CLOUD NATIVE
COMPUTING FOUNDATION

Jaeger
Deep Dive

Steve Flanders (Omnition)

CloudNativeCon China, Shanghai, 2019

Agenda

- About the Project
- New Features
- Demo
- Roadmap
- Q & A

About

Steve Flanders

Head of Product and Experience at Omnition

@smflanders | https://sflanders.net

What is Tracing & Why?

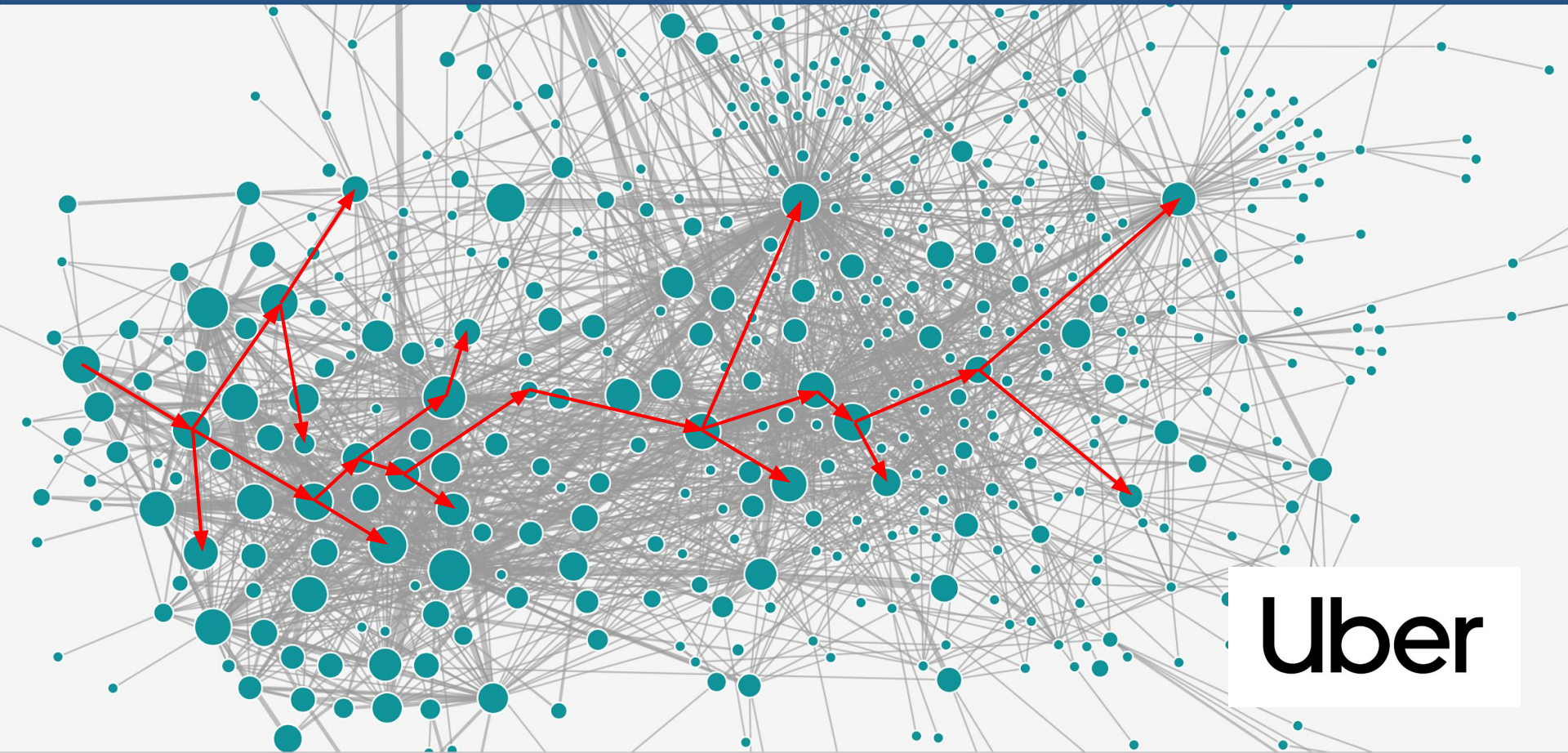
Concepts and terminology

Modern Distributed Systems are COMPLEX

Loading Netflix or Facebook home page ⇒

dozens of microservices, 100s of nodes

BILLIONS of times a day!

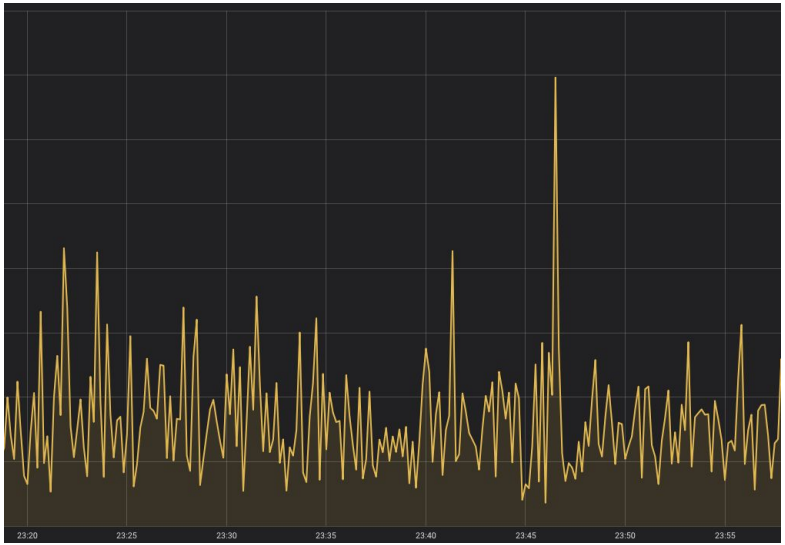


How can we tell what is going on?

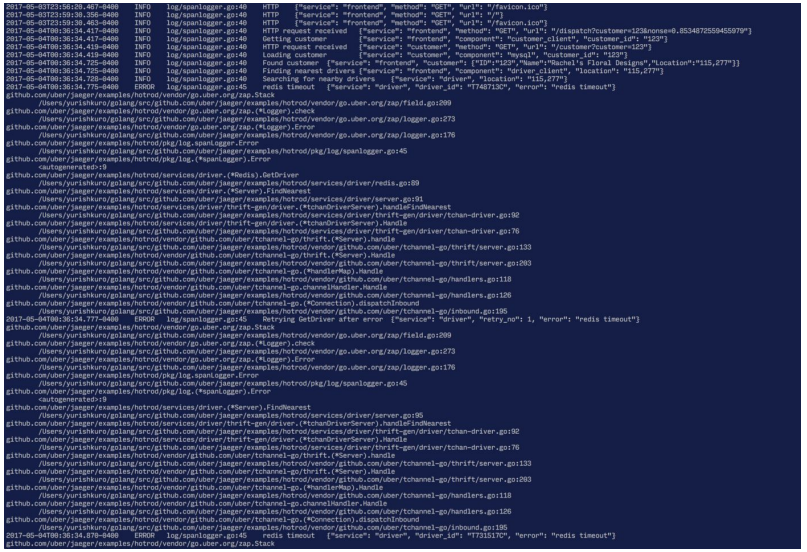
Which service is to blame

when things go wrong or become slow?

Traditional monitoring tools don't help



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



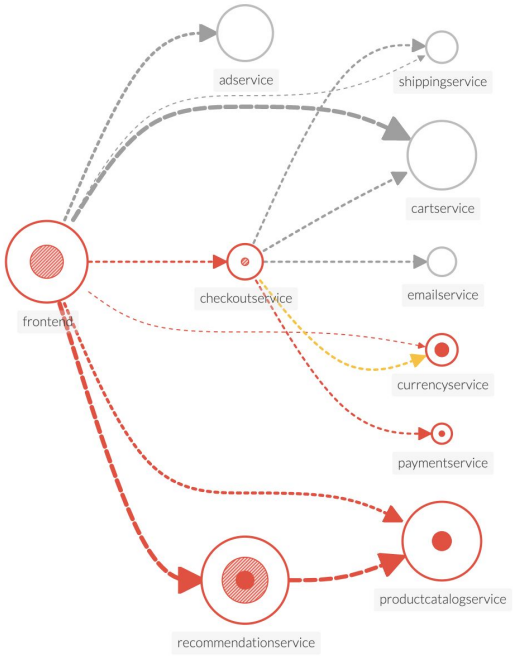
Logs are a mess: concurrent requests, multiple hosts, impossible to correlate.

Monitoring tools must tell stories!

Do you like debugging without a stack trace?

We need to monitor distributed transactions

⇒ **distributed tracing**!



Jaeger - /ˈyāgər/, *noun*: hunter

- Inspired by Google's Dapper and OpenZipkin
- Started at Uber in August 2015
- Open sourced in April 2017
- Joined CNCF in Sep 2017 (incubating)
- Applying for graduation

  https://github.com/cncf/toc/pull/171

Project & Community

- 7 maintainers, from Uber and Red Hat
- GitHub stats
  - >8,200 stars, >810 forks
  - >580 contributors
    - >220 authors of commits and pull requests
    - >350 issue creators

