

---

# EntomoModel: Understanding and Avoiding Performance Anomaly Manifestations

---

**Christopher Stewart**  
The Ohio State University

Kai Shen  
The University of Rochester

Arun Iyengar  
IBM Watson Research Center

Jian Yin  
Pacific Northwest National Labs

# Internet Services Occasionally Perform Poorly

- Internet services are very complex
  - Depend on many third party components, e.g., Apache Web server, IBM WebSphere, MYSQL
  - Complicated business logic with millions of LOC
- Today Internet services perform well most of the time, *but not all of the time*
  - Keynote Systems, July--December 2008
  - Studied 11 retailers known to offer low response times most of the time
  - **Only 3 of 11** could meet an SLA that required responding to **98%** of user requests within 2 seconds

# Performance Anomalies: One Cause of Poor Performance

- Sometimes poor performance stems from...
  - 1) subtle mis-configurations, 2) poor implementations, or 3) unexpected interactions between components
- *A **performance anomaly manifestation** occurs when these root causes degrade performance*
  - Real example:
    - System monitors miss a significant increase in resource needs **and** performance-critical components are co-located with less critical components. [Amazon S3]
  - Not examples:
    - 1) System-crash bugs, 2) Obvious under-provisioning

# Outline

---

- 1. A study of performance anomaly manifestations**
2. EntomoModel: Understanding the conditions where anomaly root causes are likely to manifest
3. Entomophobic management: Avoiding performance anomaly manifestations
4. Conclusion

# A Study of Anomaly Manifestations

1. Setup
2. Anomaly Root Causes
3. Manifestation Patterns

- Internet services support a wide range of management policies and workloads, e.g.,

Application logic

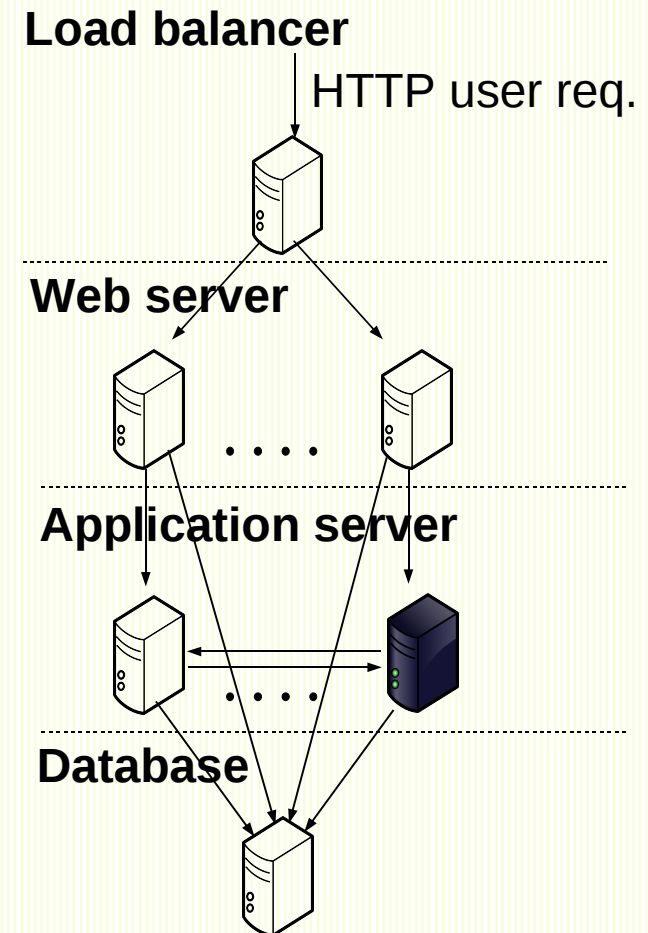
Http protocol

Placement of application logic

Request arrival intensity

Number of web servers

- We chose 9 practical parameters with over 1M possible settings
- A full specification is called a **workload-policy setting**



# A Study of Anomaly Manifestations

1. Setup
2. Anomaly Root Causes
3. Manifestation Patterns

- Procedure for detecting anomaly manifestations
  1. Randomly select a workload-policy setting
  2. Measure performance under the setting
  3. Re-insert a known, well-documented anomaly cause
  4. Measure performance under the setting again, but with the root cause(s) reinserted

