator, then the service activity vector can be interpreted as the stationary state of the system.**

Dependency matrix

$$\mathbf{D} = \begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1N} \\ D_{21} & D_{22} & \cdots & D_{2N} \\ \vdots & & \ddots & \\ D_{N1} & & \cdots & D_{NN} \end{bmatrix}$$

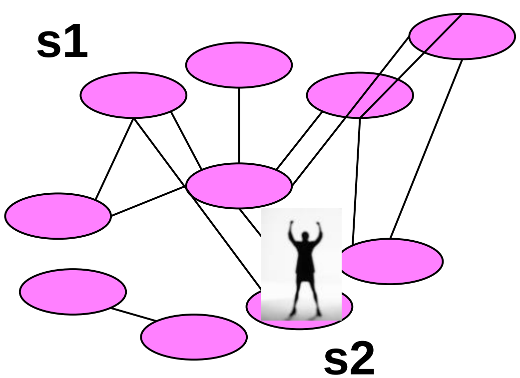
Equation of motion

$$\mathbf{x}(\tau + 1) = \mathbf{D}\mathbf{x}(\tau)$$

stationary state

$$\mathbf{u} = \mathbf{x}(\infty) / \| \mathbf{x}(\infty) \|$$

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix}$$



the probability amplitude that the system is holding the control token at the service s2.

service activity vector = summary of the system

Feature extraction:
Mathematical properties

- **SAV is invariant with respect to uniform changes in traffic.**
  - we can separate normal fluctuations in traffic from anomalies

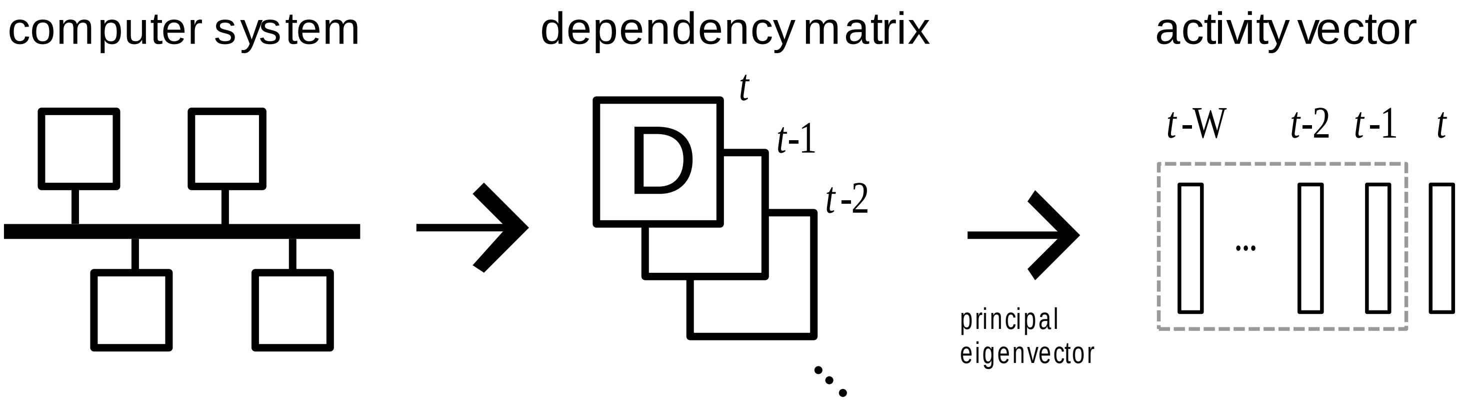
- **SAV is a positive vector.**
  - we never have negative activities.

Perron-Frobenius theorem

- **SAV has no degeneracy.**
  - we are free from such subtle problems as level crossings

Anomaly detection:
From a graph sequence to a vector sequence

- The problem was reduced to anomaly detection from a time sequence of ***directional data*** (normalized vector).