Jaeger, a Distributed Tracing Platform

trace collection backend

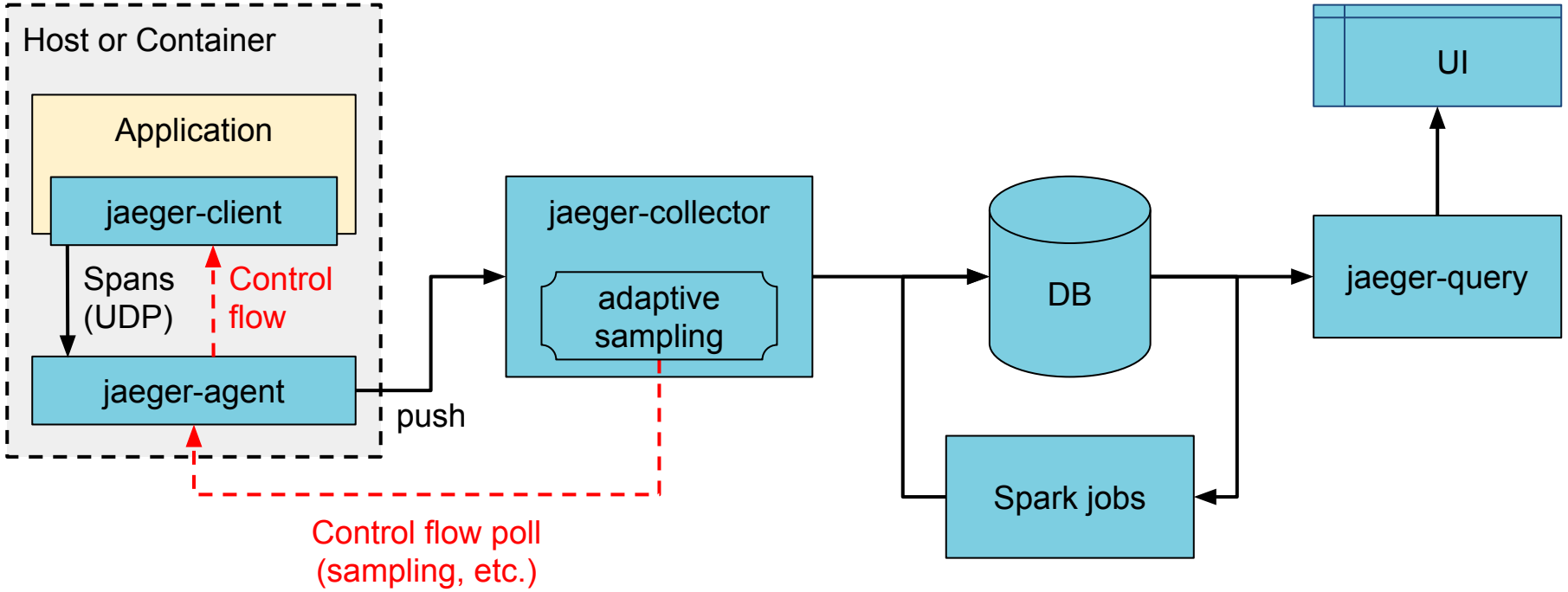
visualization frontend

instrumentation libraries

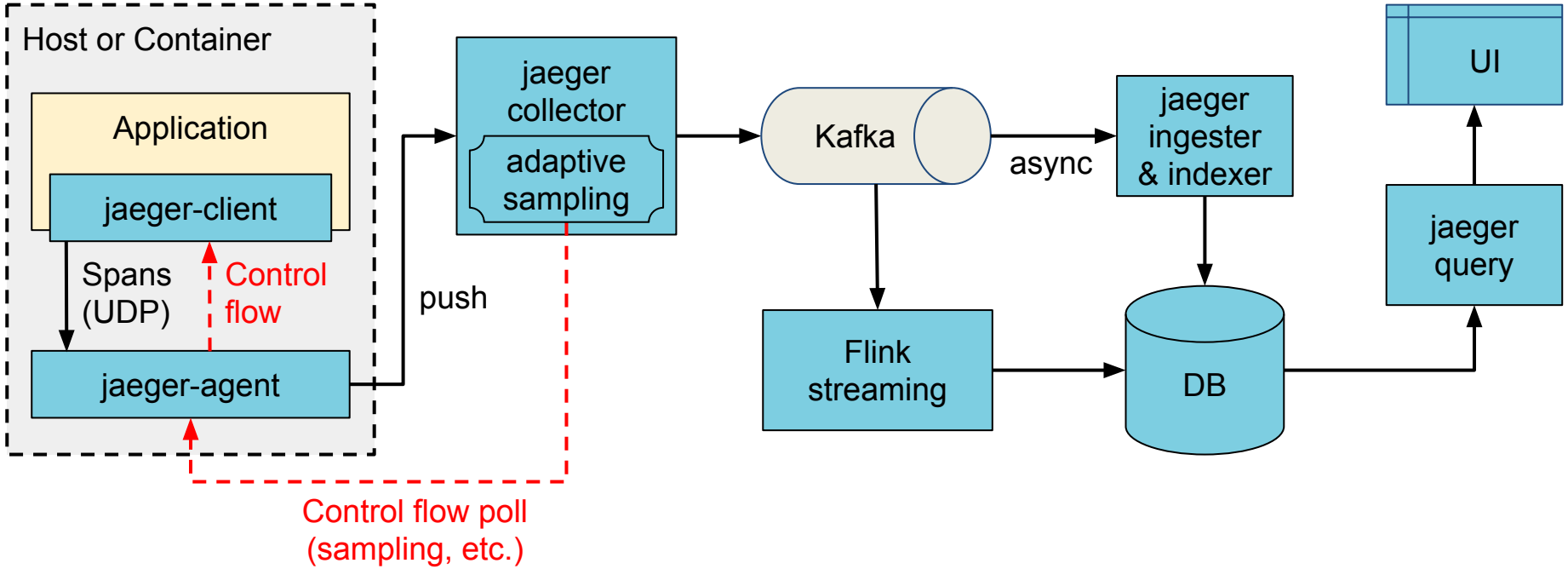
data mining platform

https://jaegertracing.io

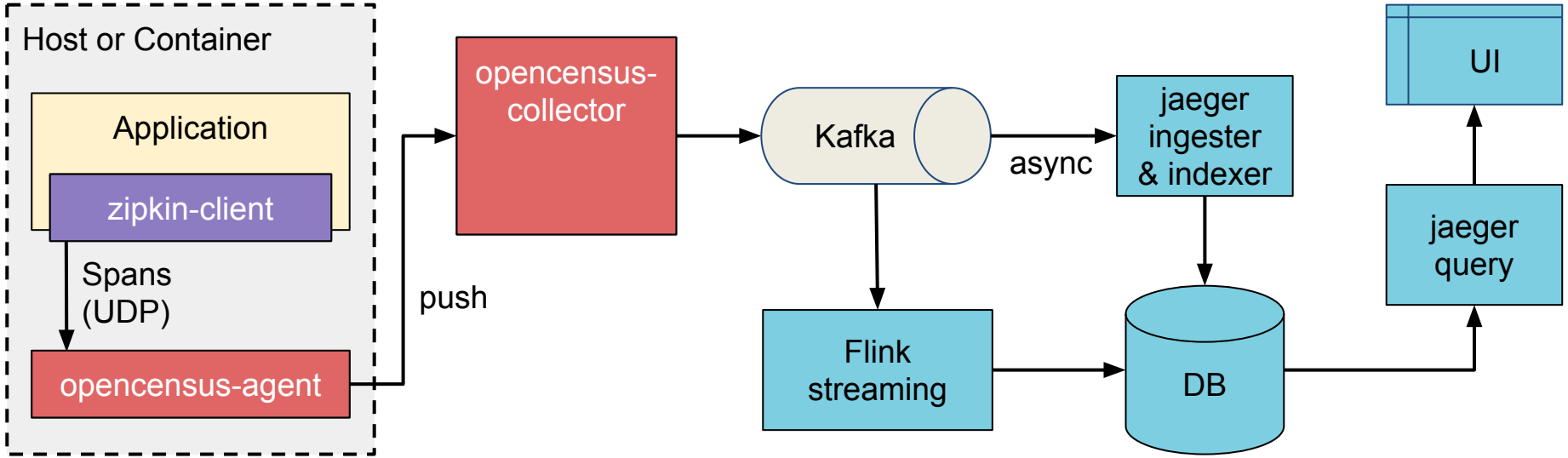
Jaeger Architecture (v1)



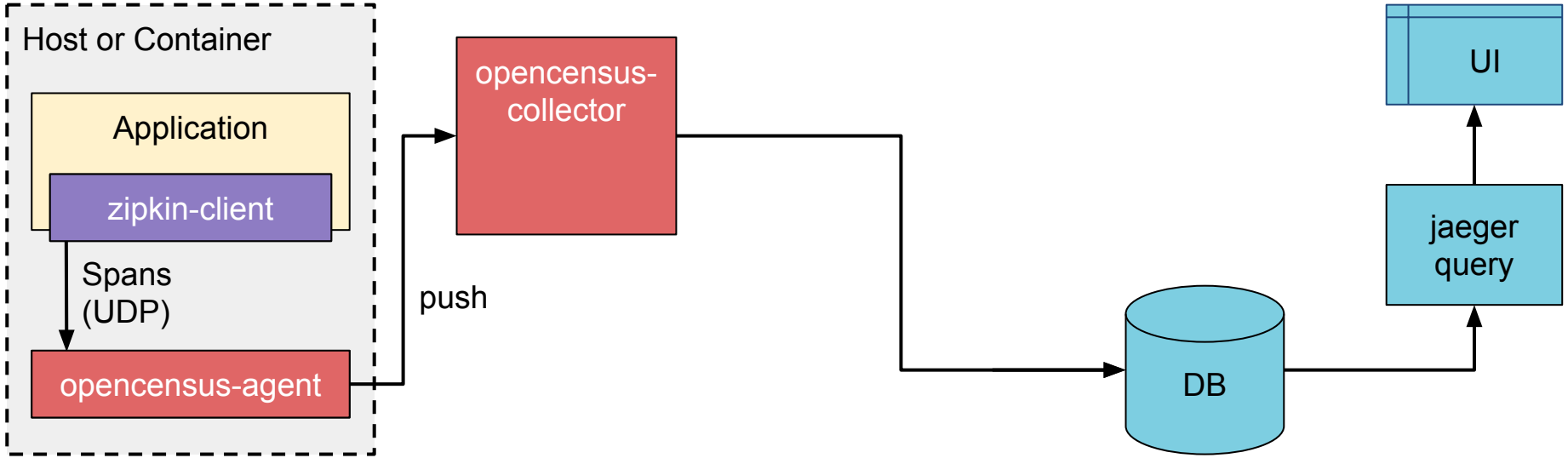
Jaeger Architecture (v2)



