Performance Debugging
for Distributed Systems
of Black Boxes

Marcos K. Aguilera, Jeffrey C. Mogul, Janet L. Wiener, Patrick Reynolds, Athicha Muthitacharoen

2012–23204  Sejin Kim
Contents

[1] Problem statement, goals, and non-goals
  [3.1] Nesting algorithm
  [3.2] Convolution algorithm
  [3.3] Comparison between Nesting and Convolution
[4] Experiments and results
[5] Summary
[1] Problem statement

- Distributed systems can be very hard to debug, especially when they exhibit poor performance. And it becomes much harder when they are composed of "black-box" components.

- Isolating performance bottlenecks is hard.
[1] Problem statement

• Causal path
  – a series of node traversals where each traversal is caused by some message from a prior node on the path

Figure 1: Example multi-tier application showing a causal path
[1] Goals

To Do
• Find high-impact causal path patterns
• Identify those nodes on high-impact patterns that, as participants on these patterns, add significant latency to the patterns

Should Do
• Require minimal knowledge
• Require no modifications to applications, middleware, messages, or workloads
• Not significantly perturb system performance
[1] Non-goals

- We are not developing tools to replace the need for programmers
- Our tools are not meant to verify correct system behavior, or diagnose the causes of faulty behavior
- Our tools are not aimed at characterizing or benchmarking system performance
- Because our tools are aimed at the debugging phase, we do not require real-time results; we are willing to use offline analysis

• Our approach relies on tracing the messages between the nodes, and using one of several offline algorithms to infer causality from these traces

• Phases
  1 Exposing and tracing communication
  2 Inferring causal paths and patterns
  3 Visualization

[3.1] Nesting algorithm

- ‘Nested’
  - B<->C call is nested in the A<->B call
  - B<->D call is nested in the A<->B call

![Figure 3: Timelines for example of Figure 2](image-url)
[3.1] Nesting algorithm

- Reconstruct call paths
  - \(A \rightarrow B \rightarrow [C;D]\)
  - \(O(mp), m:=\text{number of messages, } p:=\text{mean of per-node parallelism}\)
[3.1] Nesting algorithm

- **Medium-length delay**
  - $t_3 - t_1$ and $t_4 - t_2$

- **Long delay**
  - $t_4 - t_1$

- **Short delay**
  - $t_3 - t_2$

- **Histogram for (A,B,C) has a peak at the medium-length delay**
[3.1] Nesting algorithm

• **Pros**
  – Linear time
  – More compact algorithm than Convolution

• **Cons**
  – RPC communication

[3.2] Convolution algorithm

- Look for time-shifted similarities
- Spike position indicates time shifted

Figure 7: Example of convolution output, showing two spikes with bold lines. The x-axis represents the time shift; the y-axis roughly estimates the number of messages matching a given shift.
[3.2] Convolution algorithm

- $O(em + eS \log S)$
  - $m$ := number of messages
  - $e$ := number of edge in output graph
  - $S$ := number of time steps in trace

- **Time step size**
  - Must be shorter than delays of interest
  - Too coarse: poor accuracy
  - Too fine: long running time
[3.2] Convolution algorithm

• Pros
  – Not only RPC communication but also free-form messages

• Cons
  – Polynomial time
  – Less compact algorithm than Nesting

## [3.3] Comparison between Nesting and Convolution

<table>
<thead>
<tr>
<th></th>
<th>Nesting Algorithm</th>
<th>Convolution Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication style</td>
<td>RPC only</td>
<td>RPC or free-form messages</td>
</tr>
<tr>
<td>Rare events</td>
<td>Yes, but hard</td>
<td>No</td>
</tr>
<tr>
<td>Level of Trace detail</td>
<td>&lt;timestamp, sender, receiver&gt; + call/return tag</td>
<td>&lt;timestamp, sender, receiver&gt;</td>
</tr>
</tbody>
</table>
| Time and space complexity | Linear space  
Linear time                  | Linear space  
Polynomial time                                    |
| Visualization          | RPC call and return combined  
More compact | Less compact                                      |
[4] Experiments and results

Figure 10: Multi-tier configuration (simplified version of Figure 1)

Figure 12: Multi-tier results from the nesting algorithm

Figure 13: Multi-tier results from convolution algorithm
[5] Summary

• Looking for bottlenecks in black box systems by finding causal paths

• Suggests two different algorithms to find causal paths