

Conquering Microservices Complexity @Uber

With Distributed Tracing

Yuri Shkuro

SOFTWARE ENGINEER @ UBER

Agenda

Why Distributed Tracing

Trace as a Narrative

Trace vs. Trace

Traces vs. Trace

Data Lineage

Q & A

Yuri Shkuro



Software Engineer
Uber Technologies
shkuro.com



Founder & Maintainer
of CNCF Jaeger

jaegertracing.io



Co-founder of
OpenTracing &
OpenTelemetry



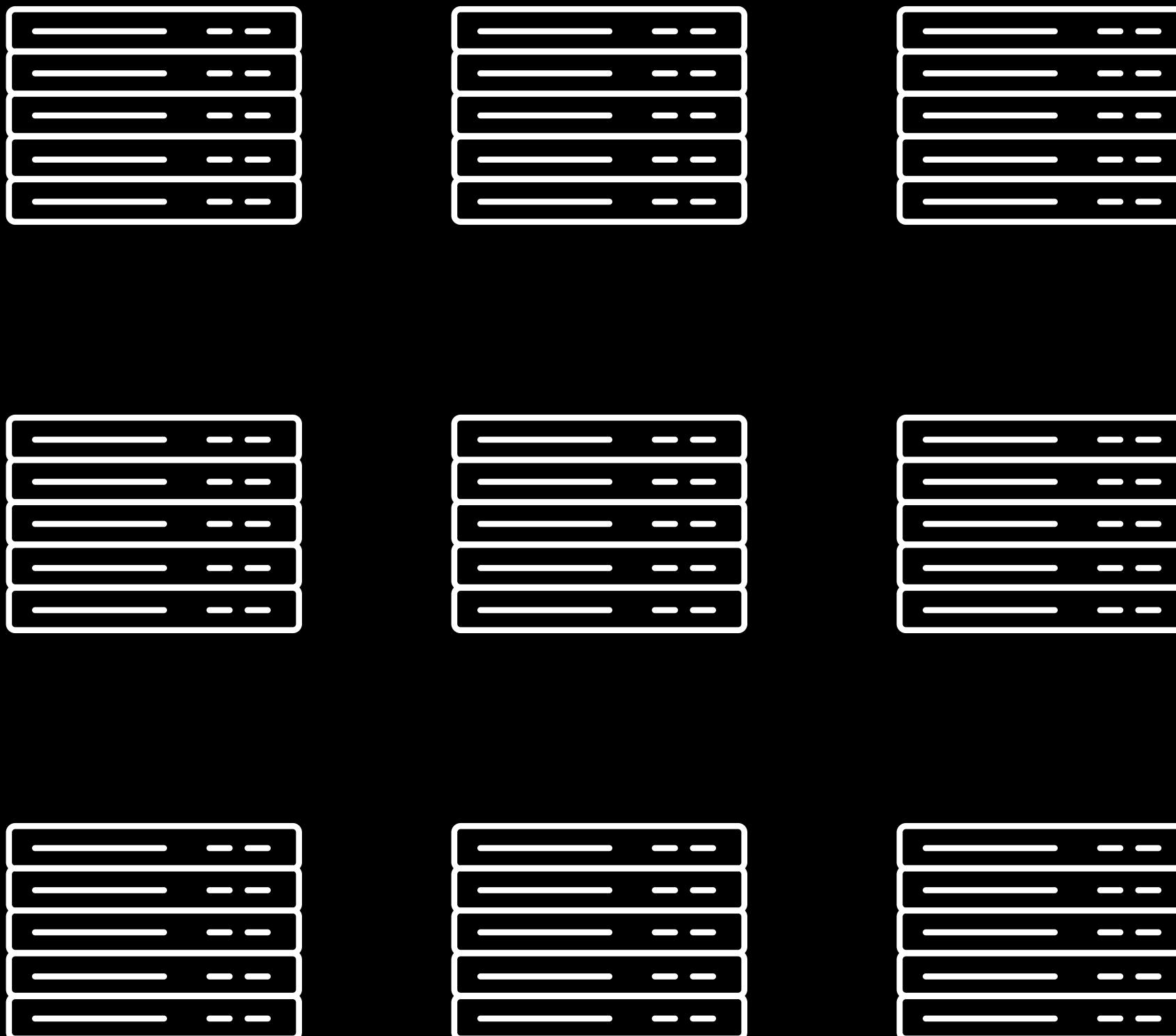
Author of "[Mastering
Distributed Tracing](#)",
by Packt Publishing

Quick Poll

Why Distributed Tracing

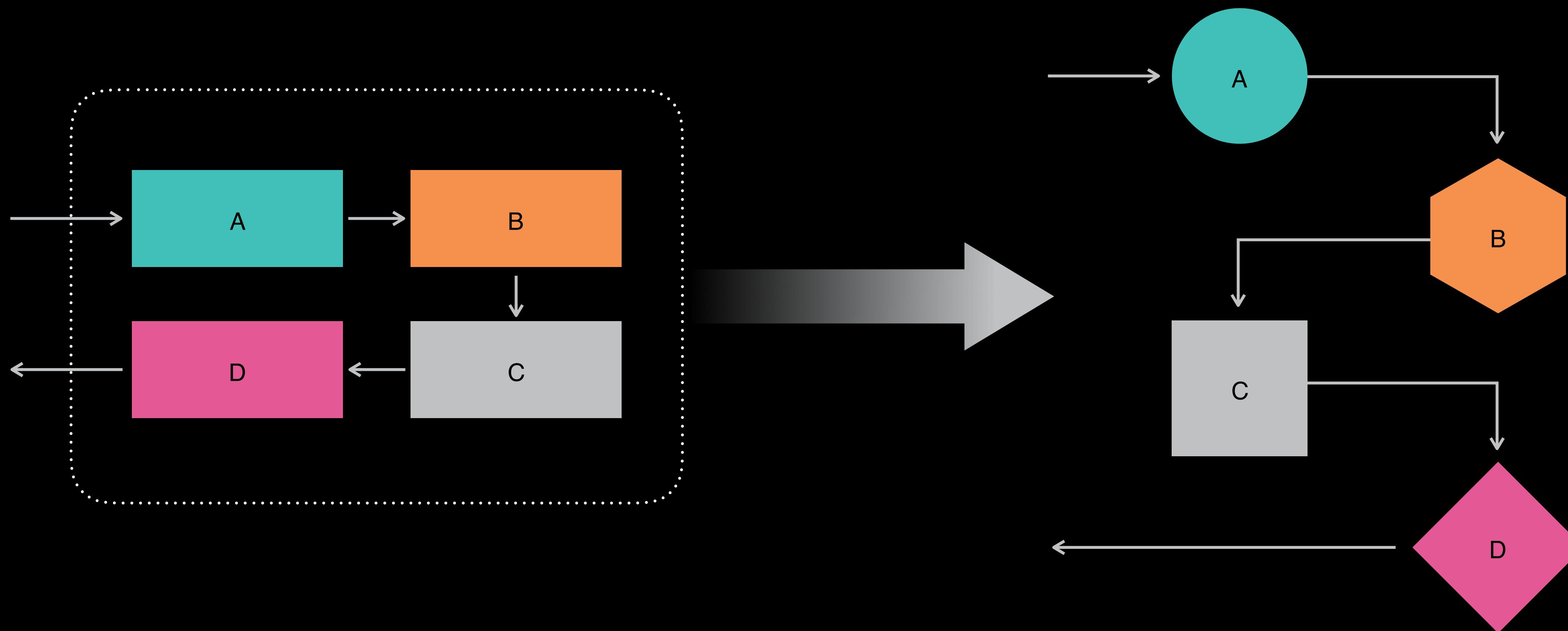
Scaling With Users

Distributed Systems



Scaling With Engineering Organization

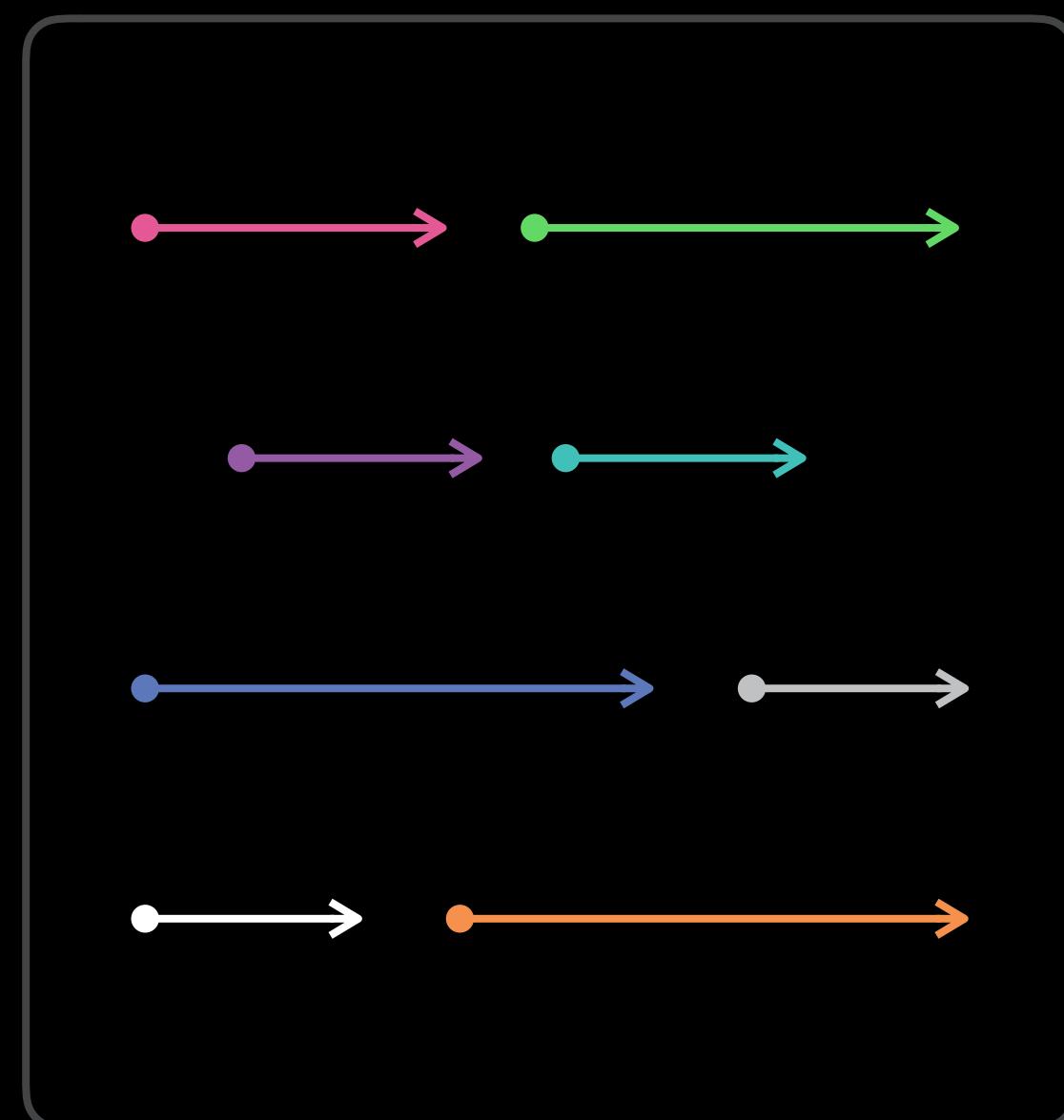
Monoliths to Microservices



Scaling With CPU Cores

Asynchronous Programming Models, Distributed Concurrency

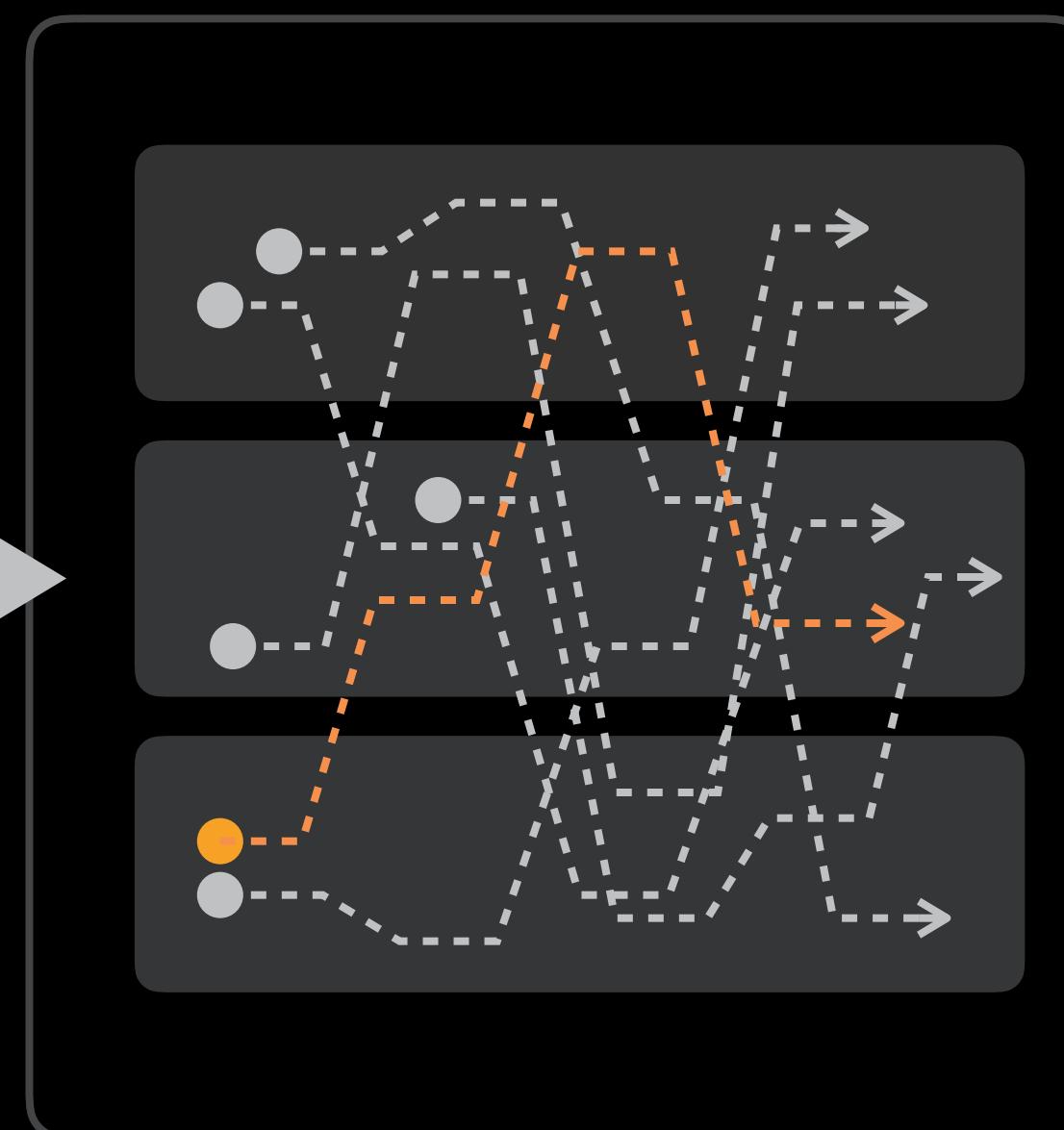
BASIC CONCURRENCY

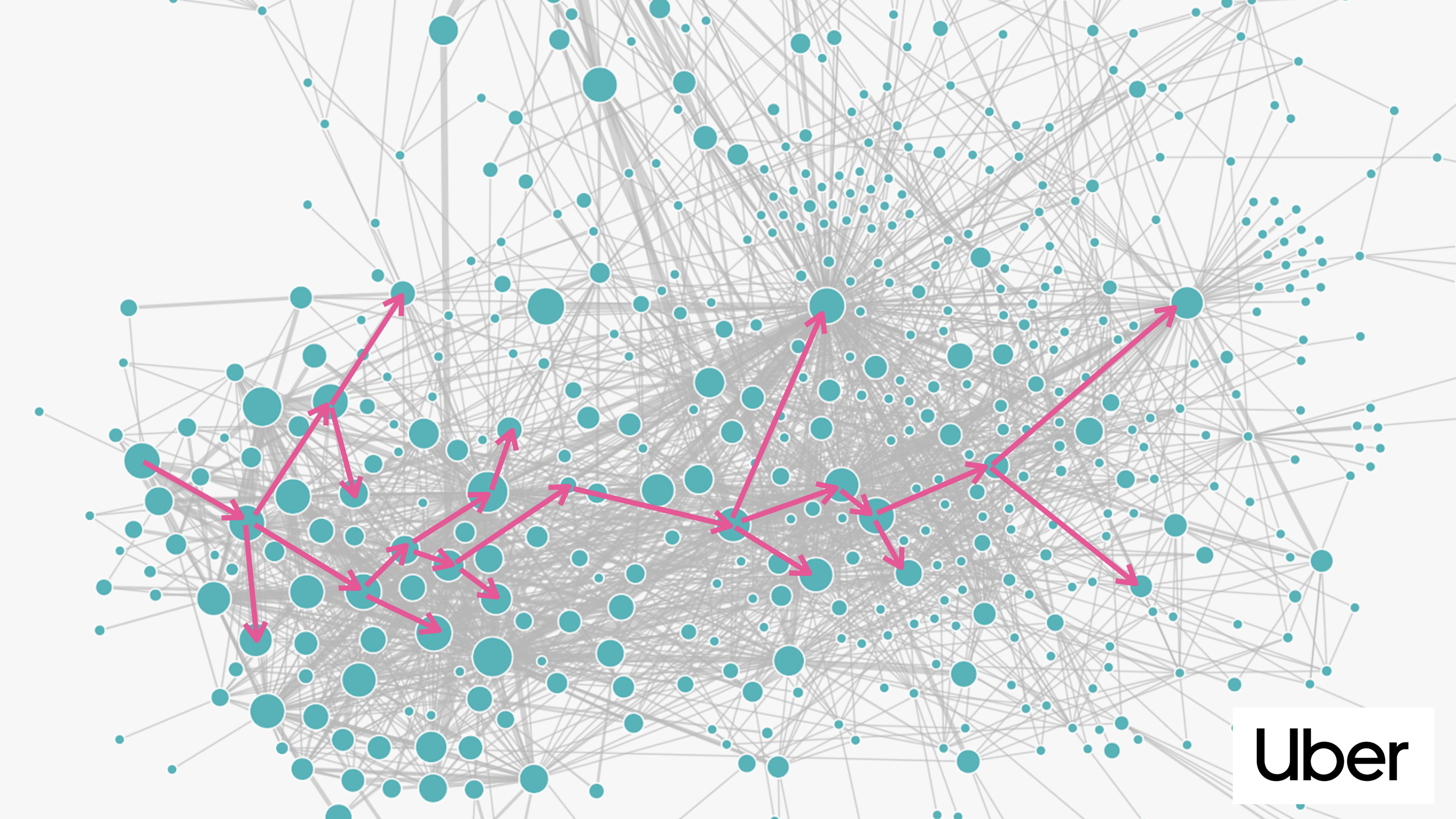


ASYNC CONCURRENCY



DISTRIBUTED CONCURRENCY





Uber

In microservices architectures
the number of failure modes
increases exponentially

Observability of
distributed transactions
is paramount!

Observability vs. monitoring

Observability vs. monitoring

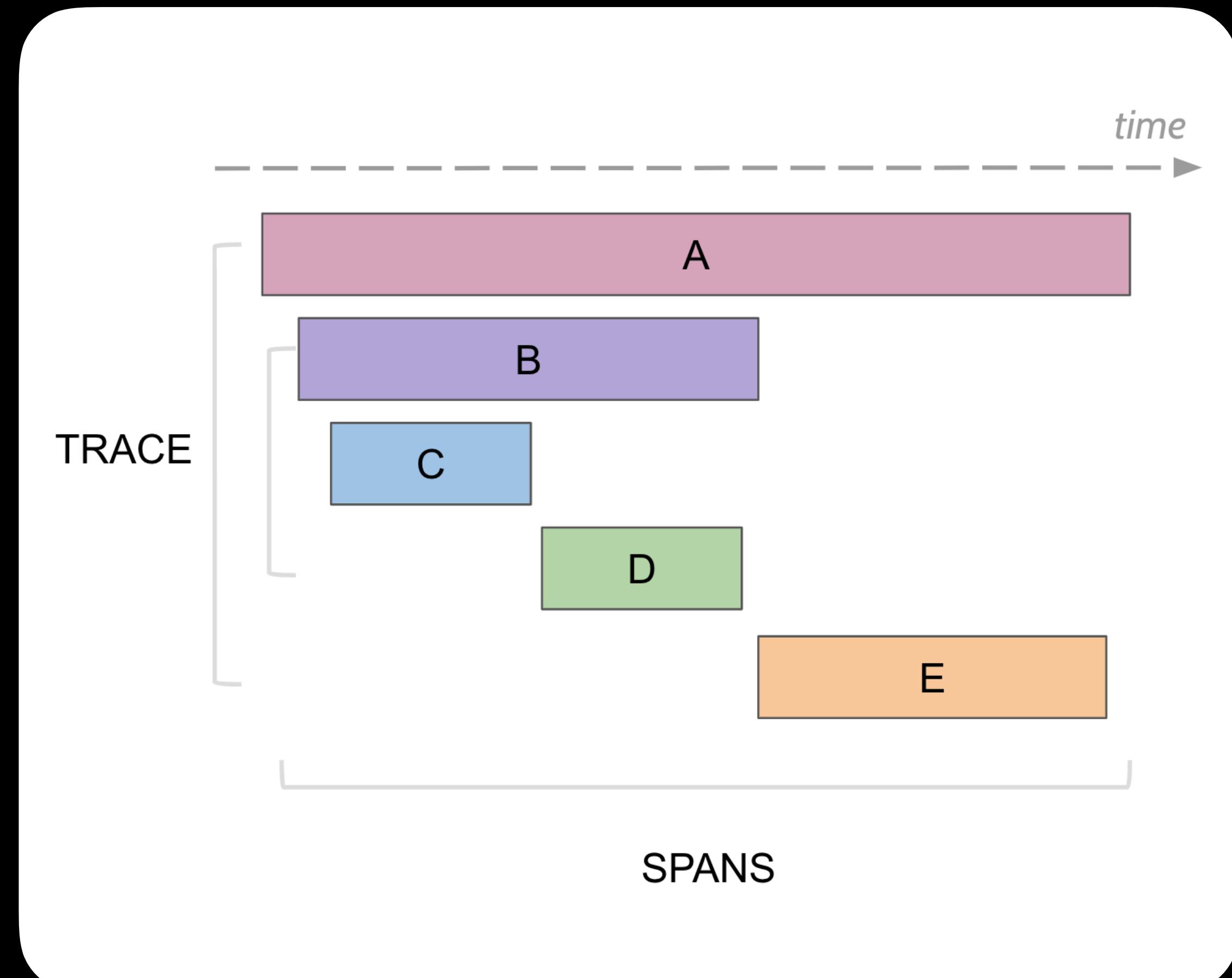
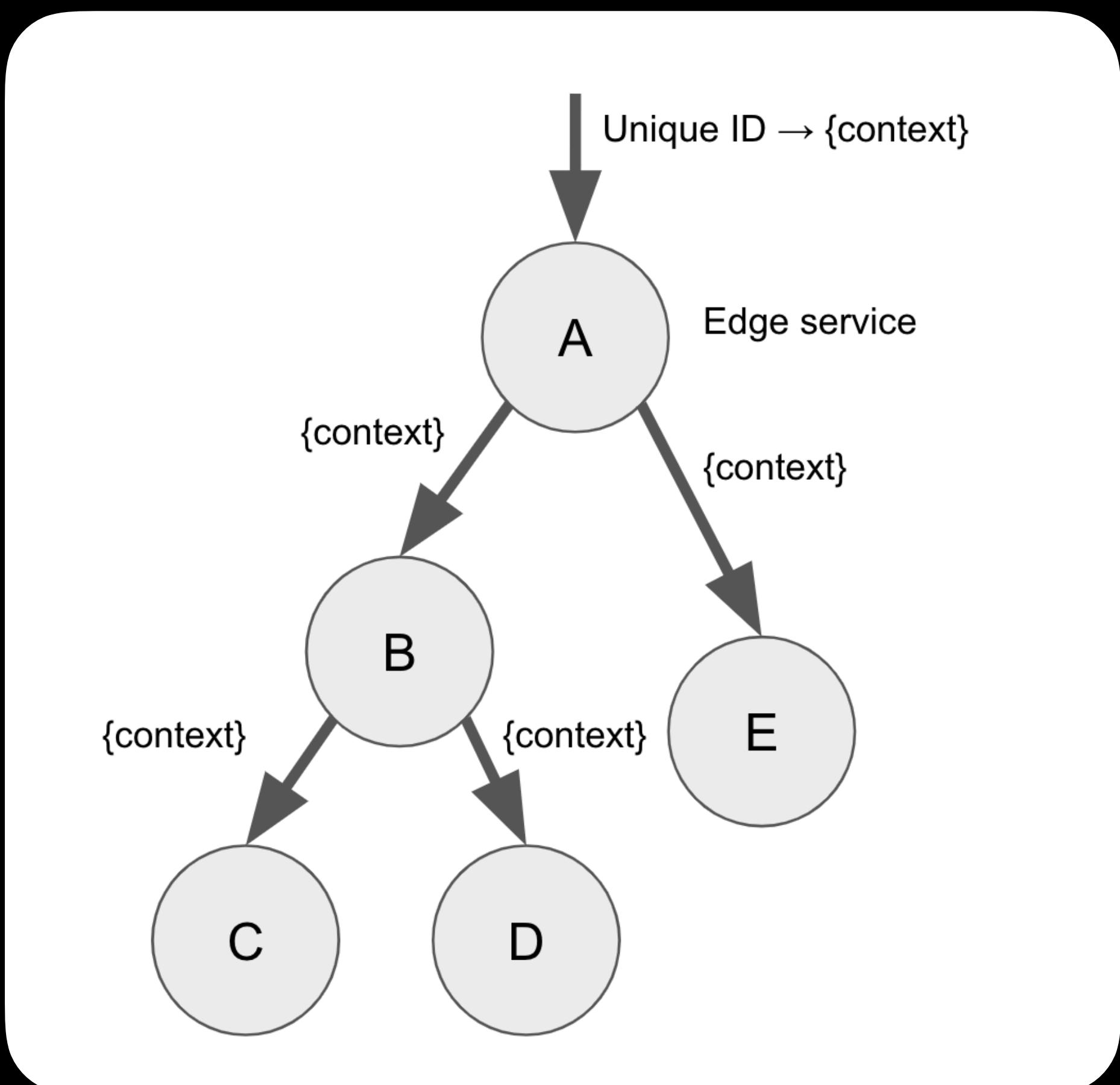
Observability