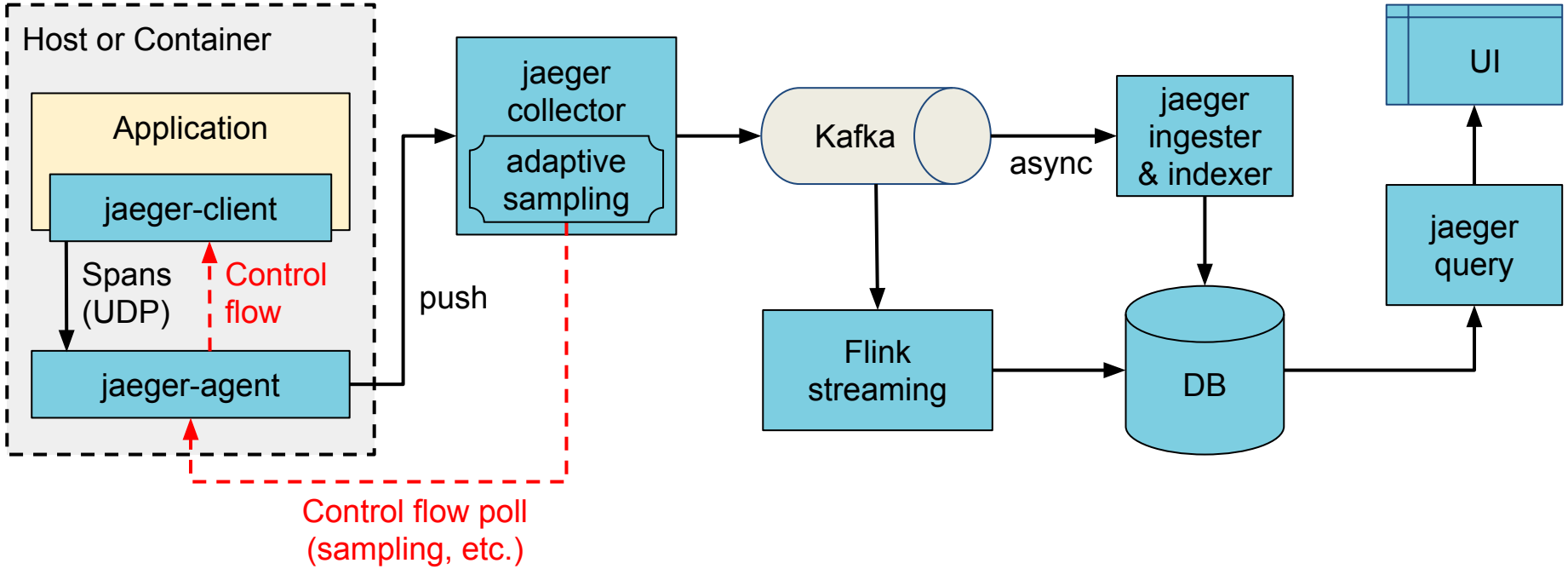
Jaeger Architecture (v2)



Jaeger Architecture (v2)



Jaeger Architecture (v2)



Jaeger <3 Open Standards

OPENTRACING + OpenCensus = OpenTelemetry

https://medium.com/jaegertracing/jaeger-and-opentelemetry-1846f701d9f2

Technology Stack

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

Integrations

Integrations

- Jaeger Operator for Kubernetes
  - https://github.com/jaegertracing/jaeger-operator
- OpenCensus libraries and agent ship with receivers/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

Jaeger 1.10 - 1.12

New Features

New Features

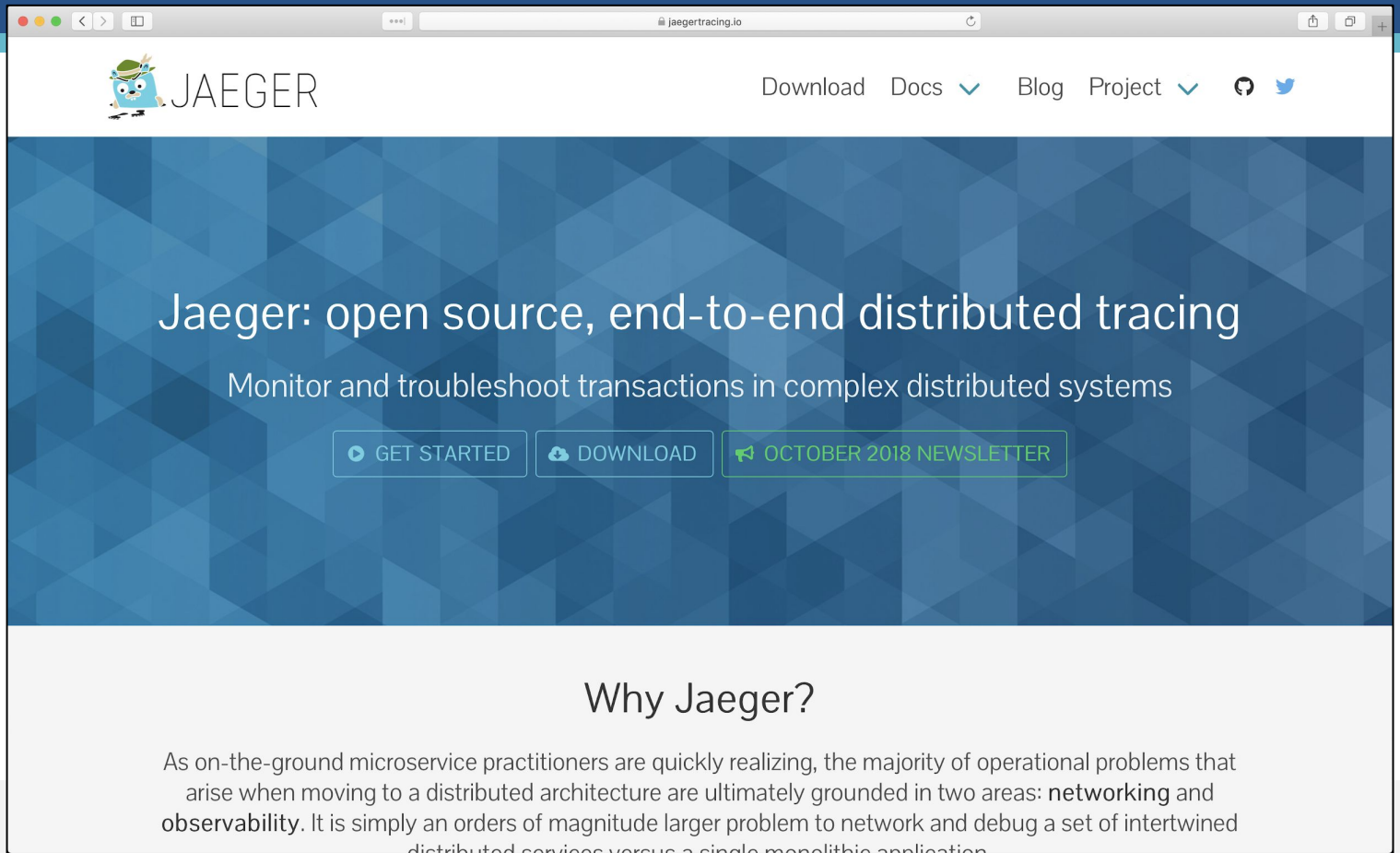
- Elasticsearch improvements (security, FindTraceIDs, archiving)
- Better gRPC support (security, retry, load balancing, external resolvers)
- Better Zipkin compatibility
- UI improvements (trace detail, find, query capabilities)

https://github.com/jaegertracing/jaeger/releases

https://github.com/jaegertracing/jaeger/blob/master/CHANGELOG.md

https://github.com/jaegertracing/jaeger-ui/blob/master/CHANGELOG.md

Website (easy to contribute)



Demo!

Demo Setup

$ > docker run -d --name jaeger -p 5775:5775/udp -p 5778:5778
-p 6831-6832:6831-6832/udp -p 16686:16686 -p 14268:14268 -p 9411:9411
-e COLLECTOR_ZIPKIN_HTTP_PORT=9411 jaegertracing/all-in-one:1.12

http://localhost:16686/search

$ > docker run --rm -it --link jaeger -p 8080-8083:8080-8083
-e JAEGER_AGENT_HOST="jaeger" jaegertracing/example-hotrod:1.12 all

