CLOUD NATIVE
COMPUTING FOUNDATION

Jaeger
Project Intro

Pavol Loffay (Red Hat), Yuri Shkuro (Uber)

CloudNativeCon NA, San Diego, Nov-19-2019

Agenda

- What is tracing
- Demo
- Project status
- New Features
- Roadmap
- Q & A

About

- Pavol Loffay (https://github.com/pavolloffay)
  - Software engineer at Red Hat
  - Maintainer of Jaeger, OpenTracing, OpenTelemetry
- Yuri Shkuro (https://github.com/yurishkuro)
  - Software engineer at Uber Technologies
  - Maintainer of Jaeger, OpenTracing, OpenTelemetry
  - Author of "Mastering Distributed Tracing" book

What is Tracing & Why?

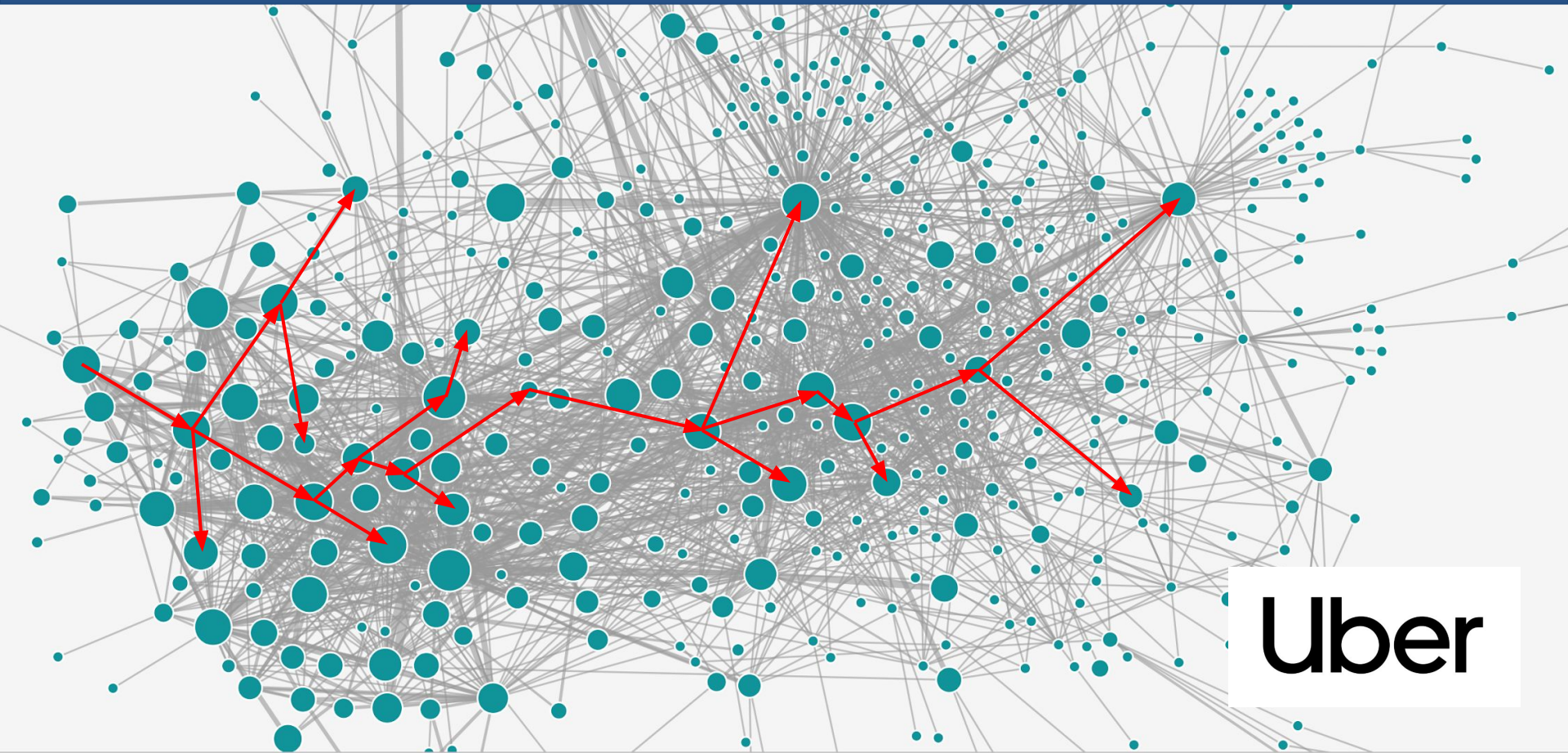
Concepts and terminology

Modern Distributed Systems are COMPLEX

Loading Netflix or Facebook home page ⇒

dozens of microservices, 100s of nodes

BILLIONS of times a day!

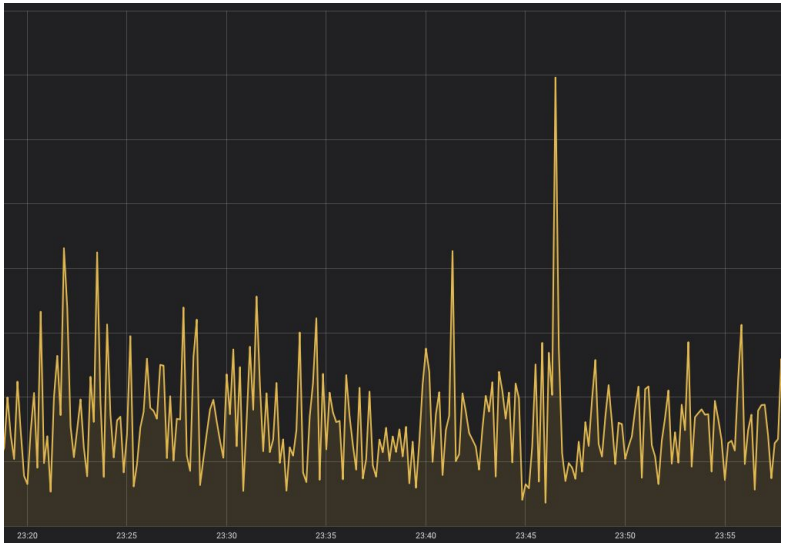


How can we tell what is going on?

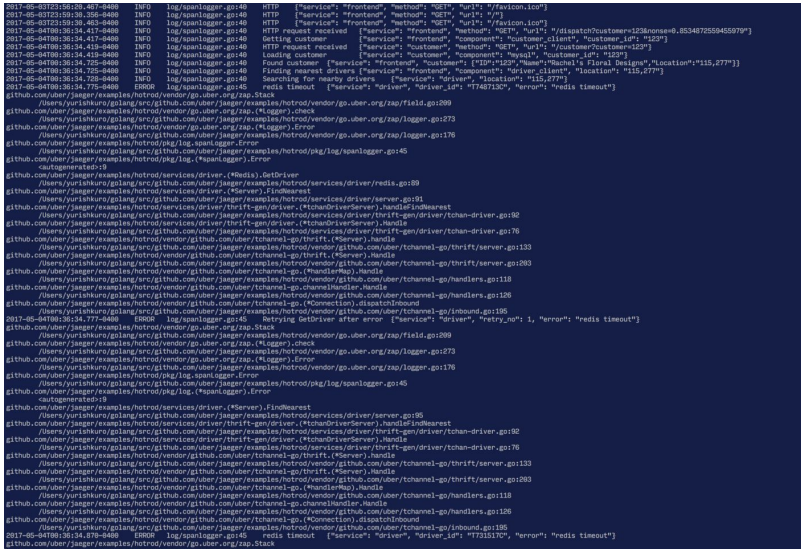
Which service is to blame

when things go wrong or become slow?

Traditional monitoring tools don't help



Metrics show something is wrong, but do not explain why.



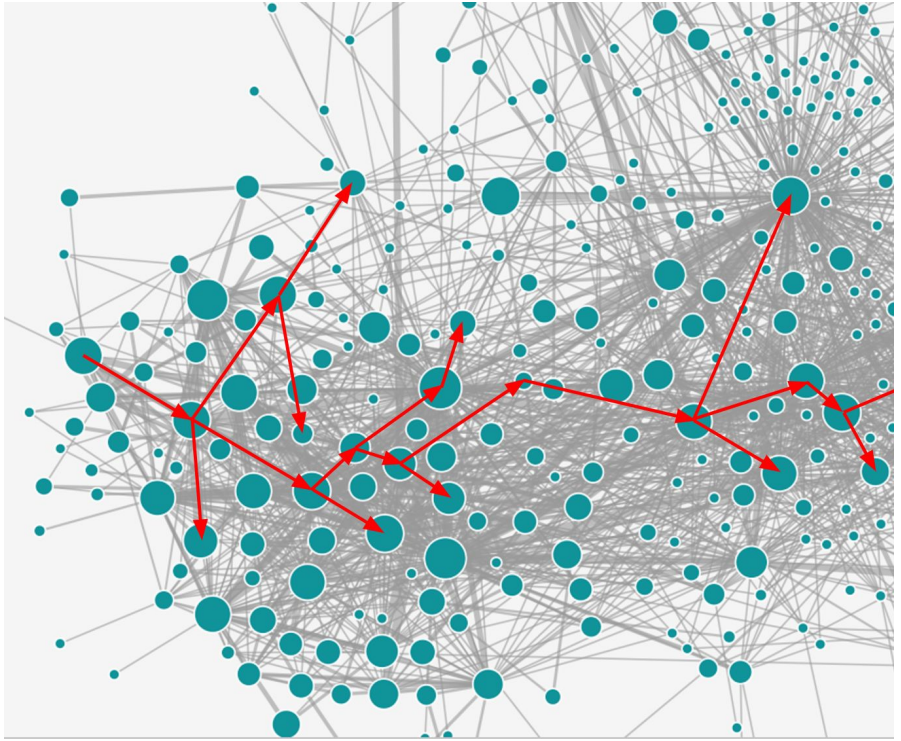
Logs are a mess: concurrent requests, multiple hosts, impossible to correlate.

Monitoring tools must tell stories!

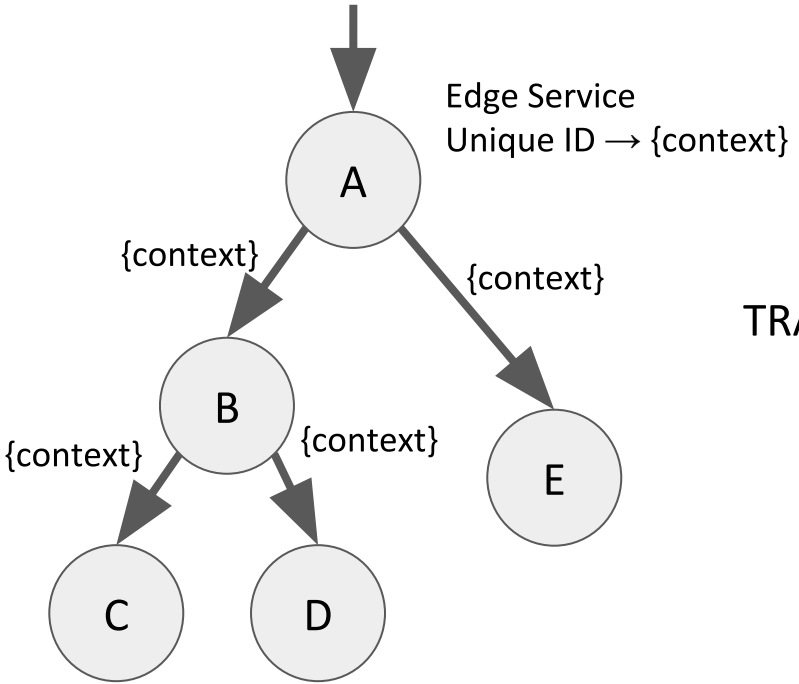
Do you like debugging without a stack trace?

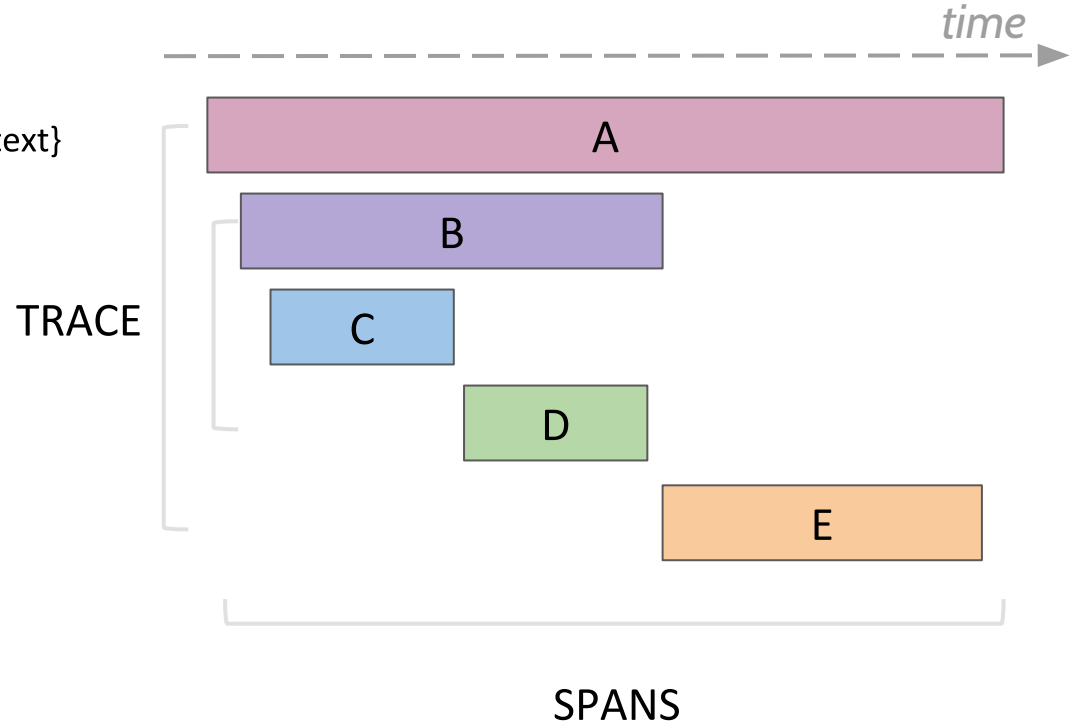
We need to monitor distributed transactions

⇒ **distributed tracing**!