System's ability to answer questions

- Which services did the request go through
- What did every service do when processing the request
- If the request was slow, where were the bottlenecks
- If the request failed, where did the errors happen
- Who should be paged
- How different was the execution from the normal system behavior
 - Structural differences
 - Performance differences
 - What was on the critical path of the request

Distributed tracing
can answer these questions
and accelerate root cause analysis

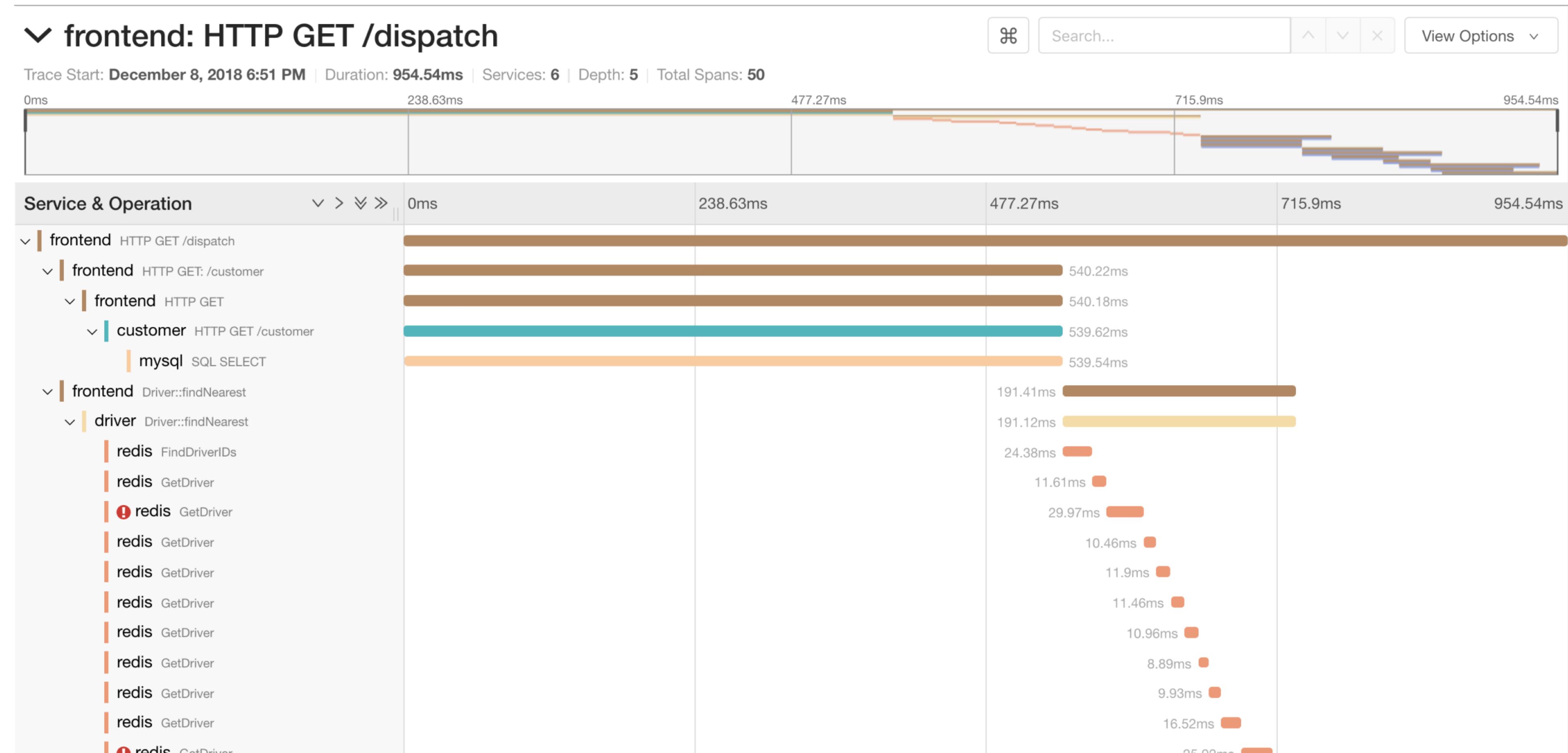
Distributed Tracing in a Nutshell



Trace as a narrative

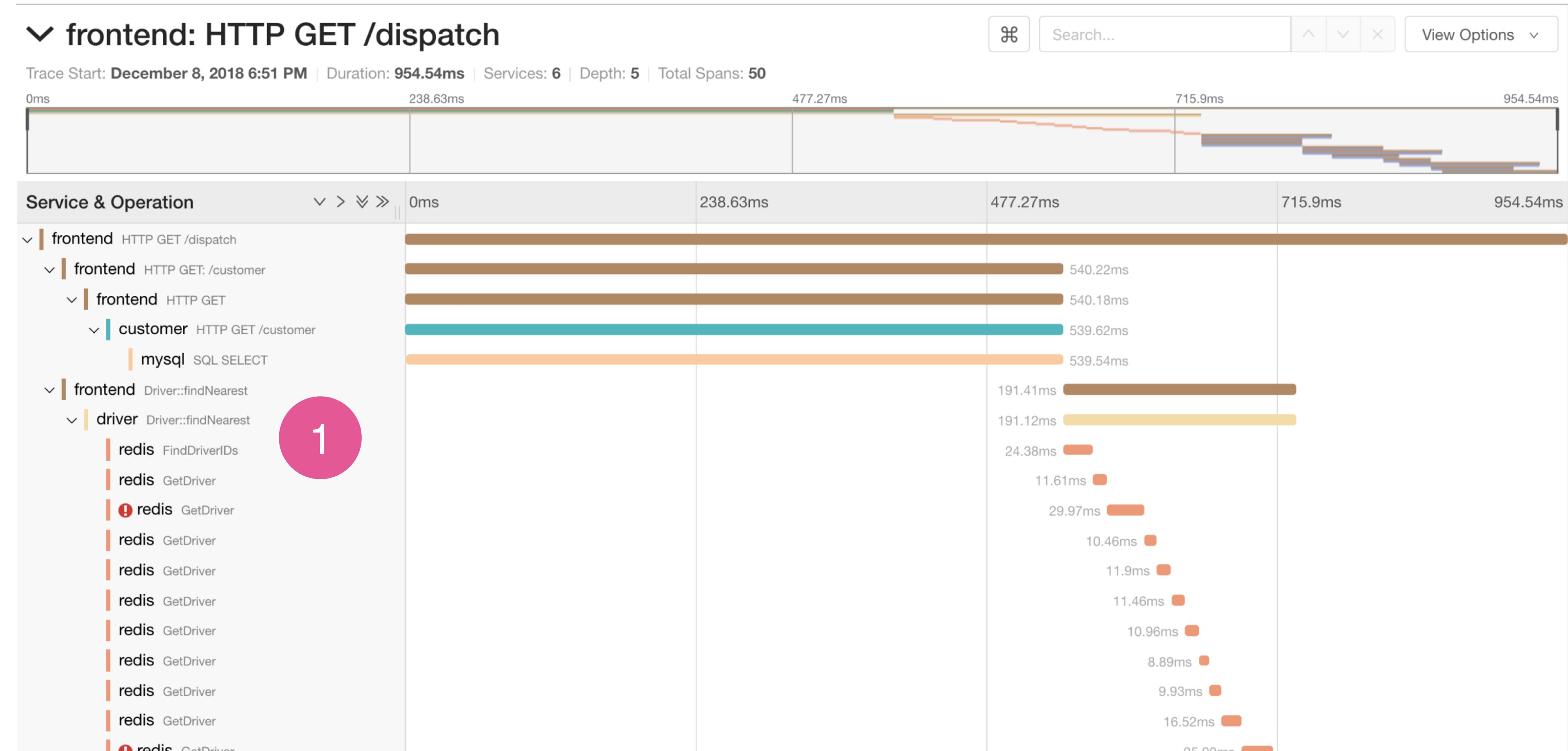
Trace Timeline

Classic trace view as Gantt chart



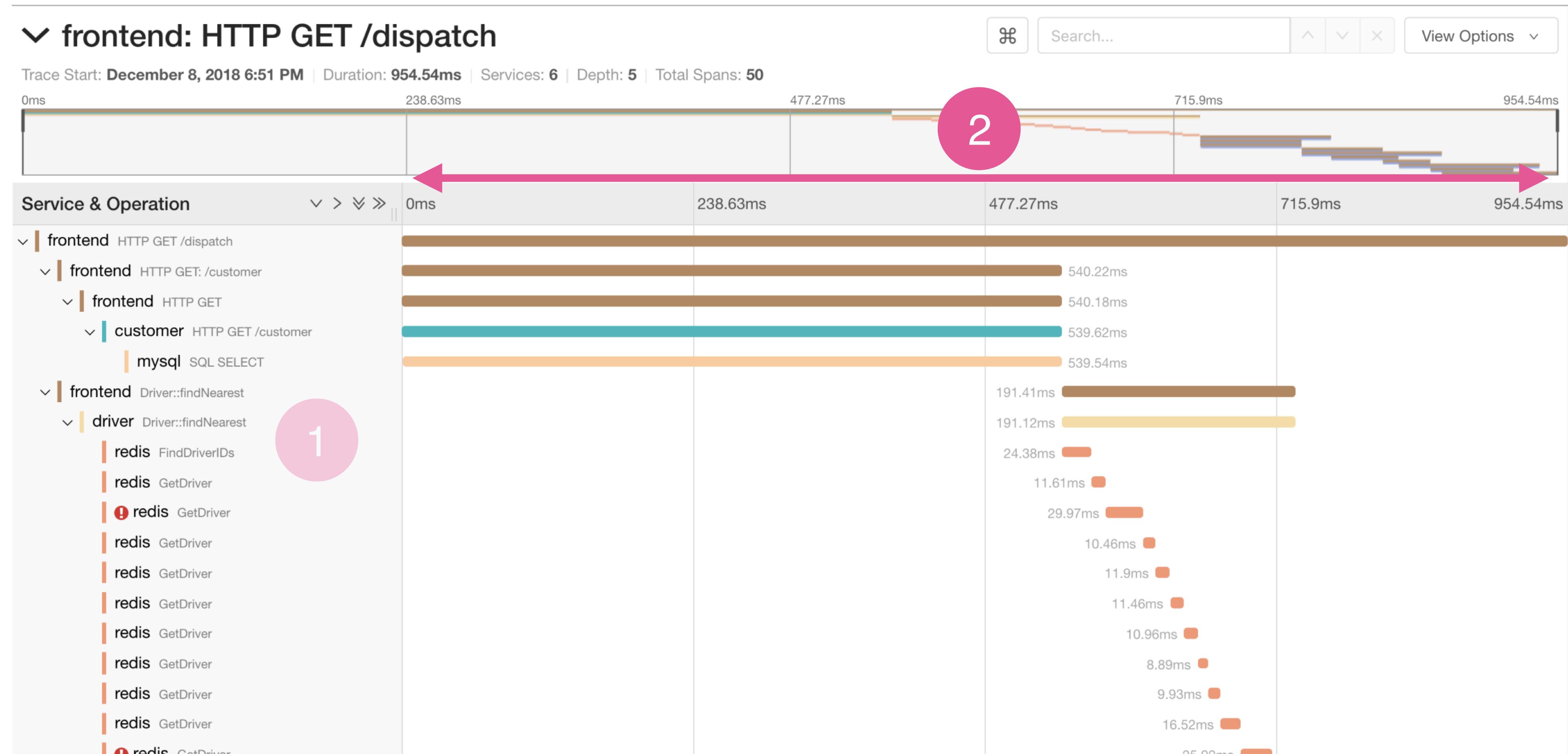
Trace Timeline

Parent → Child → Grandchild



