PREPARE: Predictive Performance Anomaly Prevention for Virtualized Cloud Systems

**Yongmin Tan**¹, Hiep Nguyen¹, Zhiming Shen¹, Xiaohui (Helen) Gu¹
Chitra Venkatramani², Deepak Rajan²

¹North Carolina State University  ²IBM T.J. Watson Research
Infrastructure as a Service (IaaS) Cloud

- Amazon’s EC2, Rackspace’s Mosso, GoGrid, Flexiscale
Cloud Systems are Error-Prone

- Recent Amazon EC2 service outage
- SLO violations

We need automated anomaly management!
Existing Approaches

- **Reactive anomaly management**
  - Perform corrections *after* an anomaly happens
  - No prevention cost but prolonged service downtime
  - Difficult to reproduce the anomaly-inducing environments

- **Proactive anomaly management**
  - Take preventive actions on *all* system components
  - Achieve better reliability but incur large overhead
  - Ignorant about anomaly causes
Virtual machines

Resource usage monitoring

Resource scaling / VM migration

Per-component online prediction models

SLO violation feedback

Anomaly cause inference

Alert

Predictive prevention actuation

Faulty components

Relevant attributes

System Design
Online Prediction Model

• *Per-component* model vs. *monolithic* model

We have $5 \times 9 = 45$ attributes

• Cannot pinpoint faulty components
• Lower prediction accuracy
Online Prediction Model (Cont.)

- **Per-component model** vs. **monolithic model**

  - More robust to prediction errors
  - Pinpoint the faulty components

\[ C_1: a_1, a_2, \ldots, a_5 \]
\[ C_2: a_1, a_2, \ldots, a_5 \]
\[ C_3: a_1, a_2, \ldots, a_5 \]
\[ C_4: a_1, a_2, \ldots, a_5 \]
\[ \ldots \]
\[ C_9: a_1, a_2, \ldots, a_5 \]
Feature Value Prediction Model

• *Two-dependent* vs. *basic* Markov model

Prediction using the basic Markov model (Gu et al., ICDE’09)
Feature Value Prediction Model (Cont.)

- *Two-dependent* vs. *basic* Markov model

Prediction using the two-dependent Markov model
Integrated Prediction Model

- Feature evolving model:
  - Predict future value with probabilities
- Anomaly classifier:
  - Identify anomaly symptoms
- Integrated predictor:
  - Estimate probability of future system normal/abnormal state

Predict an anomaly

Lead time T

The anomaly happens
Anomaly Cause Inference

- Identify the faulty components
  - Check which per-component models raise an alert
Anomaly Cause Inference (Cont.)

- Identify the mostly related attributes
- Compute the impact strength for each attribute
- Rank all the attributes based on the impact strength

\[ L = \frac{P(a_i \mid "abnormal")}{P(a_i \mid "normal")} \]
Anomaly Prevention Actuation

Elastic VM resource scaling
[Shen, Subbiah, Gu, Wilkes, SOCC’11]
Anomaly Prevention Actuation

Elastic VM resource scaling
[Shen, Subbiah, Gu, Wilkes, SOCC’11]
Anomaly Prevention Actuation

Live VM migration

Memory Usage (MB)

Time (seconds)

C5
Anomaly Prevention Actuation

Live VM migration
Anomaly Prevention Actuation (Cont.)

- Online validation
  - Check whether anomaly alerts are gone
  - Check resource usage before/after the actuation

Usage keeps increasing. This scaling is effective.
Experimental Setup

- Implemented on top of Xen
- Case study systems
  - IBM System S
  - RUBiS (EJB version) benchmark
- Data labeling
  - Match resource logs with the SLO log
IBM System S Application

![Diagram showing the flow of UDP packets from a Workload generator to various processing elements (PEs): PE1, PE2, PE3, PE6, PE4, PE5, PE7. The packets move from PE1 to PE2, then to PE3, to PE6, to PE4, then to PE5, and finally to PE7.]
IBM System S Application

[Diagram]

Workload generator

UDP packets

PE1

Memory leak

PE2 → PE3 → PE6

PE4 → PE5 → PE7
IBM System S Application

- Workload generator
- UDP packets
- PE1
- PE2
- PE3
- PE4
- PE5
- PE6
- PE7
- CPU hog