**An anomaly manifestation occurred under the setting if performance fell after reinsertion**

- Measure multiple times for confidence
- Performance metric= % of requests with response time within 2 seconds

# A Study of Anomaly Manifestations

1. Setup
- 2. Anomaly Root Causes**
3. Manifestation Patterns

- We pulled 6 anomaly root causes from bug repos of open source components, e.g., JBoss, MYSQL, etc.
- Recreated from published fixes and comments

**Load balancer:** An earlier version of our load balancer implemented round-robin load balancing poorly. The last server IP address in the setup file did not receive requests [Sourceforge project page]

**DNS system:** A DNS server did not have some IP address to a domain name mappings. Network messages to unmapped machines were delayed. [MYSQL Forums]

# A Study of Anomaly Manifestations

1. Setup
2. Anomaly Root Causes
- 3. Manifestation Patterns**

- *How frequently do performance anomalies manifest?*
- *What is their impact on performance?*
- *Are manifestations more likely under certain settings?*

	Load balancer	DNS system	Median in study
Manifestation frequency	<b>11%</b>	<b>7%</b>	<b>7.5%</b>
Avg. performance degradation	<b>65%</b>	<b>21%</b>	<b>38%</b>
Parameter with highest info gain	<b>Replica (43%)</b>	<b>HTTP (20%)</b>	

# A Study of Anomaly Manifestations

1. Setup
2. Anomaly Root Causes
- 3. Manifestation Patterns**

- **Do root causes manifest differently in combination than in isolation?**
  - For each setting, we measured performance when multiple root causes were reinserted. We compared the anomaly classification to the in isolation tests.

<b>DNS + LB root causes</b>	Normal in combination	Manifest in combination
Normal in isolation <i>(i.e., neither cause manifested by itself)</i>	<b>81%</b>	<b>2%</b>
Manifest in isolation <i>(i.e., either LB or DNS manifested by itself)</i>	<b>7%</b>	<b>10%</b>

# A Study of Anomaly Manifestations

1. Setup
2. Anomaly Root Causes
- 3. Manifestation Patterns**

	<b>Normal in combination</b>	Manifest in combination
Normal in isolation	<b>81%</b>	<b>2%</b>
<b>Manifest in isolation</b>	<b>7%</b>	<b>10%</b>

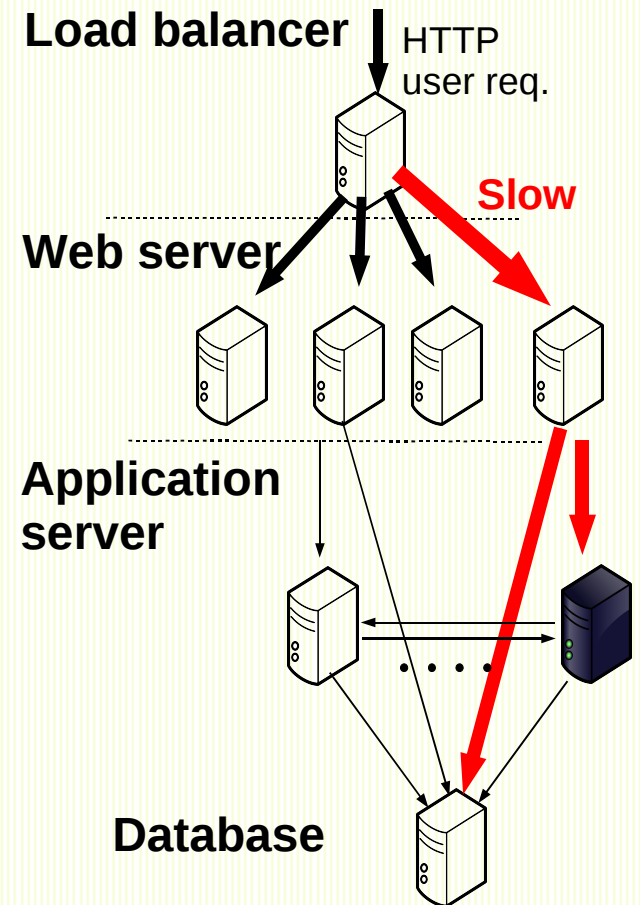
- Under some settings, root causes would manifest if they were reinserted in isolation but not in combination
- Emergent “good” behavior: How does combining two performance degrading anomalies help?**

