Dynamic Adaptation of Temporal Event Correlation Rules

Rean Griffith‡, Gail Kaiser‡
Joseph Hellerstein*, Yixin Diao*

Presented by Rean Griffith
rg2023@cs.columbia.edu
‡ - Programming Systems Lab (PSL) Columbia University
* - IBM Thomas J. Watson Research Center
Overview

- Introduction
- Problem
- Solution
- System Architecture
- How it works – Feed-forward control
- Experiments
- Results I, II, III
- Conclusions & Future work
Introduction

- Temporal event correlation is essential to realizing self-managing distributed systems.
- For example, correlating multiple event streams from multiple event sources to detect:
  - System health/live-ness
  - Processing delays in single/multi-machine systems
  - Denial of service attacks
  - Anomalous application/machine-behavior
Problem

- Time-bounds that guide event stream analysis are usually fixed. Based on “guesstimates” that ignore dynamic changes in the operating environment.
- Fixed time-bounds may result in false-alarms that distract administrators from responding to real problems.
- Issues with client-side timestamps (even with clock synchronization).
Solution

- Use time-bounds as the basis for temporal rules, but introduce an element of “fuzz” based on detected changes in the operating environment.

- To detect changes in the operating environment introduce Calibration Event Generators which generate sequences of events (Calibration frames) at a known resolution.

- Use the difference in the arrival times of calibration events to determine the “fuzz” to use.

- Only time-stamps at the receiver count.
System Architecture
How it works – Feed-forward Control

- Use the difference in the arrival times of calibration events within a calibration frame (less the generator resolution) as an observation of "propagation skew".
- Record last N observations of propagation skew.
- Sort these observations and use the median as the "fuzz" to add to timer rules.
- Using the median prevents overreaction to transient spikes.
## Experiments

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Machines</th>
<th>Operating System</th>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 3-machine</td>
<td></td>
<td>Linux 2.6</td>
<td>3GHz, 1GB</td>
<td>RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B 3-machine</td>
<td></td>
<td>Siena Event</td>
<td>N/A</td>
<td>Event Distiller</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Router</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C 2-machine</td>
<td></td>
<td>Calibration</td>
<td>N/A</td>
<td>Event Distiller</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Event Generator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D 2-machine</td>
<td></td>
<td>Calibration</td>
<td>N/A</td>
<td>Event Distiller</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Event Generator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results I – Propagation Skews

3-machine

Windows + Linux

2-machine

All Linux
Results II - Autocorrelations

3-machine

A

B

2-machine

C

D

Windows + Linux

All Linux
Results III – Sensitivity to N (Run 3 Configuration C)

Most accurate N (observation window size) depends on:
Actual conditions AND initial fuzz factor setting
Generator set to produce 241/445 “real” failures
With large N we use initial fuzz factor longer, erroneously reporting fewer “real failures” (when we’re missing real problems)

Initial fuzz factor setting = 0 ms
85%-90% accuracy with smaller N

Initial fuzz factor setting = 500 ms
80%+ accuracy with smaller N.
Conclusions

- There is more to our notion of “propagation skew” than network delays. Resource contention at the receiver on certain platforms as seen in configuration C (2-machine Linux + Windows setups) also affects our observations.

- Near optimal settings automatically achieved by managing the tradeoff between larger observation windows and the ability to respond quickly to changes in the environment.

- Feed-forward control useful in building self-regulating systems that rely on temporal event correlation.
Comments, Questions, Queries

Thank you for your time and attention.

Contact: Rean Griffith
rg2023@cs.columbia.edu
Event Package

- Events Represented as Siena Notifications of size ~80 bytes

\[ E_1 = \{ \text{FPGenGap} = "0", \text{FPResolution} = "2000", \text{FPSeqNum} = "1", \text{FPLastSeq} = "1", \text{FPTest} = "FPTest" \} \]
\[ E_2 = \{ \text{FPGenGap} = "2041", \text{FPResolution} = "2000", \text{FPSeqNum} = "1", \text{FPLastSeq} = "0", \text{FPTest} = "FPTest" \} \]