DAPA: Diagnosing Application Performance Anomalies for Virtualized Infrastructures

Hui Kang, Stony Brook University
Xiaoyun Zhu, VMware
Jennifer Wong, Stony Brook University
Performance management in cloud computing

• Actions taken by the system administrator
  – Localize the faulty application or system components
  – Find the root cause
  – Bring the system back to normal status
Existing solutions

• Domain knowledge in specific application/platforms and associated performance best practices, e.g., [B2007] (*labor intensive, non-scalable*)

• Dependency model approach, e.g., [B2003, K2005]
  – *heavy instrumentation*
  – *packet trace on each node*

• ML techniques modeling the relationship in distributed systems, [C2004] (*complexity high, unfit for online analysis*)

• Analytic performance models for adaptive resource provisioning of VMs, e.g., [P2009] (*model inputs determined by human experts, redundant and non-scalable*)

Mostly designed in the Absence of Virtualization
Characteristics of virtualized infrastructures

- **Performance loss** caused by resource contention between VMs in the same host
- Fine-grained, system-level performance counters
- VM migration
- Strong dependency between VMs and their hosts => strong correlated metrics
  - negative modeling accuracy
  - wasting computing time
DAPA – *Diagnosis And Performance Analysis*

- **System metrics**
- **Application response time**

- **Metric collection**
- **SLA monitor**

**Monitoring**

**Potential SLA violation detected!**

- **Model pre-construction**
- **Metric pre-selection**

**Modeling**

**SLA violation detected within a time window!**

- **Model clustering**
- **Model reconciliation**

**Diagnosis**

- **Suspicious metrics from VMs or hosts**
DAPA typical deployment

1. Cloud Infrastructure
2. DAPA Monitoring
3. Cloud Provider DAPA Analysis Engine
4. System Administrator
5. Virtualization Layer
6. Anomalous VM & host attributes

Cloud User

Metric Collector

SLA Monitor

DAPA

Analysis Engine

Modeling

Diagnosis

Virtualization Layer
Online monitoring

- Collection application performance and system metrics periodically
  - Transaction response time
  - cpu utilization, memory usage, etc
- Raise an alarm on a potential SLA violation
Potential SLA violation

• A more stringent condition than true SLA violation
• Always occurs prior to a true SLA violation
• Trigger of the modeling phase

Example
  • SLA violation: “more than 10% of the response time for a certain type of transaction within one minute are greater than one second”.
  • Potential SLA violation: At least 5% are greater than one second”

• Benefits – capturing the characteristics of the system in both normal and abnormal status
Modeling

- System metrics
- Application response time

- Metric collection
- SLA monitor

- Monitoring

- Potential SLA violation detected!

- Model pre-construction
- Metric pre-selection
- Model series construction

- SLA violation detected within a time window!

- Model clustering
- Model reconciliation

- Suspicious metrics from VMs or hosts

Diagnosis
Challenges

• Problems:
  – Lots of metrics and redundancy due to virtualization
  – Sensitivity of the models to data variations
  – Model overtuned

• Approach: need an appropriate modeling technique
Statistics

• Least angle regression (LARS) [E2004]
  – Input: $X_1, X_2, \ldots, X_m$ – predictor variables and $Y$ – response variable. All are $n$-dim vectors
  – Output: A set of vector of regression coefficients $\beta$ that minimize the RSS, subject to a constraint on the sum of $\beta$.

\[
\min \| Y - \hat{Y} \| = \sum_{j=1}^{n} \left( y_j - \sum_{i=1}^{m} \beta_i X_{ji} \right)^2, \text{s.t. } \sum_{i=1}^{m} |\beta_i| \leq t.
\]

 RSS \quad \text{L}_1 \text{ penalty}

Prof. Bradley Efron
(Main inventor of LARS)
LARS regression

- Example – [E2004]

LARS

Ridge regression

Coefficients become non-zero one at a time

All coefficients become non-zero at the same time
LARS in DAPA

- Data collected by DAPA
  - System metrics as the predictor variables, e.g., CPU %, disk I/O
  - Application performance as the response variable, e.g., response time of a search request

- At each step, one metric is included into the learned model, i.e., its coefficient becomes non-zero

- Determine the stop condition such that an appropriate subset of metrics are included in the model (*Metric pre-selection*)
Metric pre-selection

- A preliminary step before potential SLA violation
- $C_{p,i}$: the risk of including a metric in the model

```
1  Setup Regression stop condition;
   Setup corrThreshold;
   $C_{min} = INT\_MAX; \text{ inc } = 0$;
5  While (inc not exceed stop condition)
   Do
     $C_{p,i}$ = including $X_i$ in the set
     If $C_{p,i} < C_{min}$
       keep the metric in the output,
       improving model fit and reduce risk
10  Else
     $X_i$ increase the risk
     inc++;
     Check the correlations between $X_i$
     and other metrics already in the set;
     remove those exceeding the corrThreshold;
15  $C_{min} = C_{p,i}$;
End
```