http://localhost:8080/

Graph Visualizations

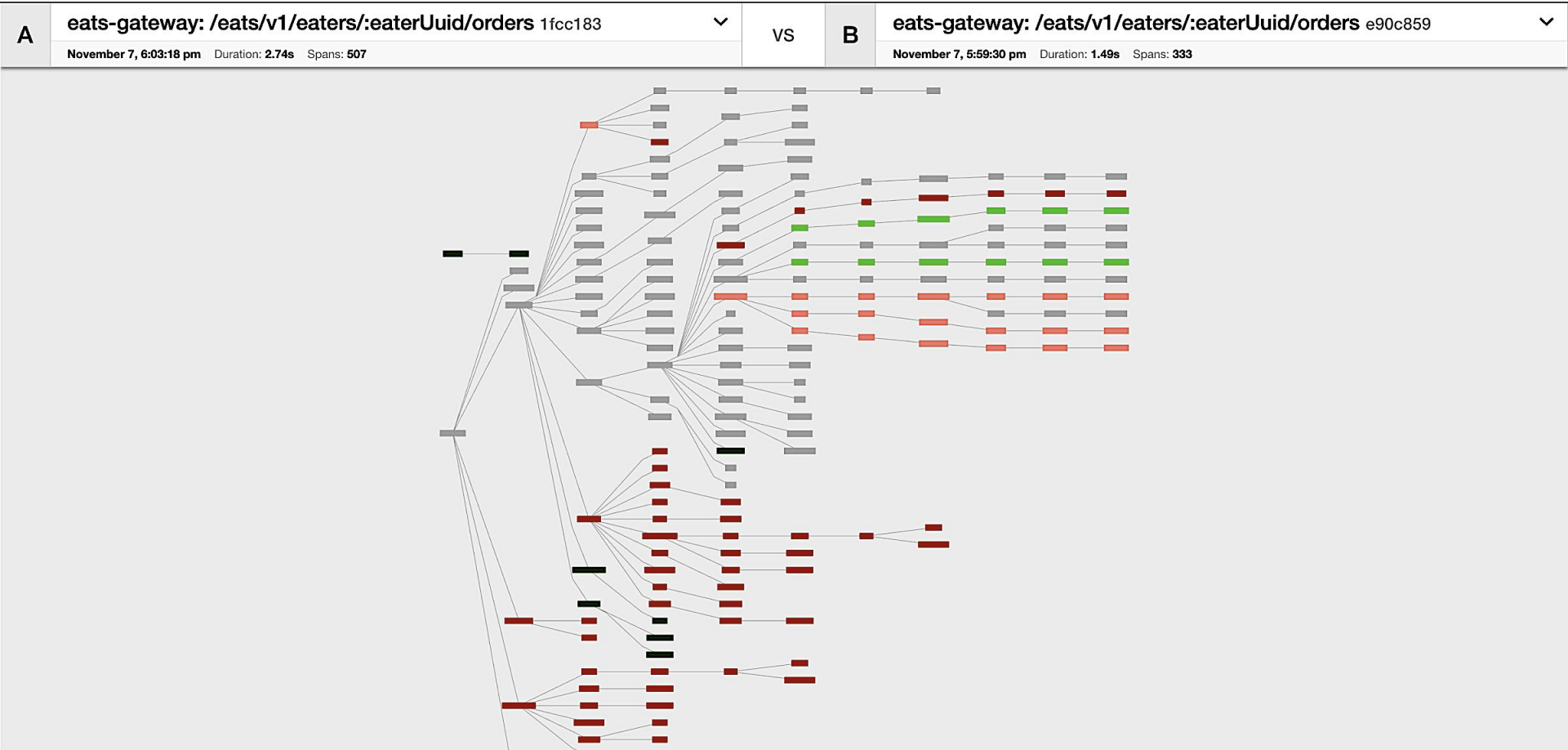
Trade Diffs and Trace Graph

Graph Visualizations

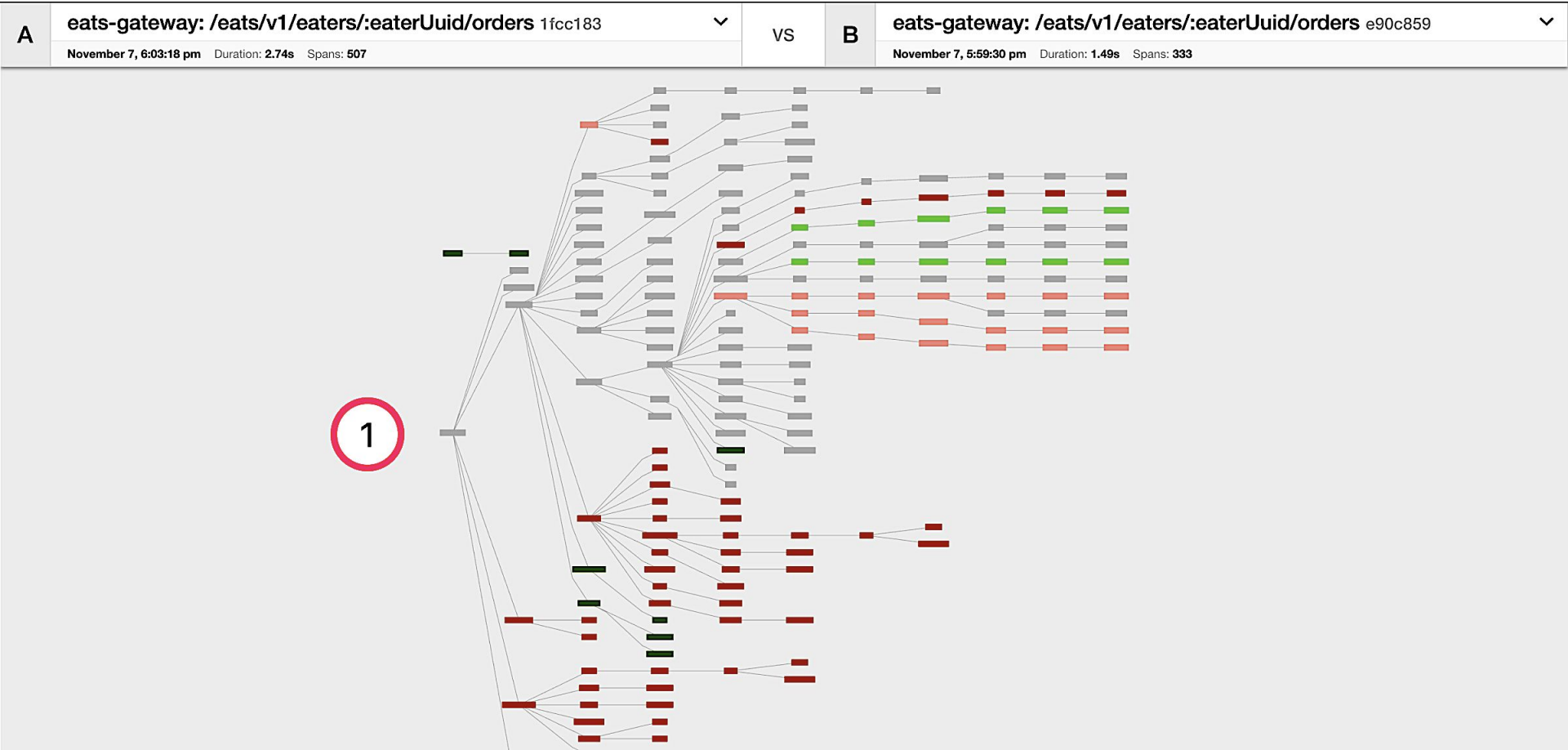
Gantt chart is not great for traces with 10s of thousands of spans

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

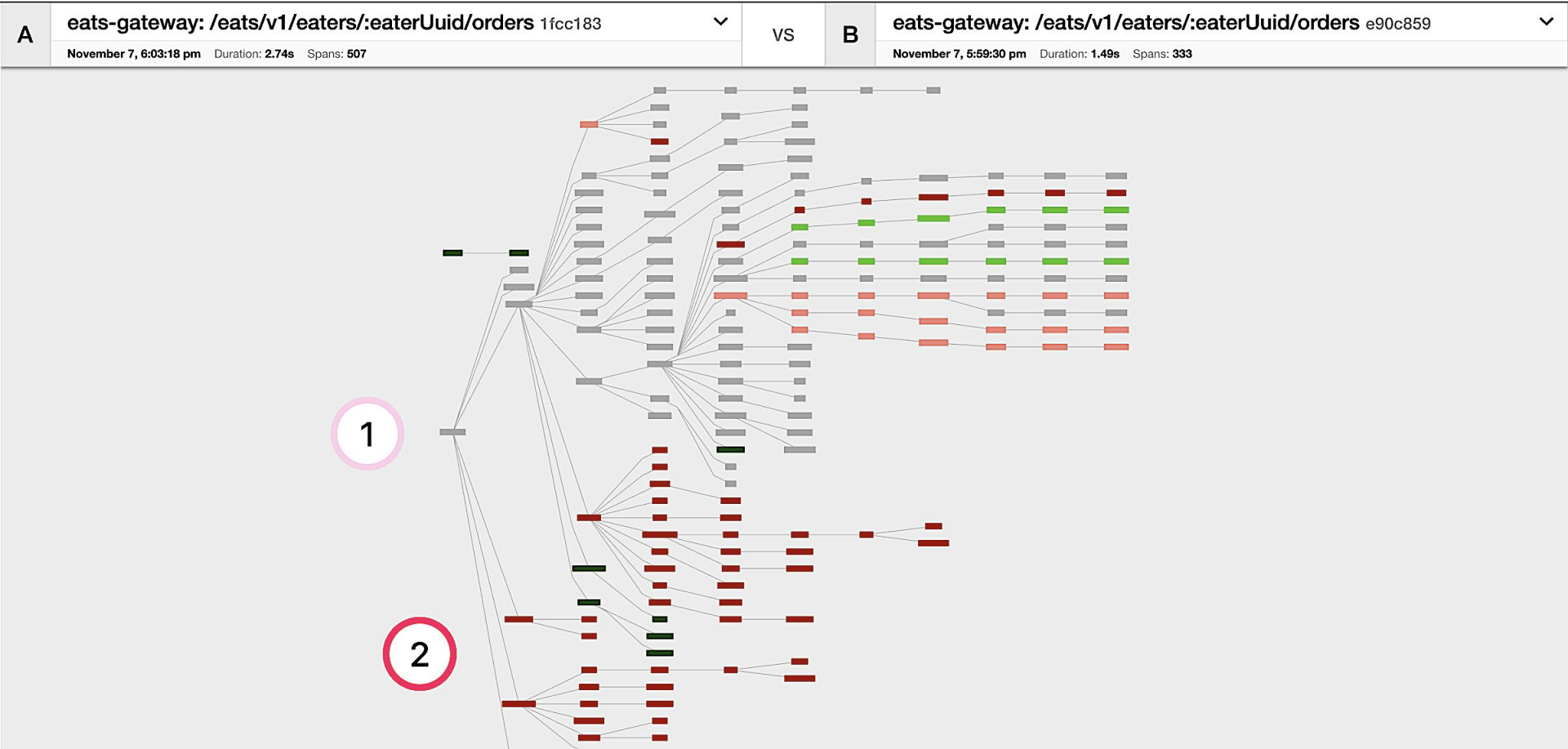
Comparing trace structures – Unified diff



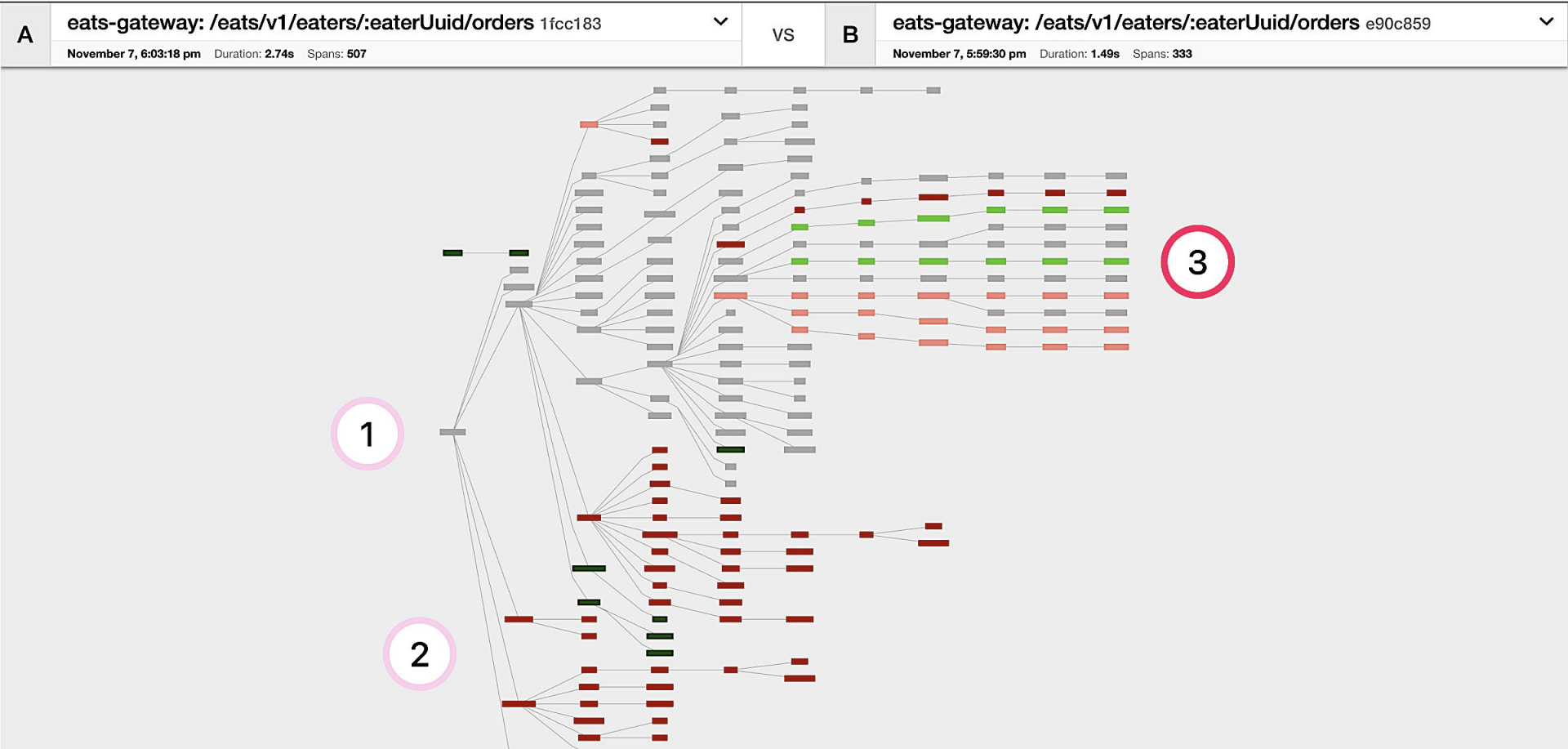
Comparing trace structures – Shared structure