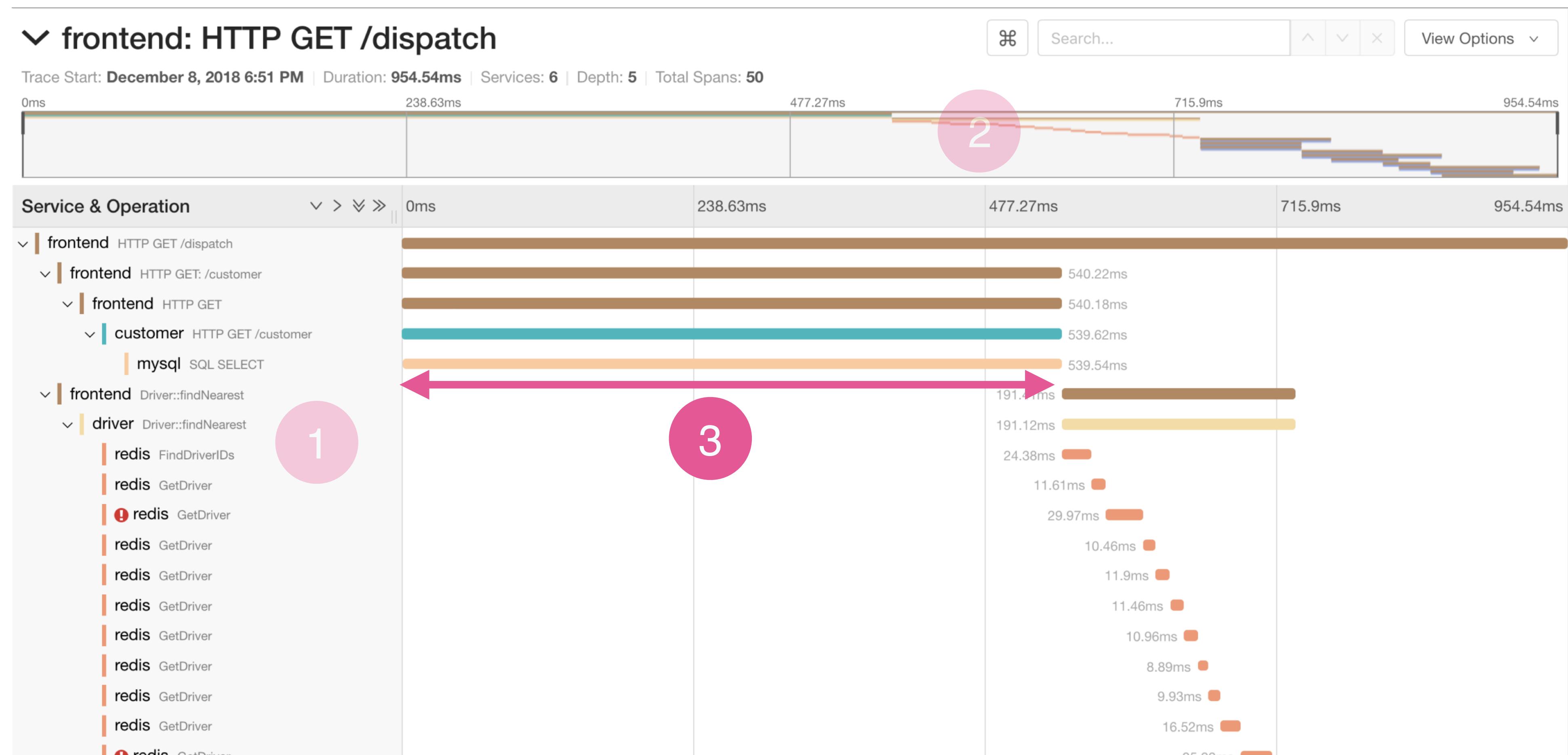
Trace Timeline

Time + Mini-Map



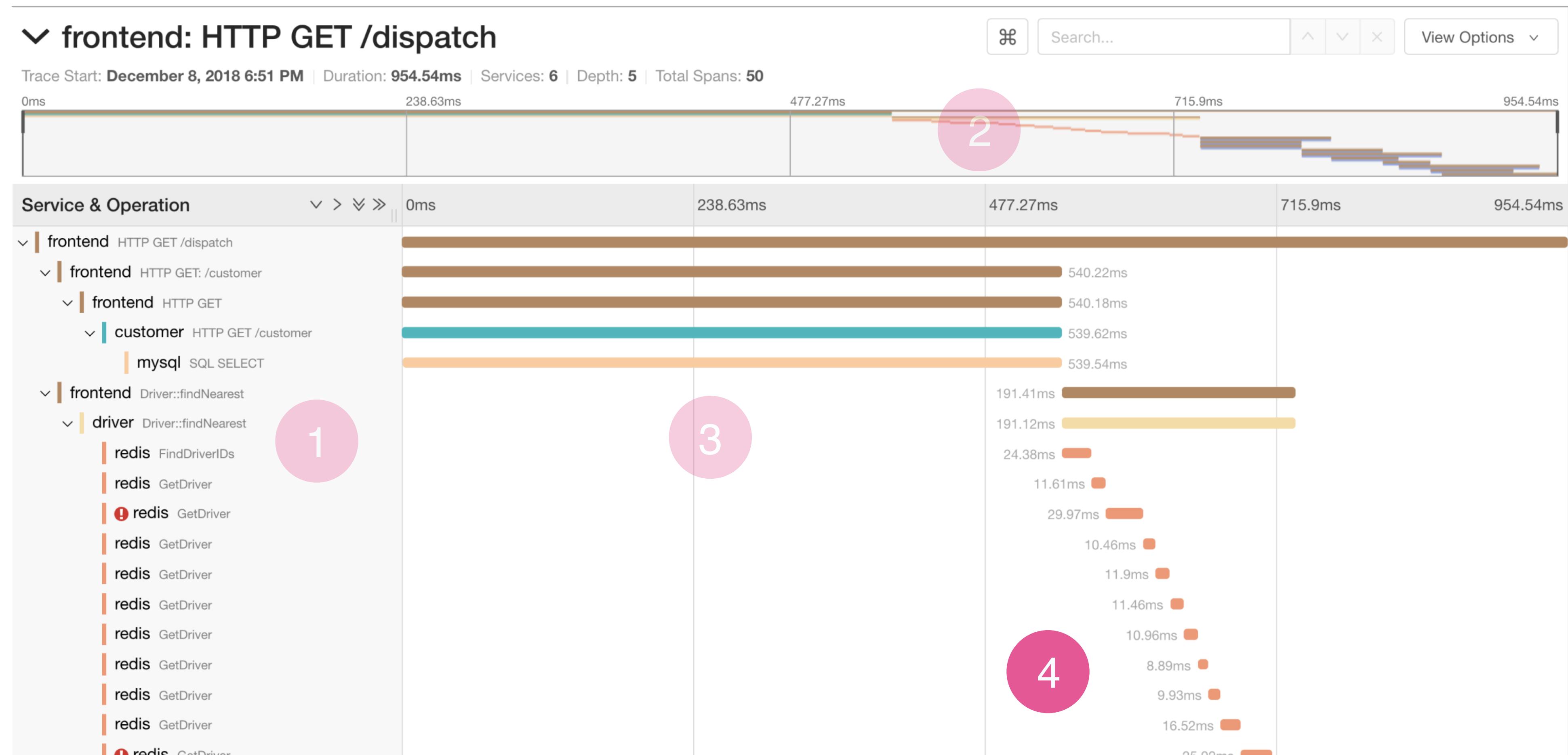
Trace Timeline

Blocking operation

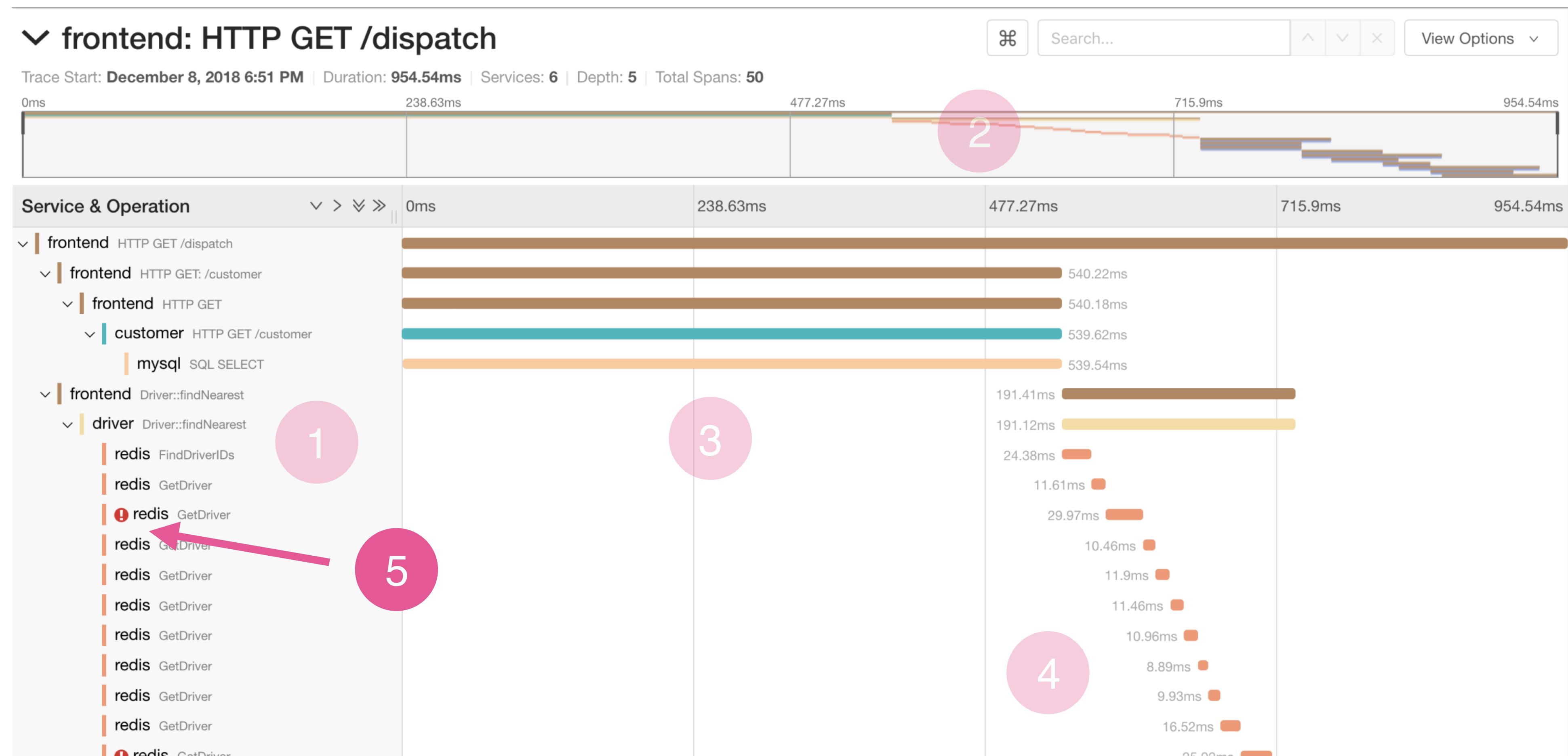


Trace Timeline

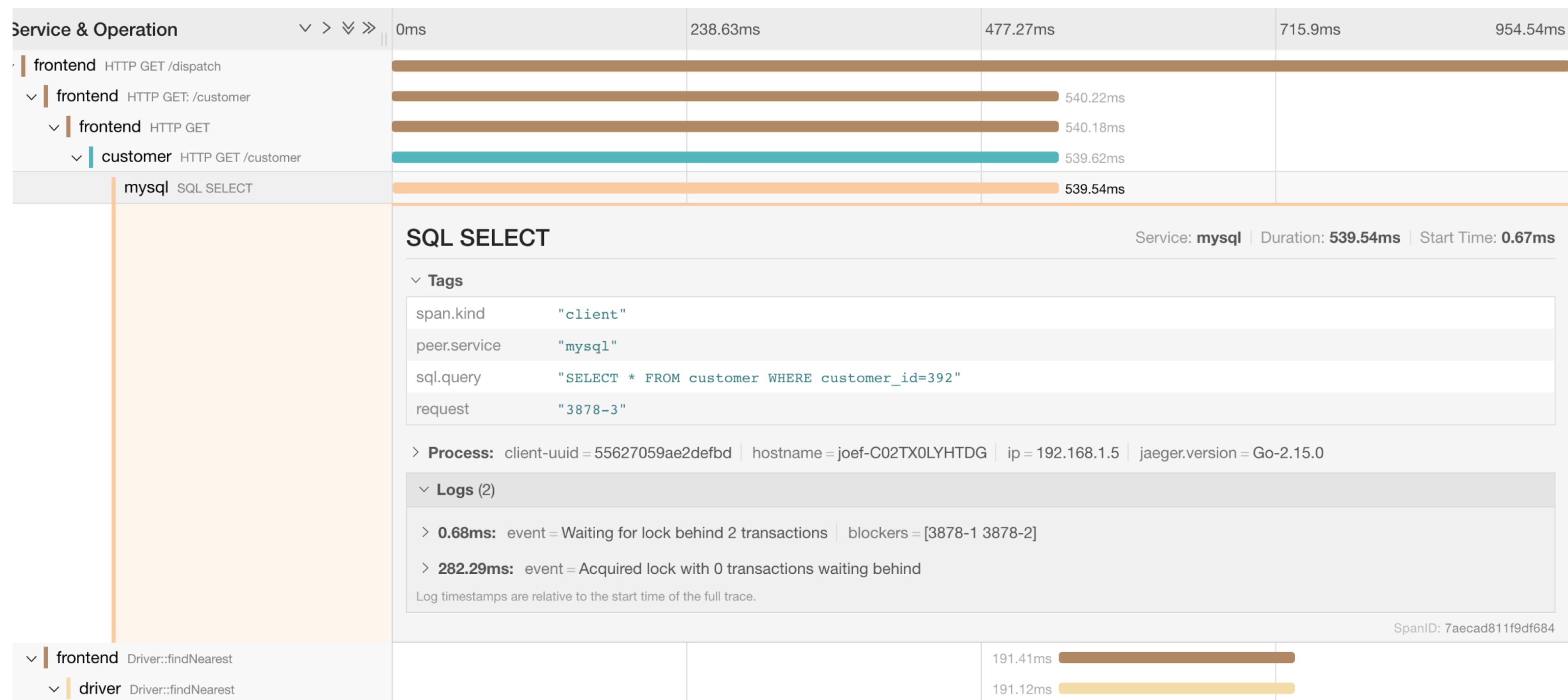
Sequential operations



Trace Timeline Errors

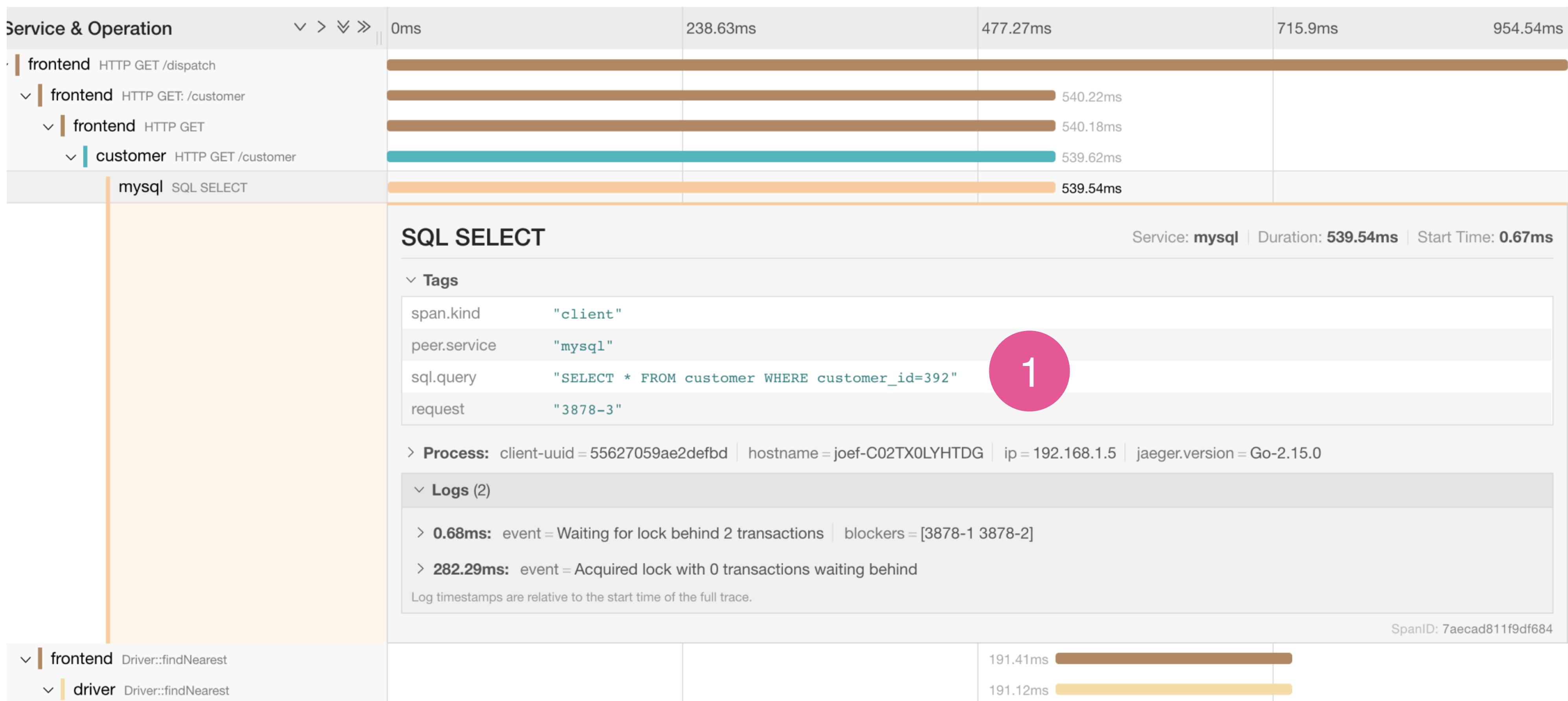


Span details



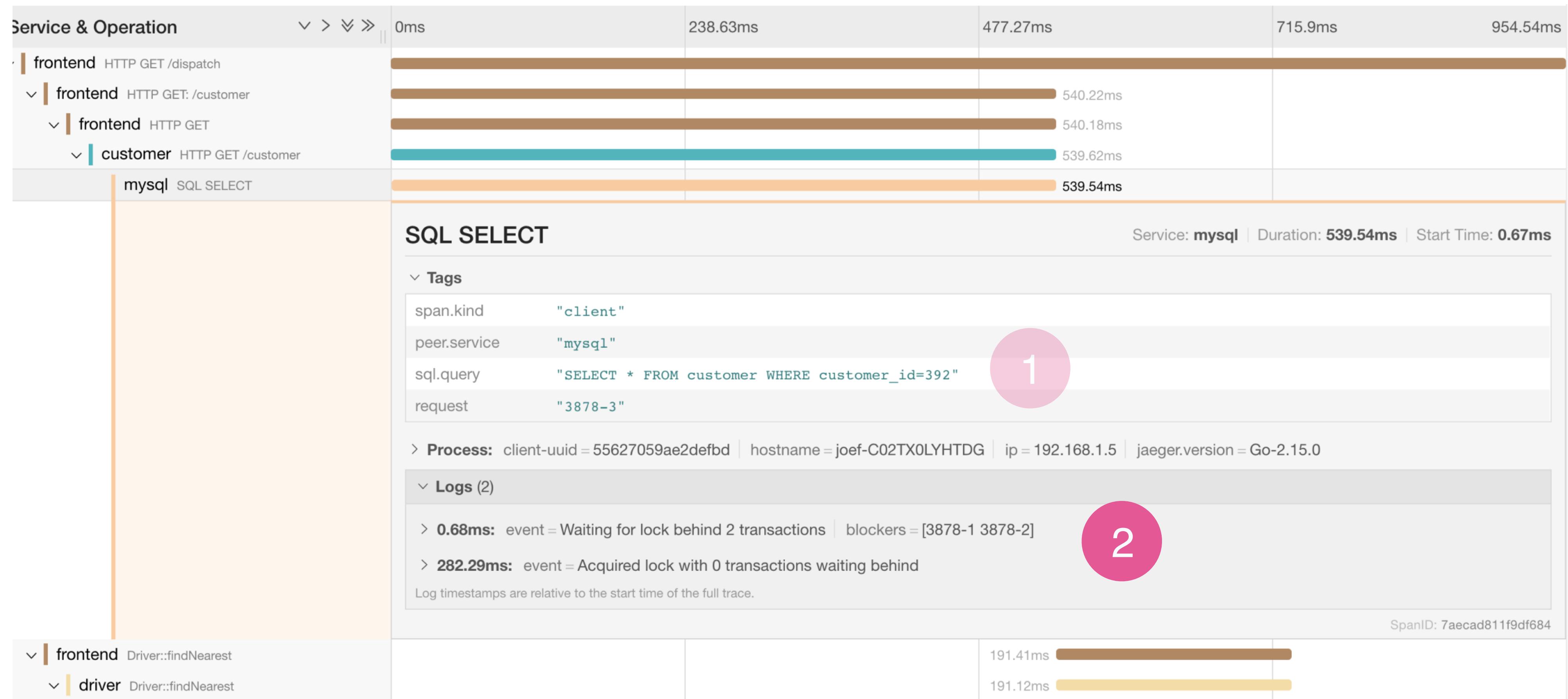
Span details

Database query



Span details

Timed events (logs)



We can also trace
asynchronous workflows

Tracing Talk Application

Mastering Distributed Tracing, Chapter 5

Tracing Talk

Nickname Ralph [EDIT](#)

Yuri Happy tracing everyone 22 minutes ago

Yuri /giphy hello 19 minutes ago



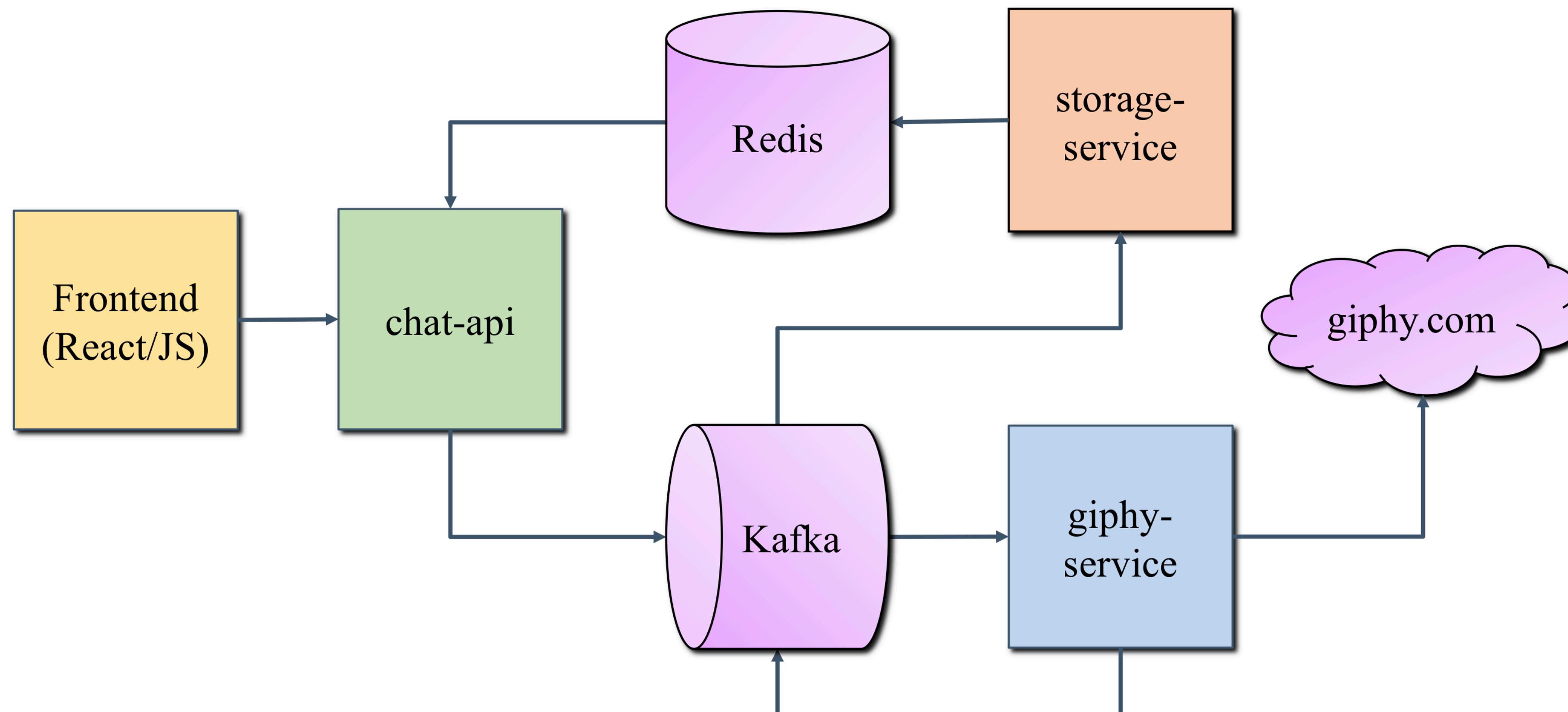
Ralph Tracing is fun! a few seconds ago

Enter message here... Try '/giphy <topic>'.