Question 1:

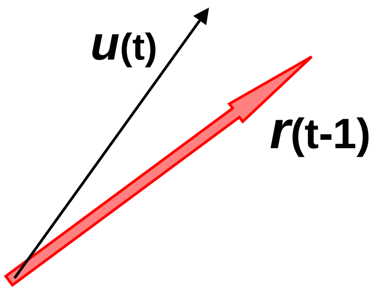
How can we define the anomaly metric?

Question 2:

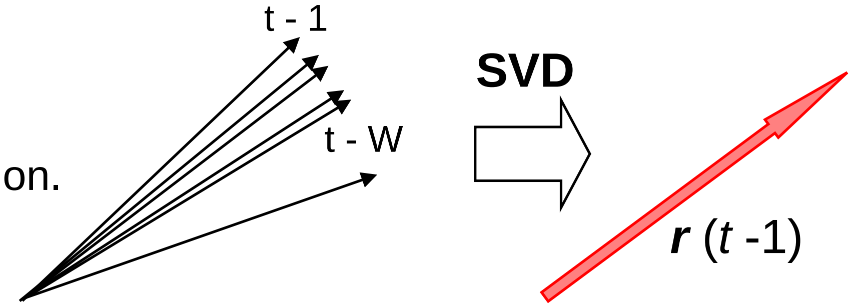
How can we determine its threshold?

Anomaly detection:
Cosine-measure-like anomaly metric

- **Definition of anomaly metric**
  - $$z(t) \equiv 1 - \boldsymbol{r}(t-1)^T \boldsymbol{u}(t)$$
    - ***u*** (*t*) : activity vector at time *t*
    - ***r*** (*t* -1) : typical activity pattern at *t* -1



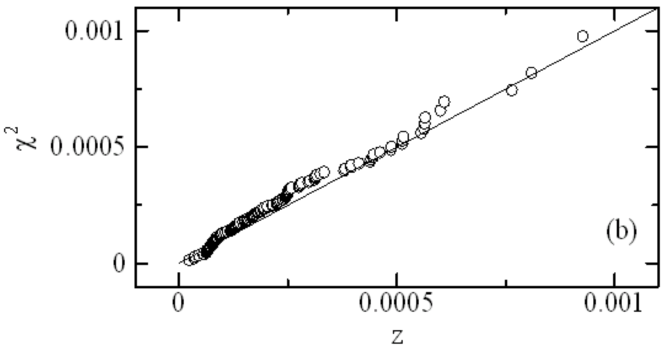
- **To find the typical activity pattern,**
  - We employ an LSI (Latent semantic indexing) like pattern extraction technique.
  - Perform SVD for
    - U = [***u*** (t -1), ***u*** (t -2), …, ***u*** (t - W)]
  - The principal left singular vector is the solution.

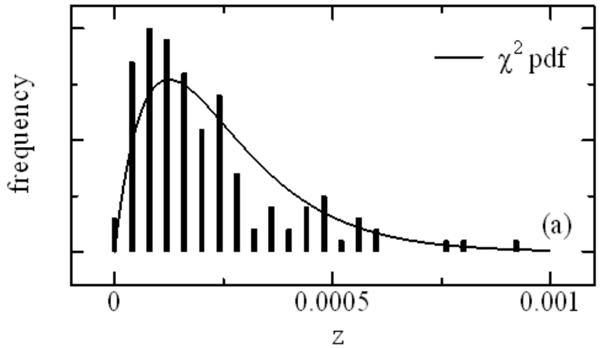


Anomaly detection:
A generative model for the anomaly metric

- **For directional data, Gaussian models do not work well**
  - The distribution of ***u***(t) degenerates on the surface of a hypersphere

- **Our starting point is the von Mises-Fisher distribution**
  - $$p(\mathbf{u}) \propto \exp\left[\frac{\mathbf{r}^T\mathbf{u}}{\Sigma}\right], \qquad \Sigma : \text{angular variance}$$
  - We can approximately derive the pdf for *z*(t) itself.





The pdf of *z* can be expressed as the chi-squared distribution with *N*-1 degrees of freedom.