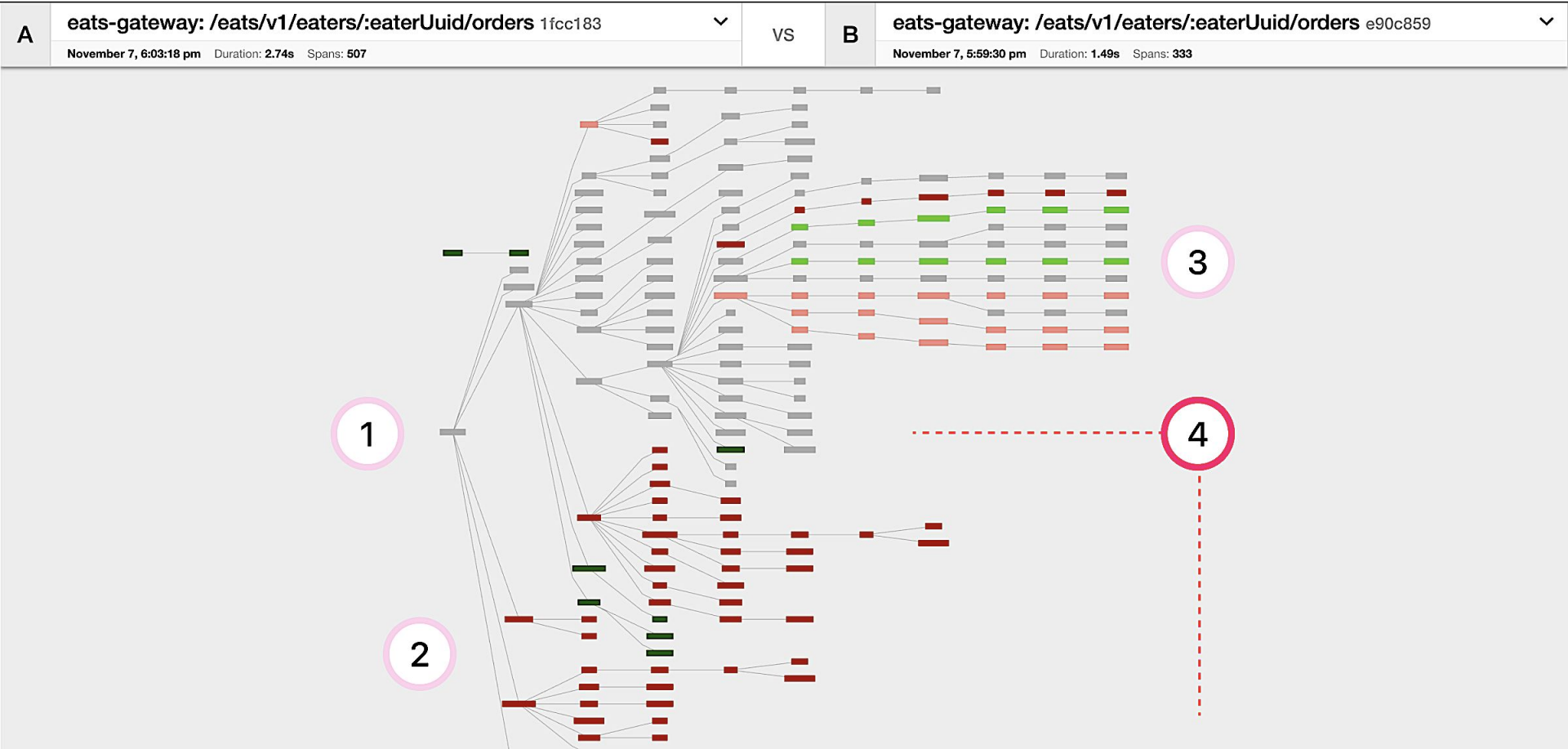
Comparing trace structures – Absent in one of the traces