Context Propagation & Distributed Tracing

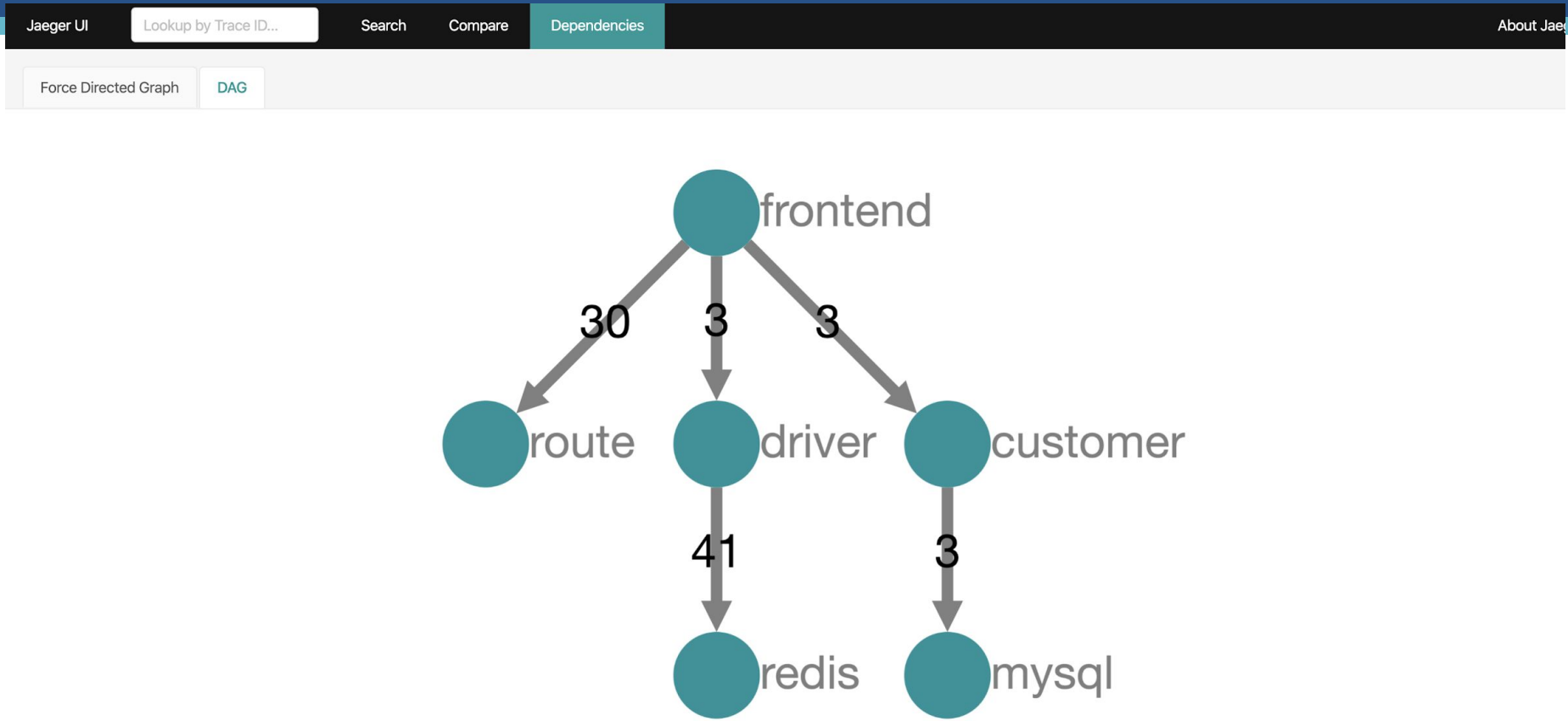




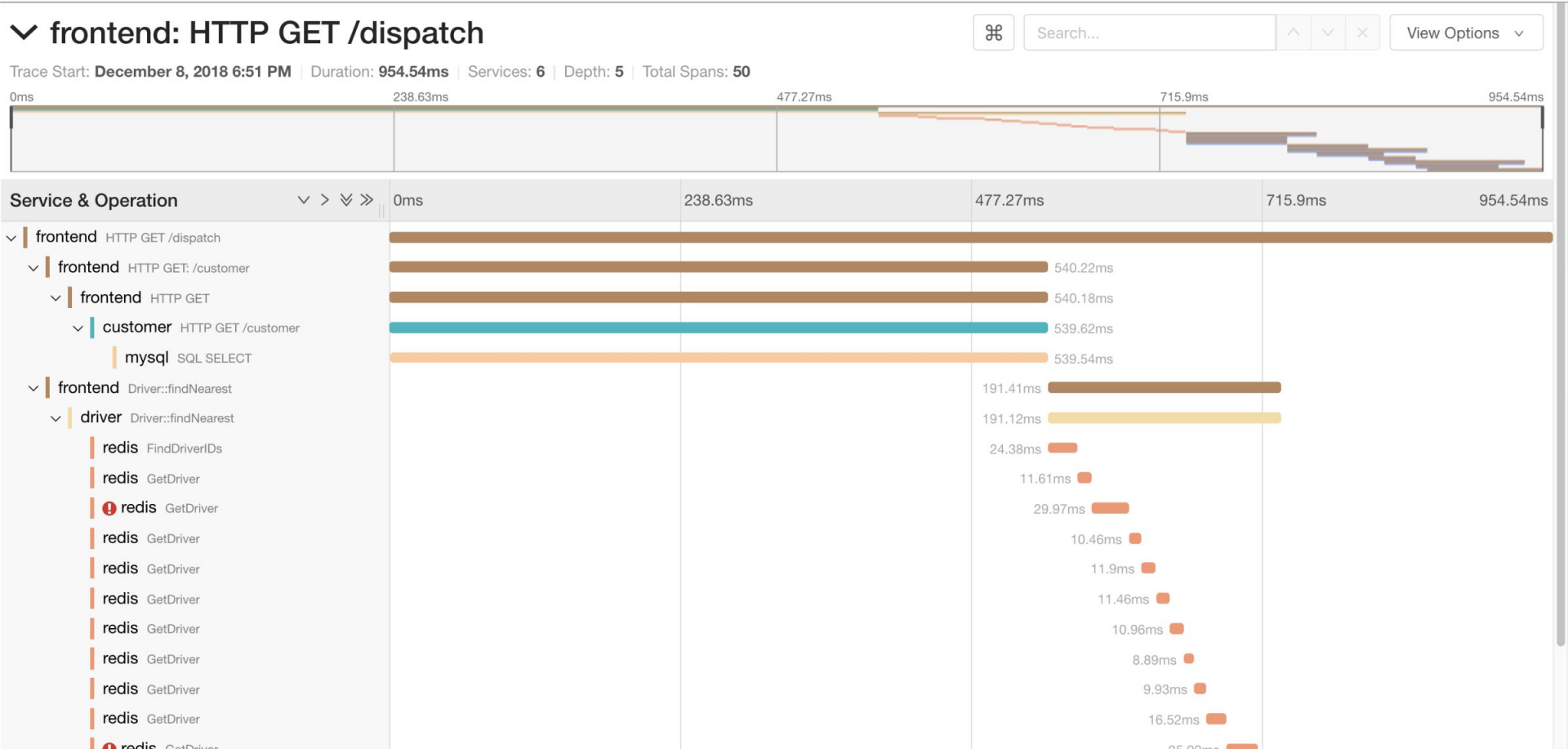
Let's look at some traces

http://bit.do/jaeger-hotrod

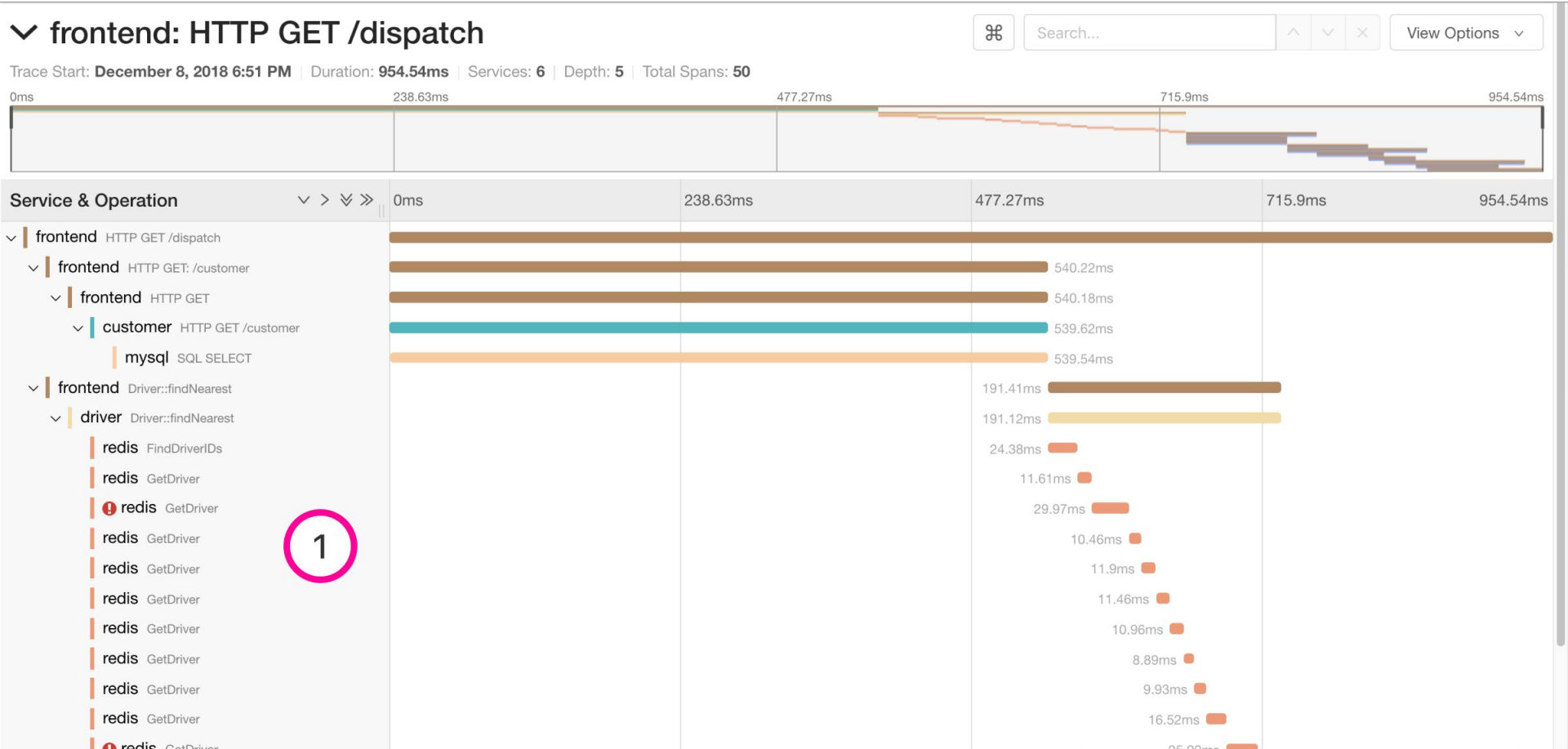
Service dependencies diagram



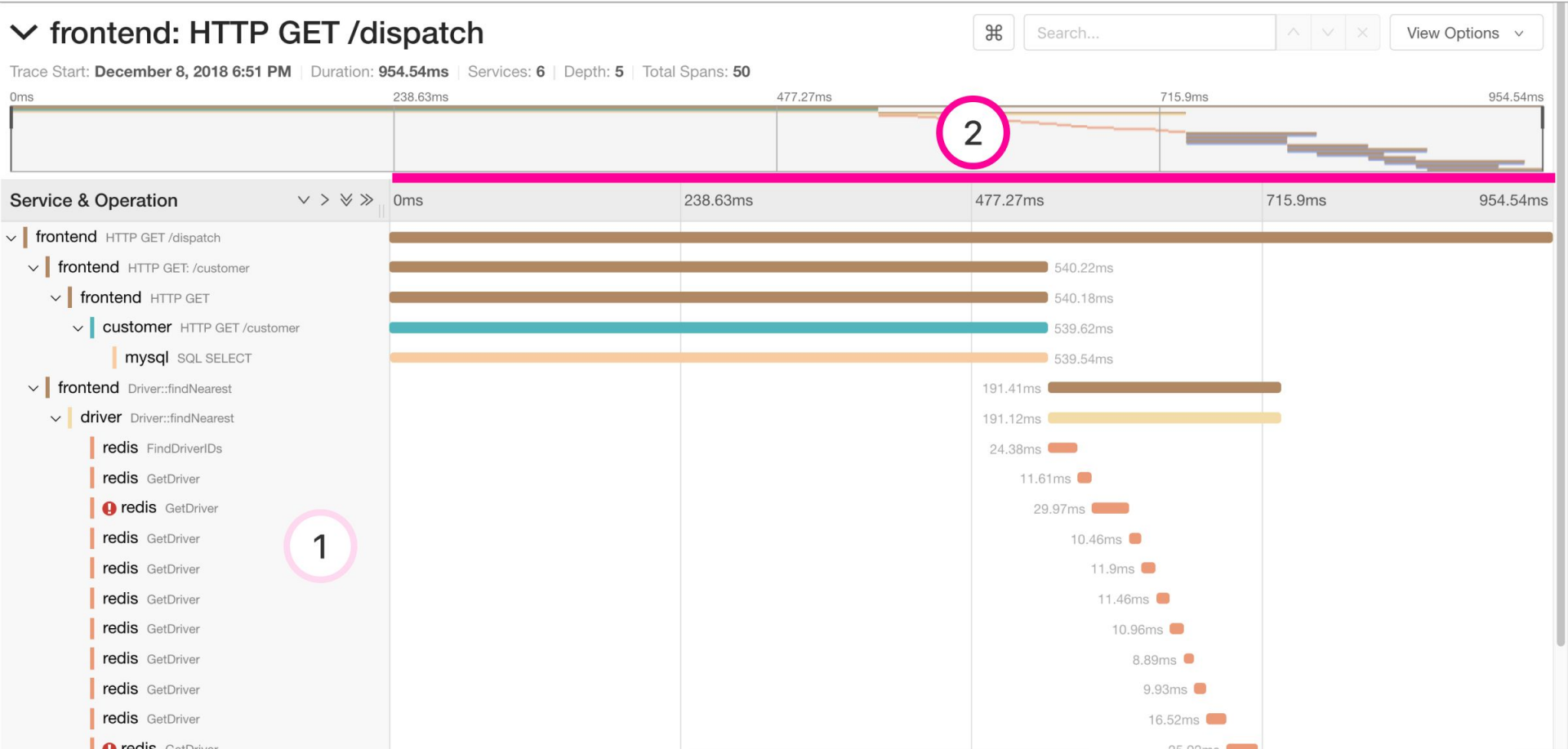
Trace timeline



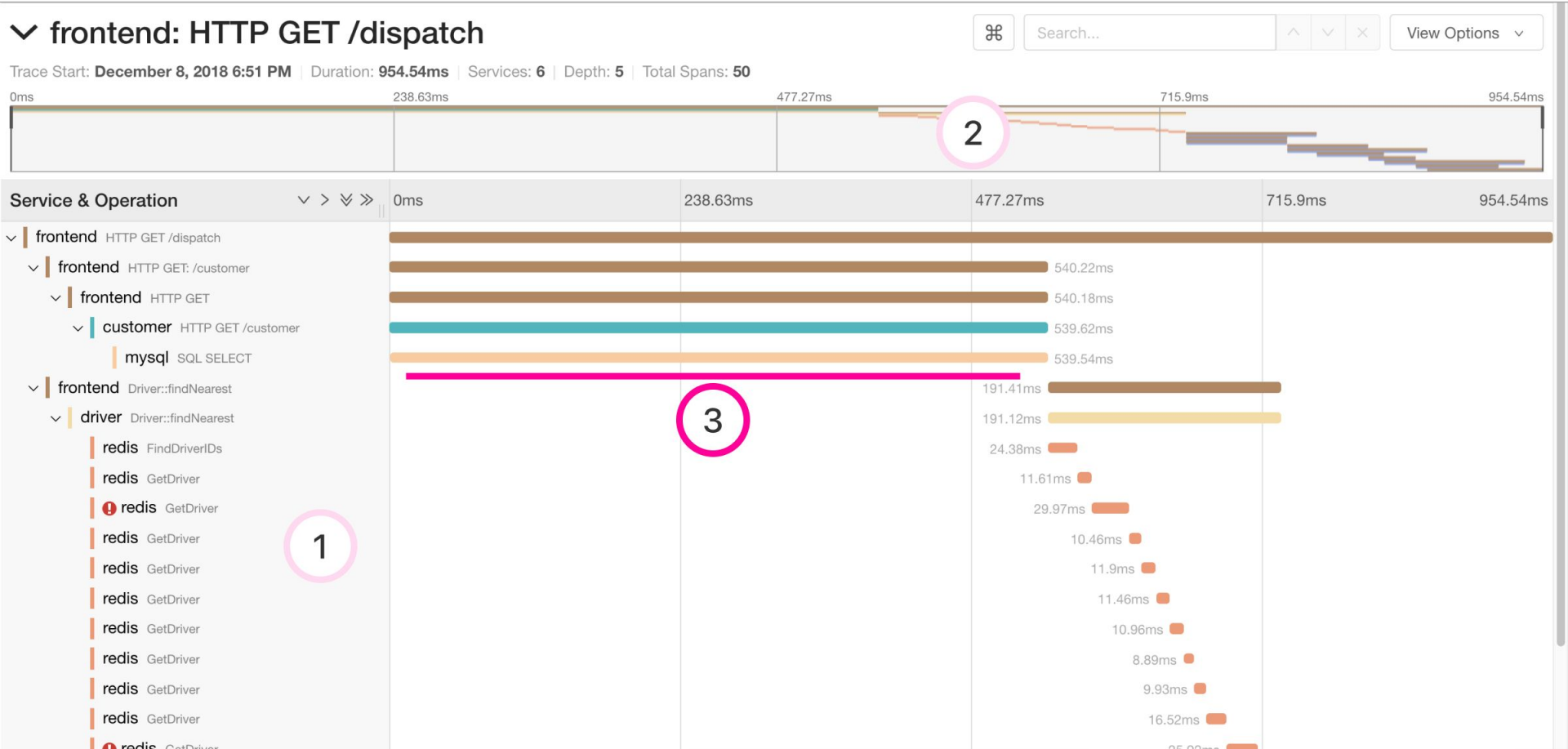
Trace timeline – Parent → Child → Grandchild