# A Study of Anomaly Manifestations

1. Setup
2. Anomaly Root Causes
- 3. Manifestation Patterns**

## 1. The DNS root cause increases the communication delay between components

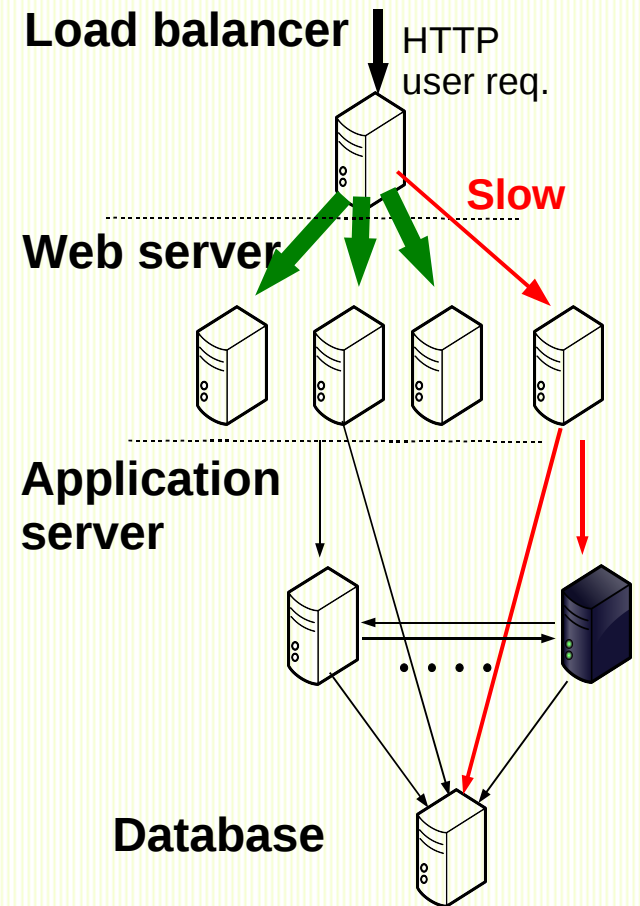
- When the root cause afflicts a web server, communication between the load balancer and web server are affected
- By itself, the increased communication delay can slow down the response time for a large portion of requests



# A Study of Anomaly Manifestations

1. Setup
2. Anomaly Root Causes
- 3. Manifestation Patterns**

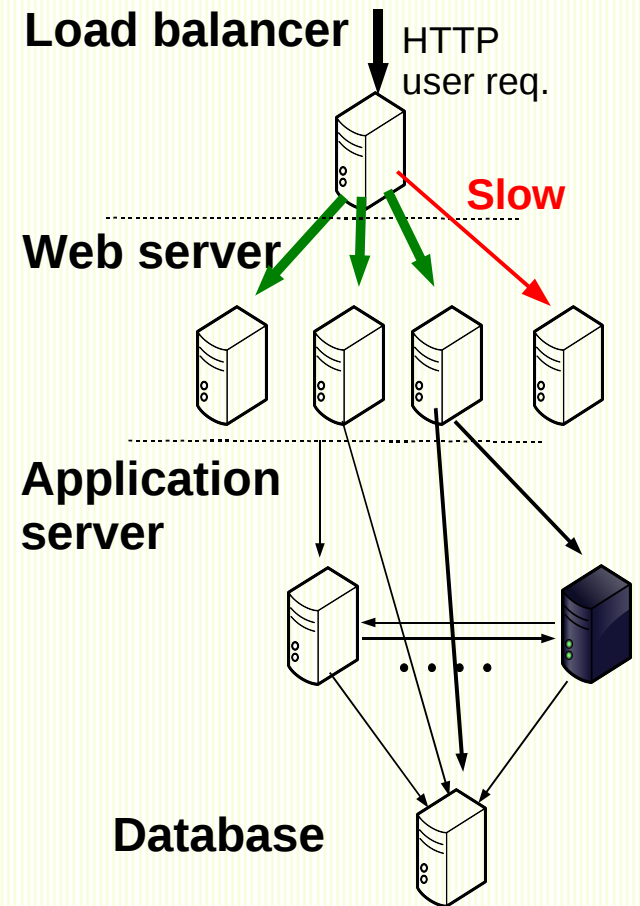
1. The DNS root cause increases the communication delay between components
2. If the load-balancer root cause afflicts the same web server, it will reduce the number of round trip communications
  - Reducing the relative impact of the DNS delay
  - But also, increasing the workload of other web servers in the system