SEND

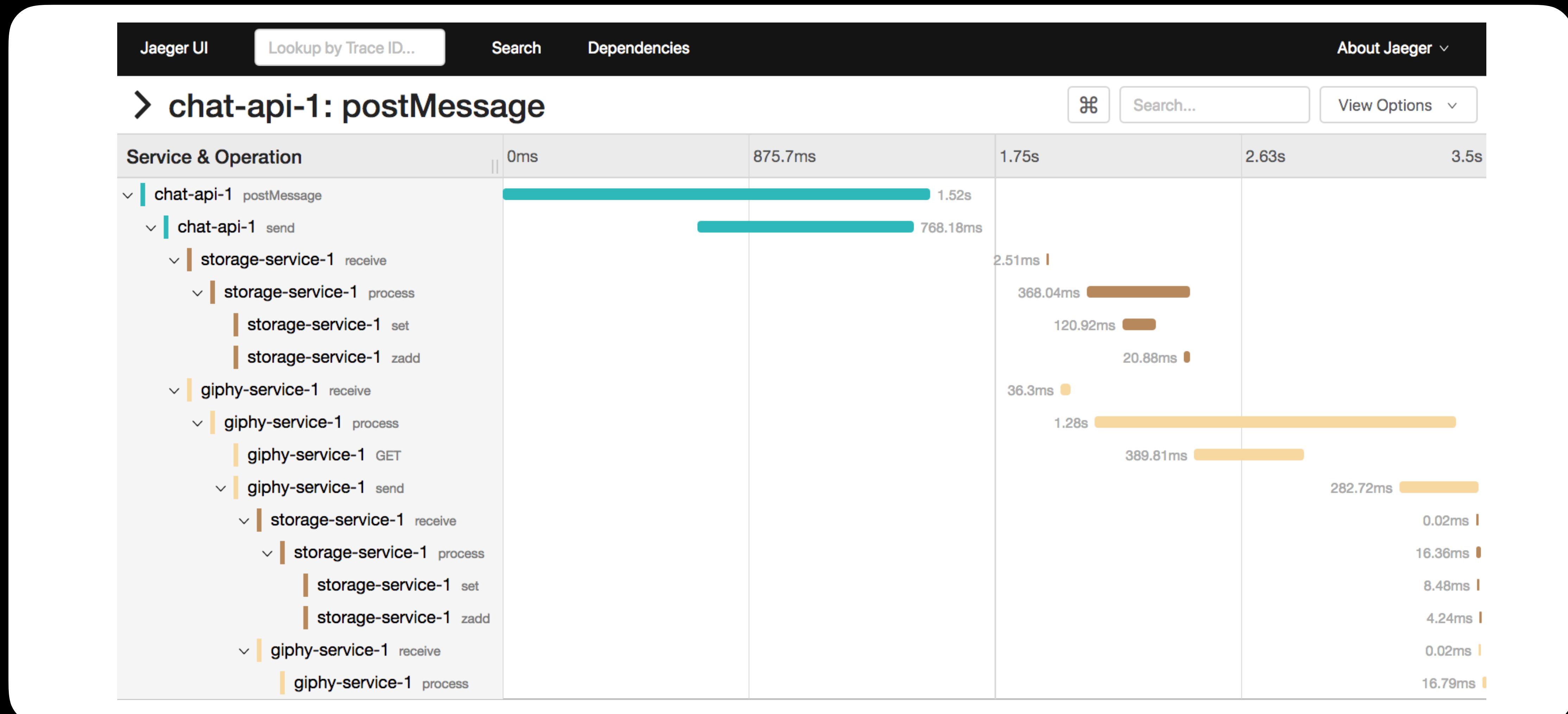
Tracing Talk Application

Architecture



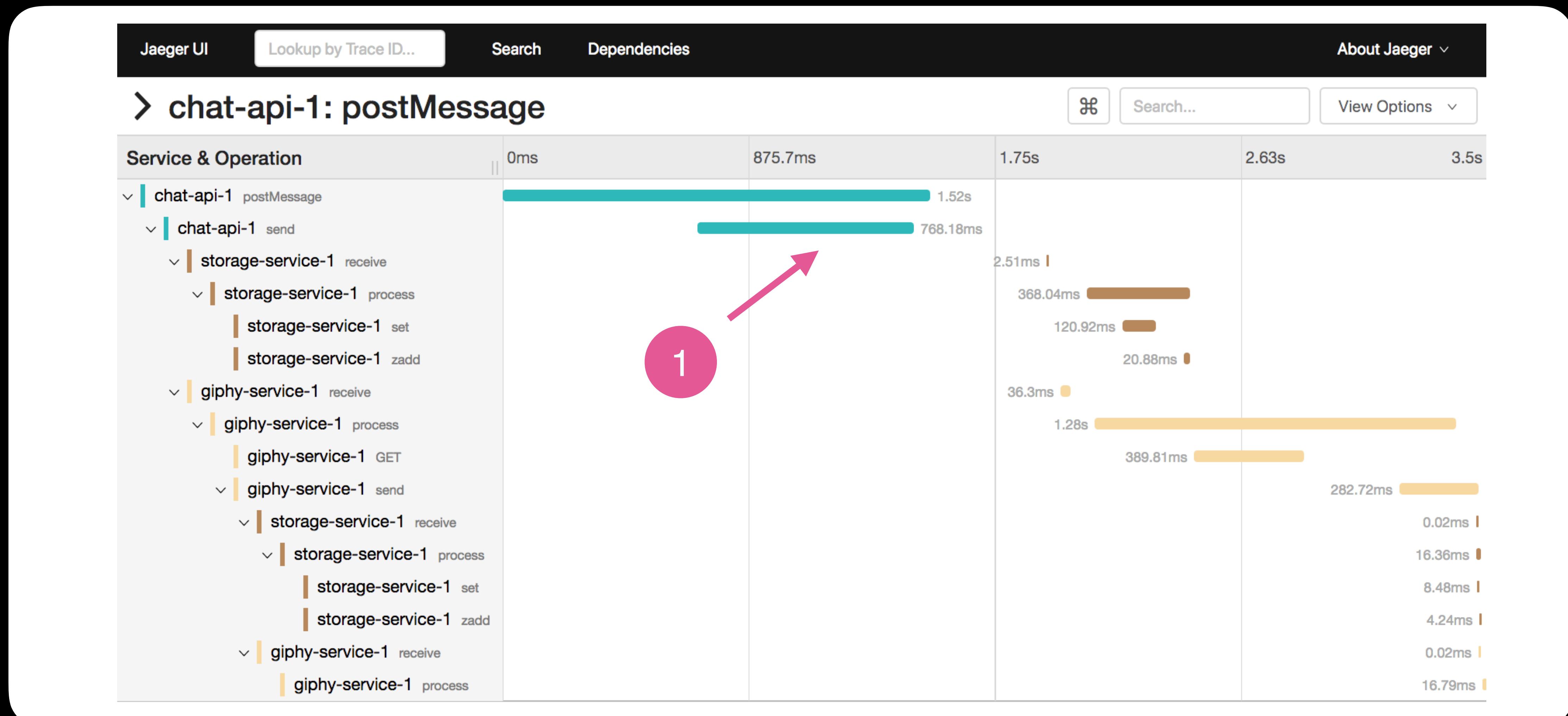
Tracing Talk Application

Request trace



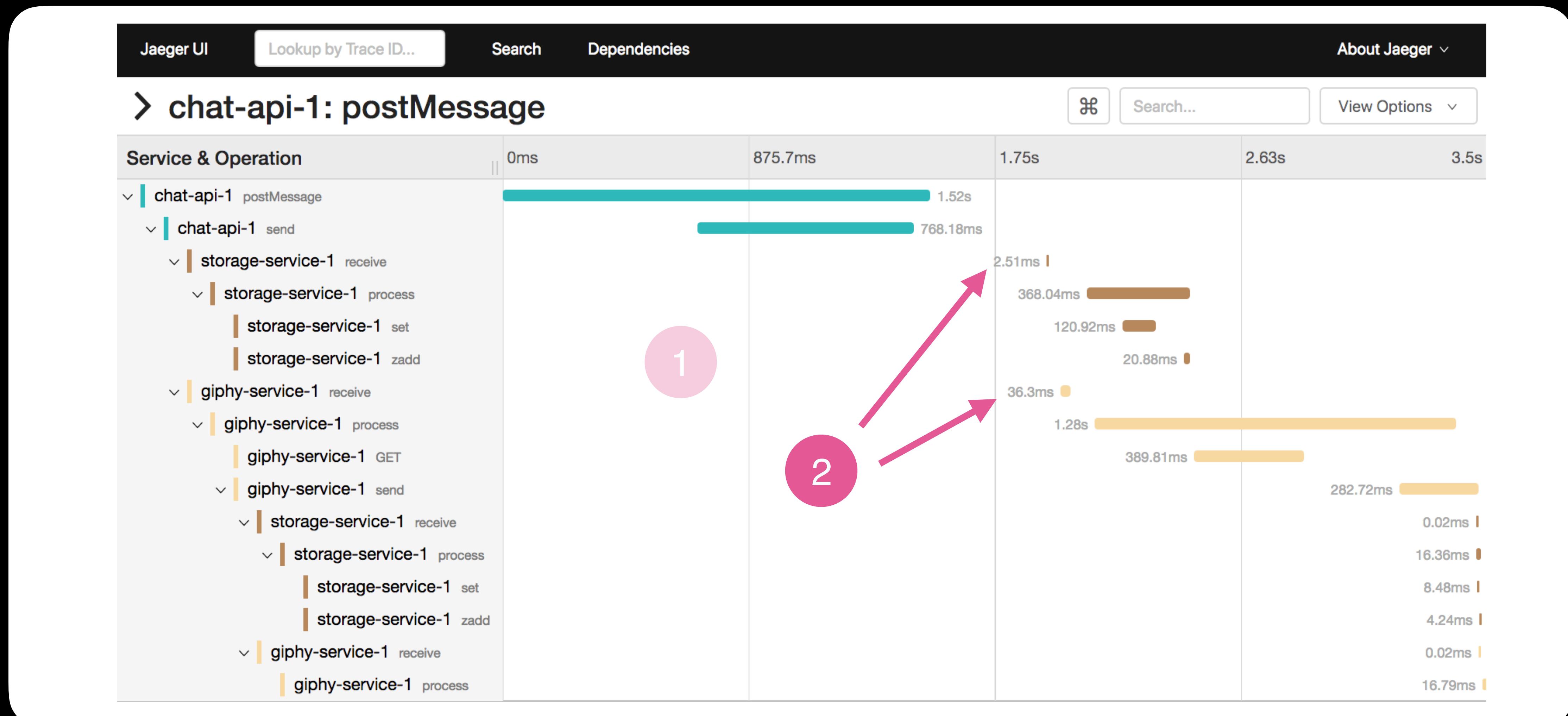
Tracing Talk Application

Message sent



Tracing Talk Application

Message received



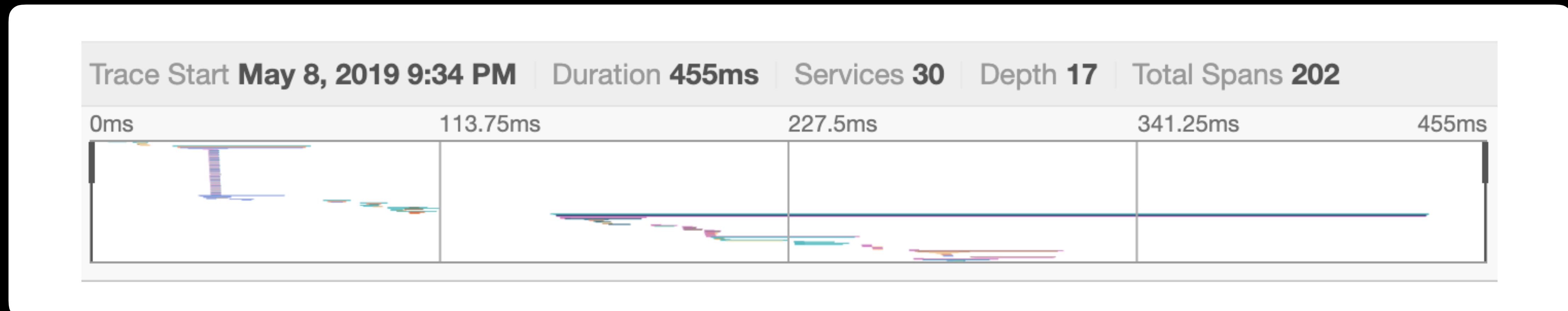
Single Trace

Pros and cons

- Tells a story about a single transaction
- Allows deep contextual drill-down
- Acts as a distributed stack trace
- Tells a story about a single transaction. What if it's an anomaly?
- One trace can be overwhelmingly complex

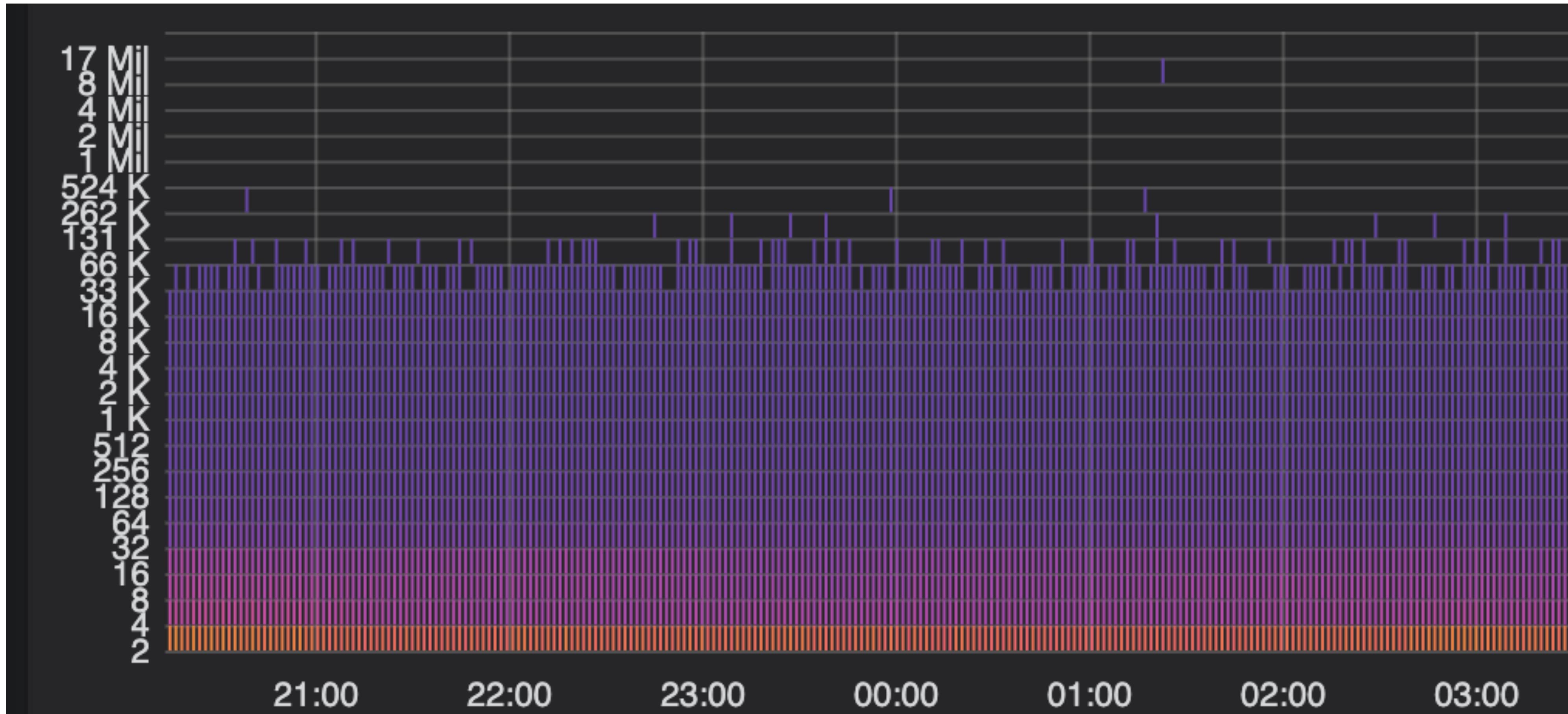
Too Much Complexity

One request - 30 services, 100+ RPCs



Too Much Complexity

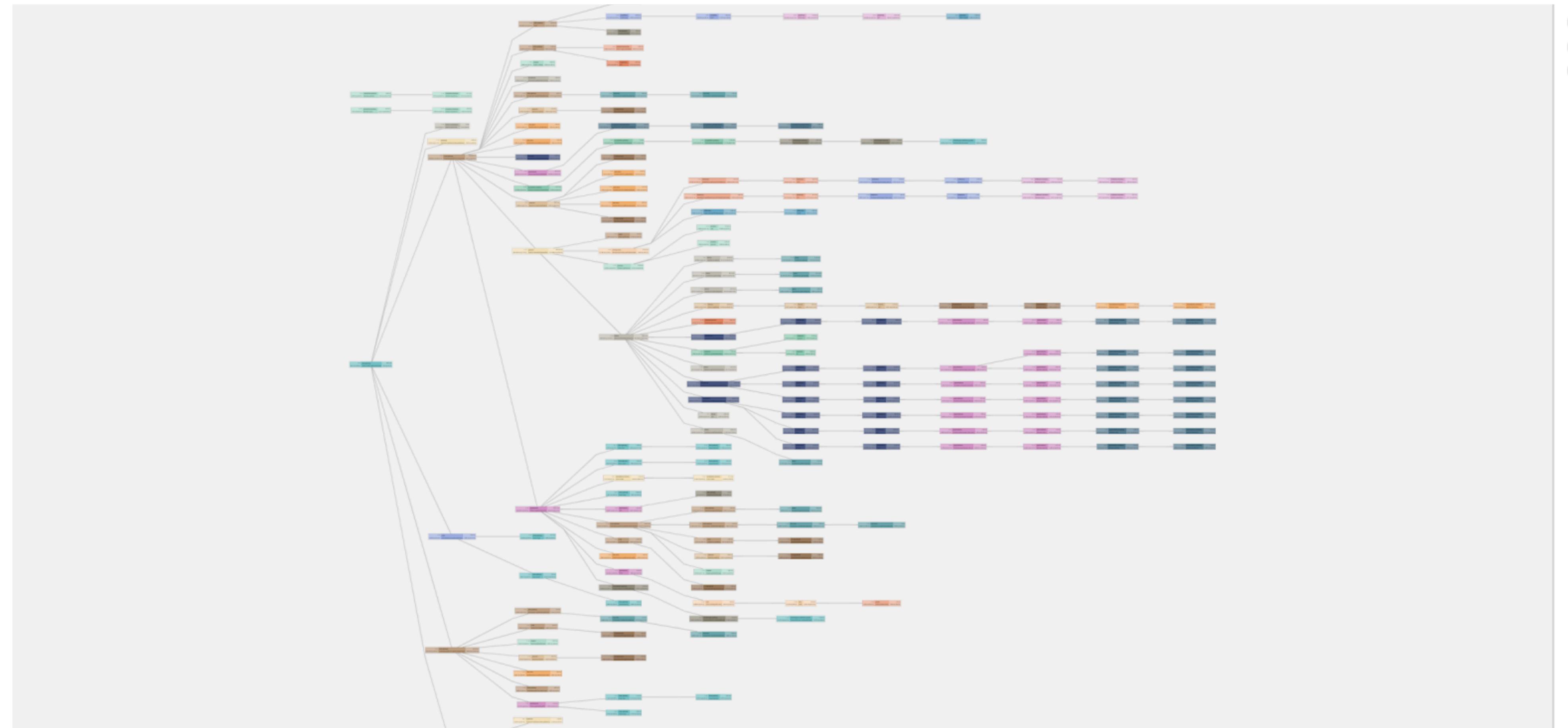
Some traces have hundreds of thousands spans