Anomaly detection:
The notion of effective dimension

- **Explicitly, the distribution of the anomaly metric, *z*, is given by**
  - $$q(z) = \frac{1}{2^{\frac{N-1}{2}}\Gamma(\frac{N-1}{2})} e^{-z/(2\Sigma)} \left(\frac{z}{\Sigma}\right)^{\frac{N-1}{2}-1} \frac{1}{\Sigma}$$

effective dimension

angular variance

- **We wish to determine a threshold of z online**
  - We need to construct an online algorithm to update the parameters
  - Seemingly, we have a single fitting parameter, ∑*,* but this model doesn't work well because of the "curse of dimension"

- **We regard *N* as a fitting parameter *n.***
  - ***n* : "effective dimension"**
    - the actual degrees of freedom in action
    - this model works well when there are inactive degrees of freedom

Anomaly detection:
A novel online algorithm to update *n* and ∑

- **Parameter estimation is still challenging**
  - Because MLE has difficulties
    - The gamma function makes everything difficult

$$q(z) = \frac{1}{2^{\frac{n-1}{2}}\Gamma\left(\frac{n-1}{2}\right)} e^{-z/(2\Sigma)} \left(\frac{z}{\Sigma}\right)^{\frac{n-1}{2}-1} \frac{1}{\Sigma}$$

- **Our approach: the moment method**
  - The chi-squared distribution has explicit expressions for the 1st and 2nd moments

$$\langle z \rangle = \int dz q(z) z = (n-1)\Sigma, \quad \langle z \rangle^2 = \int dz q(z) z^2 = 2(n^2-1)\Sigma^2$$

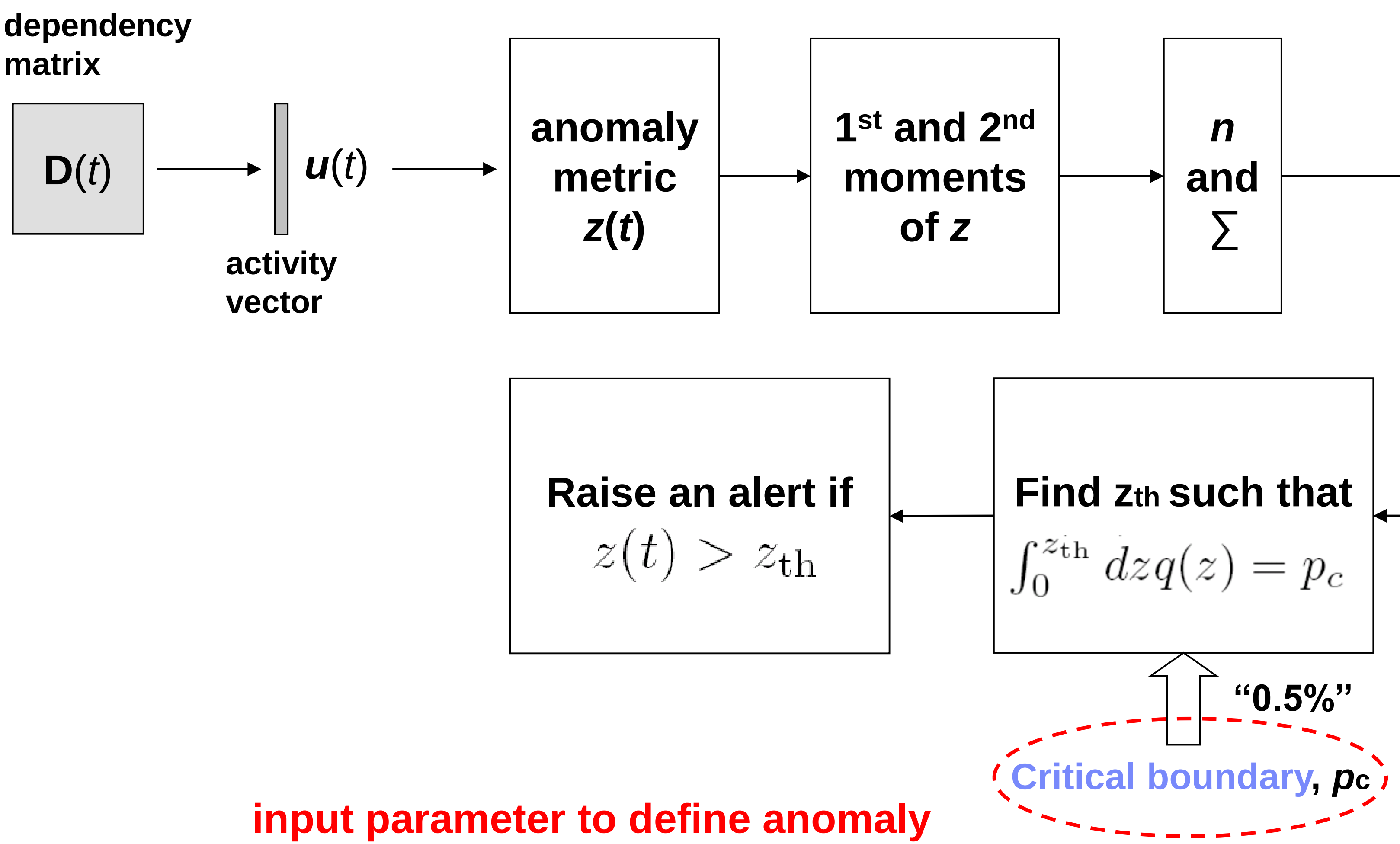
  - These can be easily solved wrt *n* and ∑