# A Study of Anomaly Manifestations

1. Setup
2. Anomaly Root Causes
- 3. Manifestation Patterns**

1. The DNS root cause increases the communication delay between components
2. If the load-balancer root cause afflicts the same web server, it will reduce the number of round trip communications
3. Well-balanced systems are often robust to single-node failures/slowdowns



# Outline

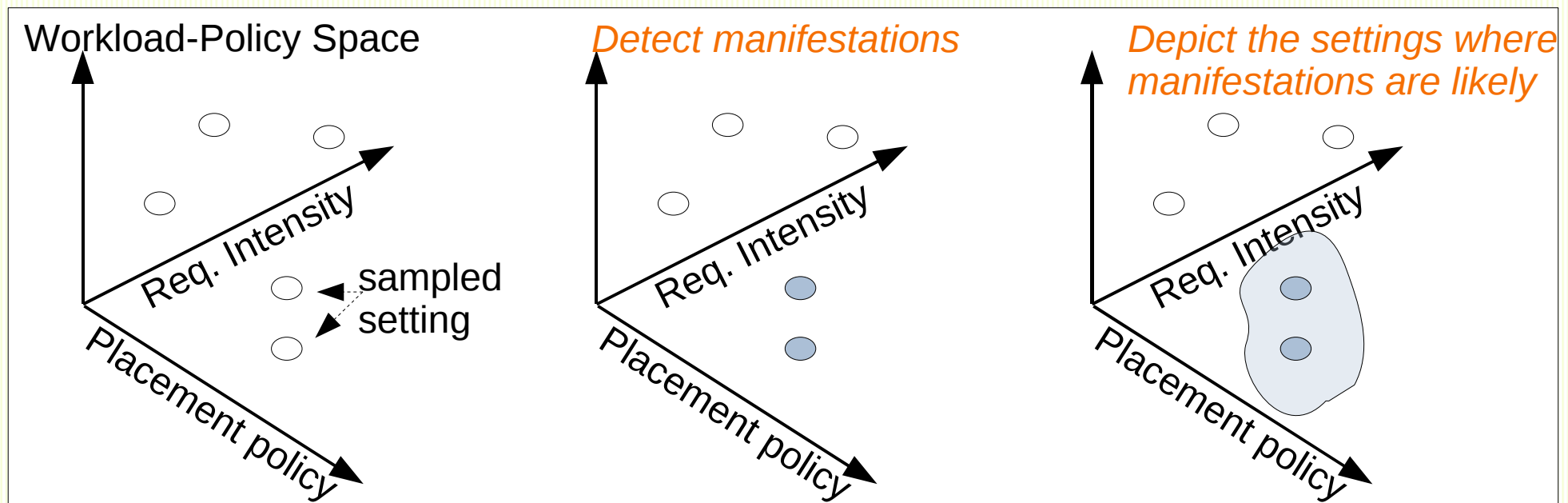
---

1. A study of performance anomaly manifestations
- 2. EntomoModel: Understanding the conditions where anomaly root causes are likely to manifest**
3. Entomophobic management: Avoiding performance anomaly manifestations
4. Conclusion

# Understanding When Anomalies Manifest

1. Approach
2. Evaluation
3. Unknown Root Causes

- Intuition: Anomaly root causes are more likely to manifest under certain settings. *Which settings?*
- EntomoModel uses 1) the ability to detect anomaly manifestations and 2) machine learning



