Correlating instrumentation data to system states: A building block for automated diagnosis and control

Ira Cohen
Ira.Cohen@hp.com

Joint work with: Moises Goldszmidt, Terence Kelly, and Julie Symons of HPLabs and Jeff Chase of CS Duke

© 2004 Hewlett-Packard Development Company, L.P.
The information contained herein is subject to change without notice
Motivation

• Complexity of deployed systems surpasses the ability of humans to diagnose, forecast, and characterize behavior
  – Scale and interconnectivity
  – Levels of abstraction
  – Changes in input, applications, infrastructure
  – Lack of closed-form mathematical characterization

• Need for automated diagnosis tools
Approaches

• A priori models of system structure and behavior
  - Difficult and costly
  - Brittle to changes and unanticipated conditions

• Models obtained automatically from historical data using induction techniques
  - Economical
  - Minimally invasive
  - Require little knowledge
  - Adapt to changes

Our approach
This work

- Study of the application of a statistical pattern recognition and probabilistic modeling for performance diagnostics and forecasting in a transactional (3-tier) service

- Goal: Analyze streams of instrumentation data collected from service components and select a focused set of metrics and thresholds that correlated to specific performance problems
Defining a performance problem: Service Level Objectives (SLO)

Which system metrics correlate with violations?
What values of these metrics correlate with the violations?
Are there different “types” of violations?
System metrics (too many to plot all)

Example metrics: CPU metrics, Memory, I/O, Network activity coming in and out of servers, Swapspace usage, Paging, etc…
Overview:

- **httpperf**
- **Apache**
- **BEA WebLogic**
- **Oracle**
- **W2K Server**
- **W2K Server**
- **W2K Server**
- **SLO compliance indicator**
- **metrics**
- **logs**
- **HP OpenView**
- **Statistical analysis/model induction engine**
- **F(M) --> SLOstate**

(Chart showing interactions and relationships between the mentioned components and processes.)
Outputs of statistical engine

In training phase:

• Subset of metrics most correlated with SLO state
  − “most correlated” = can predict SLO state with highest accuracy.
  − Feature selection is used to find the subset.

• Classifier (in the form of a probability distribution) providing a mapping from metrics to SLO state.

In usage (testing) phase:

• Accuracy of classifier in predicting SLO state
• Alarms on individual metrics when SLO is violated
The model: a classifier
F(M) \to \text{SLOstate}

- Consists of two parts
  - A probabilistic model of \( <M, \text{SLOstate}> \) (TAN): 
    \[ P(M, \text{SLOstate}) \]
  - A decision function on which state is more likely
    \[ P(S^- | M) > P(S^+ | M) \]

- Measures of success
  - Percentage of patterns captured by the model
  - Balanced Accuracy =
    \[ 0.5(\text{Prob}(F(M) = s^- | s^-) + \text{Prob}(F(M) = s^+ | s^+)) \]

Where, \( s^- = \text{SLO not violated} \), \( s^+ = \text{SLO violated} \)
TAN Classifiers

- Tree Augmented Naïve Bayes (TAN) [Friedman, Geiger, Goldszmidt 96-97]
  - Enables Interpretability (provides thresholds for individual metrics)
  - Efficiency of representation and computation
  - Takes into account correlations between metrics
  - Can fuse expert knowledge with statistical data

- Interpretability (attribution): Use log likelihood difference to determine which metric most likely contributed to an SLO violation:
  $$[\log P(m_i | \text{parent}(m_i), s-) - \log P(m_i | \text{parent}(m_i), s+)]$$
  where \(i = 1, \ldots, \text{Number of metrics},\)
TAN example

P(DB SwapspaceUsed | DB NETIF InByte, SLO state)
Selecting the metrics

- Find a subset of the metrics $M$ that maximizes the Balanced Accuracy score

- Heuristic search:
  - Start with an empty set of metrics
  - Greedily add the metric that most increases the BA score

- Use tenfold cross validation to compute BA
Example: Model Report

Input file name: Exper20  SLO Threshold: 928ms Percentile: 80%
Task: Diagnose

Top metrics ranked

<table>
<thead>
<tr>
<th>Metric</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean_IDC2_CODA_CPU_1_USERTIME</td>
<td>92.739</td>
</tr>
<tr>
<td>mean_IDC2_CODA_DISK_1_PHYSREAD</td>
<td>94.703</td>
</tr>
<tr>
<td>mean_IDC2_CODA_GBL_PAGEOUT</td>
<td>94.374</td>
</tr>
<tr>
<td>var_IDC5_CODA_NETIF_2_INPACKET</td>
<td>93.255</td>
</tr>
<tr>
<td>var_IDC2_CODA_DISK_1_PHYSBYTE</td>
<td>89.024</td>
</tr>
</tbody>
</table>