Reducing complexity by
smarter visualizations

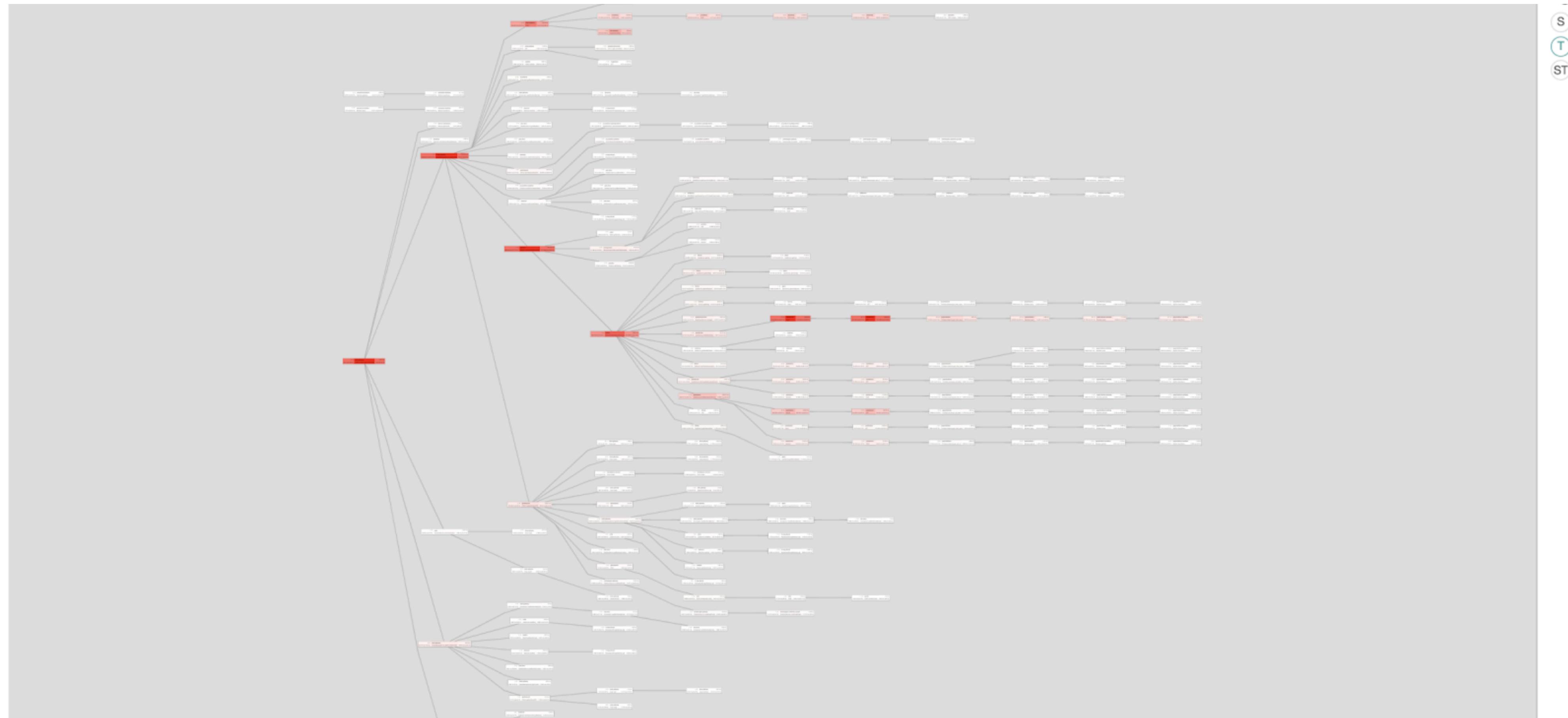
Trace graph

Time ordered, repeated edges collapsed



Trace graph

Latency heat map

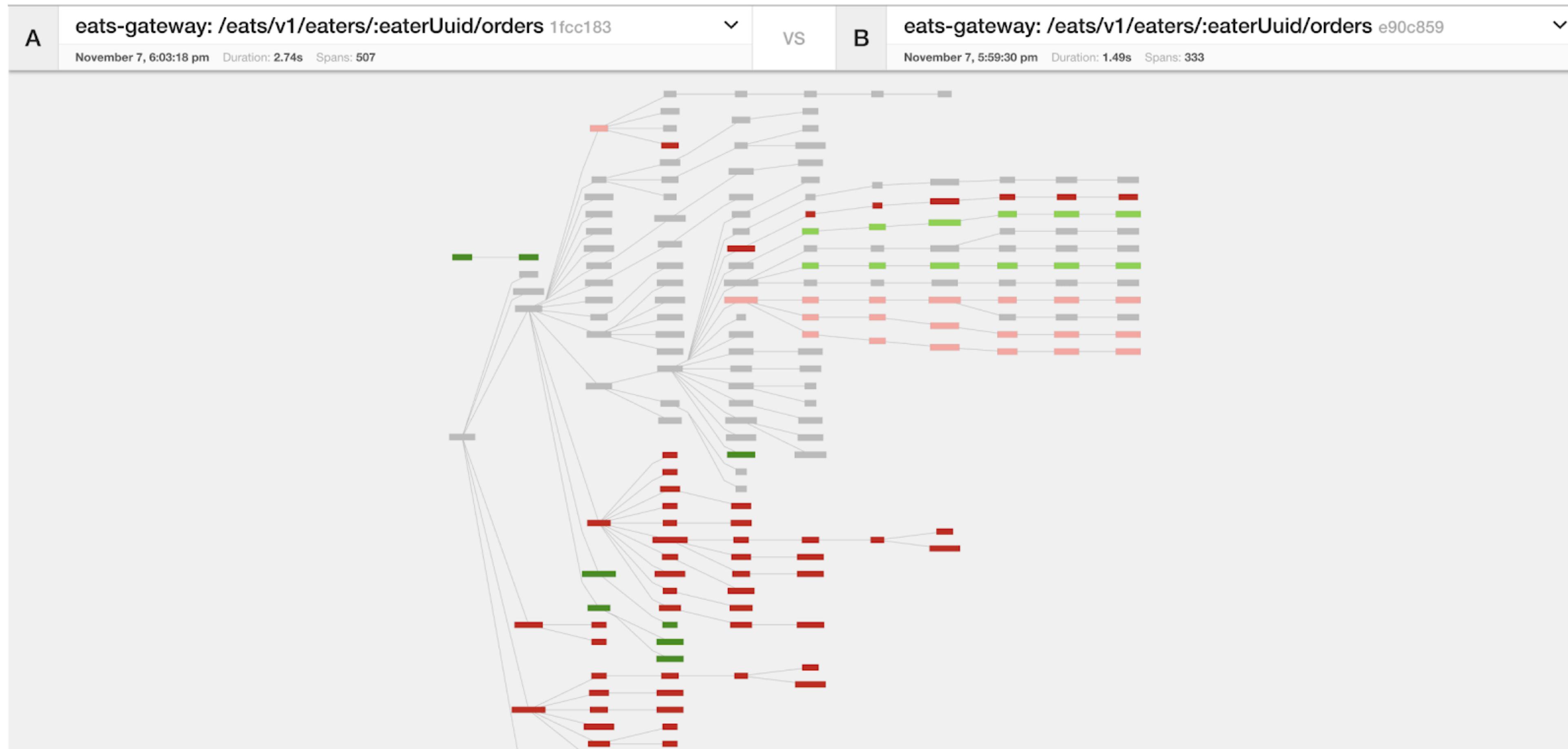


Finding anomalies is easier
when we look at differences
in performance profiles

Trace vs. Trace

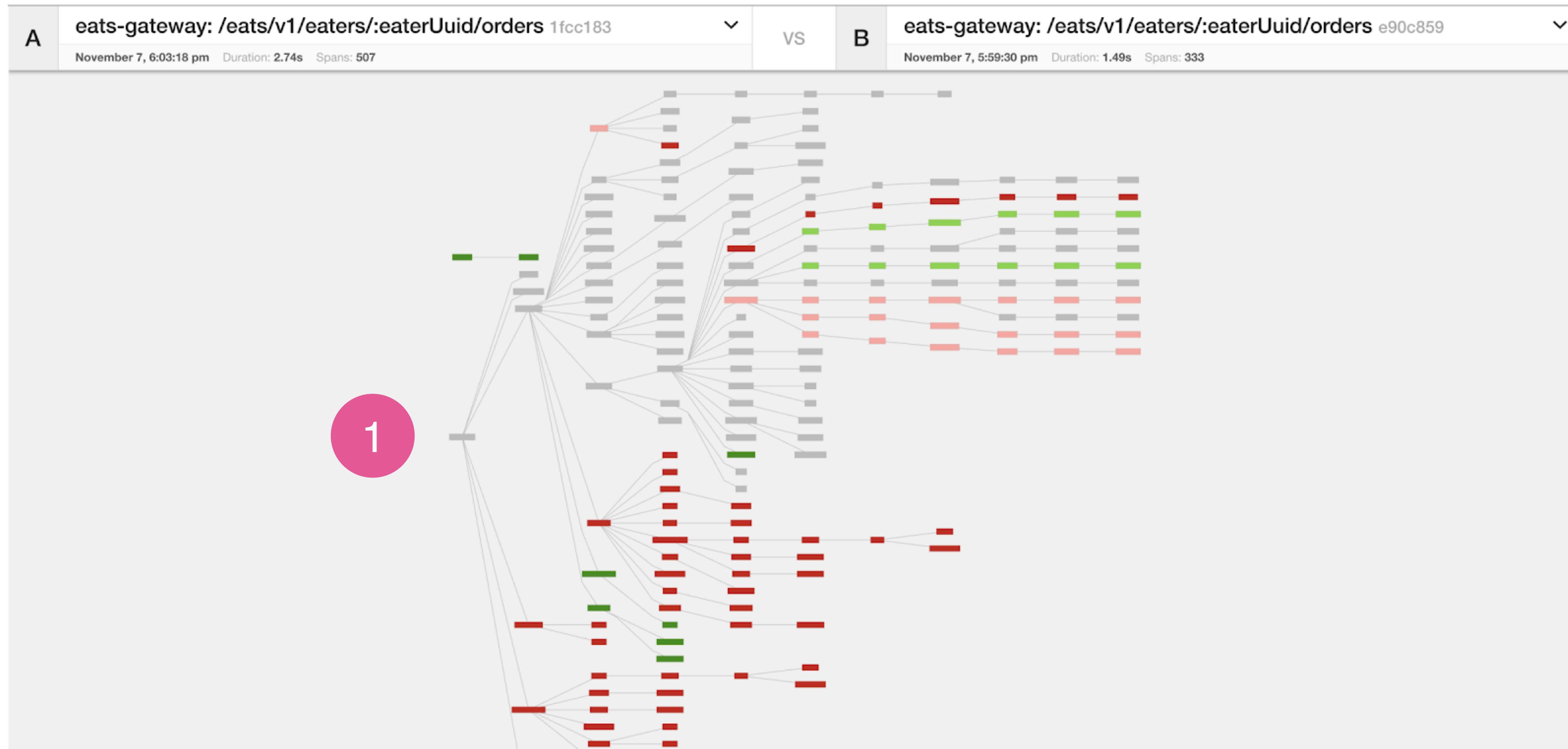
Comparing Trace Structures

Just like a Code Diff



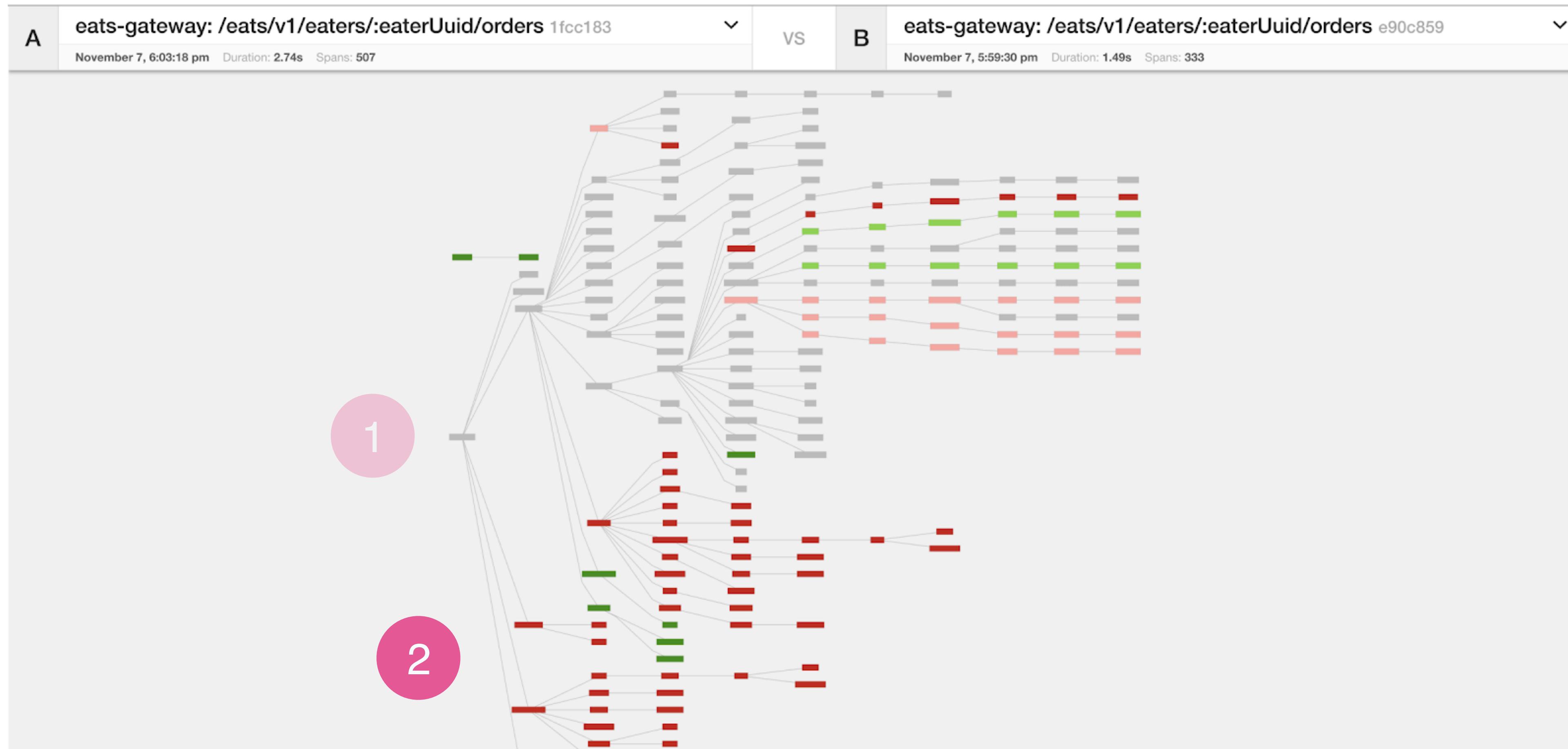
Comparing Trace Structures

Shared Structure



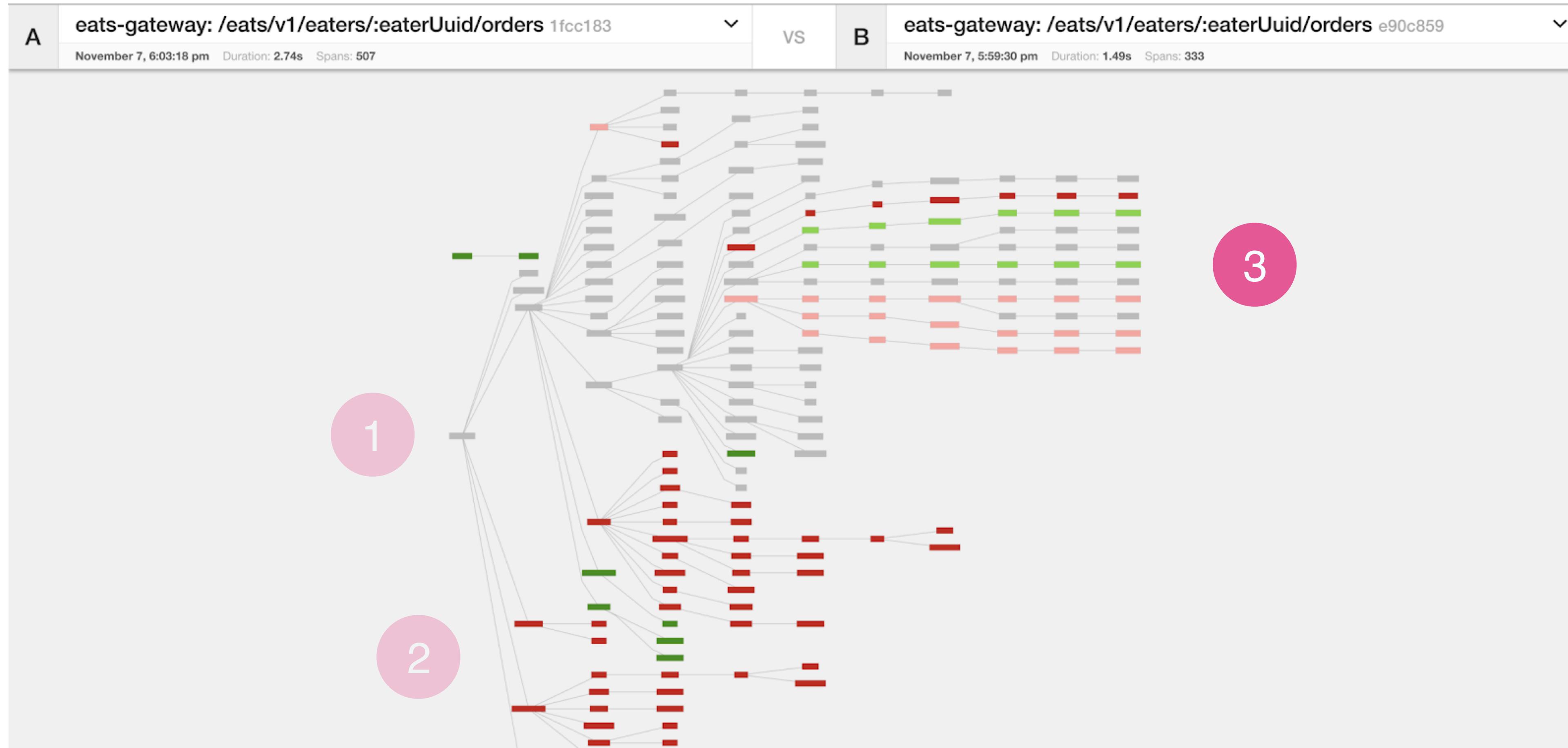
Comparing Trace Structures

Absent in One or the Traces



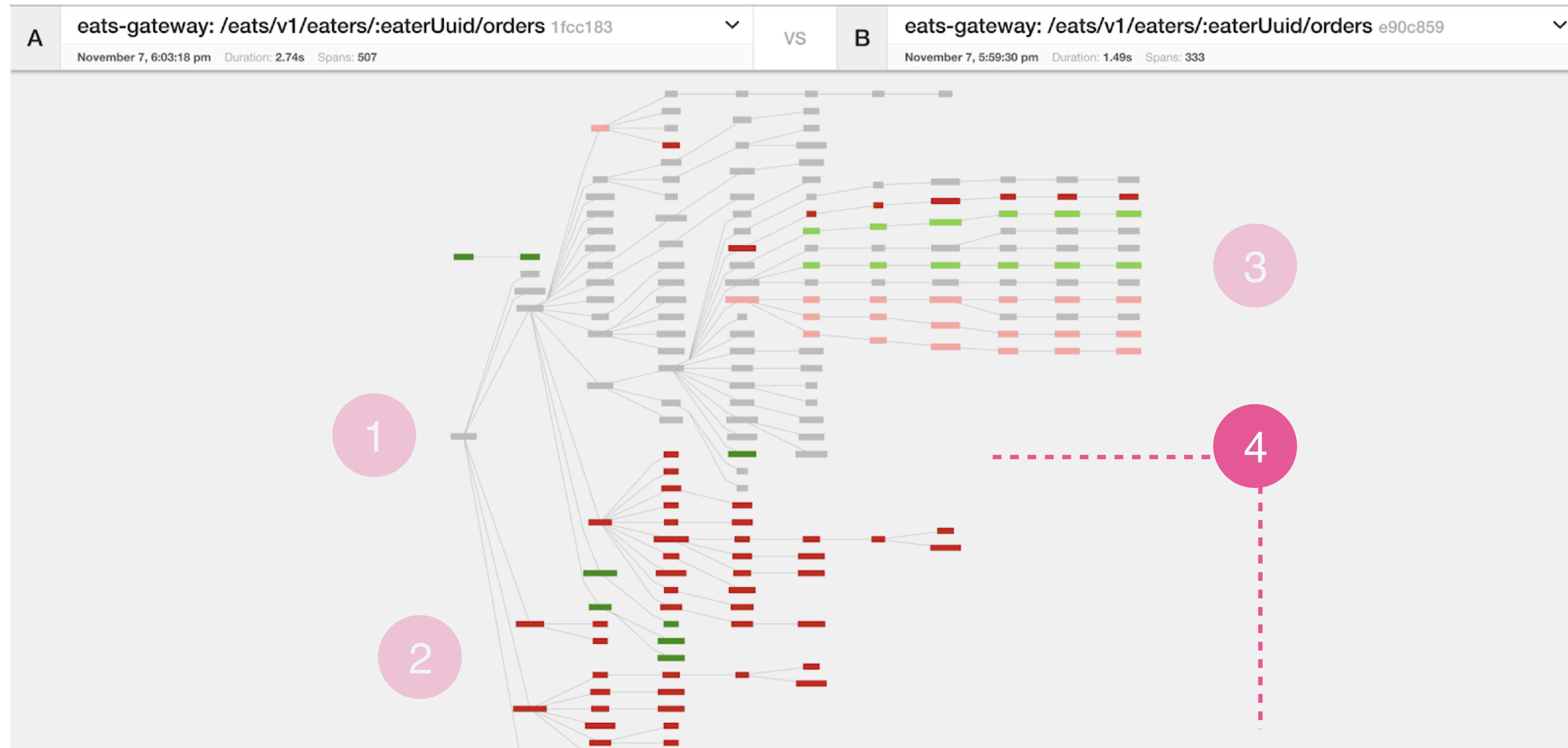
Comparing Trace Structures

More or Fewer Spans Within a Node



Comparing Trace Structures

Substantial Divergence



Deep Linking to Raw Traces & Spans

Error: "You have an outstanding balance..."

The screenshot shows the Jaeger UI interface for monitoring distributed systems. At the top, there's a breadcrumb navigation: > eats-gateway: /eats/v1/eaters/:eaterUuid/orders. Below this is a service map with several components: eats-gateway, the-menu::WasSoGood, i-got-lost::OnTheWay::ToTheJiffyStore, and abc-def::allYourBaseAreBelongToYou. The abc-def component is expanded, showing its internal structure and a duration of 1.29s. The main panel displays logs for this component, specifically for the span abc-def::allYourBaseAreBelongToYou. The logs show a single event labeled "error" with type "TChannelError". The error object contains an info field with a detailed message: "Please verify payment information to secure your account", statusCode: 403, shouldRetry: false, and a stack trace. The stack trace lists numerous paths and mountain IDs, such as 150, 74, 83, 118, 71, 36, 22, 729, 470, 458, 1269, 1030, 94, 163, 237, and 118. A pink circle with the number 5 is overlaid on the error log entry.

> eats-gateway: /eats/v1/eaters/:eaterUuid/orders

Service & Operation

eats-gateway /eats/v1/eaters/:eaterUuid/orders

eats-gateway the-menu::WasSoGood

eats-gateway i-got-lost::OnTheWay::ToTheJiffyStore

eats-gateway abc-def::allYourBaseAreBelongToYou

abc-def::allYourBaseAreBelongToYou

Tags: span.kind=client | component=THE-component | error=true

Process: ip=127.0.42.99 | jaeger.hostname=host-with-the-most | jaeger.version=version-ing | legacy-jaeger-client=42.99.99

Logs (1)

1.48s

event "error"

error.kind "TChannelError"

error.object {

info: {

message: "Please verify payment information to secure your account",
statusCode: 403,
shouldRetry: false,
stack: "*errors.errorString You have an outstanding balance due to a credit card problem. Please update your billing settings.
/there/are/many/paths/up/the/mountain:150 (0x1337b0)
/there/are/many/paths/up/the/mountain:74 (0x1337b0)
/there/are/many/paths/up/the/mountain:83 (0x1337b0)
/there/are/many/paths/up/the/mountain:118 (0x1337b0)
/there/are/many/paths/up/the/mountain:71 (0x1337b0)
/there/are/many/paths/up/the/mountain:36 (0x1337b0)
/there/are/many/paths/up/the/mountain:22 (0x1337b0)
/there/are/many/paths/up/the/mountain:729 (0x1337b0)
/there/are/many/paths/up/the/mountain:470 (0x1337b0)
/there/are/many/paths/up/the/mountain:458 (0x1337b0)
/there/are/many/paths/up/the/mountain:1269 (0x1337b0)
/there/are/many/paths/up/the/mountain:1030 (0x1337b0)
/there/are/many/paths/up/the/mountain:94 (0x1337b0)
/there/are/many/paths/up/the/mountain:163 (0x1337b0)
/there/are/many/paths/up/the/mountain:237 (0x1337b0)
/there/are/many/paths/up/the/mountain:118 (0x1337b0)

Log timestamps are relative to the start time of the full trace.