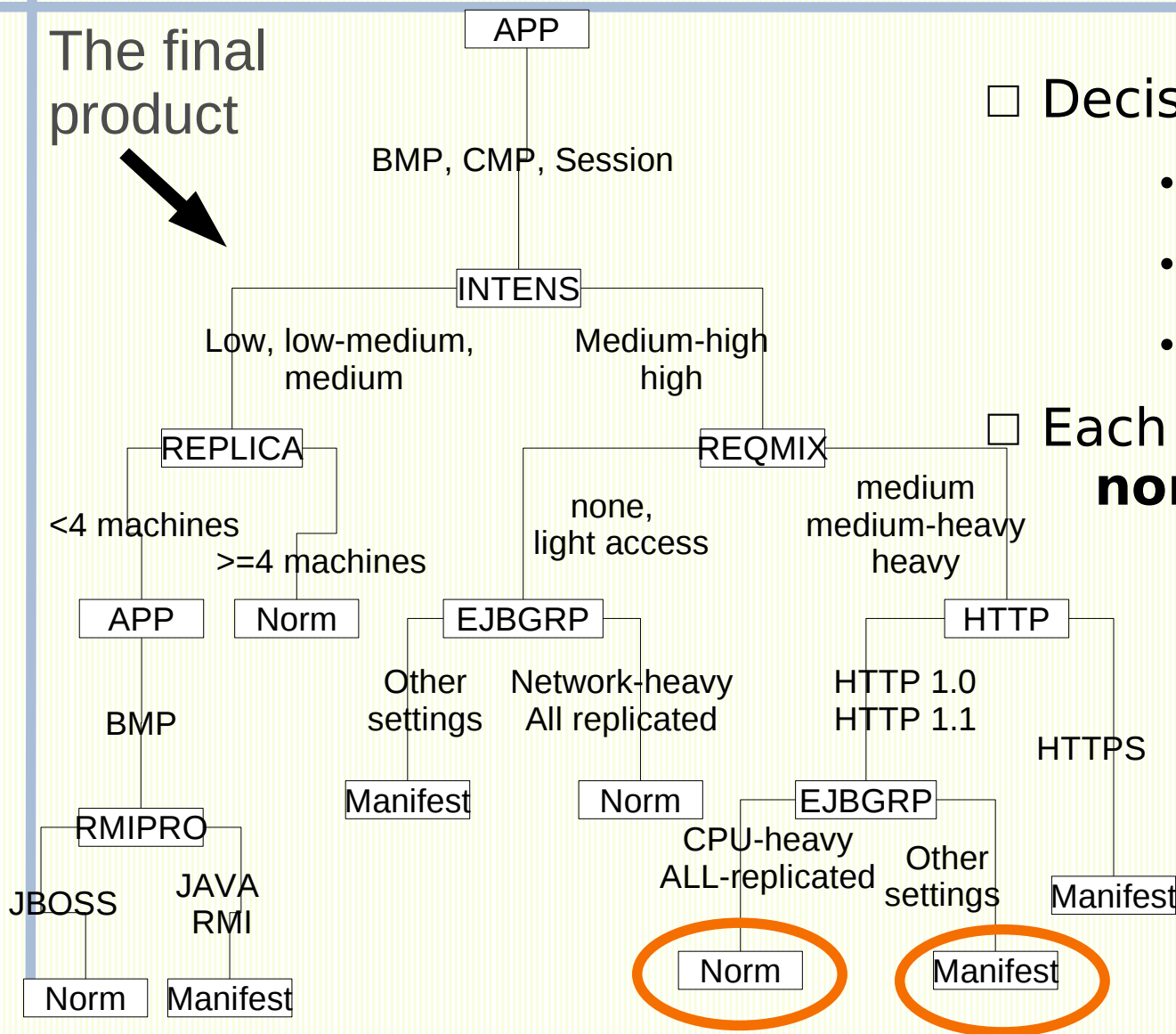
# Understanding When Anomalies Manifest

## 1. Approach

2. Evaluation

3. Unknown Root Causes

The final product



- Decision tree classifiers
  - No a-priori knowledge
  - Used in previous studies
  - Interpretable

□ Each workload setting is **normal or manifest**

# Understanding When Anomalies Manifest

1. Approach
- 2. Evaluation**
3. Unknown Root Causes

- First, we used the approach in our study to detect the manifestation of known anomalies
  - 885 samples to build decision trees, 445 to evaluate

LB Root Cause	Predicted normal	Predicted manifest
Actual normal	86%	3%
Actual manifest	1%	10%

- Impact of emergent “good” behavior
  - For DNS+LB were reinserted in combination, *emergent “good” behavior reduced misclassification by 25%*
  - Across many combinations: 1) Never hurt and 2) Up to 50% reduction in misclassification