Number of metrics chosen for classifiers: 2

<table>
<thead>
<tr>
<th>Model</th>
<th>False Alarms</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAN:</td>
<td>4.0205 +/− 2.3199</td>
<td>93.427 +/− 5.3085</td>
</tr>
</tbody>
</table>
Example: Attribution Report

mean_IDC2_CODA_CPU_1_USERTIME
mean_IDC2_CODA_DISK_1_PHYSREAD

Alarms statistics
Total instances of SLO violation: 457
Alarm Attribution combination: Missed 5
Alarm Attribution combination: Disk 14
Alarm Attribution combination: CPU 398
Alarm Attribution combination: CPU+Disk 40
Total instances where likelihood triggers alarm: 450
Example: Map of SLO violation types
Illustration: Decision boundary
Questions

• How good are the TAN models?
  − Do they capture a significant percentage of the patterns of SLO behavior?
  − Is the CPU utilization enough?

• How many metrics are needed?

• Do we need the models to adapt?
  − To different thresholds of SLO?
  − To different inputs?
  − Is this adaptation “crazy”?

• Is the output of the models helpful in determining the cause of the performance problem?
Testbeds (1) HP Labs testbed

- Three tier system
- Petstore web application running
- OpenView measurements (CODA), apache log used to extract response time.
- Workload induced performance problems
Testbeds (2)

- OpenView Germany testbed:
  - Single webserver
  - Client requests files from system – constructed to ensure disk access
  - SAR measurements
  - External application contending for system resources
Experiments on HPL testbed

Workload induced performance problems:

• Experiment 20: increasing ramp
  - Increase the number of simultaneous sessions
  - Add a new client every 20 minutes
  - Up to 100 sessions in 36 hours
  - Requests rate is a sinusoidal wave over a ramp

• Experiment 23: bursts with two workload
  - Workload 1: steady background traffic of 1k requests per minute and 20 clients
  - Workload 2: On-off every hour, with bursts of 5, 15, 20,…,50 clients (each session of 50 requests per minute)
  - Overall time = 36 hours
Output (Avg resp times) from Experiments

Experiment 20

Experiment 23
Experimental loop:

1. For each Experiment (20,23)
2. For SLO threshold = 60,61,...,90 percentile of average response time
3. Generate list of SLO violation instances using SLO threshold on the average response time
4. Perform metric selection using greedy search.
5. Learn TAN model with selected metrics.
6. Evaluate model using 10-fold cross validation: Measure balanced accuracy, false alarm and detection rates
7. Evaluate performance using only Application server CPU metric
8. if (SLO threshold == 60)
9.     Save model as Mod1
10. else
11.     Evaluate Mod1 on current SLO definition
12. Check metric attribution of current TAN model for each instance of SLO violation
Experiment 23: Balanced Accuracy
Experiment 23: False Alarms and Detection Rate

- **False Alarm Rate**
  - SLO Threshold (μsec)
  - TAN, CPU, MOD

- **Detection Rate**
  - SLO Threshold (μsec)
  - TAN, CPU, MOD
Questions answered (1)

- Does TAN capture a significant percentage of the patterns of SLO behavior?
- Is the CPU utilization enough?
- How many metrics are needed?
- Do we need the models to adapt to different thresholds of SLO?
Questions answered (2)

- Do we need the models to adapt to different workloads?
- Is the adaptation “crazy”?

<table>
<thead>
<tr>
<th>Metric/exper #</th>
<th>20</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean_IDC2_CODA_CPU_1_USERTIME</td>
<td>27</td>
<td>7</td>
</tr>
<tr>
<td>mean_IDC2_CODA_DISK_1_PHYSREAD</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>mean_IDC2_CODA_DISK_1_BUSYTIME</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>var_IDC2_CODA_DISK_1_BUSYTIME</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>mean_IDC5_CODA_DISK_1_PHYSWRITEBYTE</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>var_IDC5_CODA_GBL_SWAPSPACEUSED</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>var_IDC5_CODA_NETIF_2_INPACKET</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>mean_IDC5_CODA_GBL_SWAPSPACEUSED</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>mean_IDC5_CODA_GBL_RUNQUEUE</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>var_IDC2_CODA_CPU_1_USERTIME</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>var_IDC5_CODA_NETIF_2_INBYTE</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>var_IDC5_CODA_DISK_1_PHYSREAD</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>var_IDC2_CODA_GBL_MEMUTIL</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>numReqs</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>var_IDC5_CODA_DISK_1_PHYSWRITE</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>var_IDC5_CODA_NETIF_2_OUTPACKET</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Results from OV testbed

- External application contending to resources.
  Three tests:
  I/O bottleneck, Memory bottleneck, Disk bottleneck
## Results from OV testbed