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

  - Online estimation for the moments are easy:

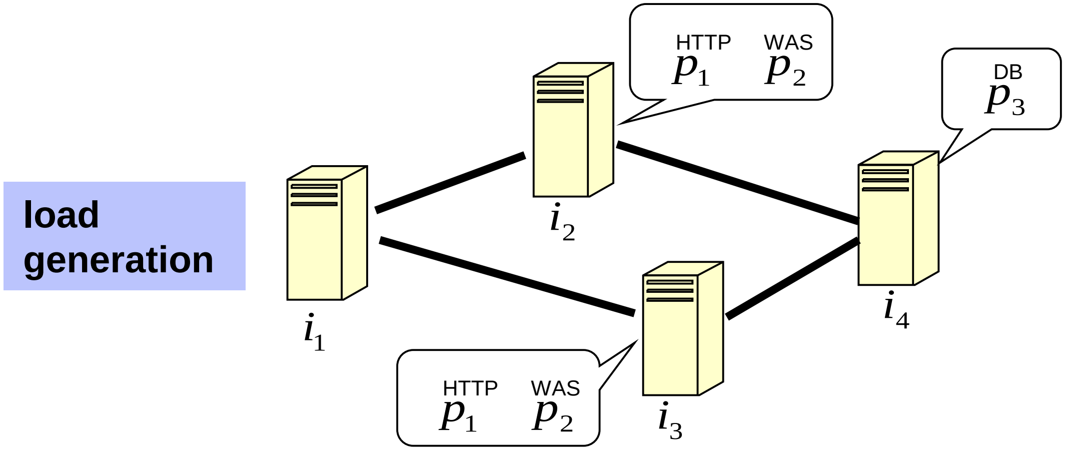
$$\langle z \rangle^{(t)} = (1-\beta)\langle z \rangle^{(t-1)} + \beta z(t) \qquad \langle z^2 \rangle^{(t)} = (1-\beta)\langle z^2 \rangle^{(t-1)} + \beta z(t)^2$$