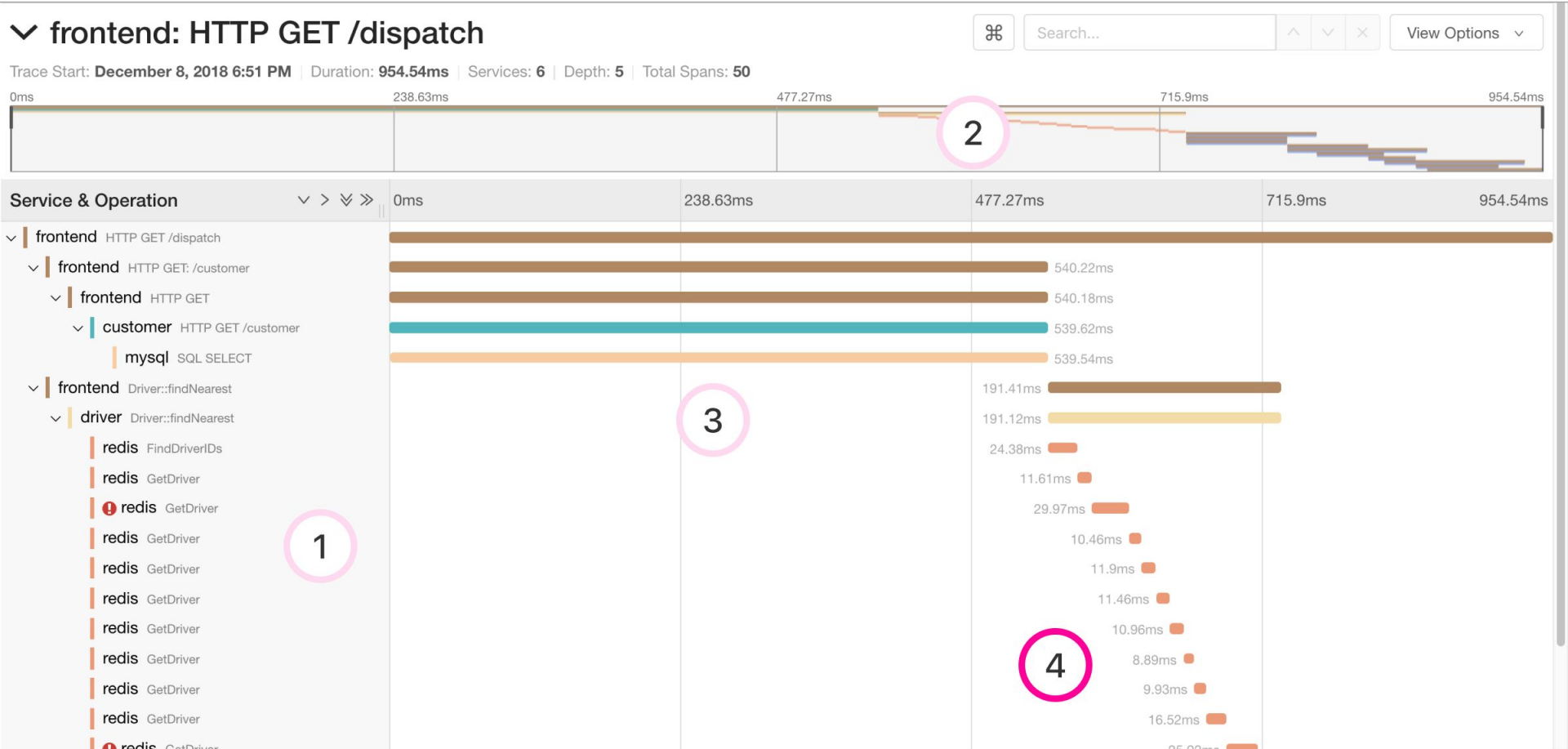
Trace timeline – Time + Mini-map



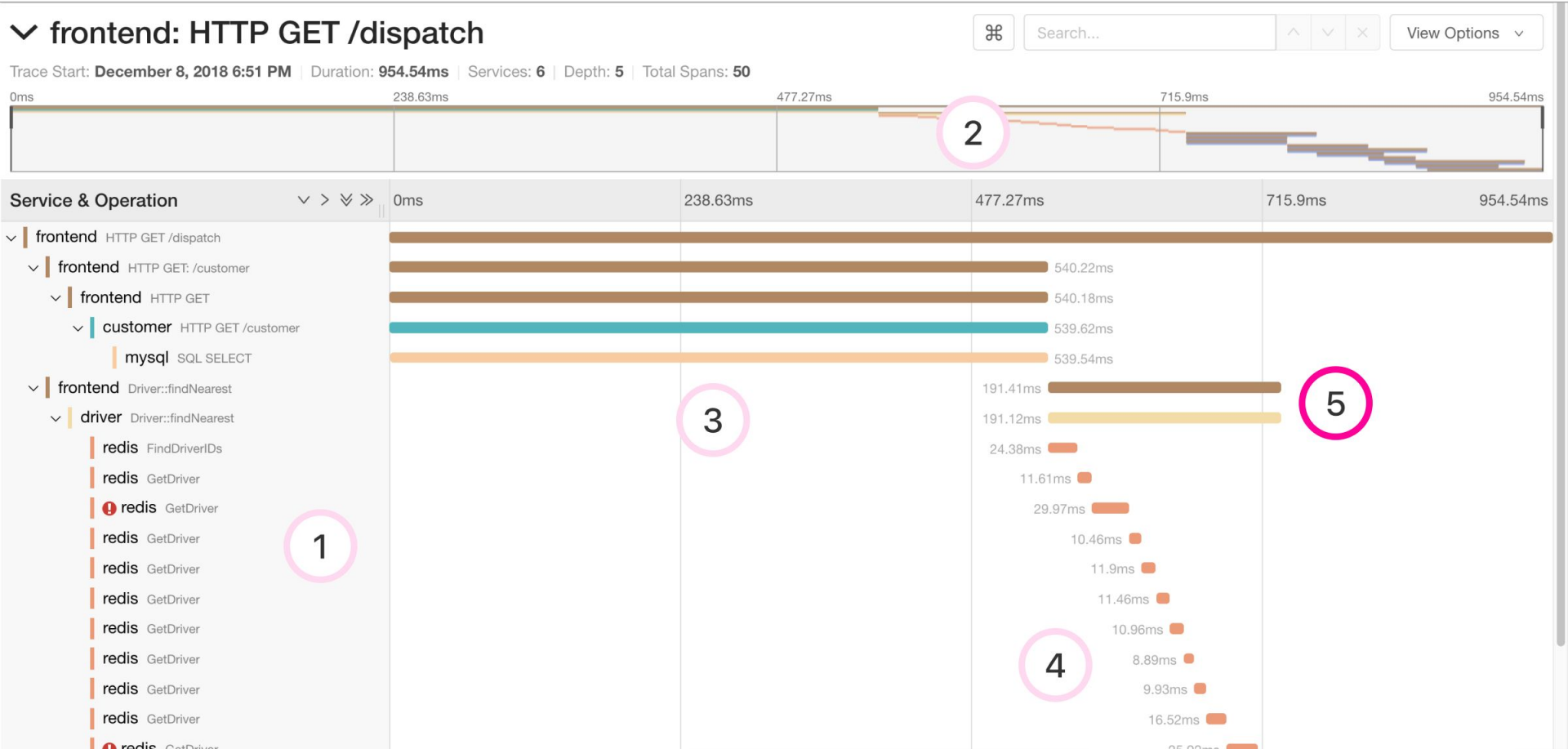
Trace timeline – A blocking operation



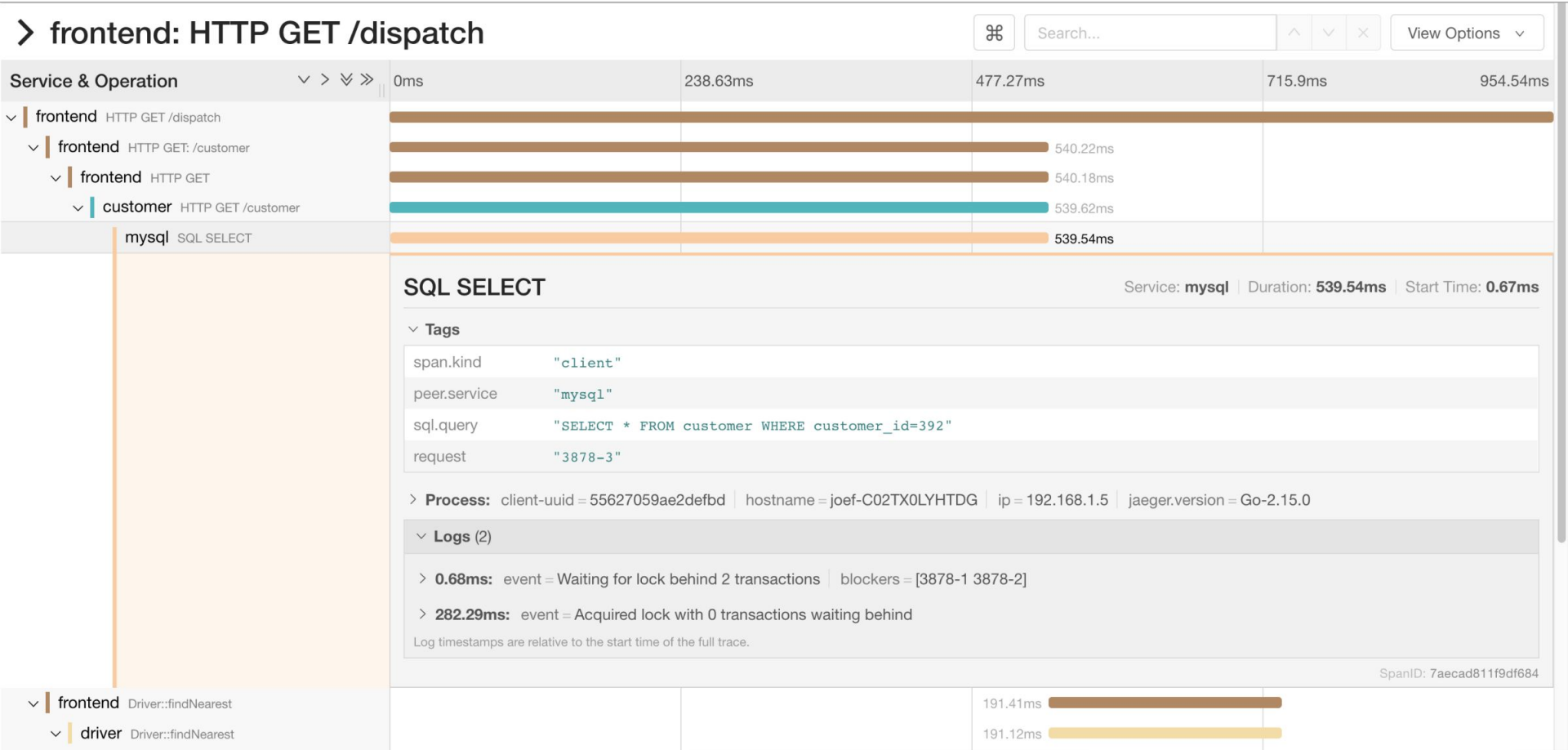
Trace timeline – Sequential operations



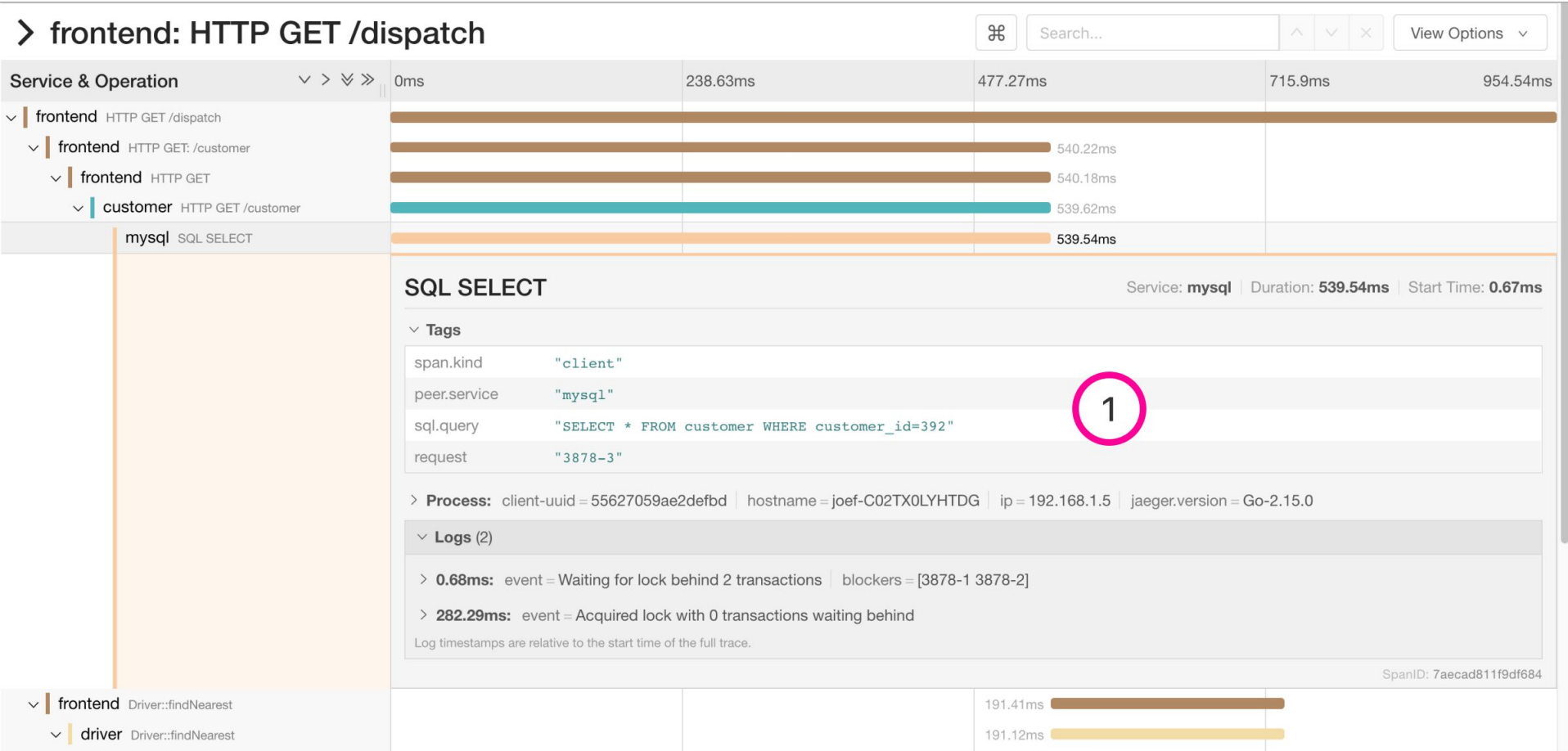
Trace timeline – Parents encompass descendents (generally)