<table>
<thead>
<tr>
<th>exper.</th>
<th>SLO</th>
<th>TAN BA</th>
<th>TAN Det</th>
<th>TAN FA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk</td>
<td>204</td>
<td>92.5</td>
<td>88.3</td>
<td>3.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±12.9</td>
<td>±15.3</td>
<td>±10.5</td>
</tr>
<tr>
<td>Mem</td>
<td>98</td>
<td>99.5</td>
<td>99.6</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.3</td>
<td>±0.01</td>
<td>±0.6</td>
</tr>
<tr>
<td>I/O</td>
<td>73</td>
<td>97.9</td>
<td>97.8</td>
<td>1.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±1.4</td>
<td>±2.4</td>
<td>±0.4</td>
</tr>
</tbody>
</table>
Is the output of the models helpful in determining the cause of the performance problem?

<table>
<thead>
<tr>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldavg-1:</td>
</tr>
<tr>
<td>plist-sz:</td>
</tr>
<tr>
<td>System load average for the last minute</td>
</tr>
<tr>
<td>Number of processes in the process list.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>pgpgout/s</td>
</tr>
<tr>
<td>txpck/s:</td>
</tr>
<tr>
<td>Total number of blocks the system paged out to disk per sec.</td>
</tr>
<tr>
<td>Total number of packets transmitted per sec (On the eth0 device)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>tps:</td>
</tr>
<tr>
<td>activepg:</td>
</tr>
<tr>
<td>Number of transfers per second that were issued to the device (I/O)</td>
</tr>
<tr>
<td>Number of active (recently touched) pages in memory.</td>
</tr>
<tr>
<td>kbbuffers:</td>
</tr>
<tr>
<td>Amount of memory used as buffers by the kernel in kilobytes.</td>
</tr>
<tr>
<td>kbswpfree:</td>
</tr>
<tr>
<td>Amount of free swap space in kilobytes.</td>
</tr>
<tr>
<td>totsck:</td>
</tr>
<tr>
<td>Total number of used sockets.</td>
</tr>
</tbody>
</table>
Conclusions from experiments

- TAN models good enough
  - They capture a significant percentage of patterns of SLO (low 90’s)
  - CPU utilization isn’t enough
- Reducing information overload: a handful of metrics are chosen (between 3-8)
- Adaptation is needed
  - To different thresholds of SLO
    - False Alarms increase otherwise
    - Mostly on the parameterization of the model
  - To workload/type of problem
    - Typically using different metrics
- Chosen metrics typically point human analysts in the right direction.

Take home: The method does not “convict the guilty”, it “clears the innocent”
Approach as a tool...

- **Diagnosis (today)**
  - Which subset of metrics correlates with SLO state?
  - What are the relevant values/regions of these metrics?
  - What are the “types” of SLO violations?

- **Root cause analysis (holy grail)**
  - What are the root causes of the violations?

- **Forecasting (in progress)**
  - See Rob Powers next week
  - Find early indicators of SLO violations

- **Control and auto tuning (starting)**
Next steps:

- From forensics to online:
  - Continuously induce new models as needed
  - Manage models running in parallel
  - Online updating of models

- Create chronicles of system behavior from output of models.

- Tier attribution: Determine the tier(s) that might be contributing to the global SLO violation
Concluding remarks

• Explored a pattern recognition and probabilistic modeling approach to diagnosing service level objective problems
  − Automated: relieves the user from inspecting hundreds of time series of metrics (their combinations!) and the possible correlations with the SLO
  − Adaptive: works for a variety of SLO definitions and with a variety of infrastructures and applications
  − Economic: does not rely on costly expert, and it is minimally invasive. Does not rely on modifications to infrastructure

• Not such a big hammer:
  − Freeware available – not worse than a regression
  − We have enough computer power in a laptop

• Could combine expert knowledge with statistical data
  − Bayesian networks
TAN Classifiers

- Metrics form a tree. Joint distribution defined as:
  \[ P(\text{Class, Metrics}) = P(\text{Class}) \prod_{i=0}^{N} P(\text{Metric } i \mid \text{Parent}(i), \text{Class}) \]

- Use Bayes rule to get classification
  \[ P(\text{Class} \mid \text{Metrics}) \propto P(\text{Class}) \prod_{i=0}^{N} P(\text{Metric } i \mid \text{Parent}(i), \text{Class}) \]

- There is an efficient algorithm for finding most likely tree (Chow-Liu algorithm)
Is it always the CPU?
A combination of metrics
Premises

• It is enough to identify system states
• Events of interest are defined externally
• Existence of repeated patterns
• Observe “enough” of these patterns
Next steps: Tier SLO and attribution

Unobservable variables

Global SLO