Comparing trace structures – More or less within a node

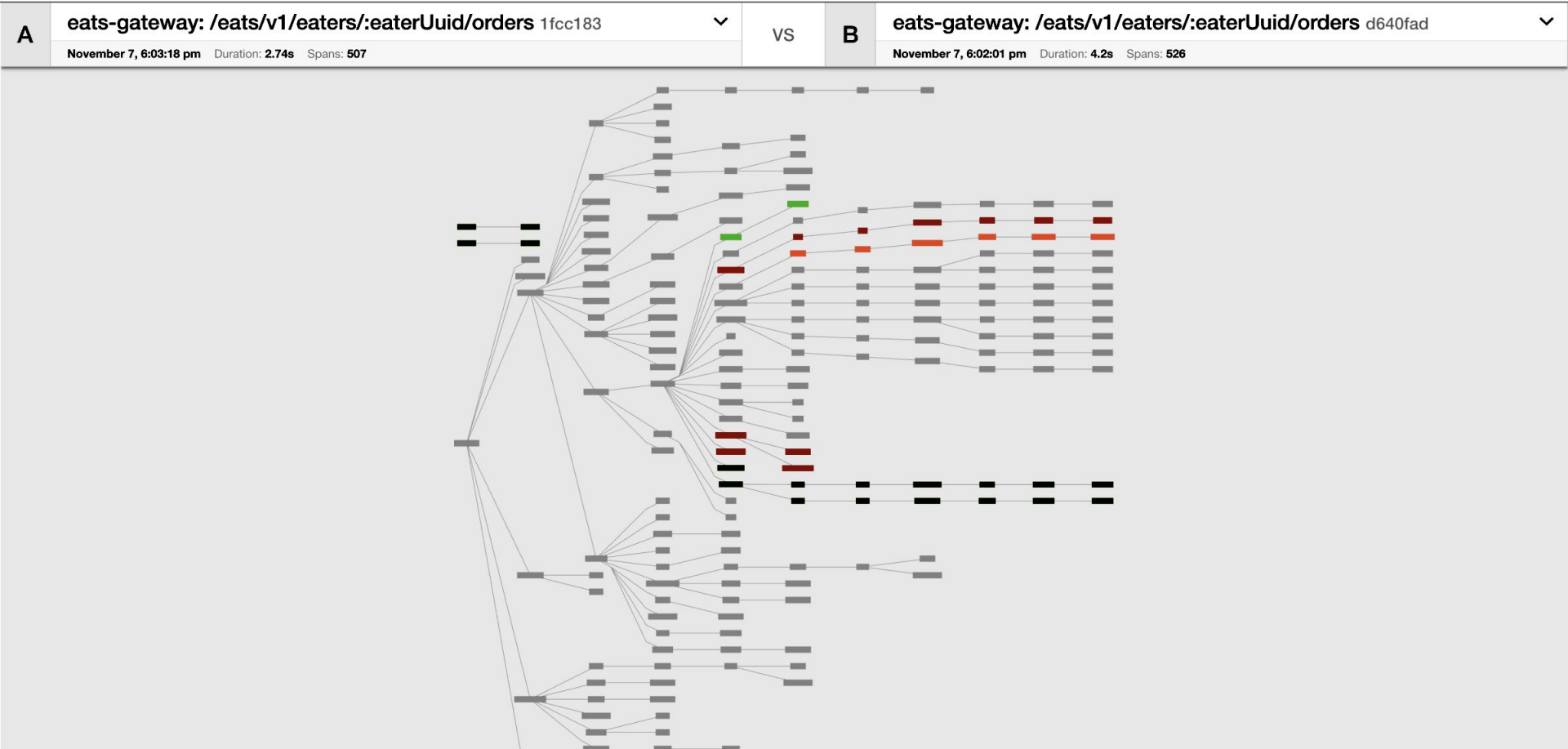


Comparing trace structures – Substantial divergence

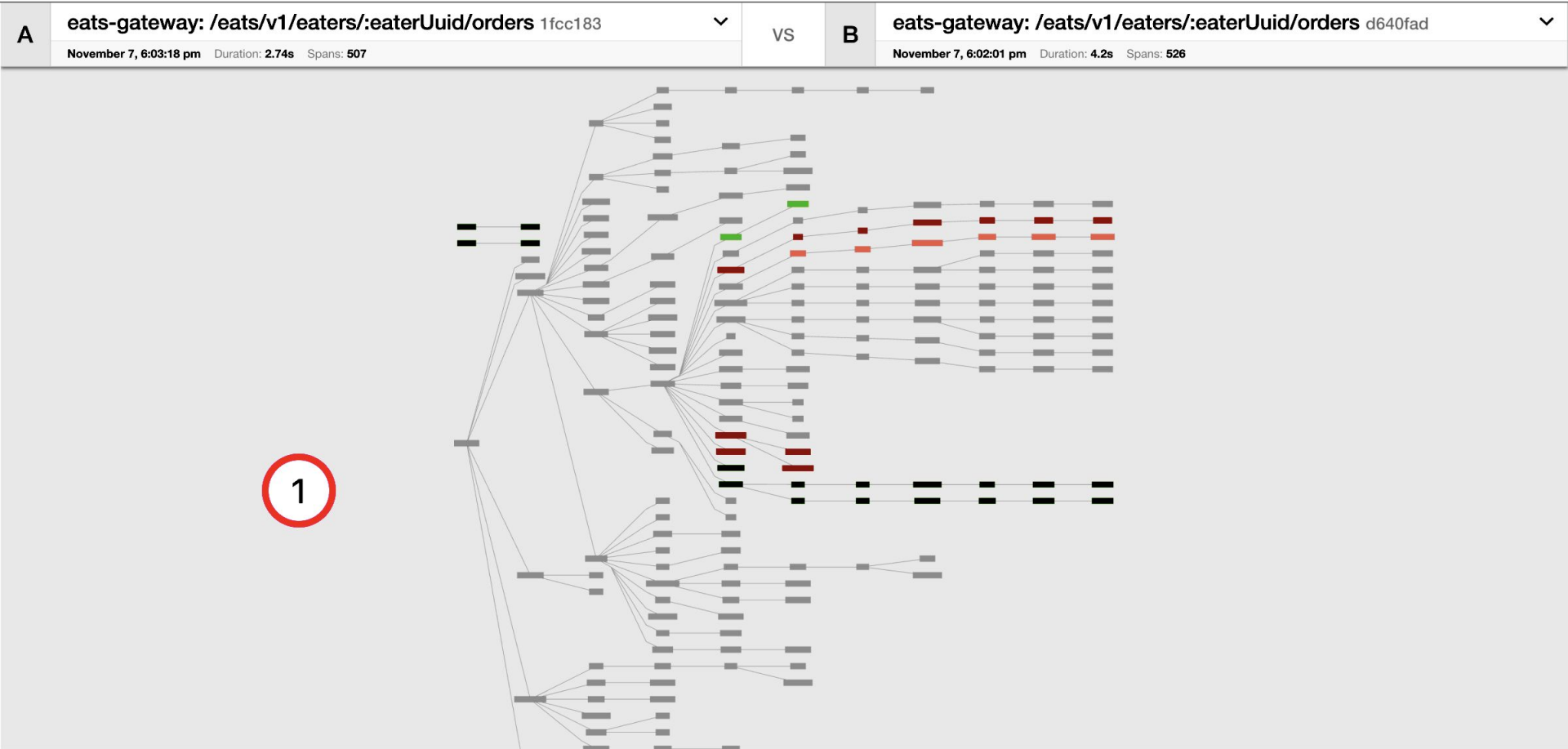


Roadmap

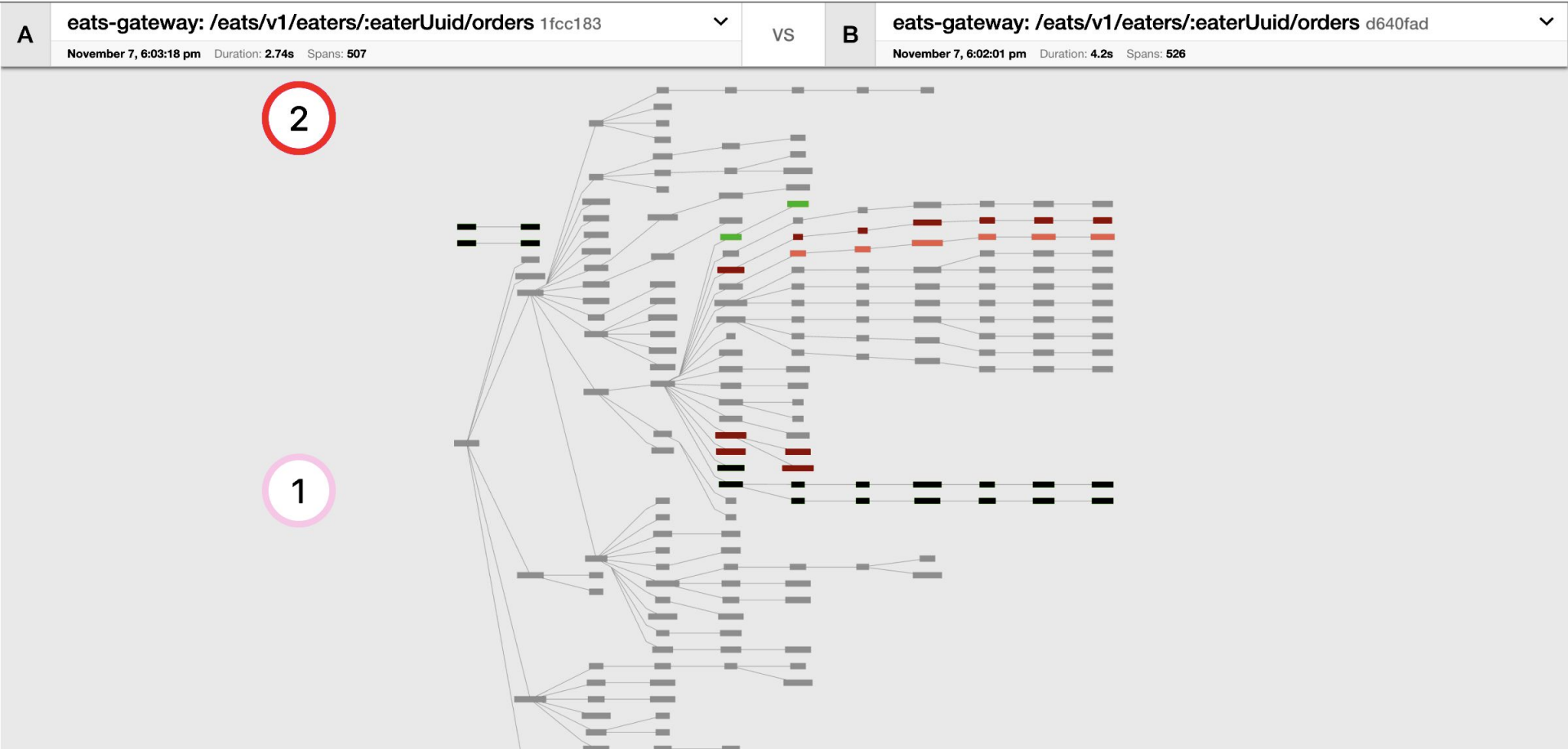
Structural vs. Time



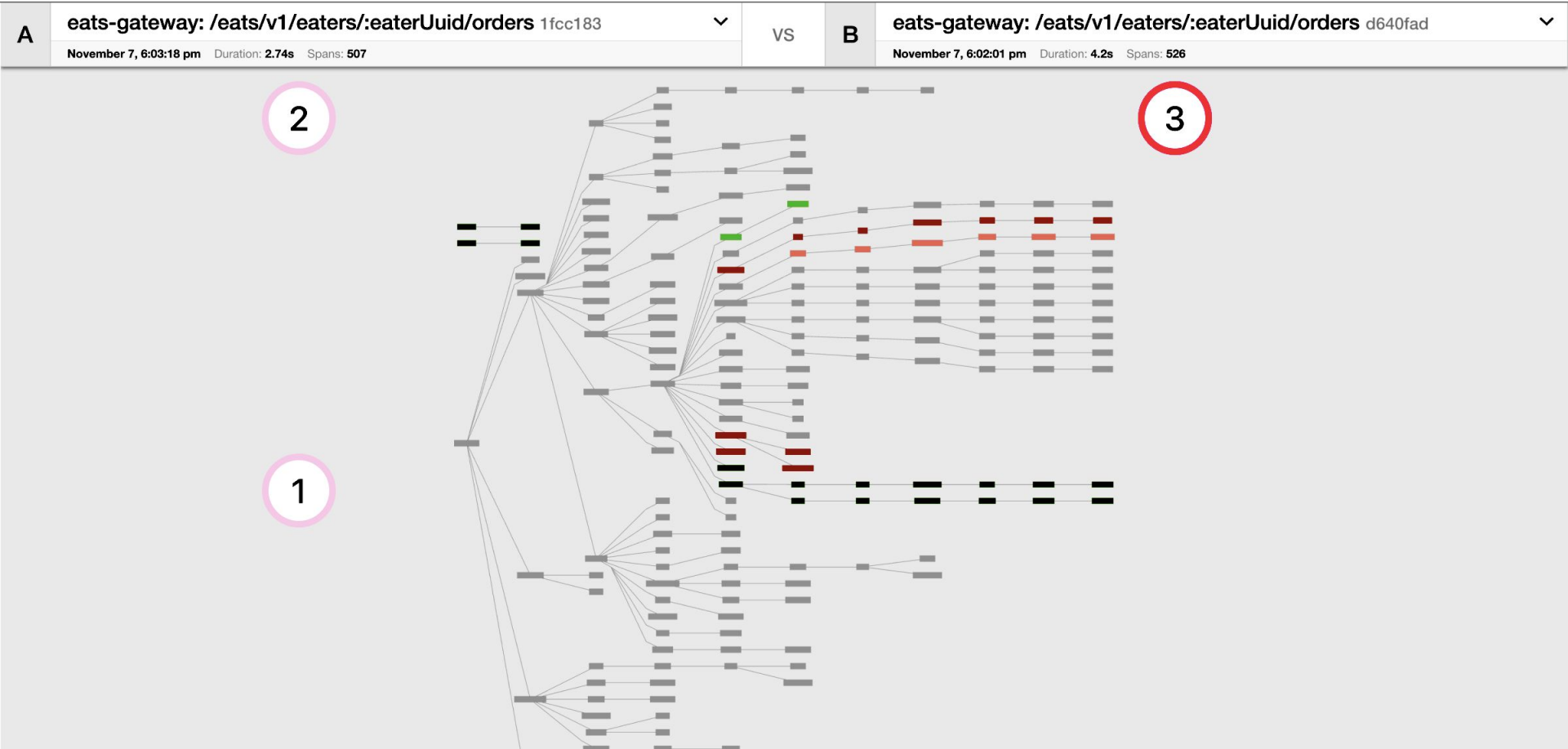
Structural vs. Time – Very similar structures