Anomaly detection:
Summary of our algorithm



Experiment:
A bug in one of the Web applications

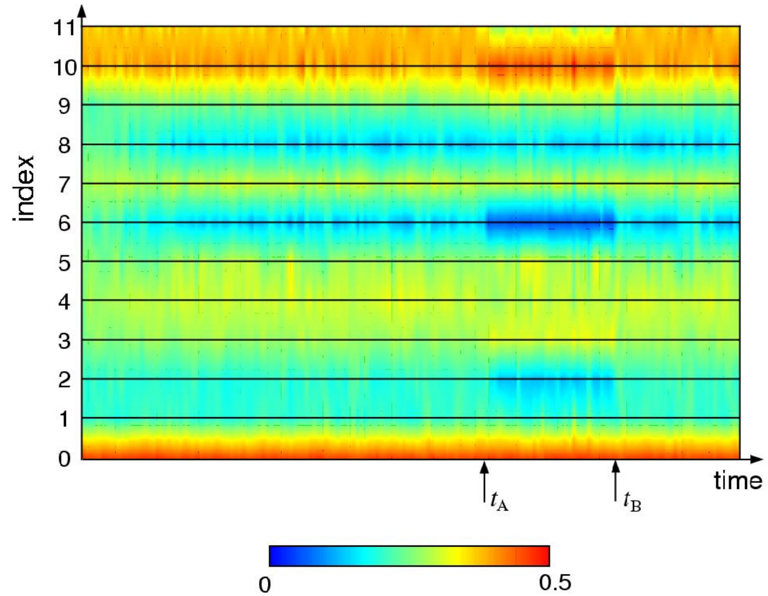
- **Settings**
  - On each of the two WAS, two applications are running ("Trade" and "Plants")
  - Service dependency matrices are generated every 20 seconds
  - The principal eigencluster has 12 services
- **A bug**
  - One of the "Trade" applications malfunctions at time t_A and recovers at t_B.
  - The server process itself continues running, so the network communication is normal at the TCP layer or below.
  - *Potentially* dangerous: The throughput is hardly affected for relatively low load

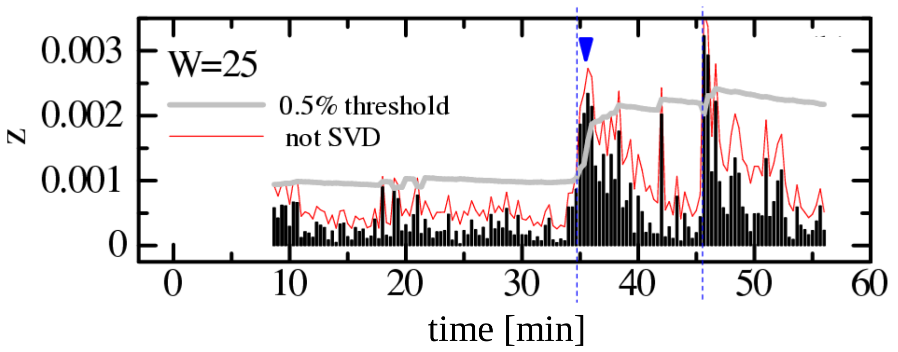


Index	I_s	I_d	P	Q
0	0.0.0.0	0.0.0.0	0	(none)
1	192.168.0.19	192.168.0.53	80	Plants
2	192.168.0.19	192.168.0.54	80	Plants
3	192.168.0.19	192.168.0.53	80	Trade
4	192.168.0.19	192.168.0.54	80	Trade
5	192.168.0.54	192.168.0.53	5558	JMS
6	192.168.0.53	192.168.0.54	9081	Plants
7	192.168.0.53	192.168.0.54	9081	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

Experiment:
The malfunction could be detected

- **The malfunction started at t_A and finished at t_B**
- **Time evolution of SAV**
  - clearly visualizes the malfunction period
  - the malfunction of the single service (#11) causes a massive change
- **Anomaly metric**
  - Two features clearly indicate the malfunction period
    - The latter is the evidence that the online calculation works well
- **Calculated threshold value**
  - dynamically adapted to the situation
  - n ~ 4 is much smaller than N=12





Summary

- We have considered the issue of anomaly detection from a highly dynamic graph sequence.

- We have introduced several new concepts

service activity vector	cosine-measure-like anomaly metric
effective dimension	moment method

- We demonstrated the utility of our approach in a benchmark system.