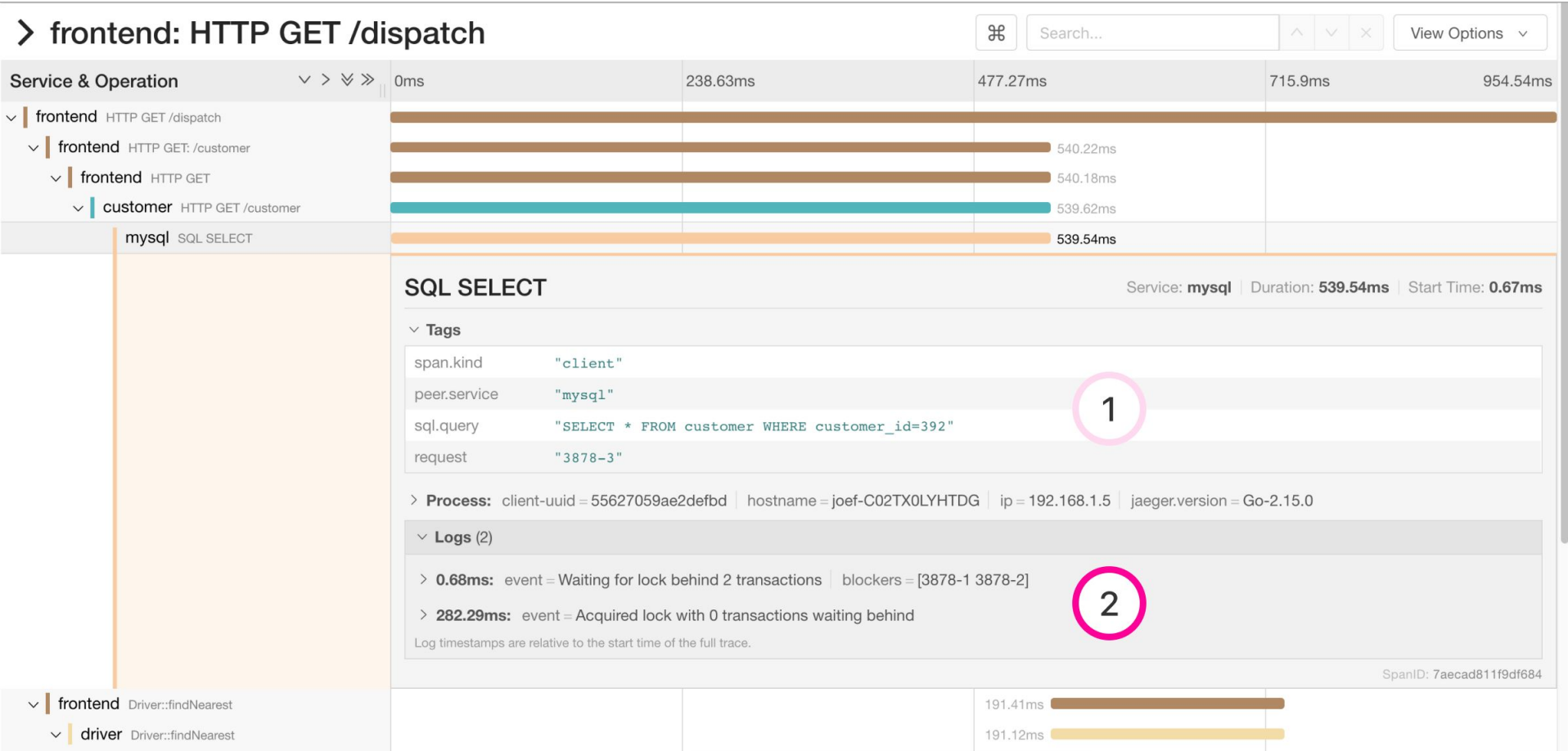
Span details



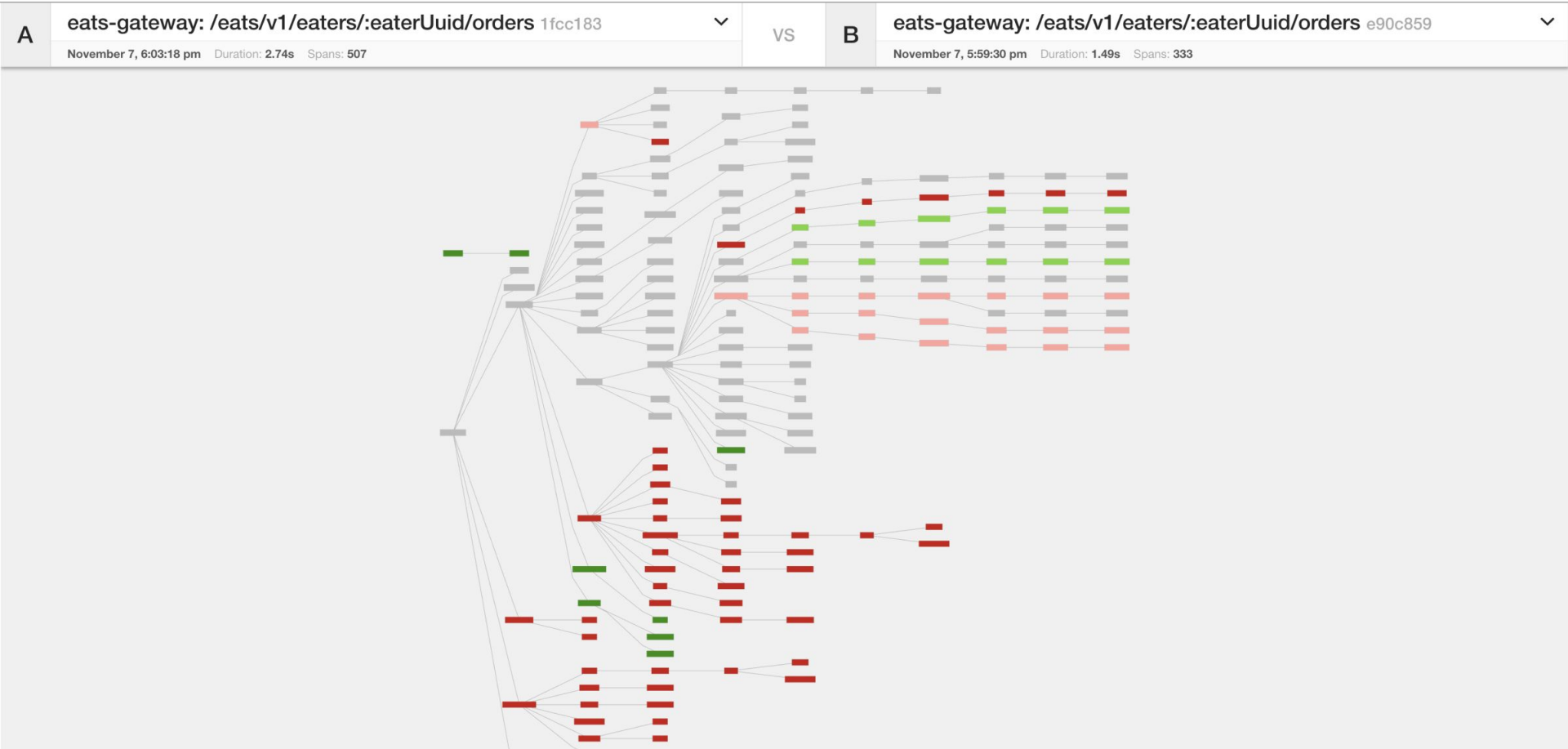
Span details – Database query



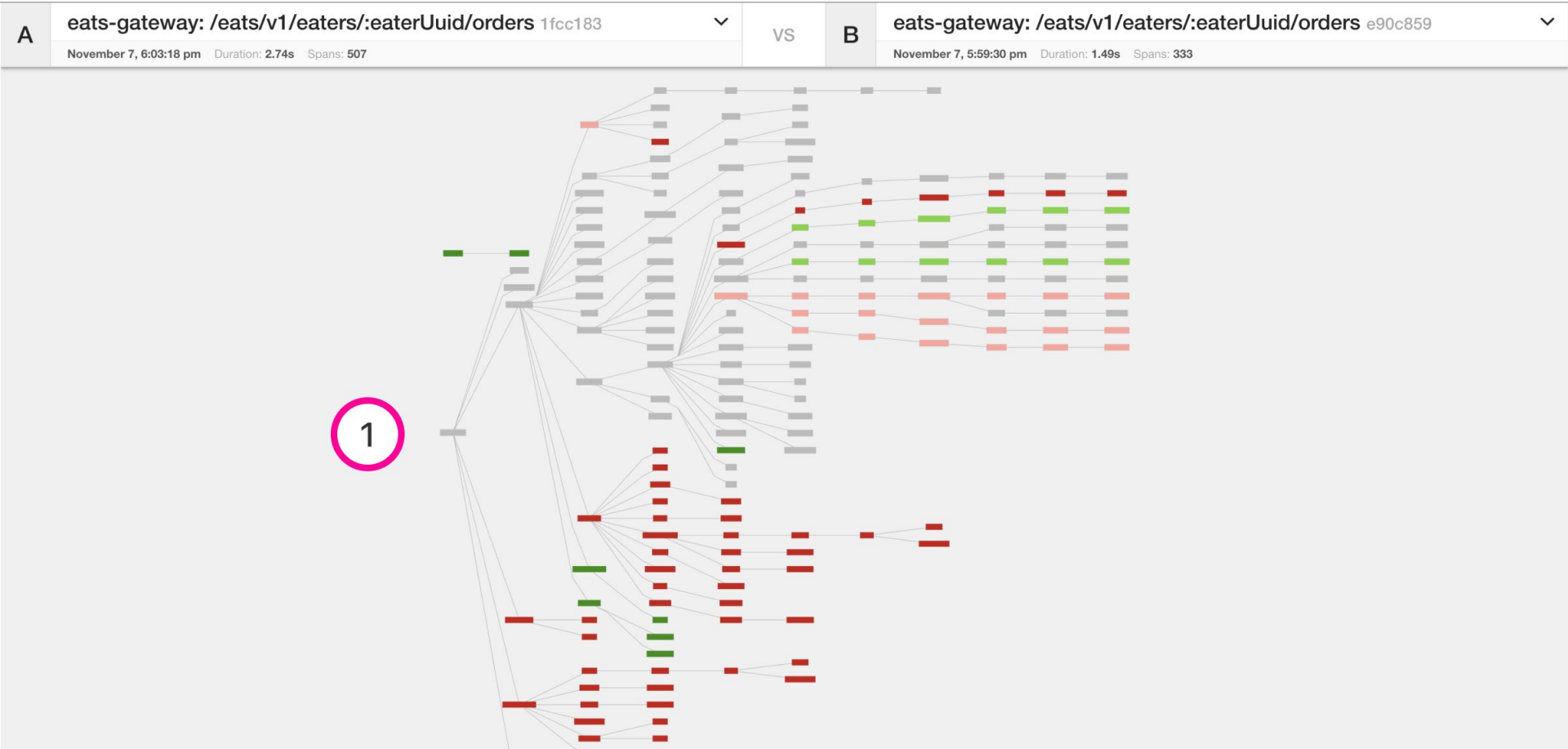
Span details – Lock contention



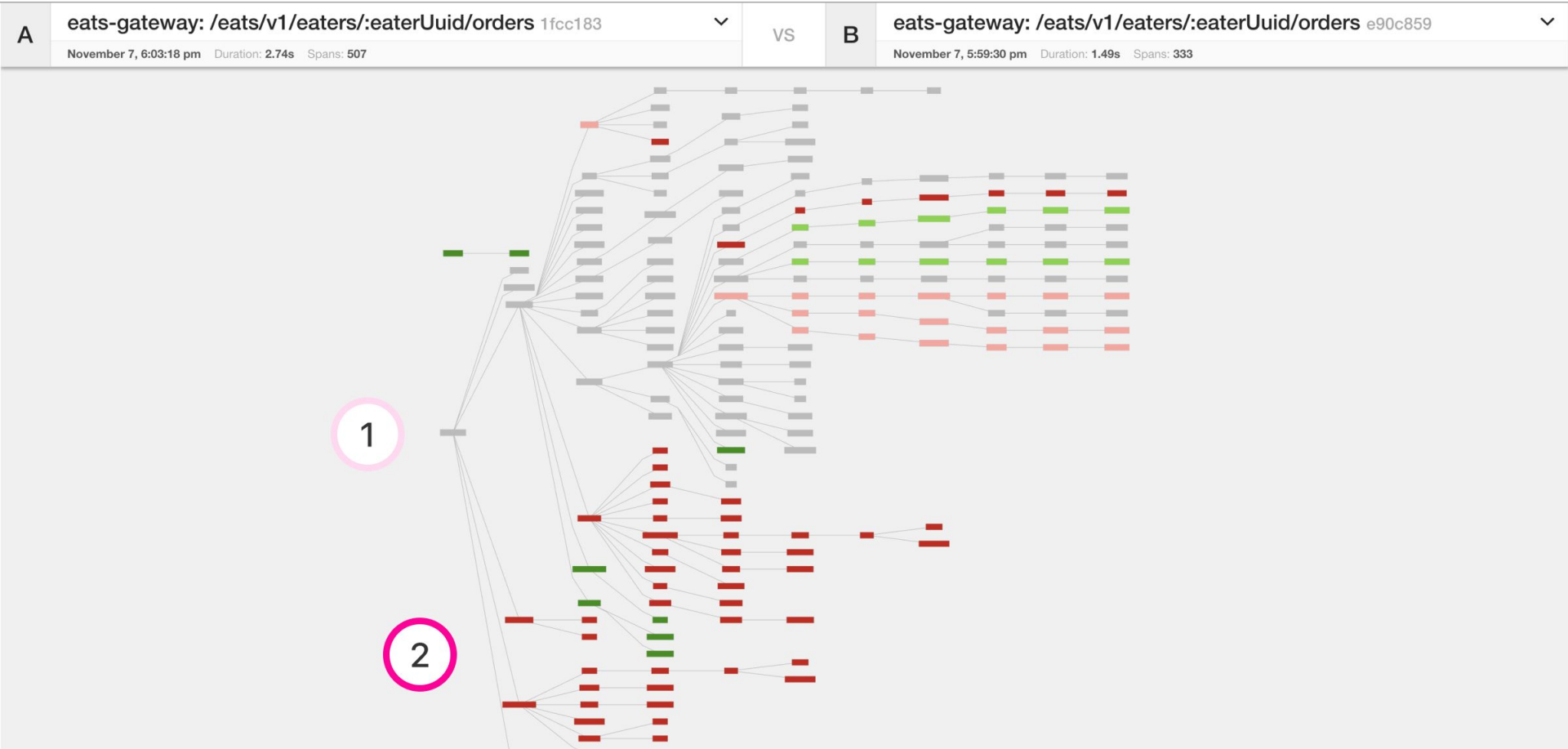
Comparing trace structures – Unified diff