# Understanding When Anomalies Manifest

1. Approach
2. Evaluation
- 3. Unknown Root Causes**

- Unknown, active root causes will probably manifest under different settings than past, fixed anomalies

**Challenge:** How can we detect the manifestations of root causes that we can't reinsert?

**Observation:** Performance anomaly manifestations cause unexpected and significant performance degradation

**Our approach:** Build a performance model that captures the systems expected performance---*A manifestation occurs when actual performance falls below the model's expectations*

- [Stewart-NSDI-2005, Urgaonkar-Sigmetrics-2005, Stewart-Usenix-2008, Thereska-Sigmetrics-2008]

# Understanding When Anomalies Manifest

1. Approach
2. Evaluation
- 3. Unknown Root Causes**

**Our approach:** Build a performance model that captures the systems expected performance---*A manifestation occurs when actual performance falls below the model's expectations*

- 885 samples (of performance) and model predictions to build decision tree; 445 to evaluate
- Decision tree shown earlier

No Known Root Causes Inserted	Predicted normal	Predicted manifest
Actual normal	82%	7%
Actual manifest	2%	9%

# Outline

---

1. A study of performance anomaly manifestations
2. EntomoModel: Understanding the conditions where anomaly root causes are likely to manifest
- 3. Entomophobic management: Avoiding performance anomaly manifestations**
4. Conclusion

# Avoiding Manifestations

1. Entomophobic Man.
2. Results

- Entomophobia is the fear of bugs
  - Option 1: Extermination---find and remove all anomaly root causes
  - Option 2: Avoidance [qin-sosp-2005]---operate under only the settings where anomalies don't manifest

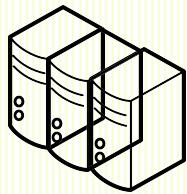
*Entomophobic management uses EntomoModel to identify workload and policy settings to avoid*

- Note, entomophobic management is intended to complement other online management techniques

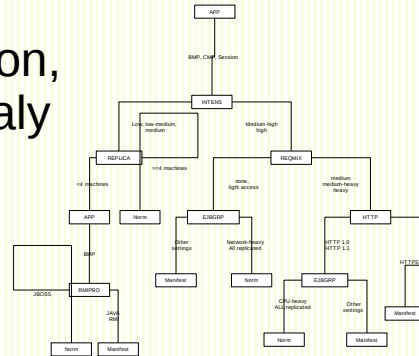
# Avoiding Manifestations

## 1. Entomophobic Man.

## 2. Results



Was this SLA violation, caused by an anomaly manifestation?