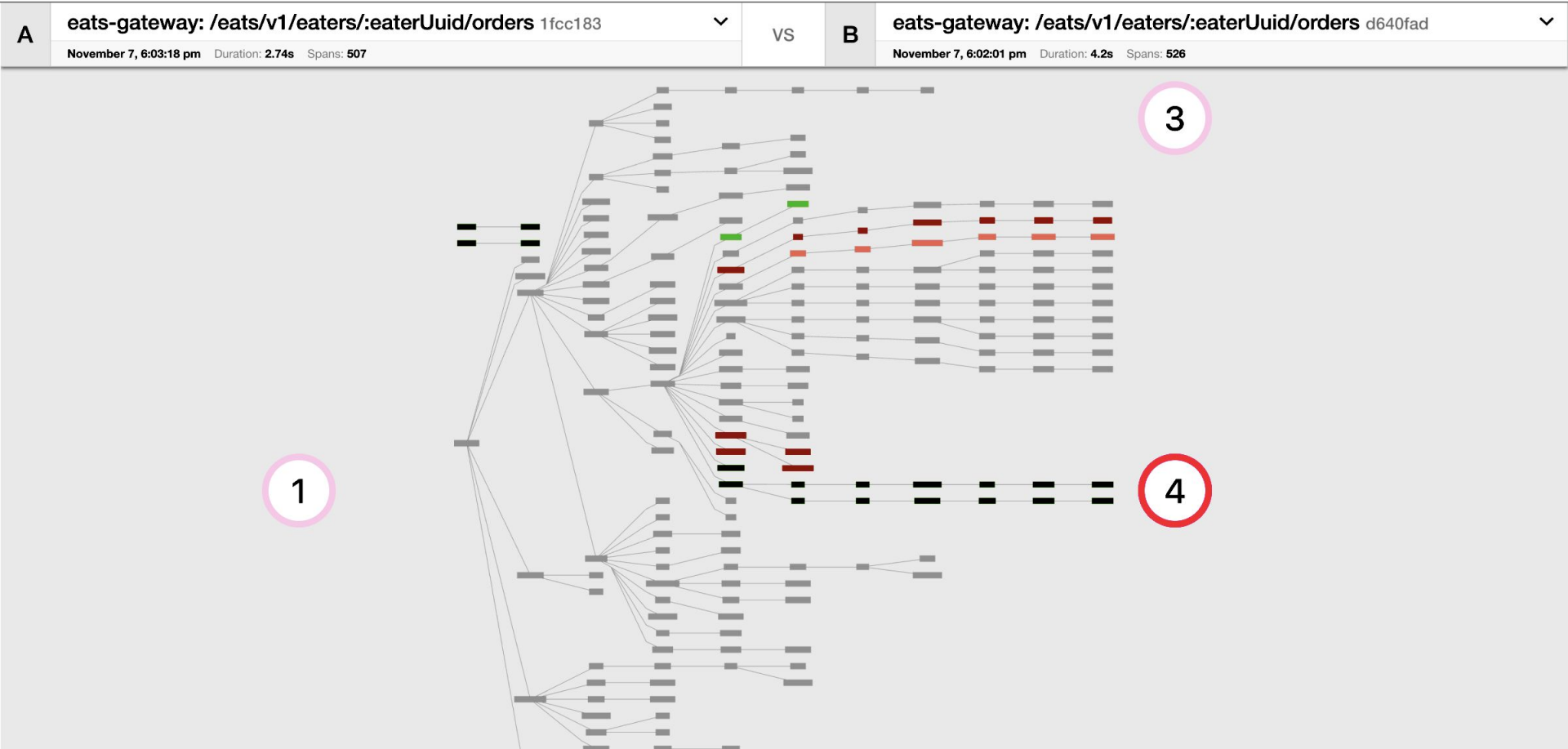
Structural vs. Time – 2.74 seconds



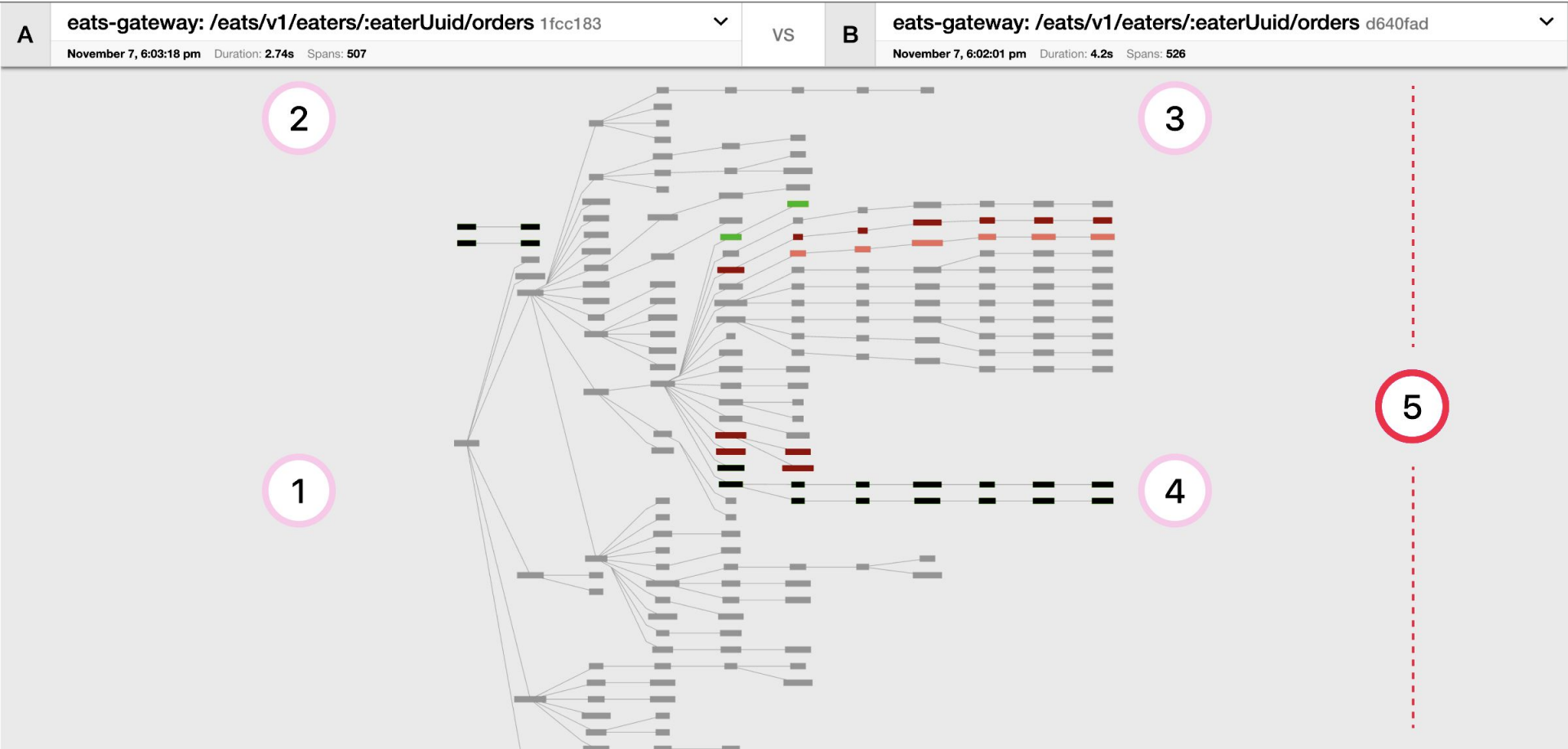
Structural vs. Time – 50% increase in duration



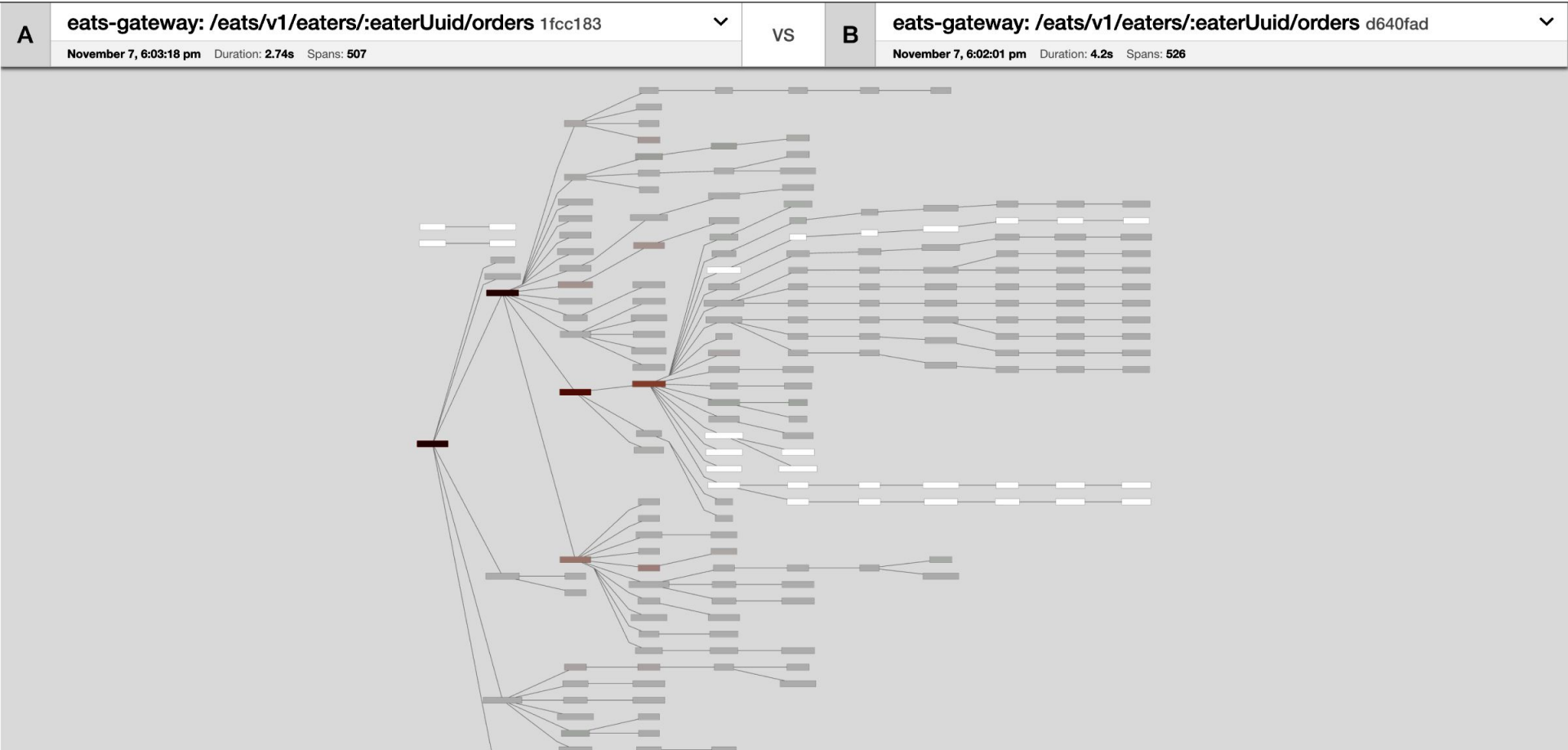
Structural vs. Time – Are these new spans to blame?