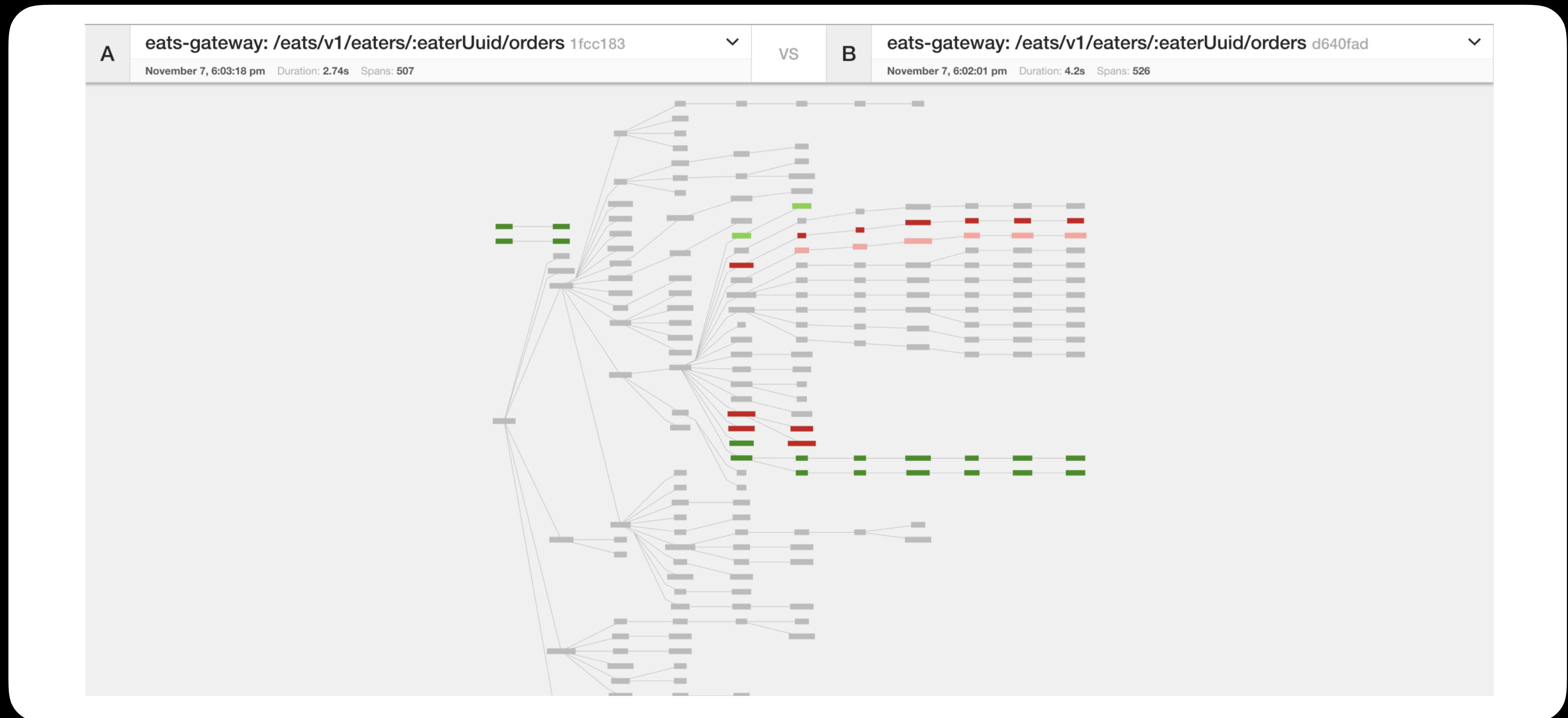
Production story

Migrating services to a **nearby** datacenter

Request latency **doubles**

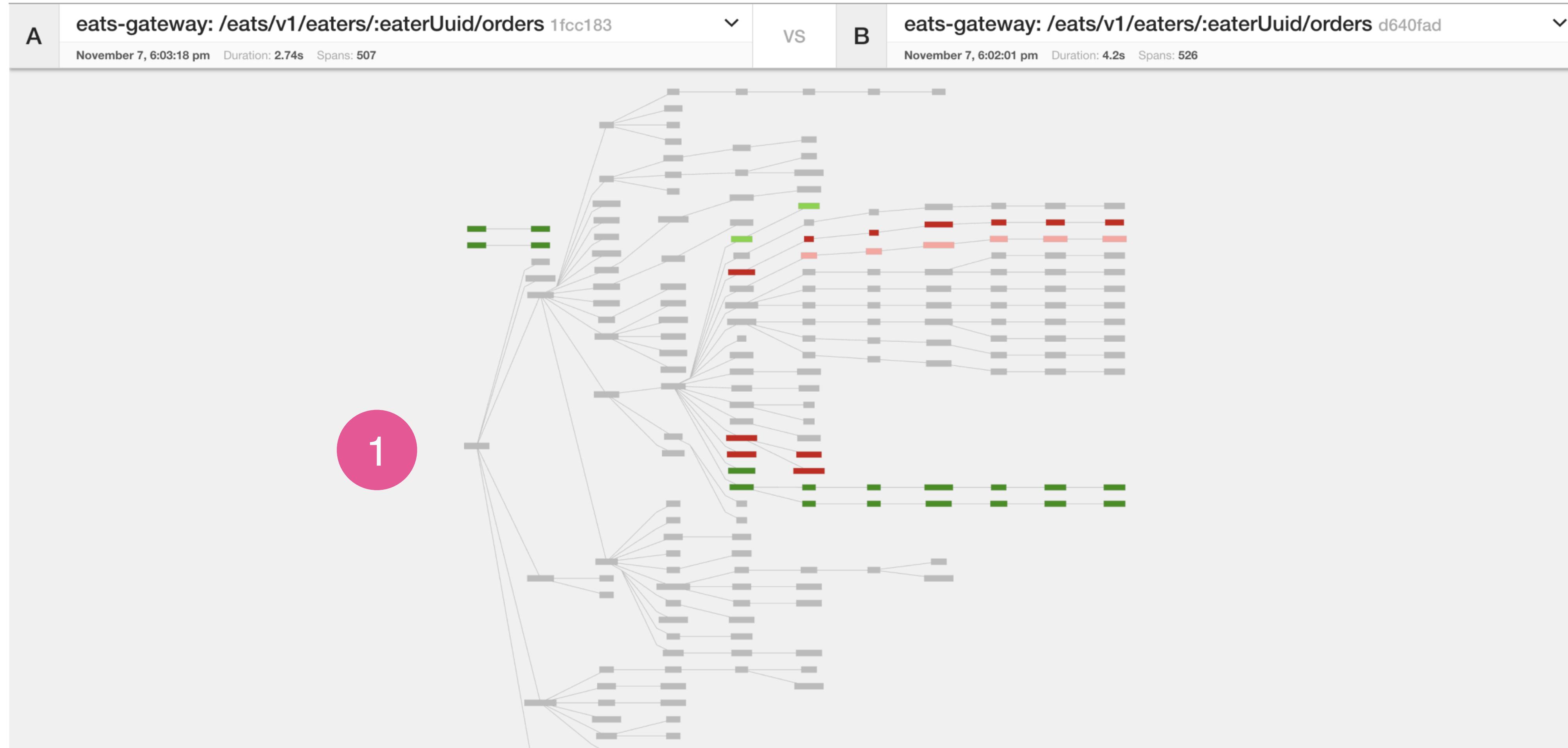
Investigating latency

Structural comparison not always useful



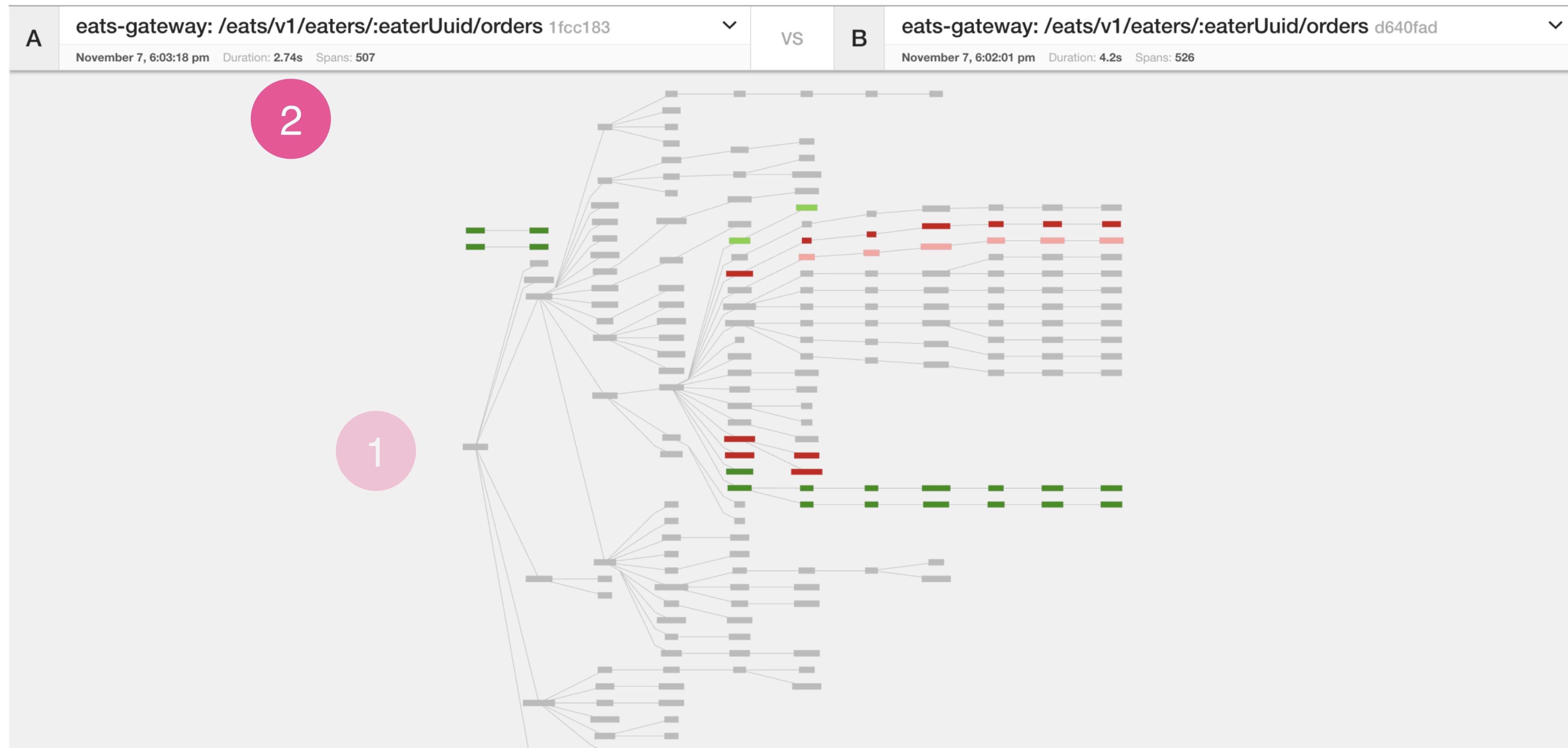
Investigating latency

Very similar structure



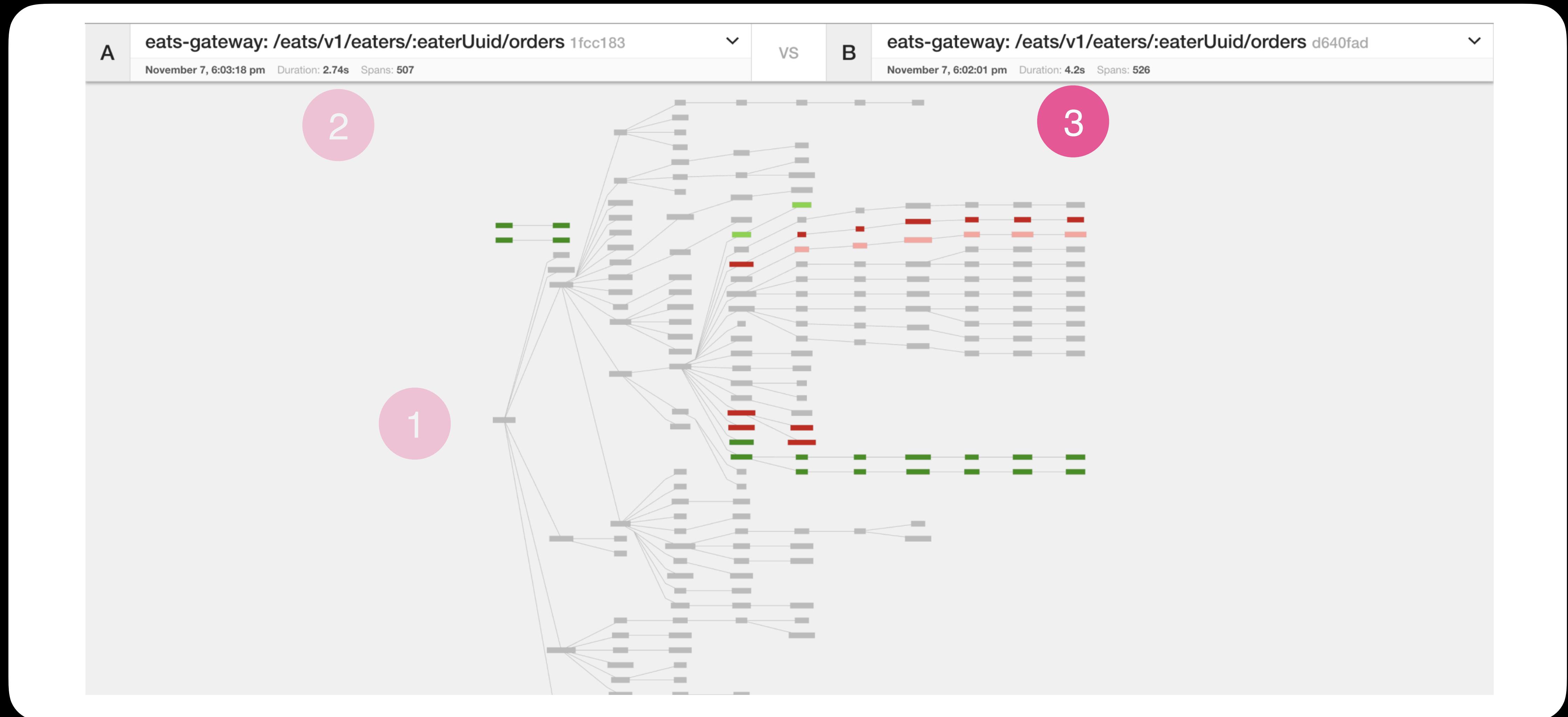
Investigating latency

Left trace 2.74 seconds



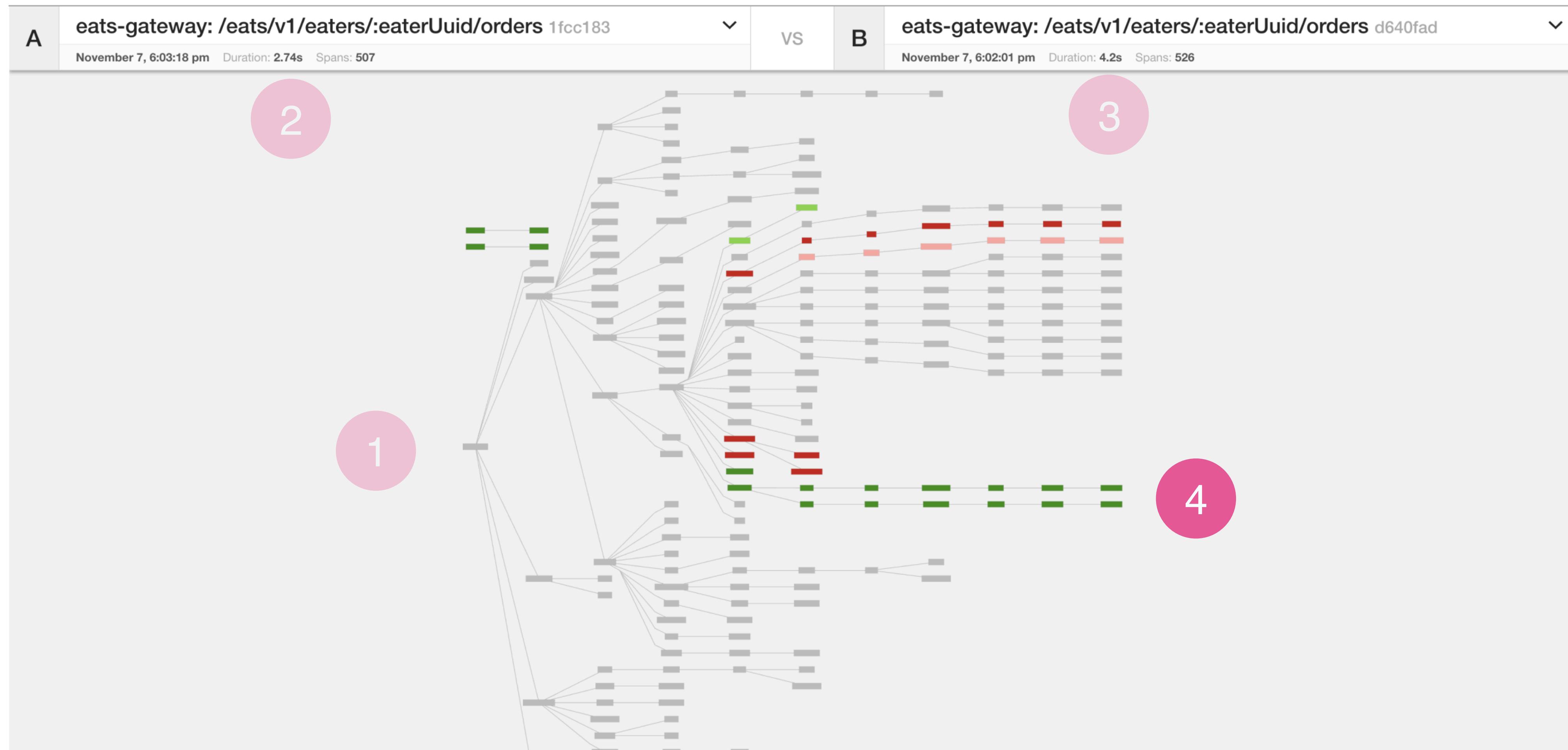
Investigating latency

Right trace 4.2 seconds



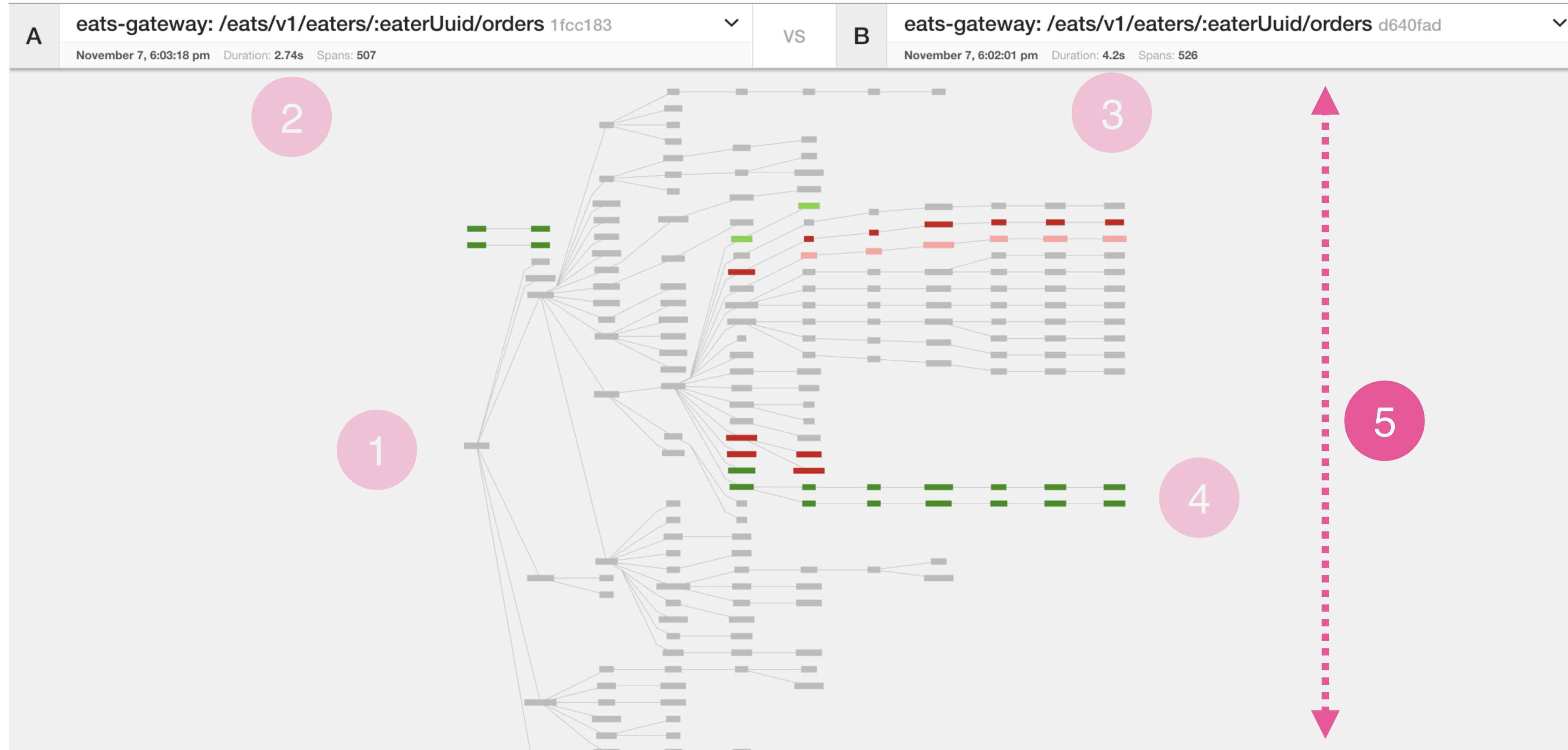
Investigating latency

Due to structural differences?



Investigating latency

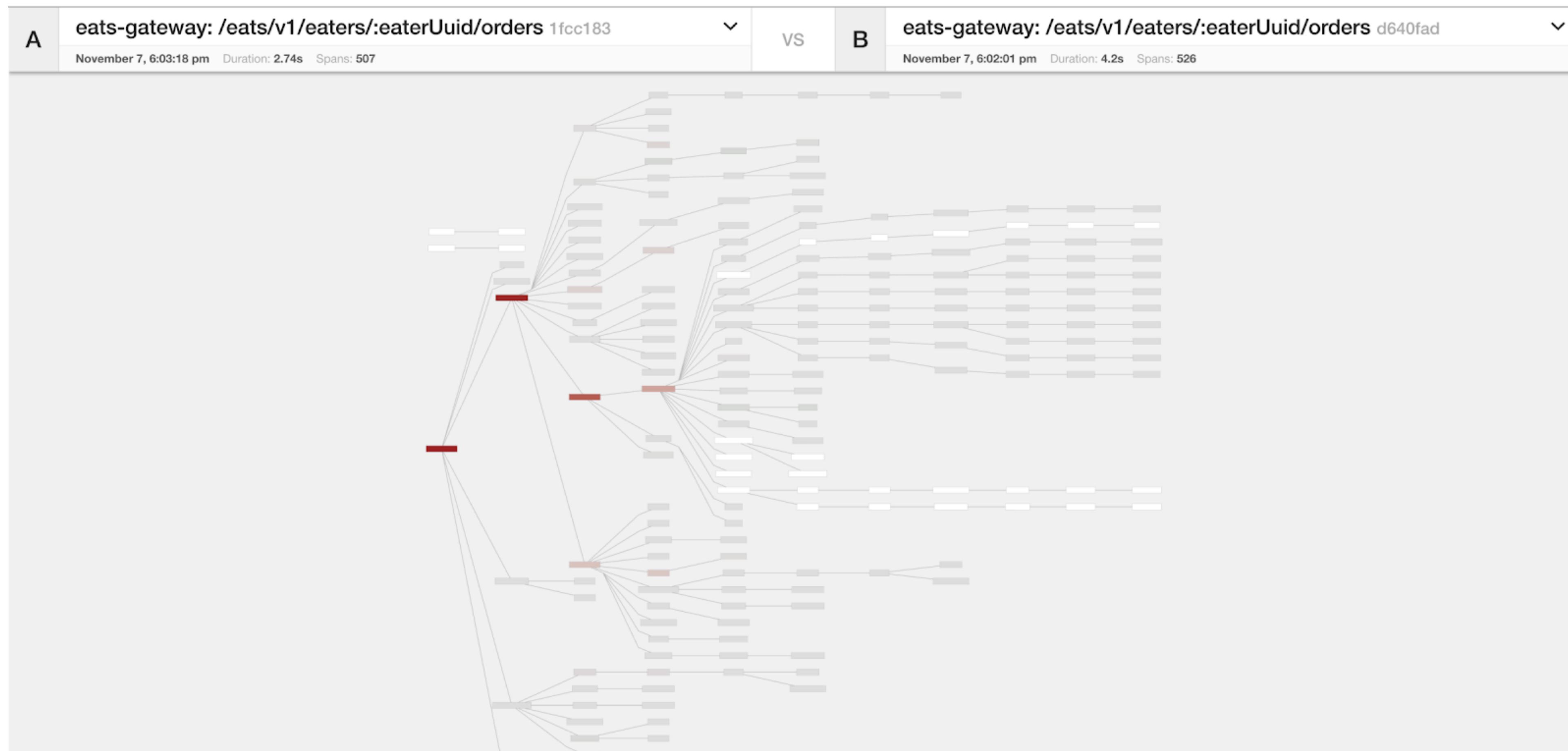
Or dispersed contributors?



Heat-maps!

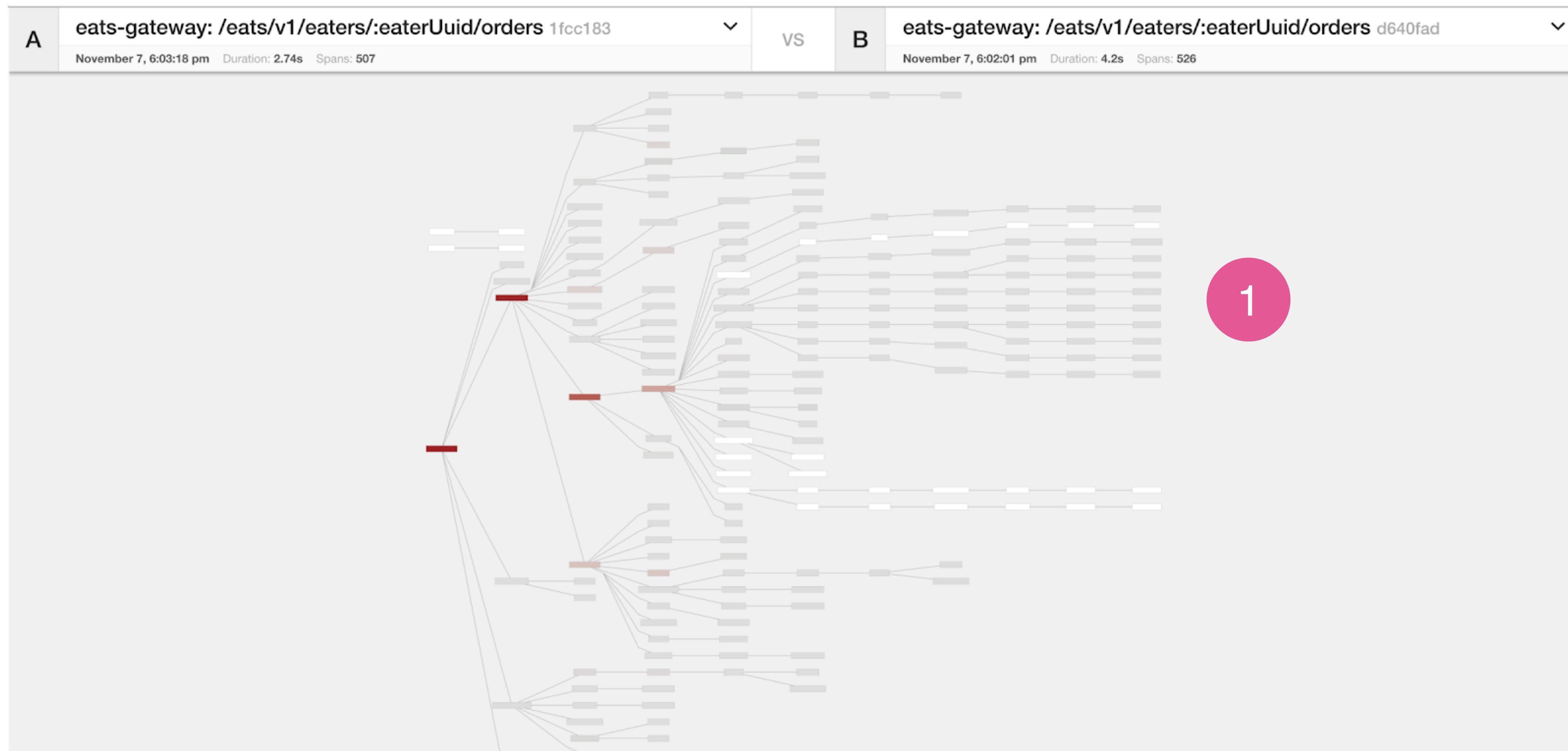
Comparing trace durations

Heat-map of latencies



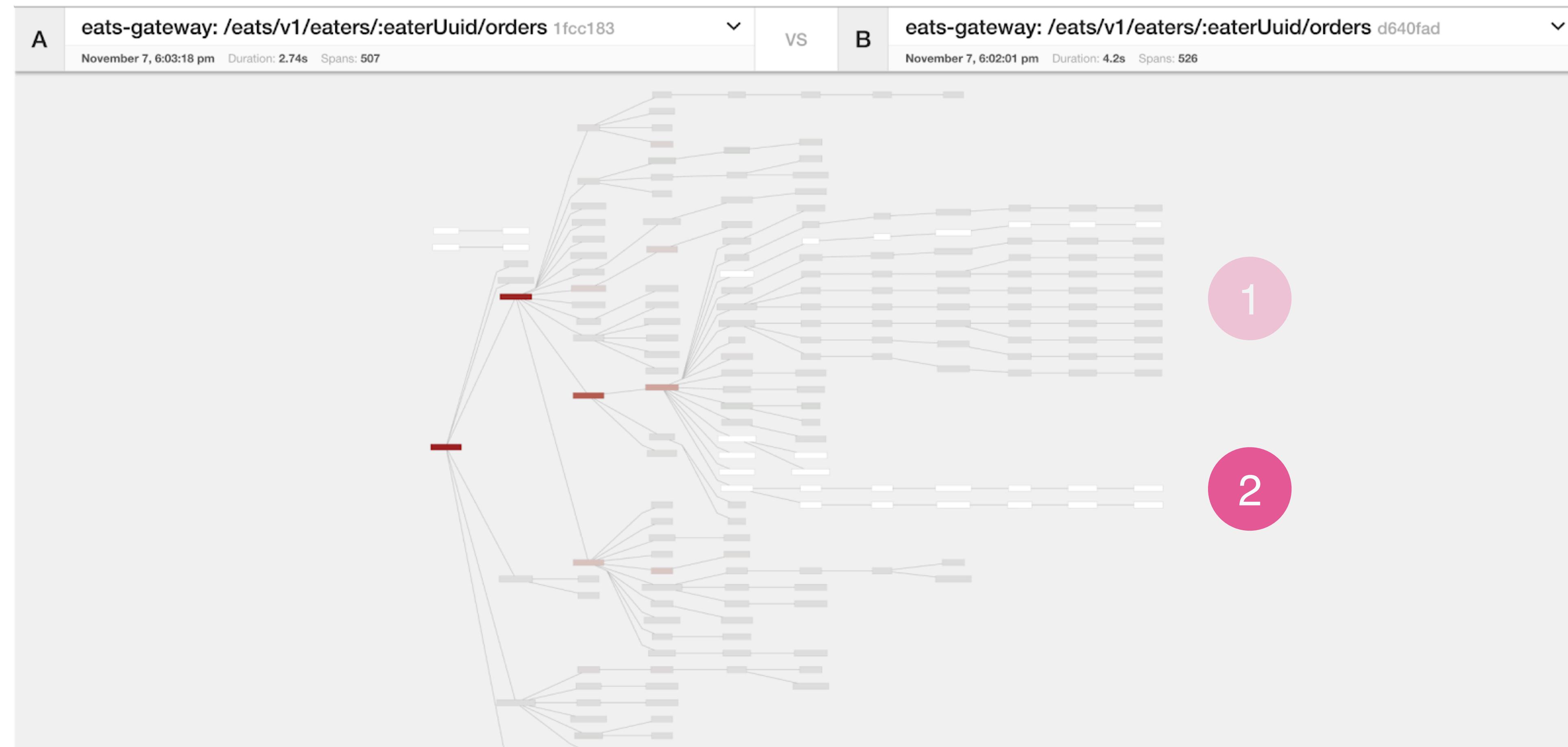
Comparing trace durations

Similar durations (grey)