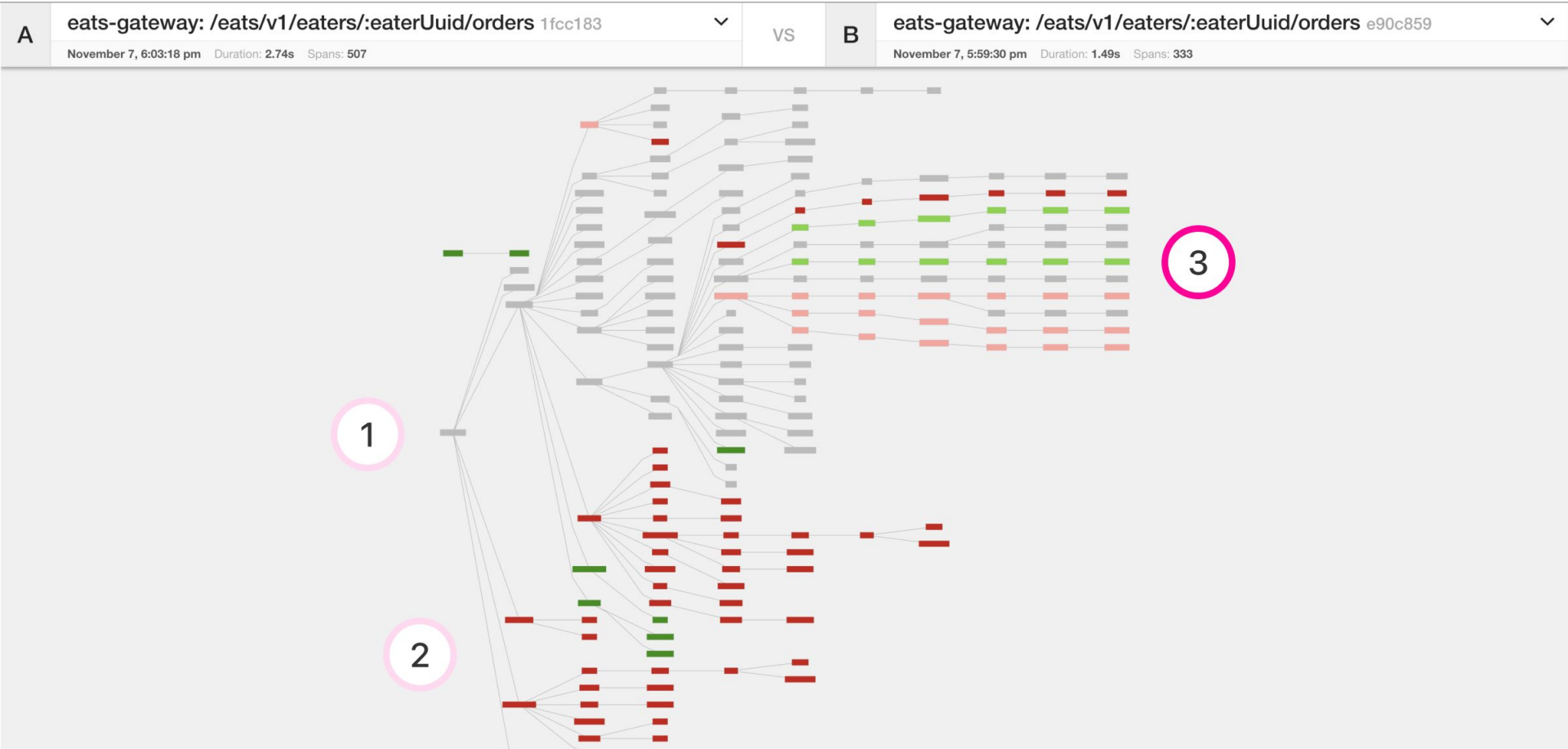
Comparing trace structures – Shared structure



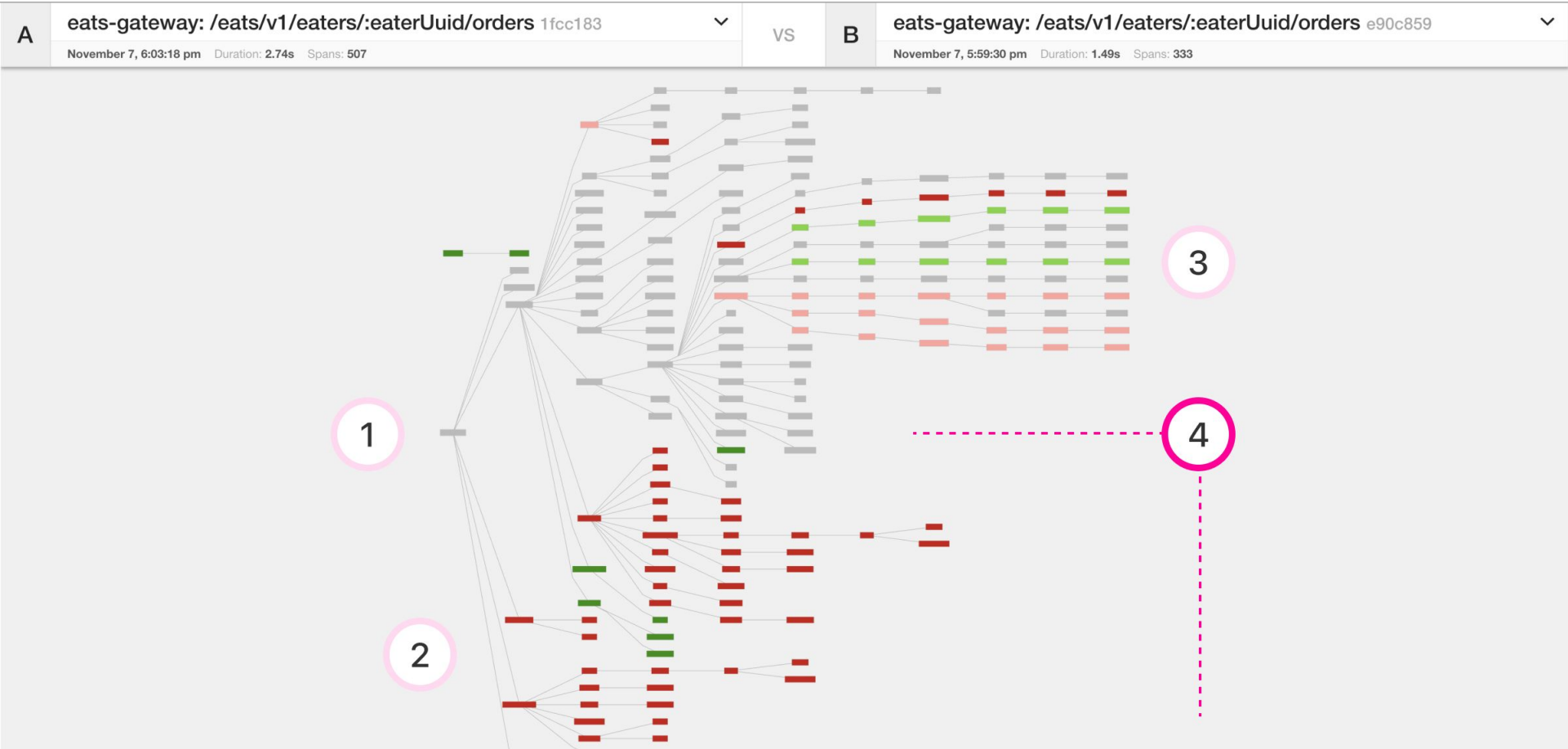
Comparing trace structures – Absent in one or the traces



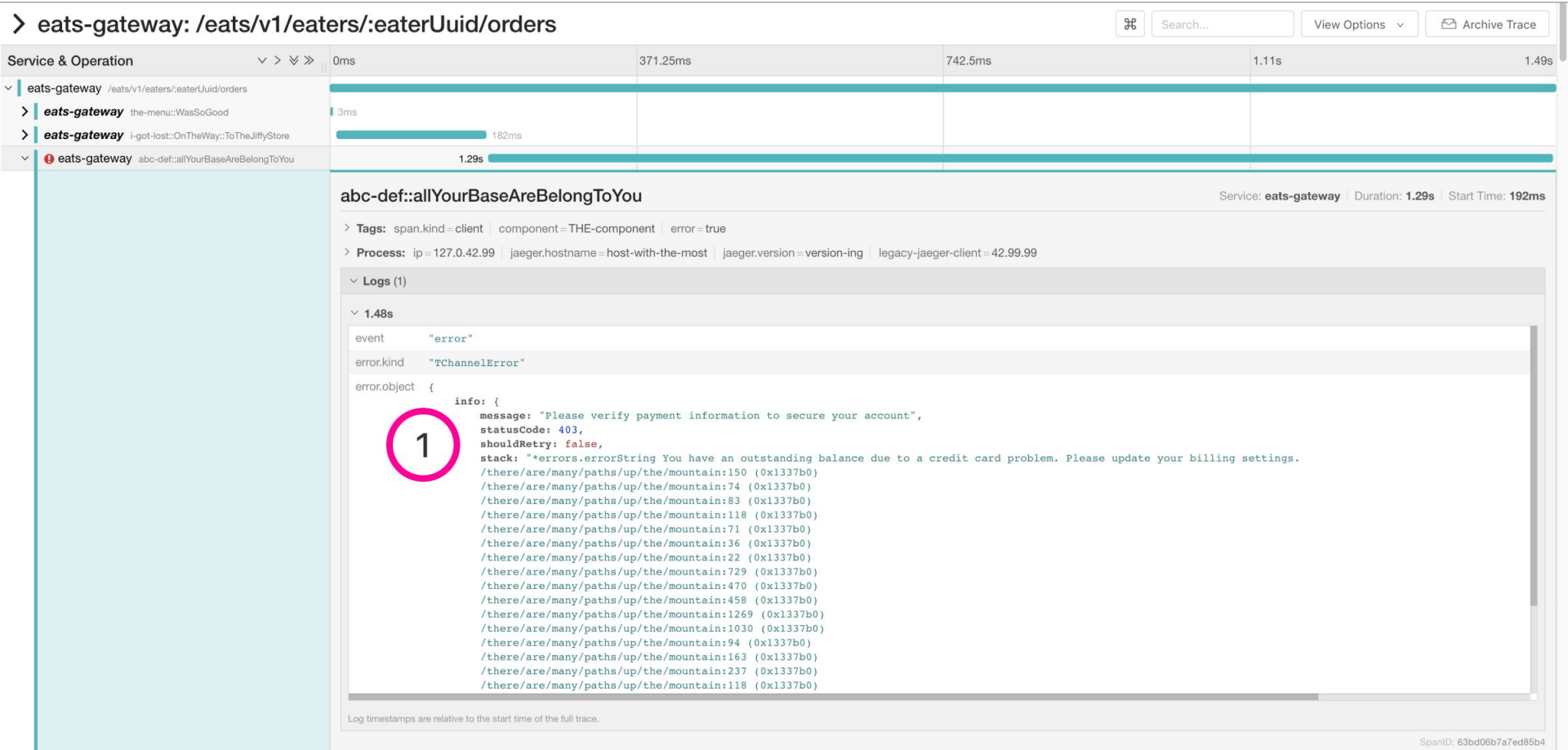
Comparing trace structures – More or less within a node



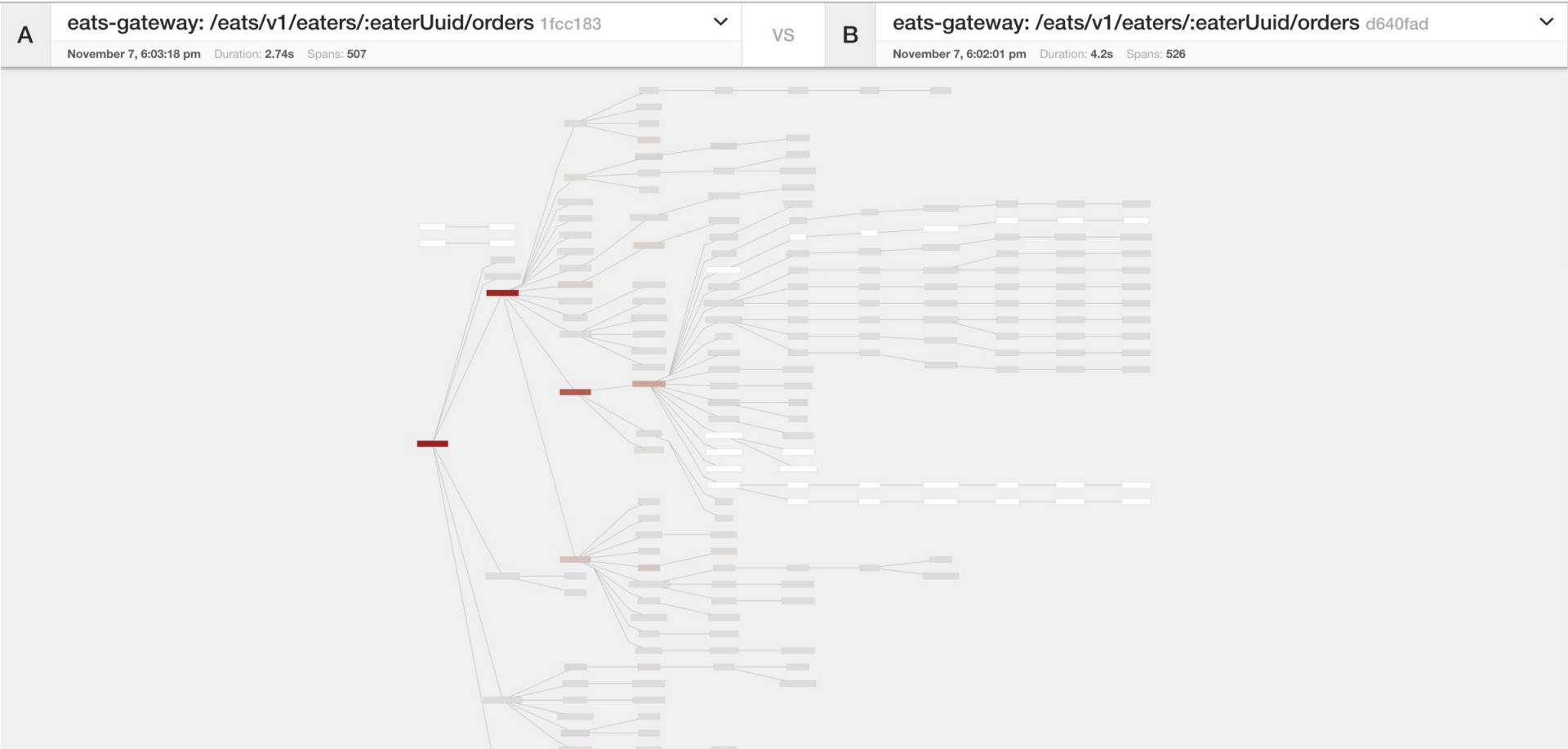
Comparing trace structures – Substantial divergence



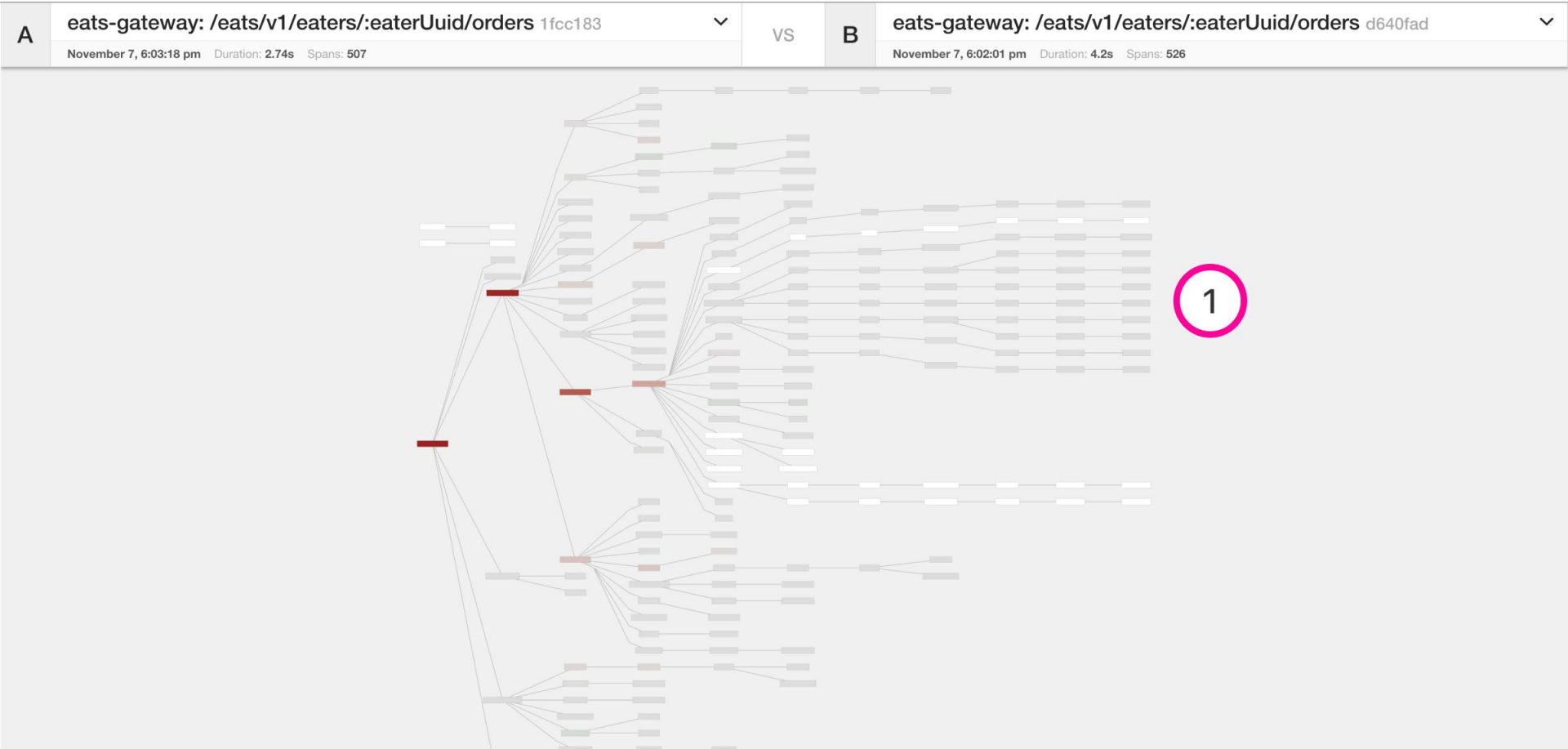
"You have an outstanding balance…"



