Locating Regions of Uncertainty in Distributed Systems Using Aggregate Trace Data

Joey Dong & Anshul Rastogi
Mentors: Darby Huye, Max Liu, Dr. Raja Sambasivan
Distributed system - a bunch of computers working together. To help debug we trace our systems, but don’t trace everything. Here are two traces where there’s uncertainty, and we need more implementation (tracepoints). To do that we look at the exclusive and inclusive latencies.

Use one whole example throughout the whole presentation, and don’t just make things A, B, C, use real world examples

Tell them more about HOW you got the ex and in values, animate it, your calculations
Distributed Systems: What Are They?

Distributed systems are networks of devices/machines (servers and computers, for example) that communicate with one another to complete tasks.

Examples:
- Google
- Facebook
- Cellular networks

Distributed systems are essential to the modern world!
Instrumentation & Distributed Tracing

- **Distributed tracing**: a way to observe and record how application requests get propagated through a distributed system.
- Traces are useful for developers to see the architecture of and interactions within the system
  - Have a clearer idea of what’s going on when modifying the system codebase identifying errors
- Essential for distributed systems, given their complex structure

- Instrumentation often consists of **tracepoints** embedded into code.
- They monitor and log information.
Span Model of Tracing

Generally, applications have thousands of spans.
HotROD example
Quite different end-to-end latencies across traces → moderate uncertainty

**Uncertainty:** lack of predictability of behavior of a system based on given information

To reduce uncertainty, increase the amount of given information

→ more instrumentation!

But where to instrument?
- Parent?
- Child?

Place with more uncertainty!
Inclusive/Exclusive Latencies

**TRACE 1**

**Exclusive Latency**

**Inclusive Latency**

Assuming that parent does nothing but wait for the child’s response after creating the child:

Exclusive latency tells us how much latency the parent is responsible for.
Calculation of ex/inclusive latencies

- **Inclusive latency**: the whole length of the span
  - (end of span) - (start of span)
  - $9 - 0 = 9$

- **Exclusive latency**: the length of the span minus the time spent on waiting for other spans
  - inclusive latency - (end of child span - start of child span)
  - $9 - 0 - (4 - 1) = 6$
Overview of our algorithm

A developer notices their program has latency issues, but isn’t sure which operation caused it. Our algorithm:

1. Calculate the exclusive latency and inclusive latency values for all spans within all traces
2. For spans with same operation name, combine information across traces to get uncertainty measures.
3. User select an uncertain measure as their rubric to judge the operations
4. Operations are ranked based on this measure -> the higher ranked an operation is, the more implementation is needed there!
Our Uncertainty Measures

Two of our uncertainty measures are:

1. **Standard deviation in exclusive latency**
   a. Shows us the deviation of exclusive latency values - how much does it vary? How much uncertainty is here?
   b. Exclusive latency: Which operations are doing the most work themselves?

2. **Coefficient of variation in exclusive latency**
   a. Coeff. Of Variation is (standard deviation)/(mean) * 100%
   b. Shows us the relative dispersion of exclusive latency
Algorithm: Step 1

TRACE 1

Frontend

Database

In          Ex
Frontend    9-0 = 9     9-(4-1)-0 = 6
Database    4-1 = 3     4-1 = 3

TRACE 2

Frontend

Database

In          Ex
Frontend    13-0 = 11   13-(5-1) = 9
Database    9-5 = 4     9-5 = 4
## Algorithm: Step 2

For spans with same operation name, combine information across traces to get uncertainty measures.

<table>
<thead>
<tr>
<th></th>
<th>Trace 1 exclusive latency</th>
<th>Trace 2 exclusive latency</th>
<th>Standard Deviation in Ex</th>
<th>Coeff. Of Variation in Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontend</td>
<td>6</td>
<td>9</td>
<td>2.1</td>
<td>33%</td>
</tr>
<tr>
<td>Database</td>
<td>3</td>
<td>4</td>
<td>0.71</td>
<td>20%</td>
</tr>
</tbody>
</table>
Algorithm: Step 2 cont.

User select one from the given values as their rubric.

In this case, the user selects the std. div. in exclusive latency as their rubric.

<table>
<thead>
<tr>
<th></th>
<th>Trace 1 exclusive latency</th>
<th>Trace 2 exclusive latency</th>
<th>Standard Deviation in Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontend</td>
<td>6</td>
<td>9</td>
<td>2.1</td>
</tr>
<tr>
<td>Database</td>
<td>3</td>
<td>4</td>
<td>0.71</td>
</tr>
</tbody>
</table>
Algorithm: Step 3

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Operation Name</th>
<th>Standard deviation in Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>Frontend</td>
<td>2.828</td>
</tr>
<tr>
<td>2nd</td>
<td>Database</td>
<td>0</td>
</tr>
</tbody>
</table>

rank areas of uncertainty according to the rubric of their choosing, in this case, the std. div. in Ex
- Frontend ranked higher than database → more instrumentation added to frontend
- Developer may discover an error in the frontend that caused a cache miss to sometimes return slowly prior to reaching for the database to search for the user query.
- Knowing this would allow them to address the error
Limitations

Our algorithm...

- Assumes the presence of “skeleton” trace points to build span model (minimal instrumentation)
- Repeated span names become an issue
- Limited to informing developer about inclusive/exclusive parts of spans
Future Ideas

- Analysis of more of the architecture itself (how is one microservice connected to another, etc.) to better determine uncertainty sources

- There are multiple ways of representing traces
  - We use a span-based model
  - There are also segment-based models and event-based models, which explore dimensions of distributed systems that our span-based model inherently cannot.
Acknowledgements to:

- MIT PRIMES, for the opportunity
- Our mentors – Darby Huye, Max Liu, and Dr. Raja Sambasivan
Image Sources

https://twitter.com/google
https://twitter.com/jackkleeman/status/1190354757308862468
http://clipart-library.com/question-mark-images.html
https://jeffmcclung.com/2013/11/11/limitations/