Yes, avoid it by changing this policy

Yes, but you can't avoid it

No apply other techniques, e.g, add nodes

### Experimental Setup

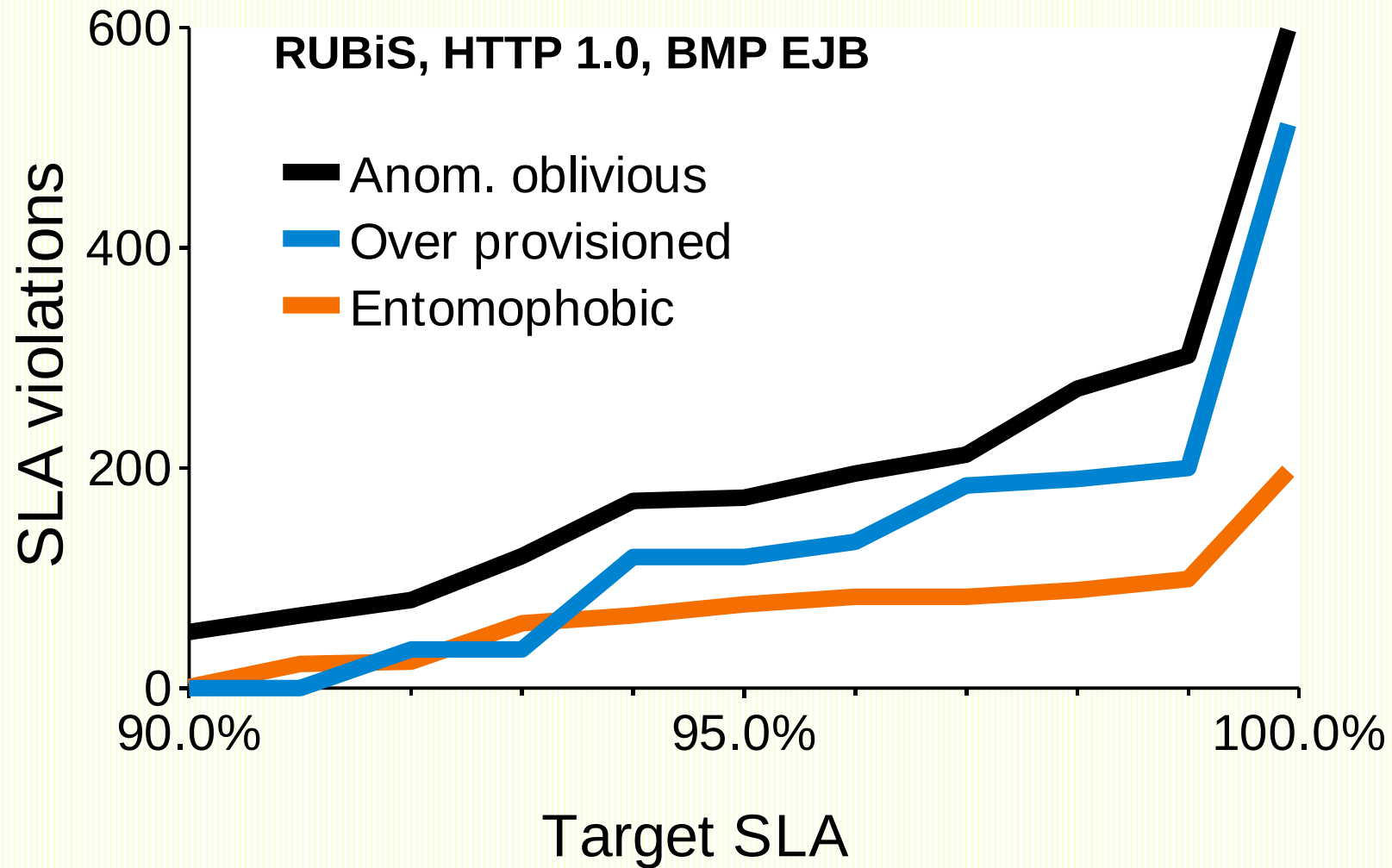
- Adapted a nonstationary request trace from a real enterprise application (stewart-eurosys-2007)
- Checked SLA status every 3 minutes

Compare 3 competing approaches

- Increase number of nodes on SLA violations
- Always use max nodes
- Entomophobic management uses EntomoModel

# Avoiding Manifestations

1. Entomophobic Man.
- 2. Results**



# Outline

---

1. A study of performance anomaly manifestations
2. EntomoModel: Understanding the conditions where anomaly root causes are likely to manifest
3. Entomophobic management: Avoiding performance anomaly manifestations
- 4. Conclusion**

# Conclusion

## 1. Related Work

2. Future Work

3. Take Aways

- Jeff Mogul (mogul-eurosys-2006) warned about emergent behavior in complex computer systems, citing examples in networking and OS
- Eno Thereska et al. (thereska-sigmetrics-2008) created a hybrid human-crafted/machine-learned performance model.
- Kai Shen et al. (shen-sigmetrics-2009) proposed reference-driven anomaly detection---future direction
- Much work on online adaptive system management, e.g., padala-eurosys-2009

# Conclusion

1. Related Work
- 2. Future Work**
3. Take Aways

- How anomalous is your IAAS provider?
  - In this study, we learned that anomalies are more likely under certain management policies
  - *What is the role of the third-party infrastructure provider in avoiding anomaly manifestations?*
  - *Can an infrastructure provider trick monitoring components to inflate IAAS performance results?*

**Approach:** 1) Understand the effect of IAAS on performance dependability. 2) Create performance anomaly detection that can't be cheated.

# Conclusion

1. Related Work
2. Future Work
- 3. Take Aways**

- Performance anomalies manifest infrequently but their impact is significant
- Anomaly root causes manifest differently in combination than in isolation.
  - Under some settings, these interactions actually block manifestations (*emergent “good” behavior*)
- Anomaly manifestations are more likely under certain workload-policy settings, allowing machine-learned models (*EntomoModel*) to predict and avoid them.