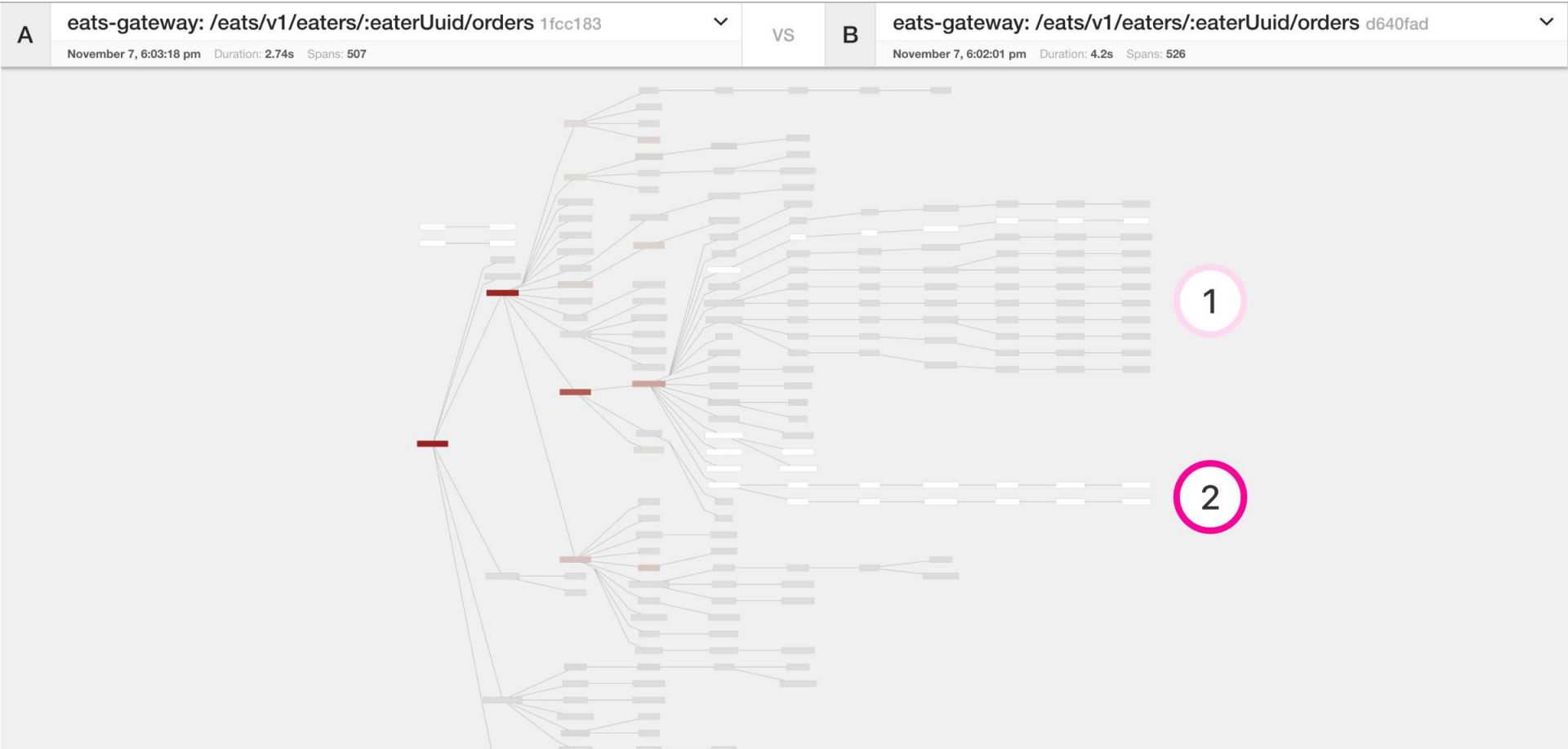
Comparing span durations



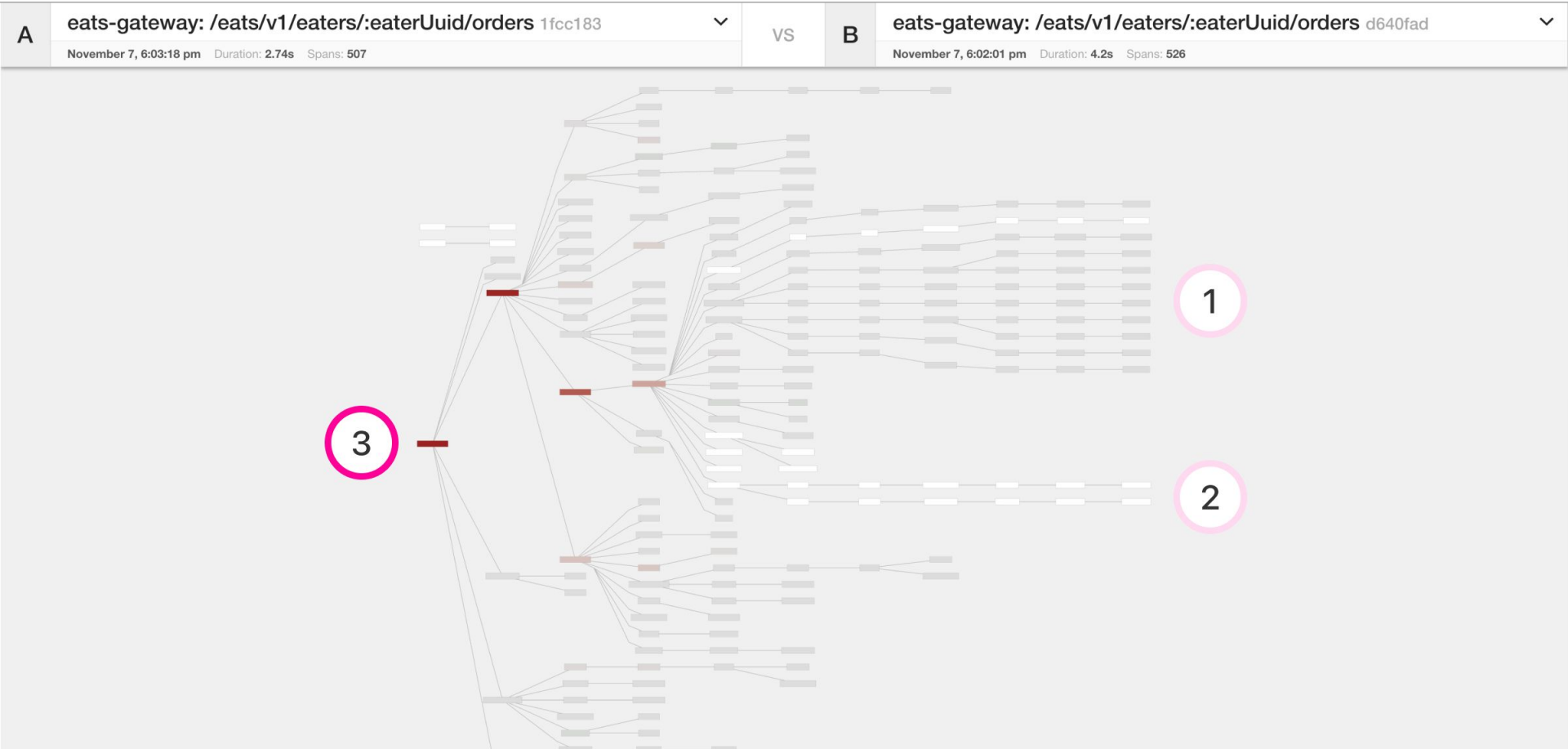
Comparing span durations – Similar durations



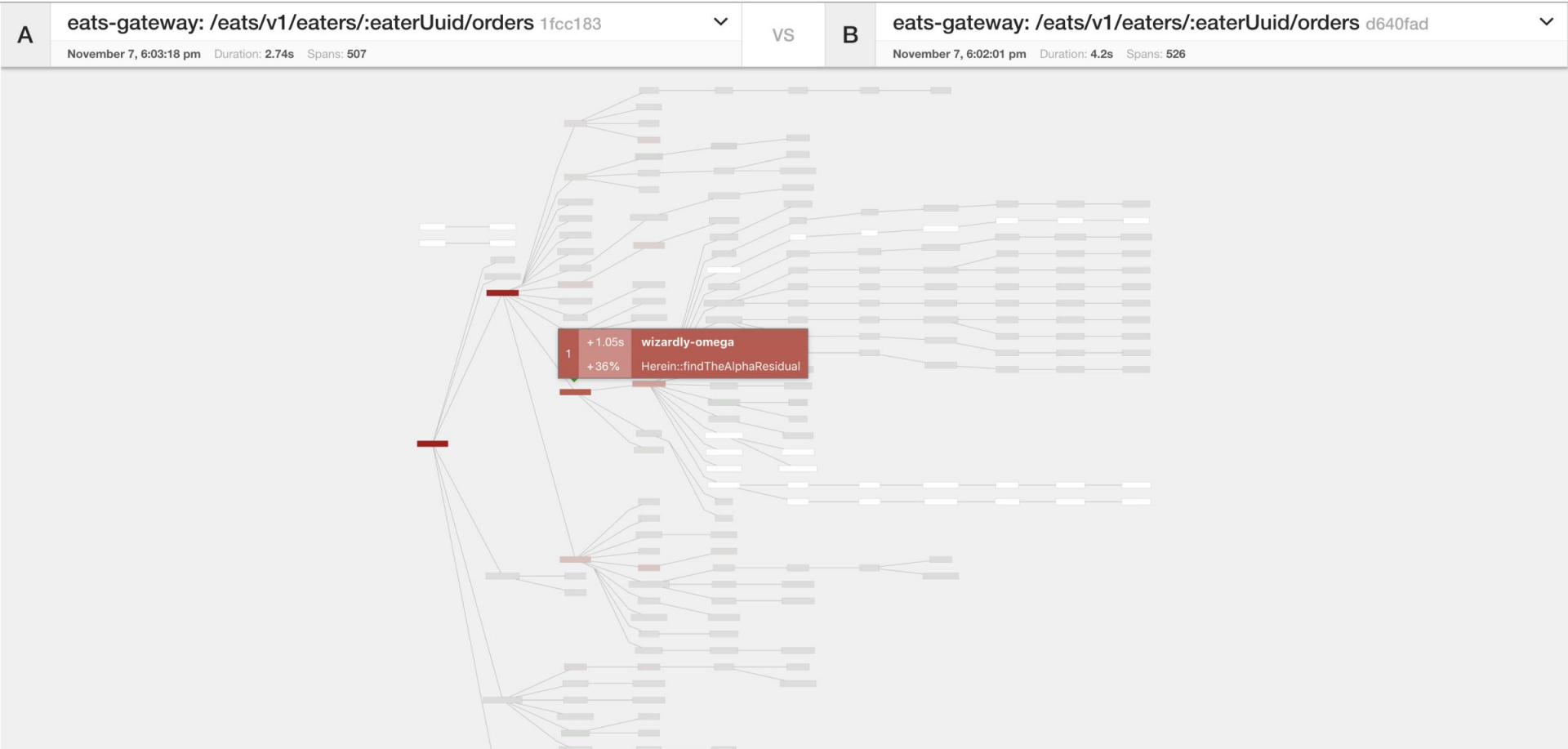
Comparing span durations – Nodes that aren't shared



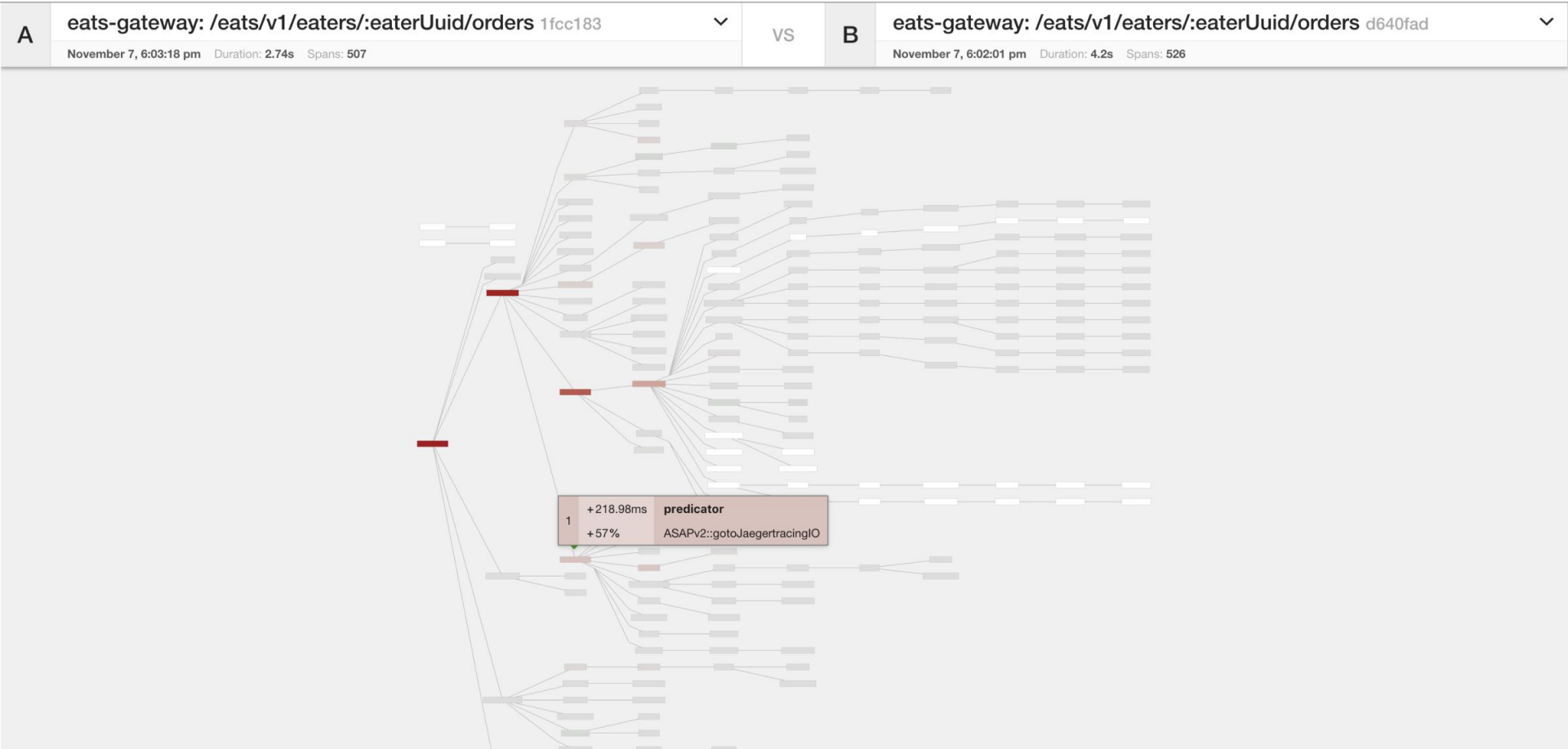
Comparing span durations – Follow the slower nodes