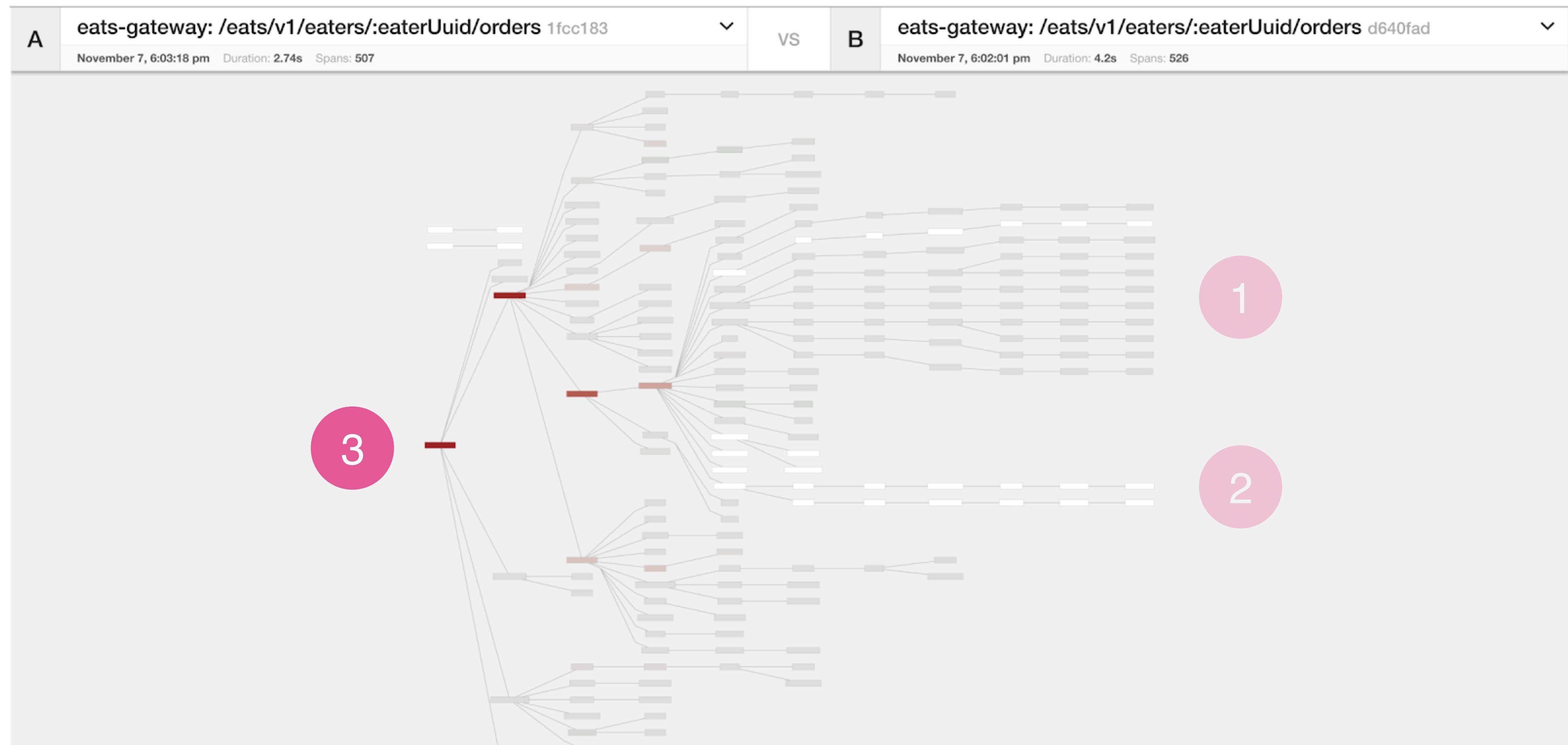
Comparing trace durations

Nodes that are not shared (white)



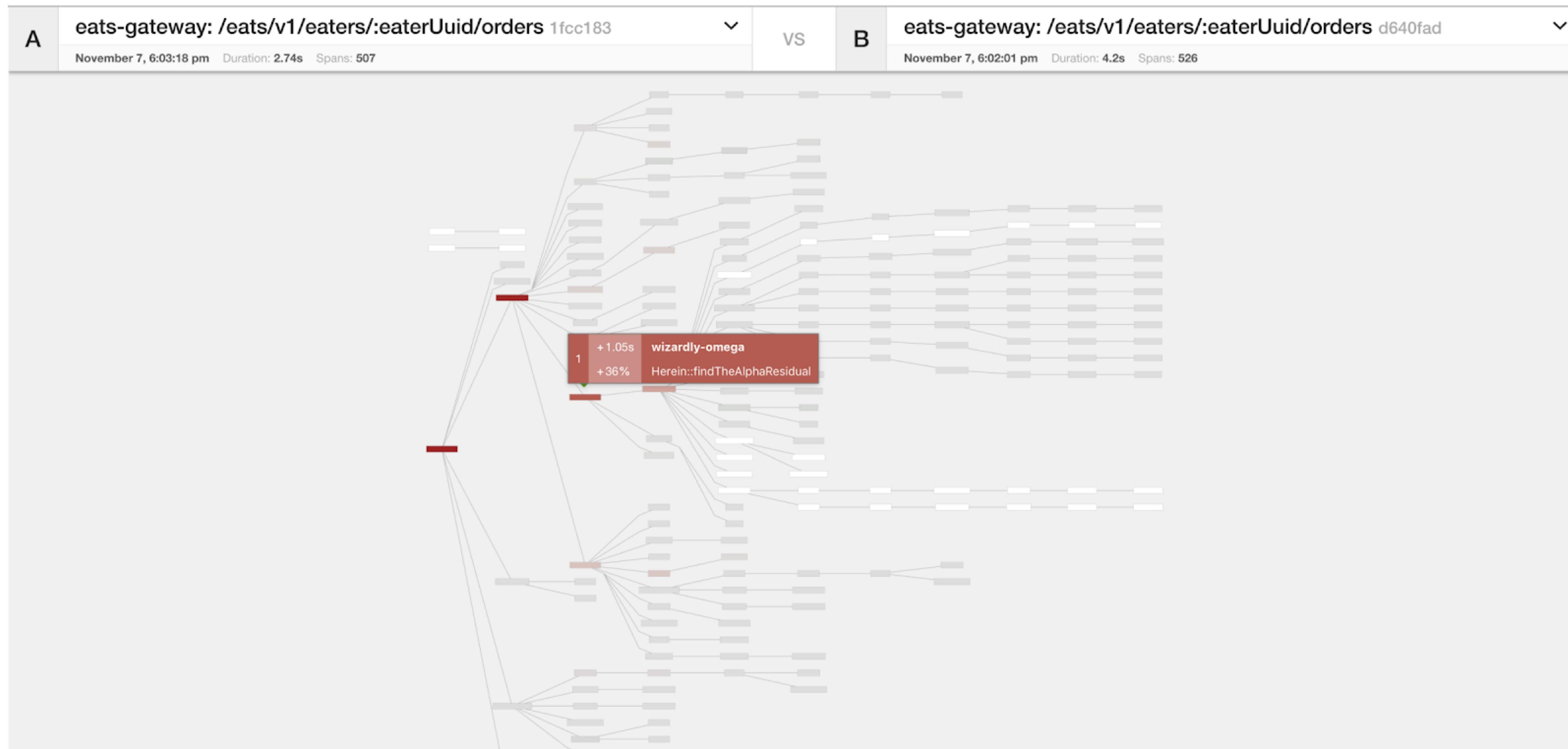
Comparing trace durations

Red heat-map for latency differences



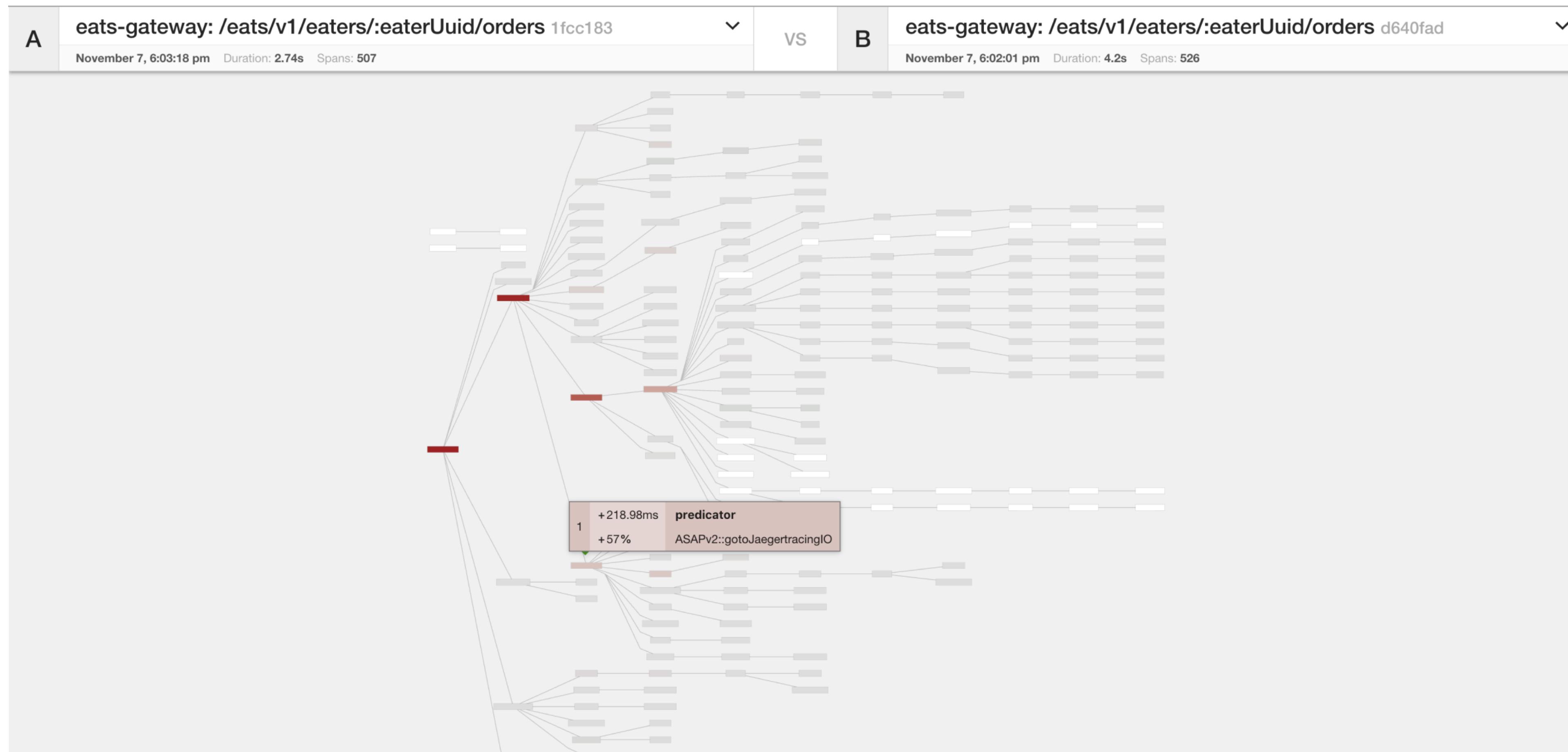
Comparing trace durations

Details on Mouse-Over



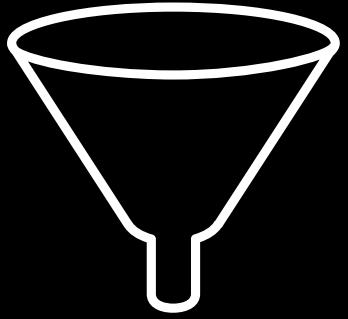
Comparing trace durations

Details on Mouse-Over

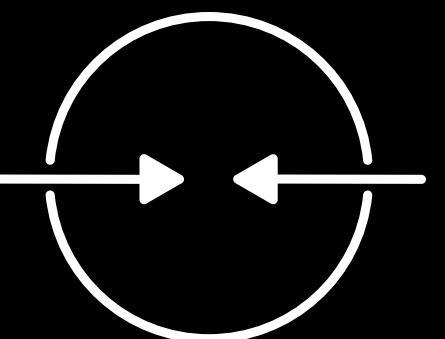


How Are These Approach Different?

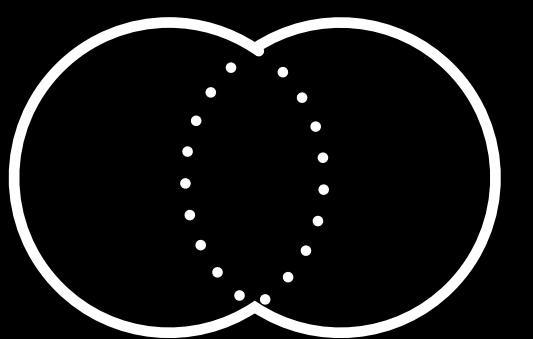
Summary



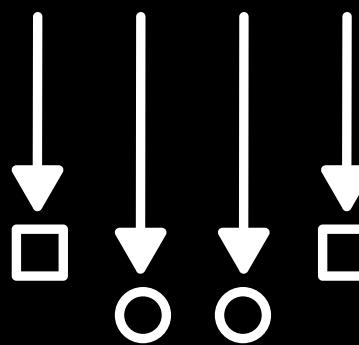
Surface less
information



Condense
the structural
representation



Emphasize
the differences



Distinct comparison
modes simplify
the comparisons

Challenges

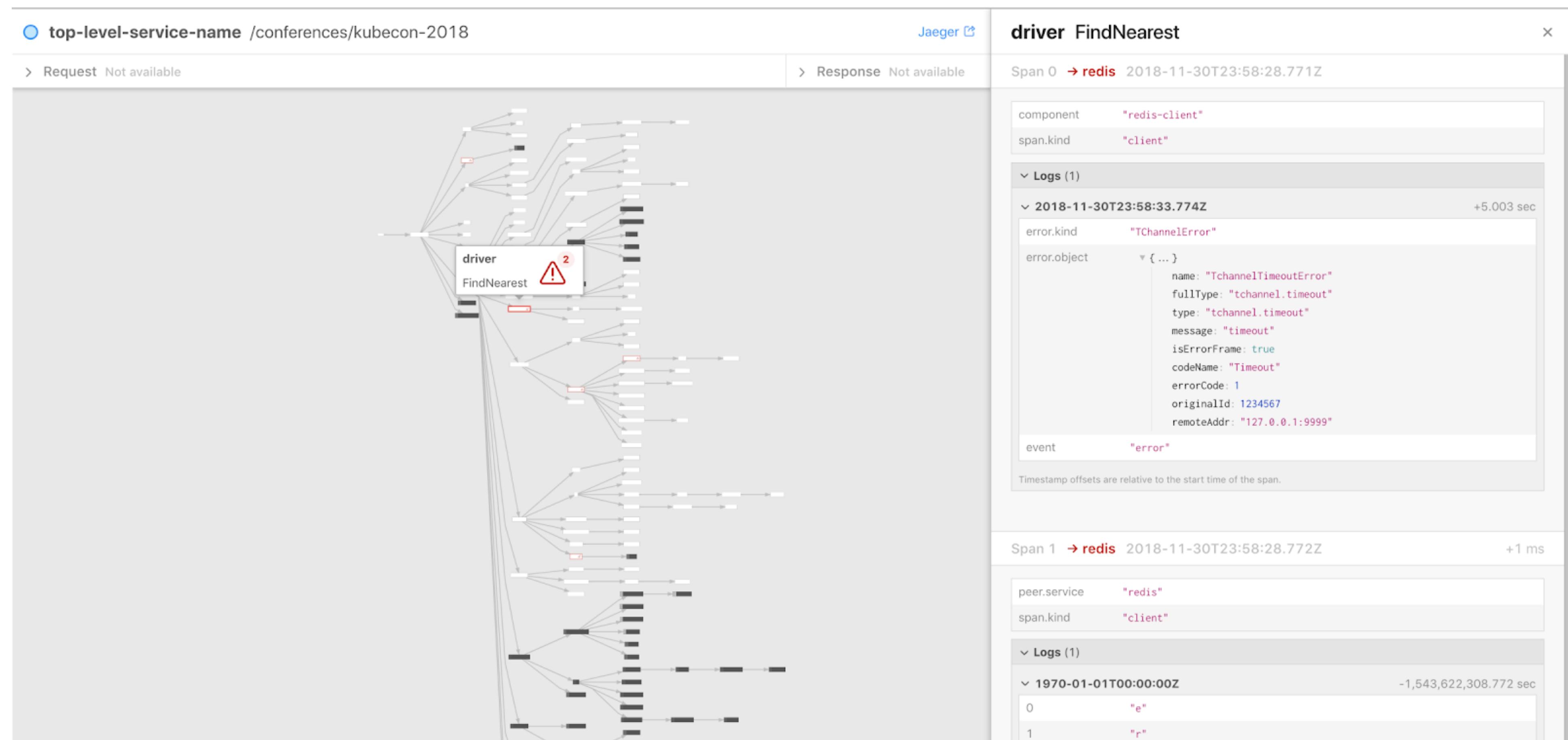
Individual traces can be an **outliers**.

User must find the **right** baseline.

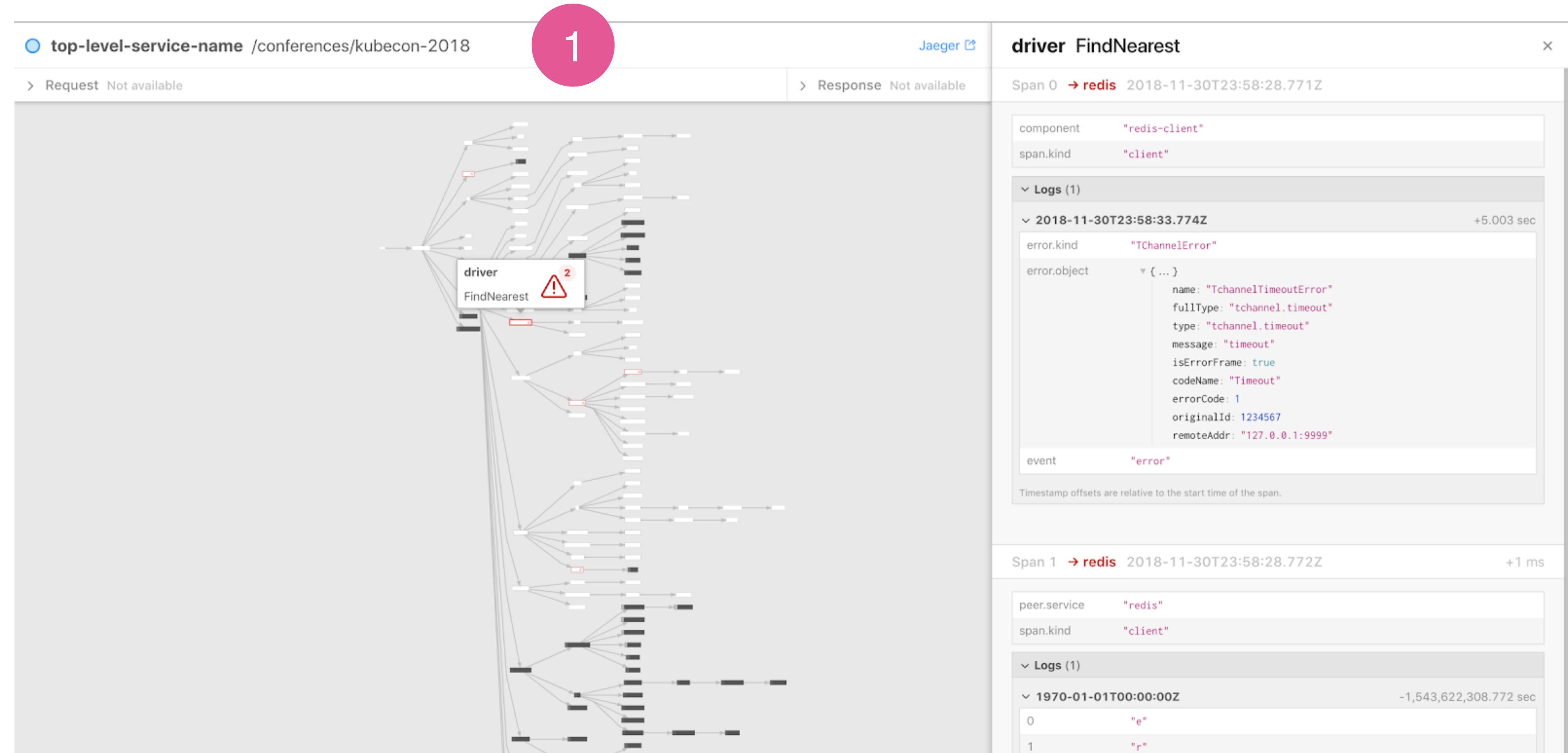
Traces vs. Trace

What Went Wrong?