Output the most important metrics for modeling the system
DAPA - Diagnosis

- Build a series of models using pre-selected metrics
- A true SLA violation triggers the diagnosis phase

Model pre-construction

Metric pre-selection

Model series construction

Modeling

System metrics

Application response time

Suspicious metrics from VMs or hosts

Model clustering

Model reconciliation

SLA violation detected within a time window!

detected!
Model clustering and reconciliation

- $k$-means clustering ($k=2$) classifies models into two clusters: SLA compliance and violation

- Aggregate the data samples of the SLA violation cluster and create a new LARS model
  - Metrics ordered by their impacts to the SLA violation
Experiment setup

• Deploy DAPA prototype in the virtualized environment
• Inject several synthetic faults into the system, causing SLA violations
• Metric pre-selection is crucial
• Localize the resource bottleneck in a component related to the SLA violation
System architecture

- Xen 4.0 hypervisor and all VM Fedora 13
- Olio – a multi-tier social network application
- Trace driven workloads [A2000]
Data collection

- Application performance
- Critical system metrics
  - CPU, memory, disk, network, OS
- Runs in 5 second interval

- Application performance metrics is the tag search transaction response time

- SLA violation – “> 10% of the response time within one minute are greater than one second”

<table>
<thead>
<tr>
<th>Category</th>
<th>Metrics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>uc</td>
<td>CPU utilization in user mode</td>
</tr>
<tr>
<td></td>
<td>sysc</td>
<td>CPU utilization in kernel mode</td>
</tr>
<tr>
<td></td>
<td>wait</td>
<td>CPU idle waiting for outstanding I/O requests</td>
</tr>
<tr>
<td>Memory</td>
<td>act</td>
<td>the amount of active memory</td>
</tr>
<tr>
<td></td>
<td>inact</td>
<td>the amount of inactive memory</td>
</tr>
<tr>
<td></td>
<td>si</td>
<td>amount of memory swapped in from disk</td>
</tr>
<tr>
<td></td>
<td>so</td>
<td>amount of memory swapped out to disk</td>
</tr>
<tr>
<td>Disk</td>
<td>bo</td>
<td># blocks sent to the block device</td>
</tr>
<tr>
<td></td>
<td>bi</td>
<td># blocks received from the block device</td>
</tr>
<tr>
<td>Network</td>
<td>rxp</td>
<td>transmitted packets</td>
</tr>
<tr>
<td></td>
<td>txp</td>
<td>received packets</td>
</tr>
<tr>
<td>OS</td>
<td>ctx</td>
<td># context switch</td>
</tr>
<tr>
<td></td>
<td>in</td>
<td># interrupts</td>
</tr>
</tbody>
</table>
Synthetic faults injected

- Inject one single fault at a time
  - I/O intensive processes in an app-tier VM
  - Ballooning down the available memory of a DB VM
  - Add CPU-intensive VM to contend with an app-tier VM
- Each LARS model built on 120 data samples
- 10 models (100 minutes run) for each fault

![Application response time of tag search transactions under I/O fault case](image)

**Potential SLA violation**

**Fault injected**

**SLA violation**
Results – metrics pre-selection

- Reduce number of metrics by 1/3, from 54 to 32
- Eliminate highly correlated metrics
- The scales of Y-axis are different
## Results – selected metrics

<table>
<thead>
<tr>
<th>Injected faults</th>
<th>Top three metrics from DAPA</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk</td>
<td>io $\text{wait}_{vm1}$, bo$\text{vm}_1$, cxt$\text{vm}_3$</td>
<td>IO wait, block out, and cxt switch selected; intensive I/O activities on vm1 and vm3 affected</td>
</tr>
<tr>
<td>Memory</td>
<td>act$\text{vm}_4$, so$\text{vm}_4$, si$\text{vm}_4$</td>
<td>Active memory, swap out/in selected; vm4 short of memory</td>
</tr>
<tr>
<td>CPU</td>
<td>hc$\text{vm}_1$, hc$\text{vm}_2$, htx$\text{vm}_1$</td>
<td>Host CPU resource contention between vm1 and vm2;</td>
</tr>
</tbody>
</table>
Conclusion

• DAPA: a model-based performance diagnosis framework
  – Consider metric variations and scalability
  – Customized modeling techniques
  – Output ranked metrics related to performance loss

• Deploy with a multi-tier web application in a virtualized infrastructure

• Localize several bottlenecks causing the SLA violations
Future work

• Incorporate internal knowledge from various system layers of the virtualization platform
  – Gray-box approach
  – Differentiate root causes between misconfiguration of VM and interference from other VMs

• Global knowledge e.g., network topology

• Automatically finding the causal relationship between VM instances in the hosted application
Thank you! 😊
References


Results – metrics ranking (I/O fault case)