Comparing span durations



Comparing span durations



Graph Visualizations

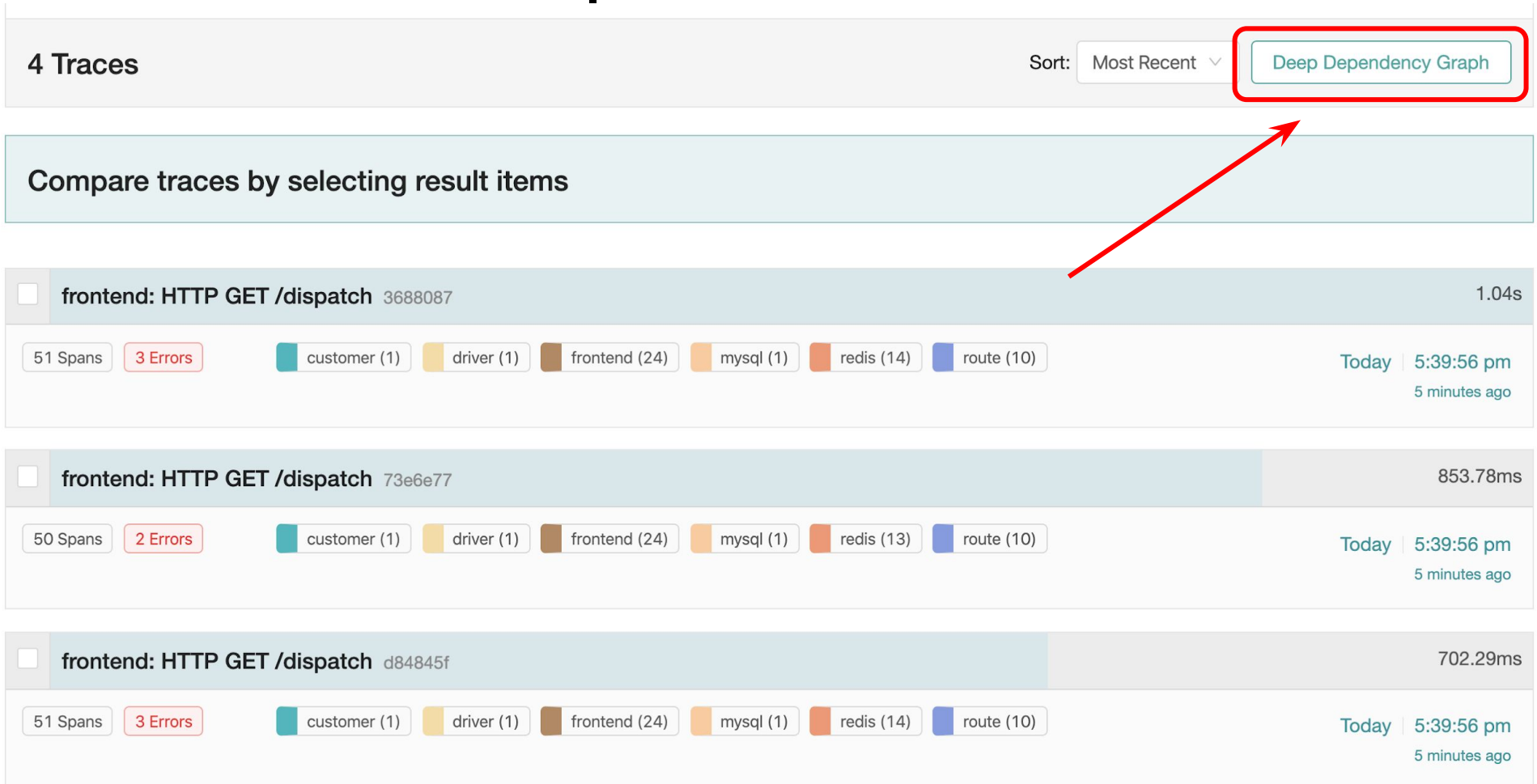
Gantt chart is not great for traces with many 100s of spans

- Trace Diffs
  - Compare two traces
  - Compare one trace against a group of traces (coming soon)
- Trace Graph
  - Call graph visualization with mini-aggregations
  - Showing paths rather than individual RPCs

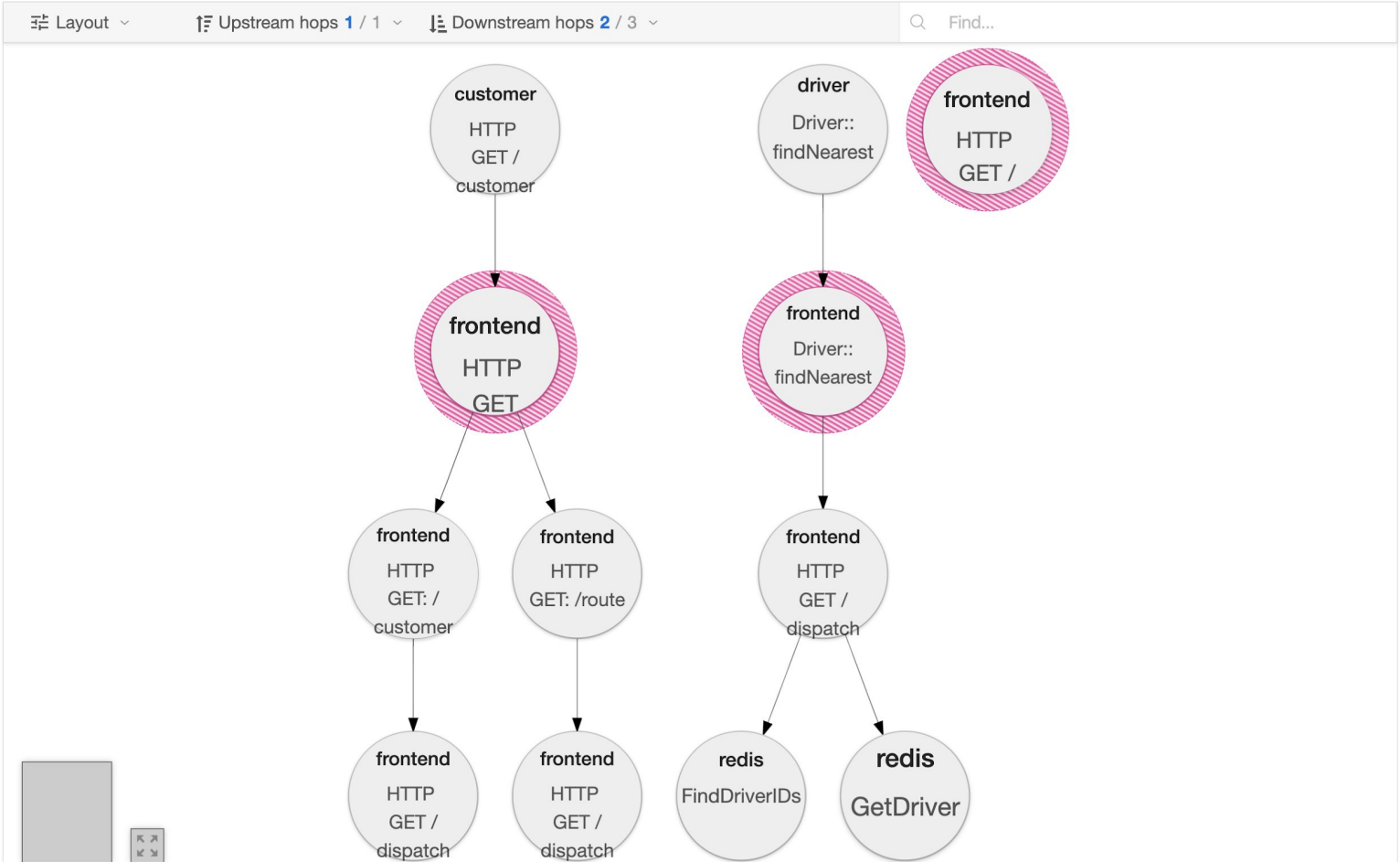
Graph Visualizations

- Surface less information
- Condense the structural representation
- Emphasize the differences
- Distinct comparison modes simplify the comparisons

Transitive Service Graphs



Transitive Service Graphs



Distributed Tracing Systems

distributed transaction monitoring

performance and latency optimization

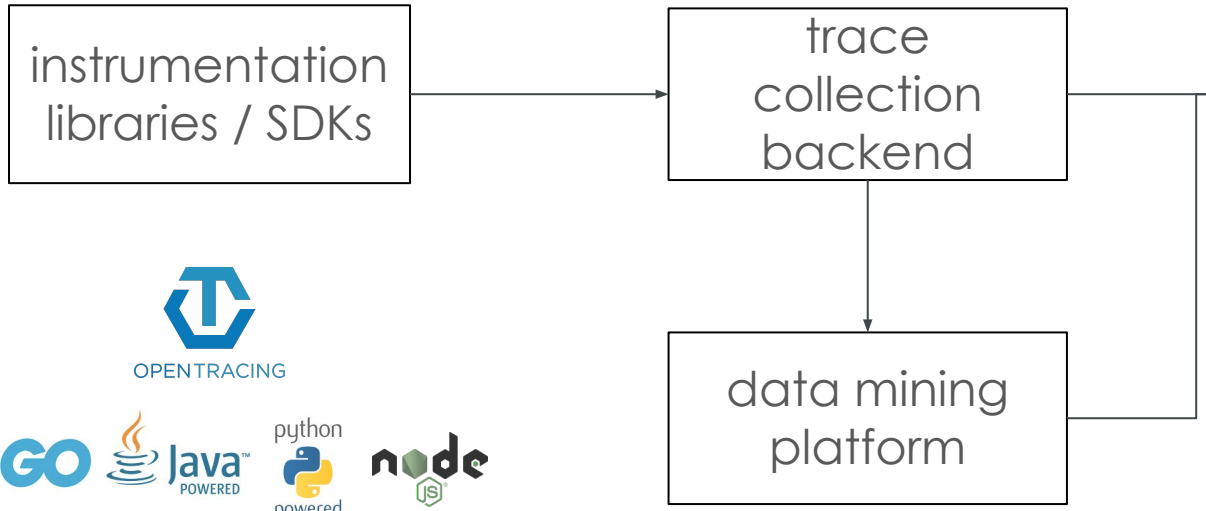
root cause analysis

service dependency analysis

distributed context propagation

Jaeger

Jaeger, a Distributed Tracing Platform



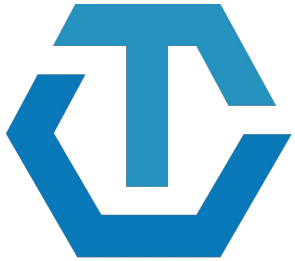
Jaeger - /ˈyāgər/, *noun*: hunter

- Inspired by Google's Dapper and OpenZipkin
- Created at Uber in August 2015
- Open sourced in April 2017
- Joined CNCF in Sep 2017 (incubating)
- Graduated to top-level CNCF project

  Oct 31, 2019 (CNCF announcement)

OpenTracing

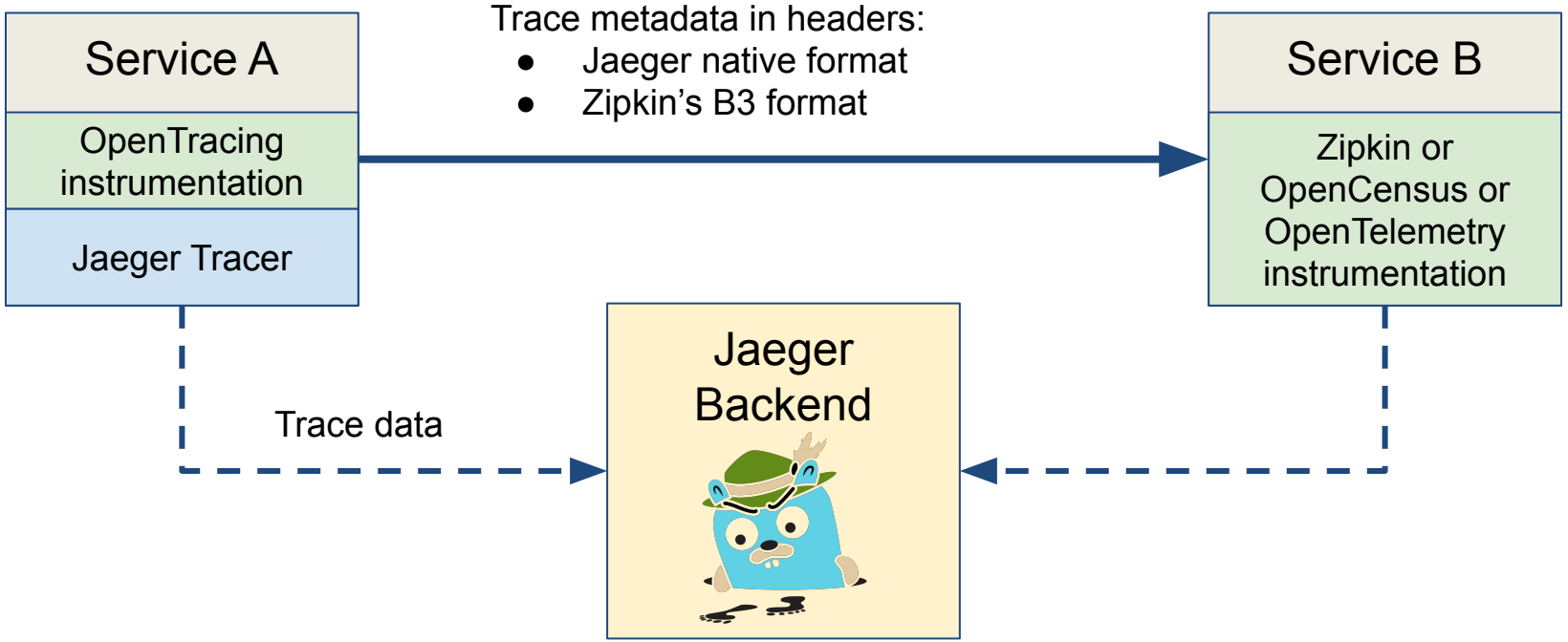
- **Instrumentation API**
  - Context propagation
  - Distributed tracing
  - Contextualized logging
  - Contextualized metrics
- Vendor neutral
- Cross language
- CNCF top-level project