Root Cause Analysis

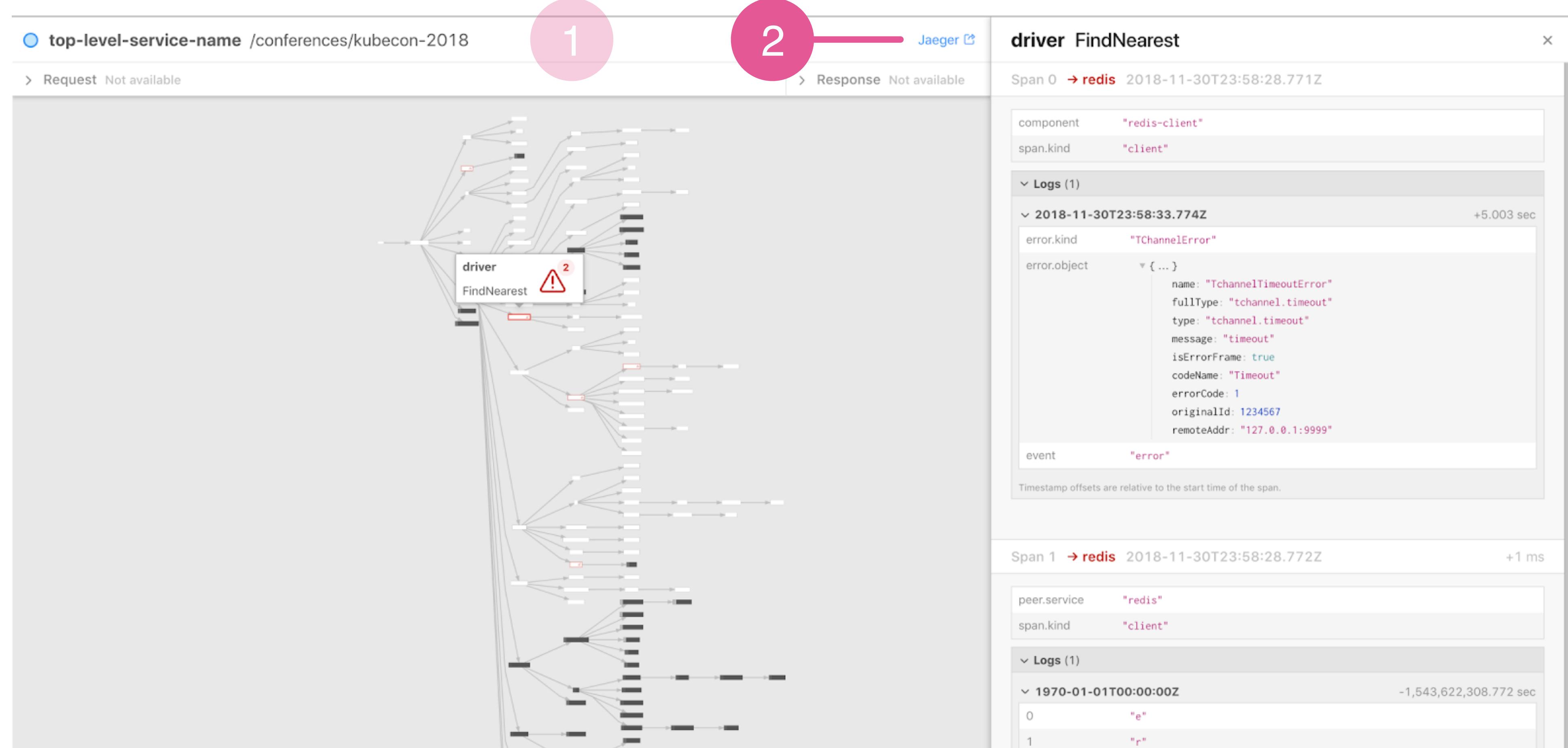


Top Level Outcome Including Request/Response Payloads



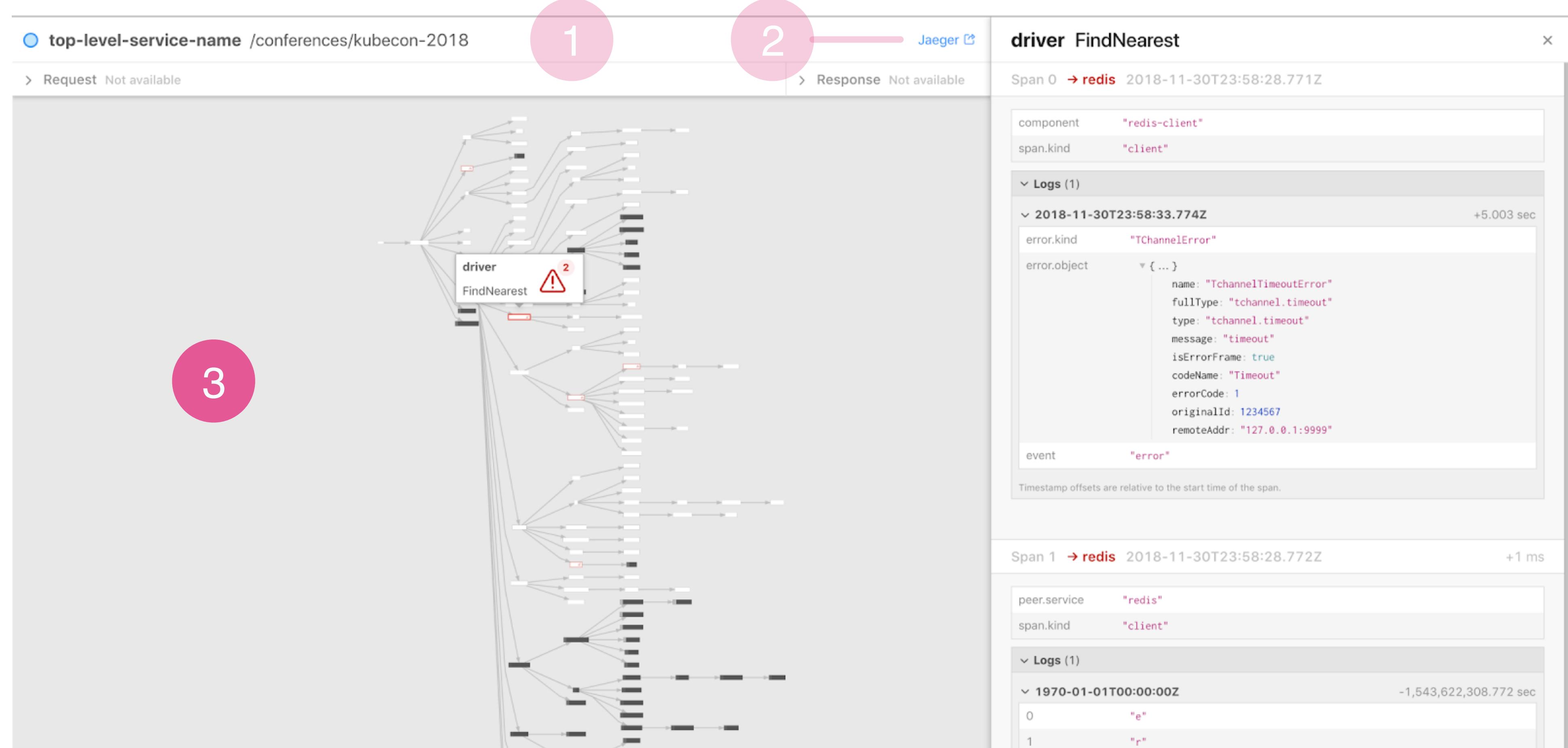
Link to the Trace

Can Always Go Back to Raw Data



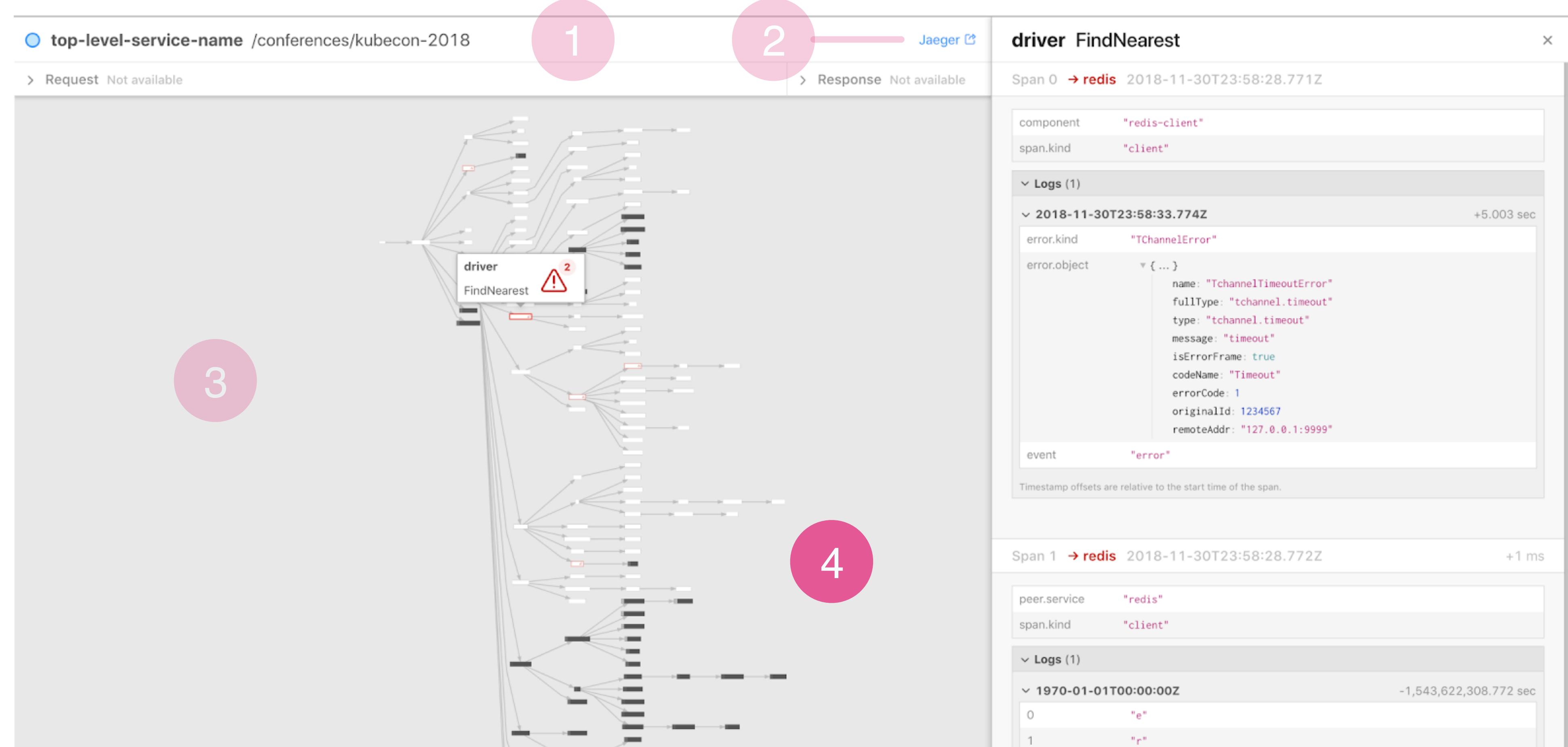
Trace Structure

Nodes Are Sorted Chronologically

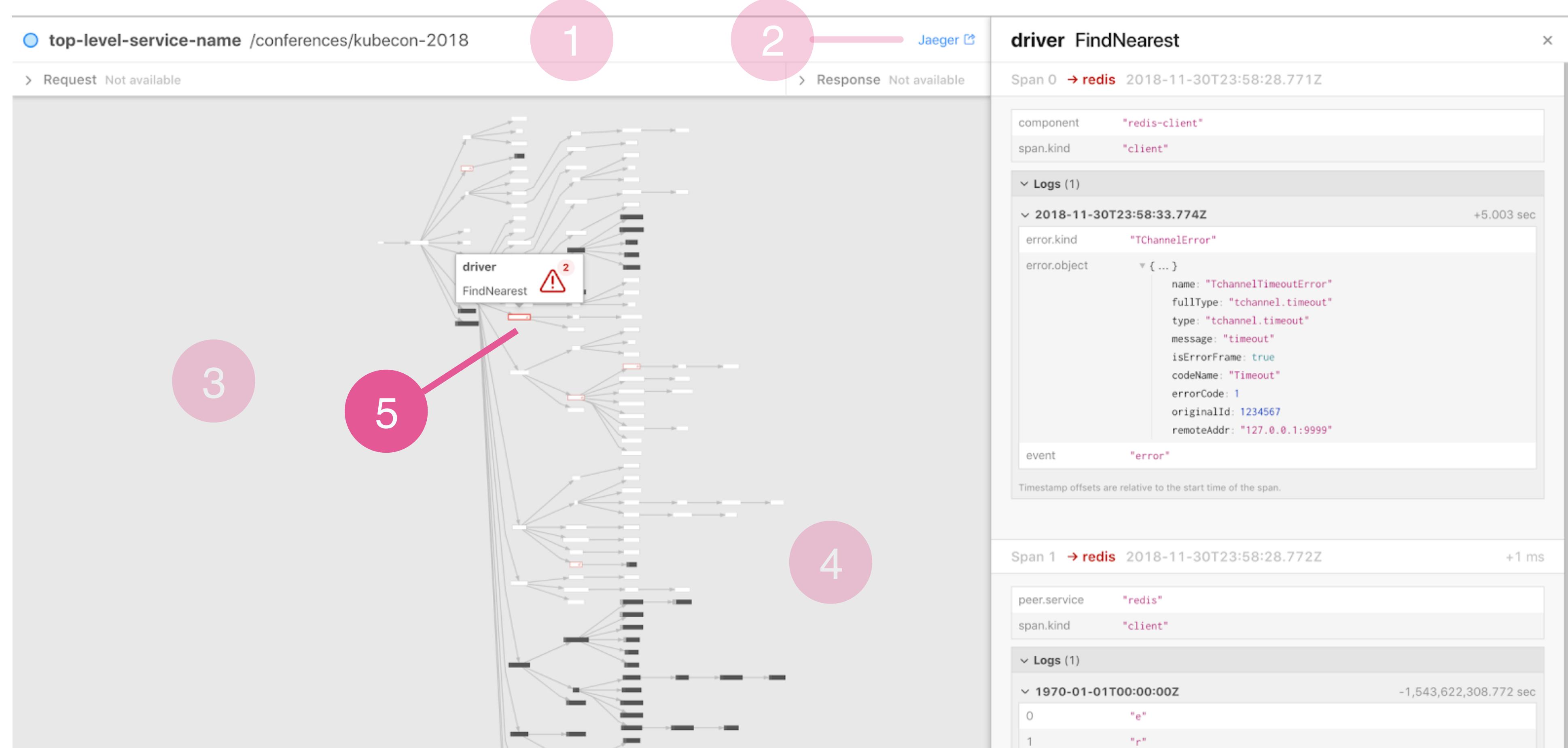


Present and Missing Nodes

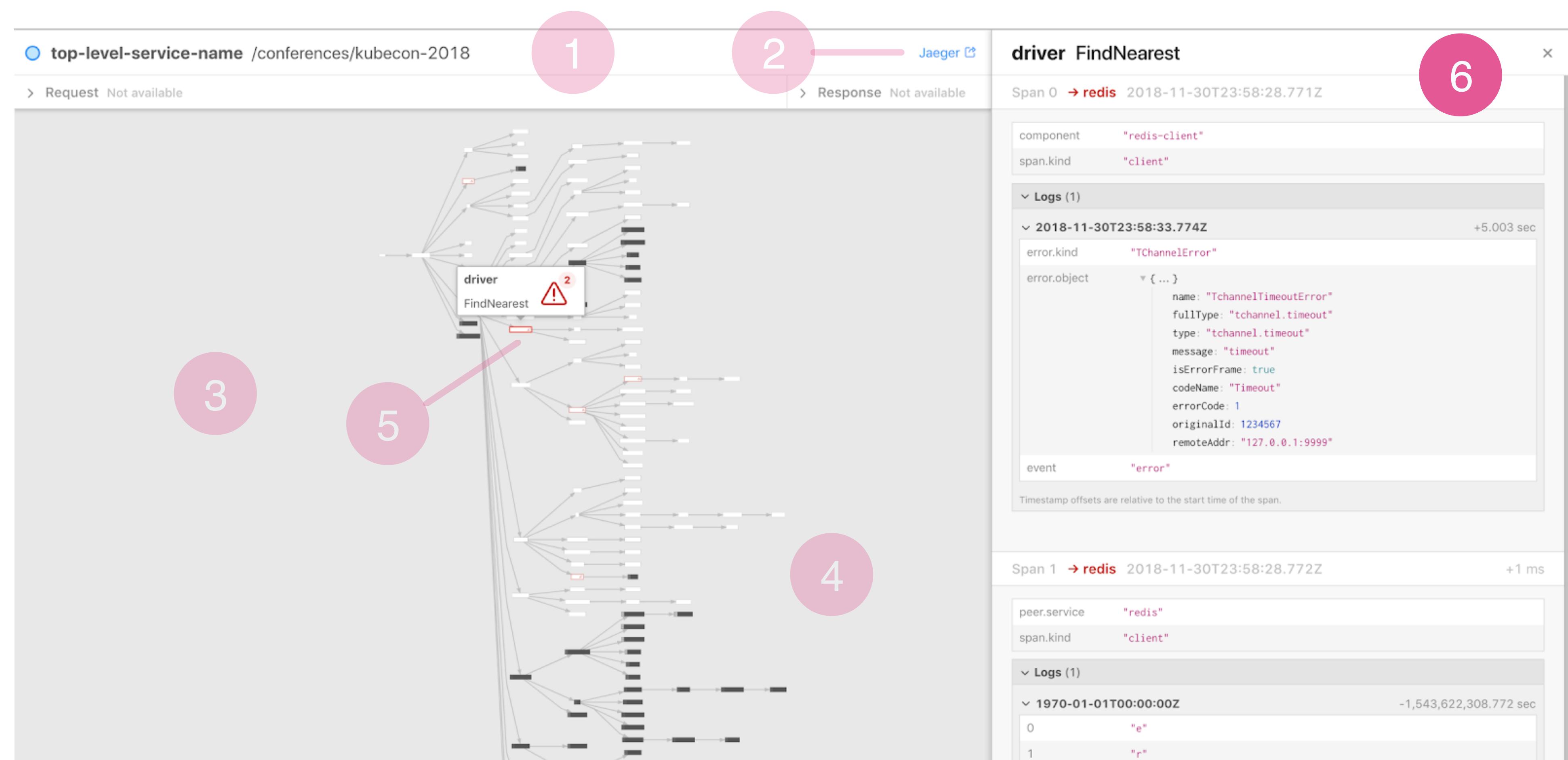
Color-Coding



A Node With Error Data

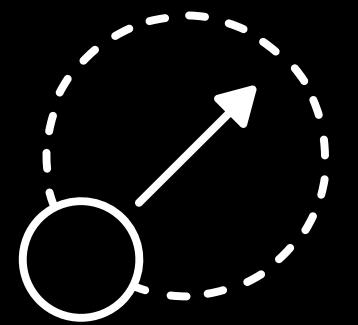


Error Data Panel



How Is This Approach Different?

Summary



Much broader
context:
aggregate vs.
one trace



One purpose: **root cause analysis** of reliability issues

Tackling Data Complexity

Uber is a data company

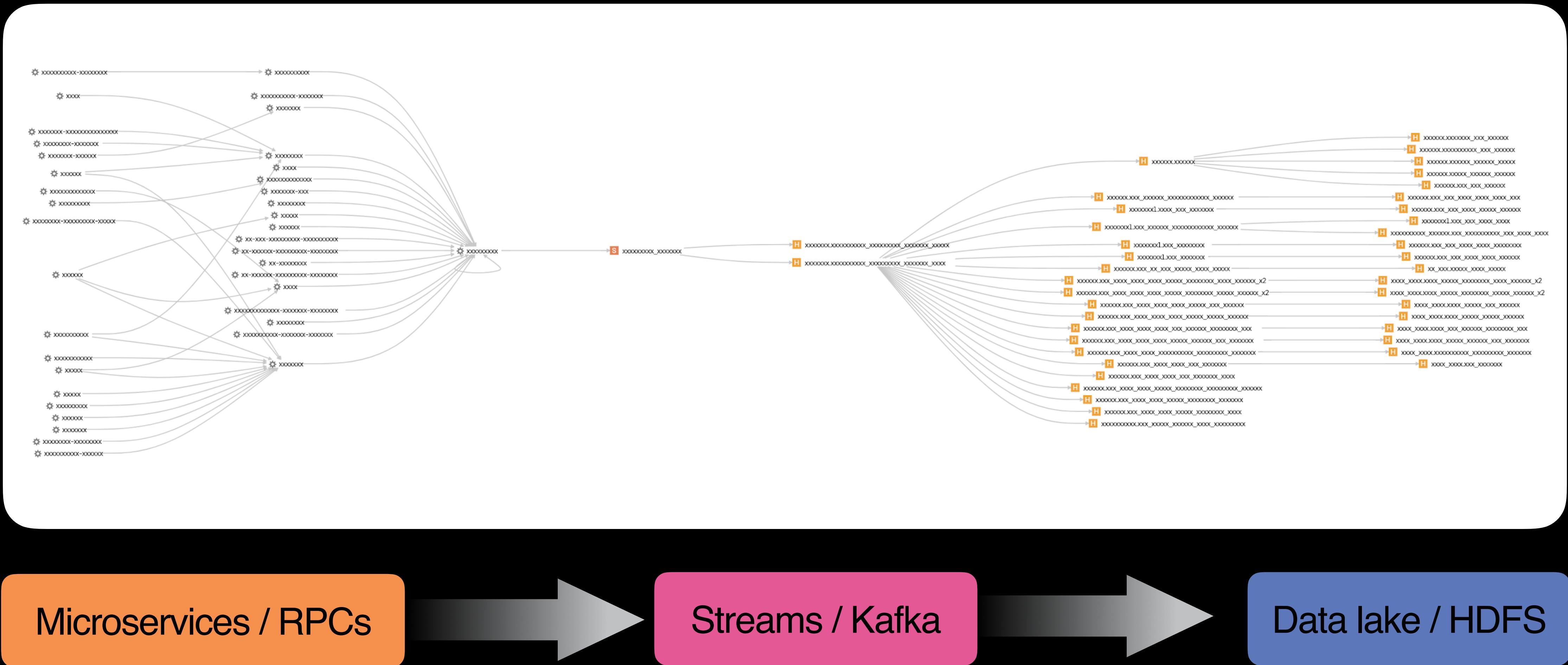
OK, and a transportation company



- Data undergoes many **transformations**
- More data is **derived** from other data
- Debugging **data quality** is difficult

Data Lineage

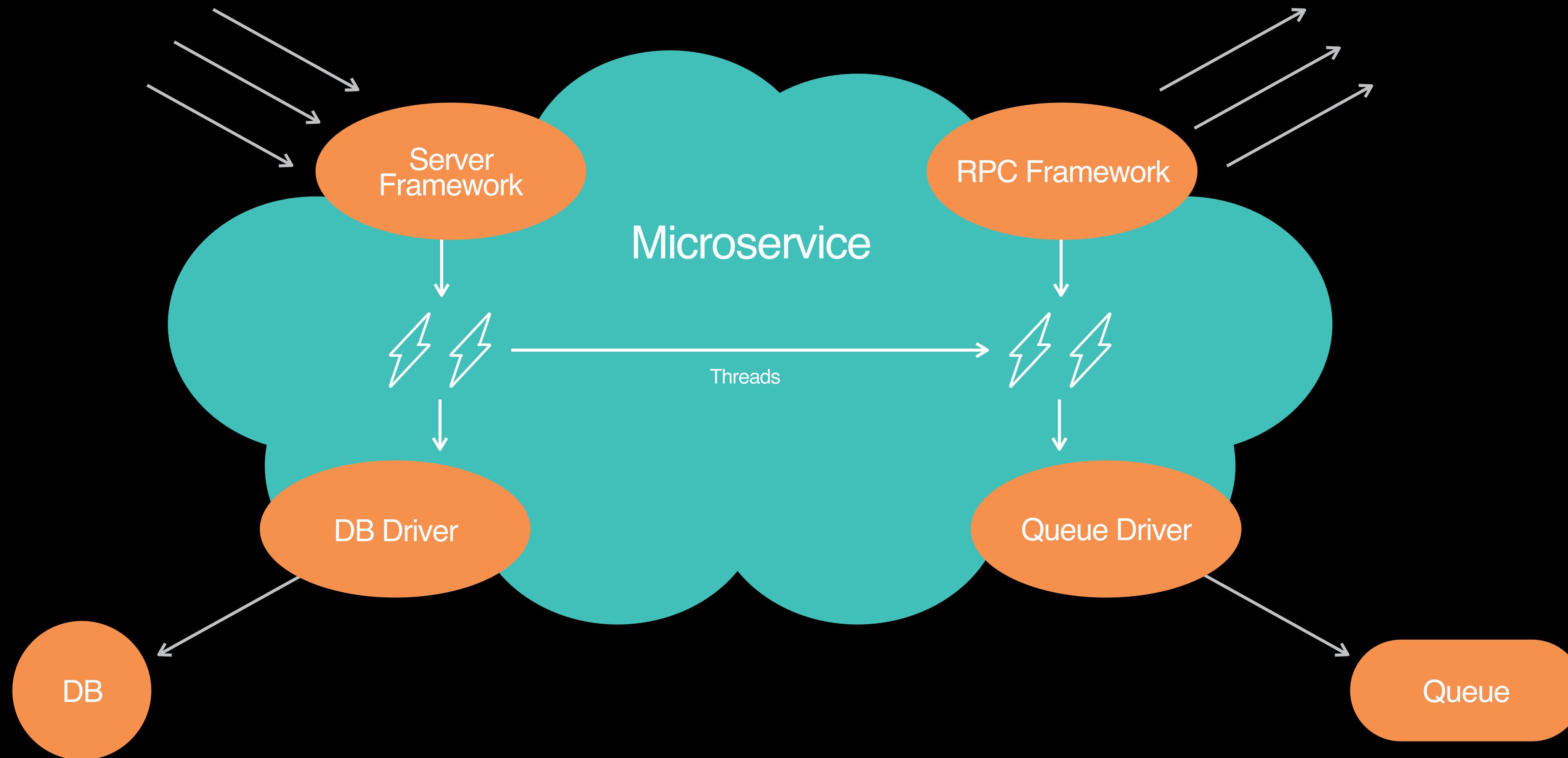
Debugging Data Quality



Observability requires
high quality instrumentation.

Our Software Is Highly Composable

Often from Open Source Components



Tracing **breaks** if components
don't understand each other.

Standardization Efforts

Instrumentation and Data Formats



OpenTelemetry

- Effective observability requires high-quality telemetry.
- OpenTelemetry makes robust, portable telemetry a built-in feature of cloud-native software.



- Distributed Tracing Working Group
- Data formats for on-the-wire trace context & correlation-context, and out-of-band trace data.

In Summary

Distributed tracing helps us
to deal with the overwhelming
complexity of microservices

In Summary

Creative visualizations
are essential
in performance analysis

In Summary

Distributed tracing empowers
unparalleled insights
into our distributed systems

Thank You

Find me @ shkuro.com

Q&A