21
IBM System S Application

Workload generator

Increase workload

UDP packets

Processing bottleneck
RUBiS Application

- Workload generator
- Web server
- App server1
- App server2
- DB server

Http requests
RUBiS Application

Workload generator

Web server

Http requests

App server1

Memory leak

App server2

DB server
RUBiS Application
RUBiS Application

增加工作负载

工作负载生成器

Web服务器

HTTP请求

数据库服务器

应用程序服务器1

应用程序服务器2

处理瓶颈
Evaluation Methodology

• Compare with some alternative schemes
  - Reactive intervention
  - Without intervention

• Evaluation metrics
  - SLO violation time
  - SLO metric traces (throughput trace or average response time trace)
  - Anomaly prediction accuracy (detection rate and false alarm rate under different look-ahead windows)
Experiment Results: SLO Violation Time

- Without intervention
- Reactive intervention
- PREPARE

**a)** Memory leak
- System S: 152 seconds
- RUBiS: 196 seconds

**b)** CPU hog
- System S: 301 seconds
- RUBiS: 301 seconds
Experiment Results: SLO Metric and Resource Traces (Memory Leak in RUBiS)

Without intervention

SLO violation threshold

Reactive intervention

SLO violation threshold

PREPARE

SLO violation threshold
Experiment Results: Prediction Accuracy

CPU hog (RUBiS)

per-component model vs. monolithic model
Experiment Results: Prediction Accuracy

Two-dependent Markov model vs. basic Markov model
Discussions and Future Work

• Use the supervised learning method
  - Need labeled training data
  - Use unsupervised learning (Dean, Hiep, Gu, ICAC’2012)

• Perform coarse-grained diagnosis
  - Try to alleviate, but not remove the faults
  - We can integrate with fine-grained debugging tools

• Anomaly manifestation should be observable
  - True for most faults causing performance anomalies
  - We can perform white-box diagnosis
Conclusions

• PREPARE: a *predictive* performance anomaly prevention system
  - Predict recurrent anomalies
  - Pinpoint faulty components and infer relevant attributes
  - Perform just-in-time anomaly prevention actions

• Deployed on IBM System S and RUBiS
  - Effectively prevent anomalies caused by several faults
  - Achieve high prediction accuracy
  - Light-weight and non-intrusive
Acknowledgments

• Anonymous reviewers
• This work was sponsored in part by NSF CNS0915567 grant, NSF CNS0915861 grant, NSF CAREER Award, CNS1149445, U.S. Army Research Office (ARO) under grant W911NF-10-1-0273, IBM Faculty Awards and Google Research Awards.

THANK YOU

http://dance.csc.ncsu.edu
Backup Slides
Our Approach: Predictive Anomaly Prevention

C1 VM

C4 VM

C7 VM

C2 VM

C5 VM

C8 VM

C3 VM

C6 VM

C9 VM

Memory cap

Memory usage

Average response time

SLO violation!
Our Approach: Predictive Anomaly Prevention

- **Performance anomaly prediction**
  - **Anomaly alerts**
  - **Cause inference**
  - **Inference results**
  - **Trigger memory scaling**
  - **Anomaly prevention**

- **Resource usage**

- **Memory cap**

- **Memory usage**
Our Approach: Predictive Anomaly Prevention

Average response time

SLO conformance!

C1 VM
C2 VM
C3 VM
C4 VM
C5 VM
C6 VM
C7 VM
C8 VM
C9 VM

Memory cap
Memory usage
Online Prediction Model (Cont.)

- CPU usage (%) metric ranged from 0 to 100 with two states

Two-dependent Markov model

Basic Markov model

- K-step prediction: compute K-step state transition matrices

\[
\begin{pmatrix}
0.4 & 0.6 & 0 & 0 \\
0 & 0 & 0.3 & 0.7 \\
0.8 & 0.2 & 0 & 0 \\
0 & 0 & 0.4 & 0.6 \\
\end{pmatrix}^K
\]

\[
\begin{pmatrix}
0.4 & 0.6 \\
0.1 & 0.9 \\
\end{pmatrix}^K
\]