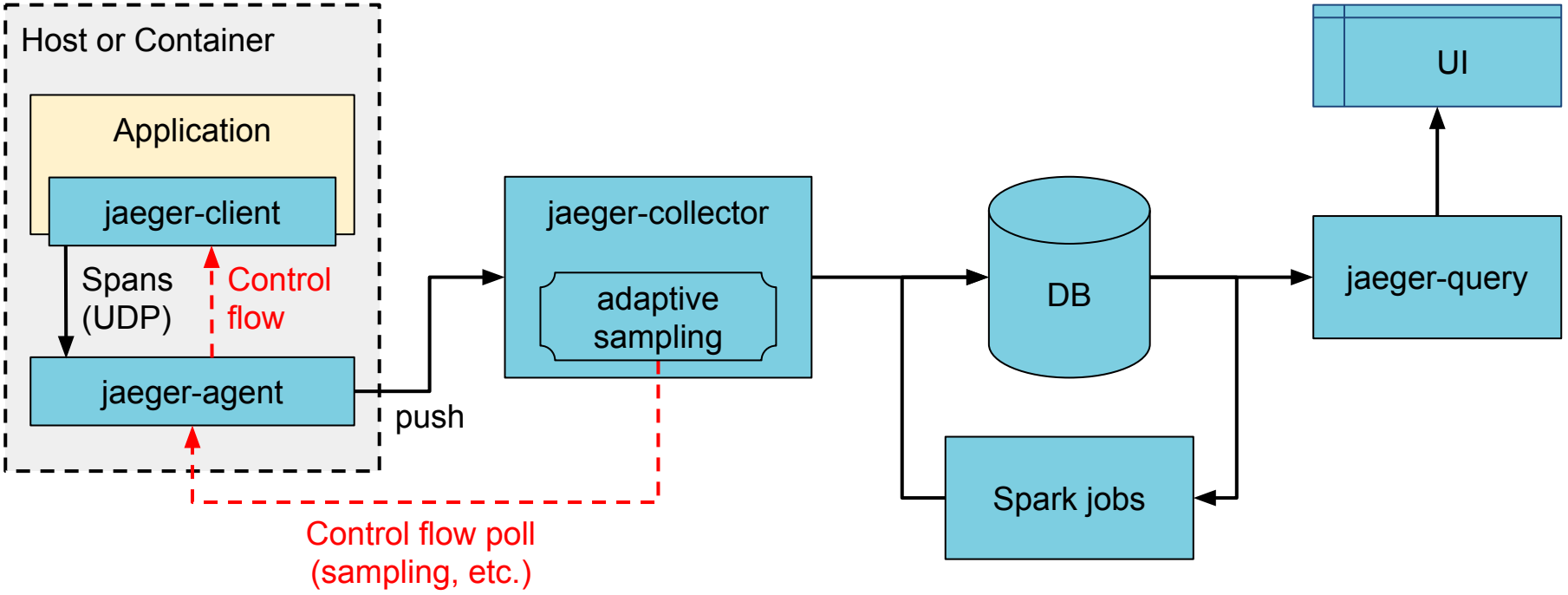
OPENTRACING

http://opentracing.io

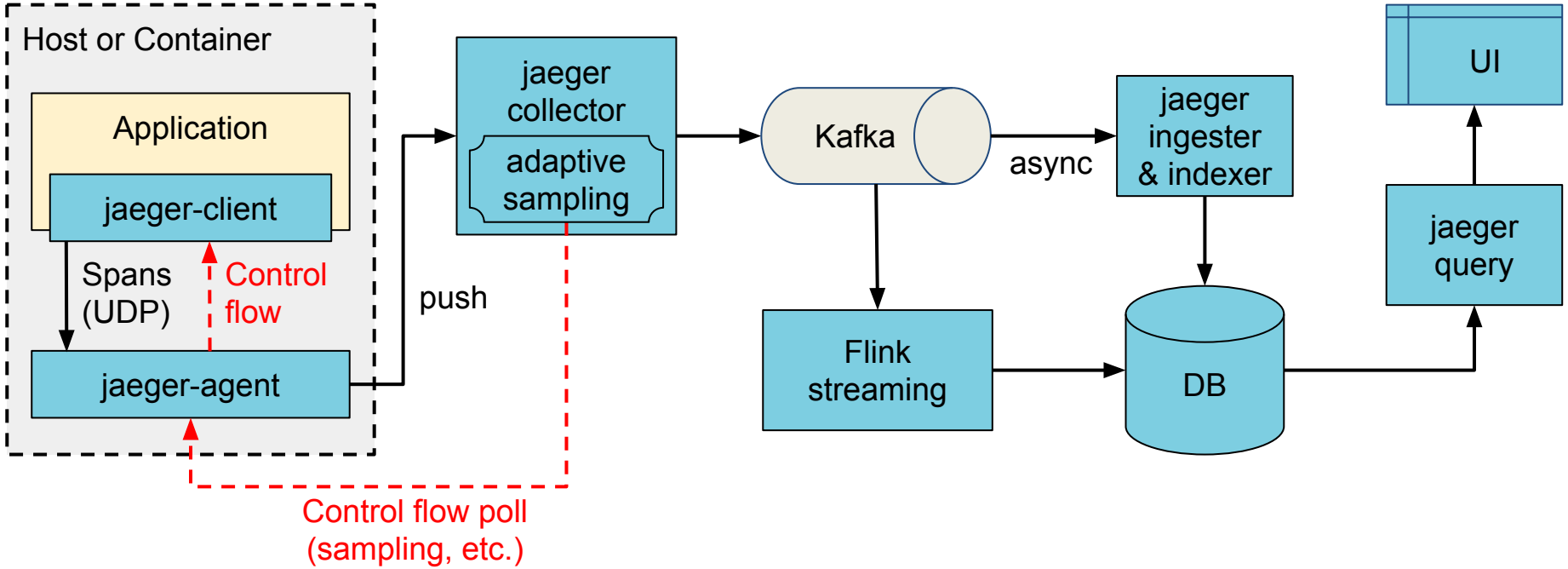
Jaeger Architecture



Architecture 2017: Push



Architecture now: Push+Async+Streaming



Technology Stack

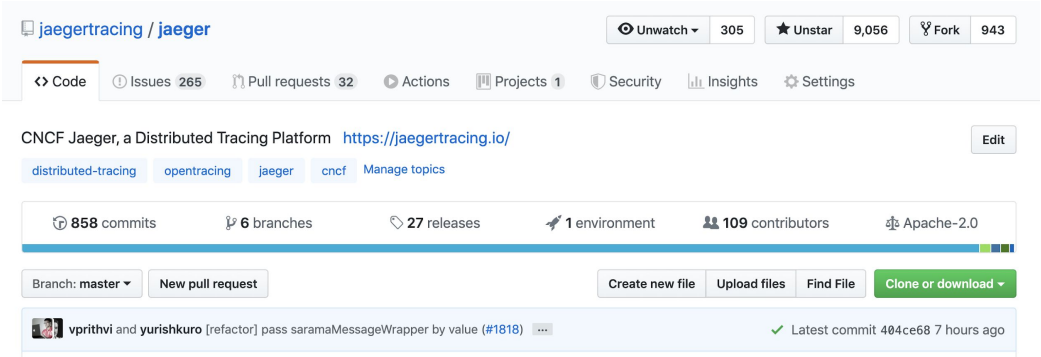
- Go backend
- Pluggable storage
  - Cassandra, Elasticsearch, badger, memory
- React/Javascript frontend
- OpenTracing Instrumentation libraries
- Integration with Kafka, Apache Flink

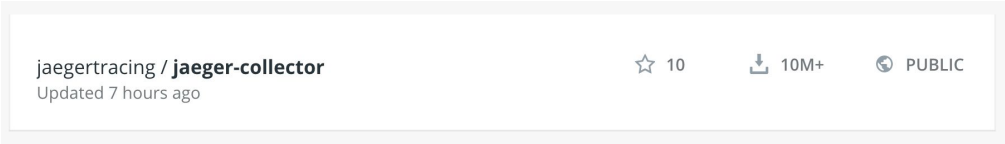
OPENTRACING

Go

Project & Community

- **9,000+ GH stars**
- **1200+ contributors**
- **375 authors of commits and pull requests**
- **15 maintainers** across all components from 5+ companies (backend: 7 and 3 respectively)
- **815** Gitter channel members
- **2,800+** Twitter followers
- **15 releases** since incubation
- **10M+** Docker pulls





Jaeger 1.15

New Features

New Features

- Kubernetes Operator
- Badger storage
- Storage plugins: Couchbase, InfluxDB
- Visual trace comparisons
- Security improvements
  - TLS with gRPC, Kafka, Elasticsearch

Documentation Website

- Releases & Downloads
- Architecture
- Deployment
- Command line options
- Client features

Integrations

- Jaeger Operator for Kubernetes
  - https://github.com/jaegertracing/jaeger-operator
- OpenTelemetry libraries and collector ship with exporters for Jaeger
  - https://opencensus.io/guides/exporters/supported-exporters/java/jaeger/
- Istio comes with Jaeger included
  - https://istio.io/docs/tasks/telemetry/distributed-tracing/
- Envoy works with Jaeger native C++ client
  - https://www.envoyproxy.io/docs/envoy/latest/start/sandboxes/jaeger_native_tracing
- Eclipse Trace Compass incubator supports importing Jaeger traces
  - https://github.com/tuxology/tracevizlab/tree/master/labs/303-jaeger-opentracing-traces

Asynchronous span ingestion

- Push model was struggling to keep up with traffic spikes
  - Because of sync storage writes
  - Collectors had to drop data randomly
- Kafka is much more elastic for writes
  - Just raw bytes, no schema, no indexing
  - A lot less overhead on the write path
- Data in Kafka allows for streaming data mining & aggregations
- Two new components: `jaeger-ingester` and `jaeger-indexer`

Protobuf & gRPC

- Internal data model generated from Protobuf IDL
- gRPC connection between `jaeger-agent` and `jaeger-collector`

Why

- gRPC plays better with modern routing than TChannel
- Path to official data model and collector/query APIs
- Protobuf-based JSON API
- Unblock development of storage plugins
- (Thrift still supported for backwards compatibility)

Zipkin Compatibility

- Clients
  - Zipkin `B3-***` headers for context propagation
  - Interop between Jaeger-instrumented and Zipkin-instrumented apps
- Collector
  - Zipkin Thrift, Protobuf, and JSON v2 span format
  - Use Zipkin instrumentation (e.g. Brave) to send traces to Jaeger
- Kafka

Roadmap

http://bit.do/jaeger-roadmap

Roadmap

- Trace DSL, jupyter notebooks and where we are heading
- Delayed & ad-hoc sampling
- Tail-based sampling
- OpenTelemetry

Adaptive Sampling

Problem

- APIs have endpoints with different QPS
- Service owners do not know the full impact of sampling probability

Adaptive Sampling is per service + endpoint,

decided by Jaeger backend based on traffic

Adaptive Sampling Status

- Jaeger clients support per service/endpoint sampling strategies
- Can be statically configured in collector
- Pull requests for dynamic recalculations

Data Pipeline

- Based on Kafka and Apache Flink
- Support aggregations and data mining
- Examples:
  - Pairwise service graph (dependencies diagram)
  - Path-based service graphs
  - Latency histograms

Getting in Touch

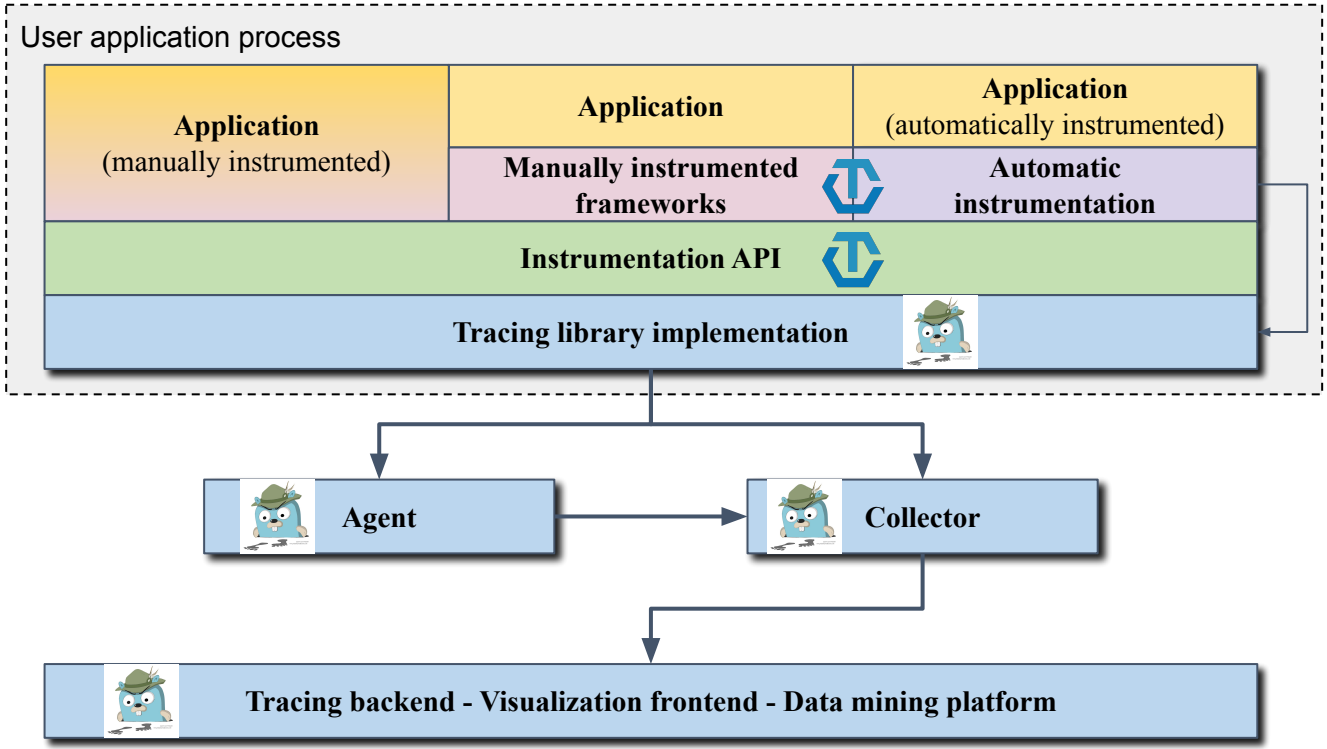
- GitHub: https://github.com/jaegertracing
- Chat: https://gitter.im/jaegertracing/
- Mailing List - jaeger-tracing@googlegroups.com
- Blog: https://medium.com/jaegertracing
- Twitter: https://twitter.com/JaegerTracing
- Bi-Weekly Community Meetings

Q&A

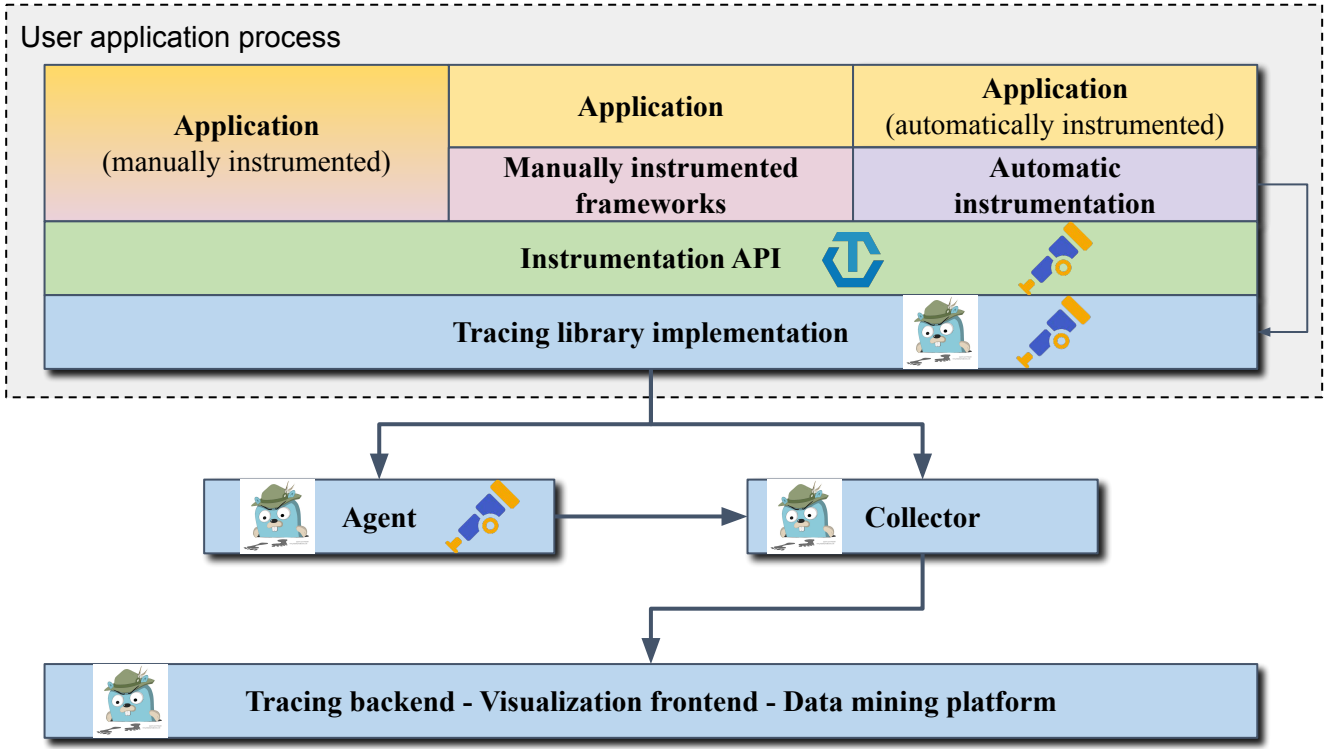
- Jaeger Deep Dive - Wed, November 20, 2:25pm

https://jaegertracing.io

Jaeger vs. OpenTracing



Jaeger vs. OpenTracing, OpenCensus, OpenTelemetry



OpenTelemetry

contrib

API

SDK

Agent &
Collector

Learn More

Website: jaegertracing.io/

Blog: medium.com/jaegertracing