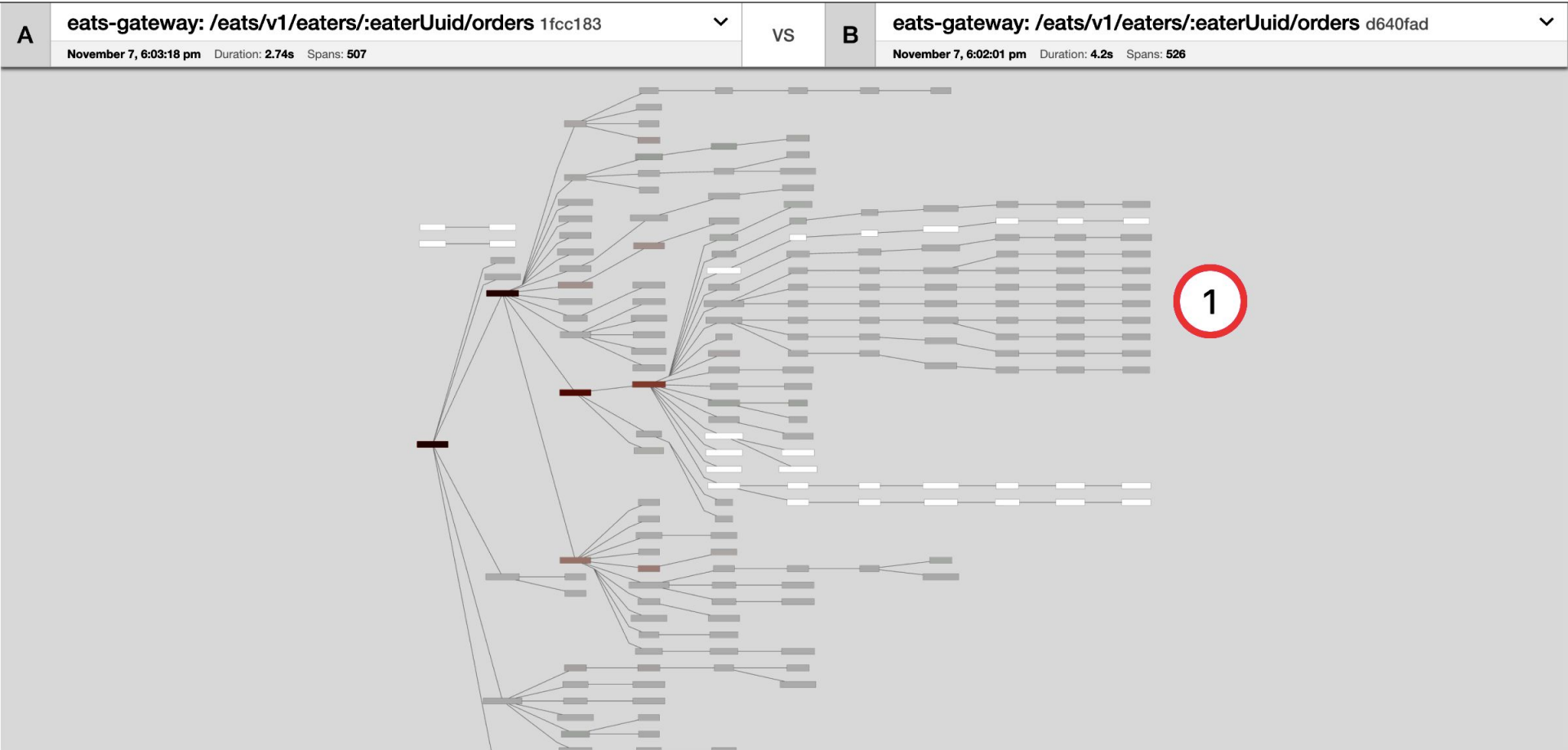
Structural vs. Time – Or is the lag increased throughout?



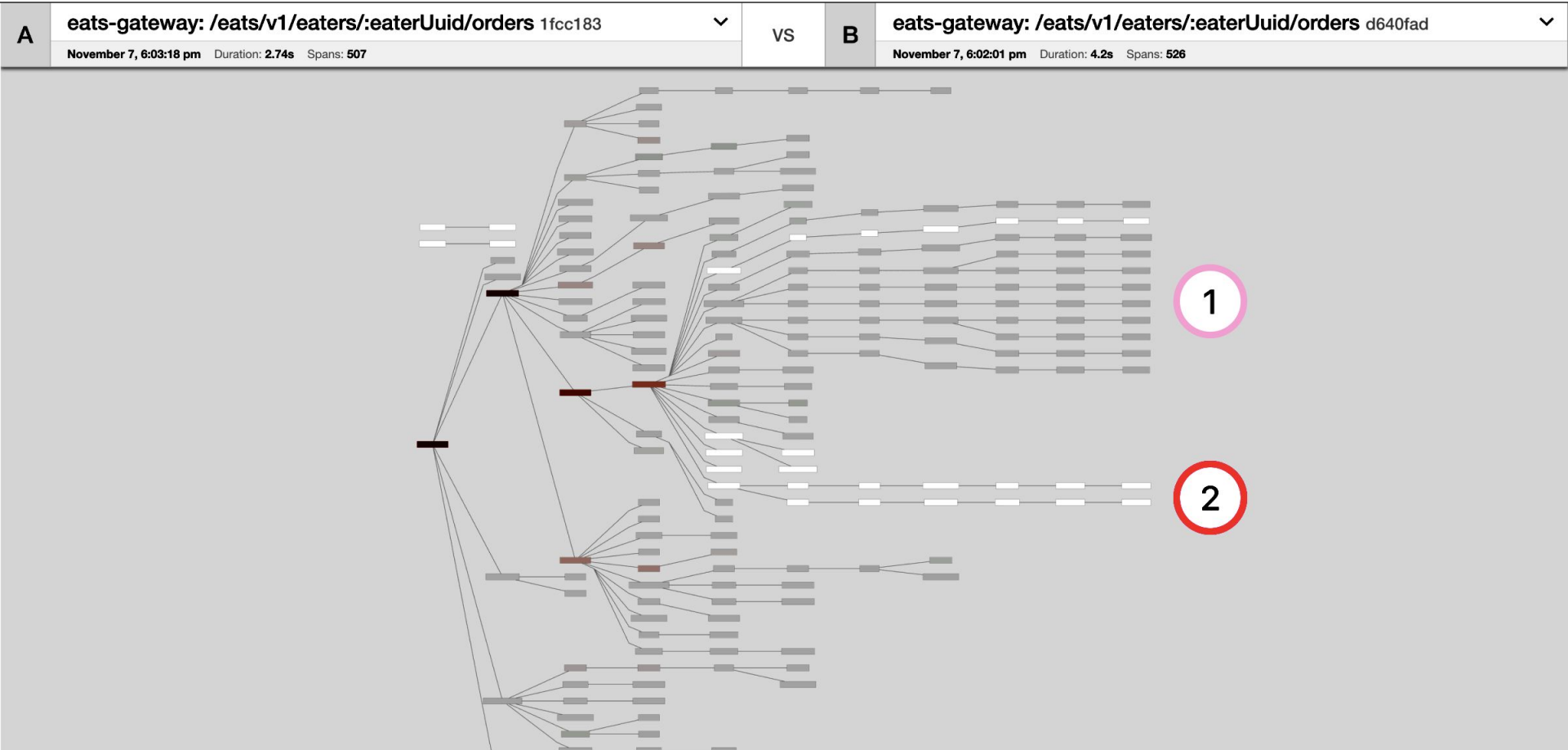
Comparing span durations – Roadmap Item



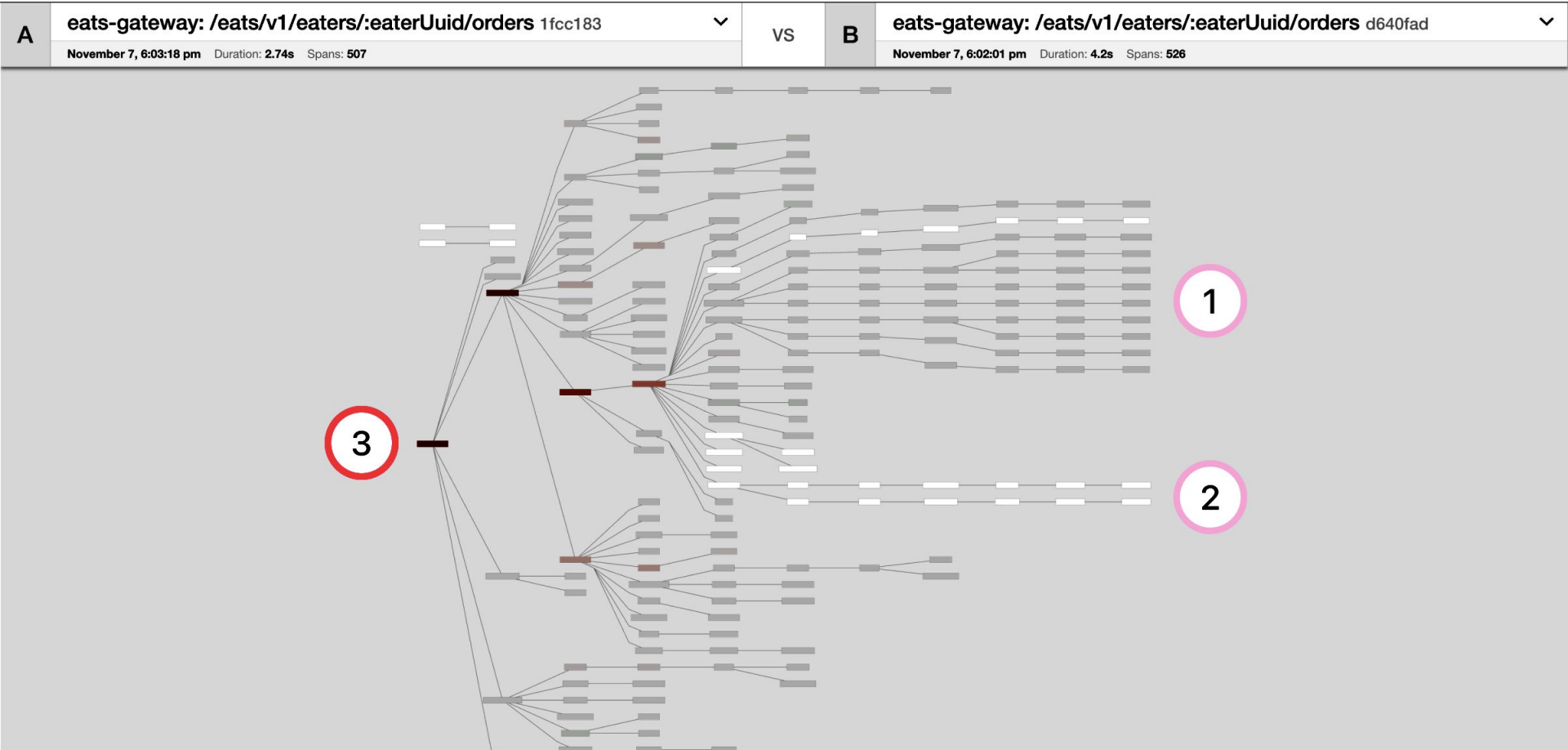
Comparing span durations – Similar durations



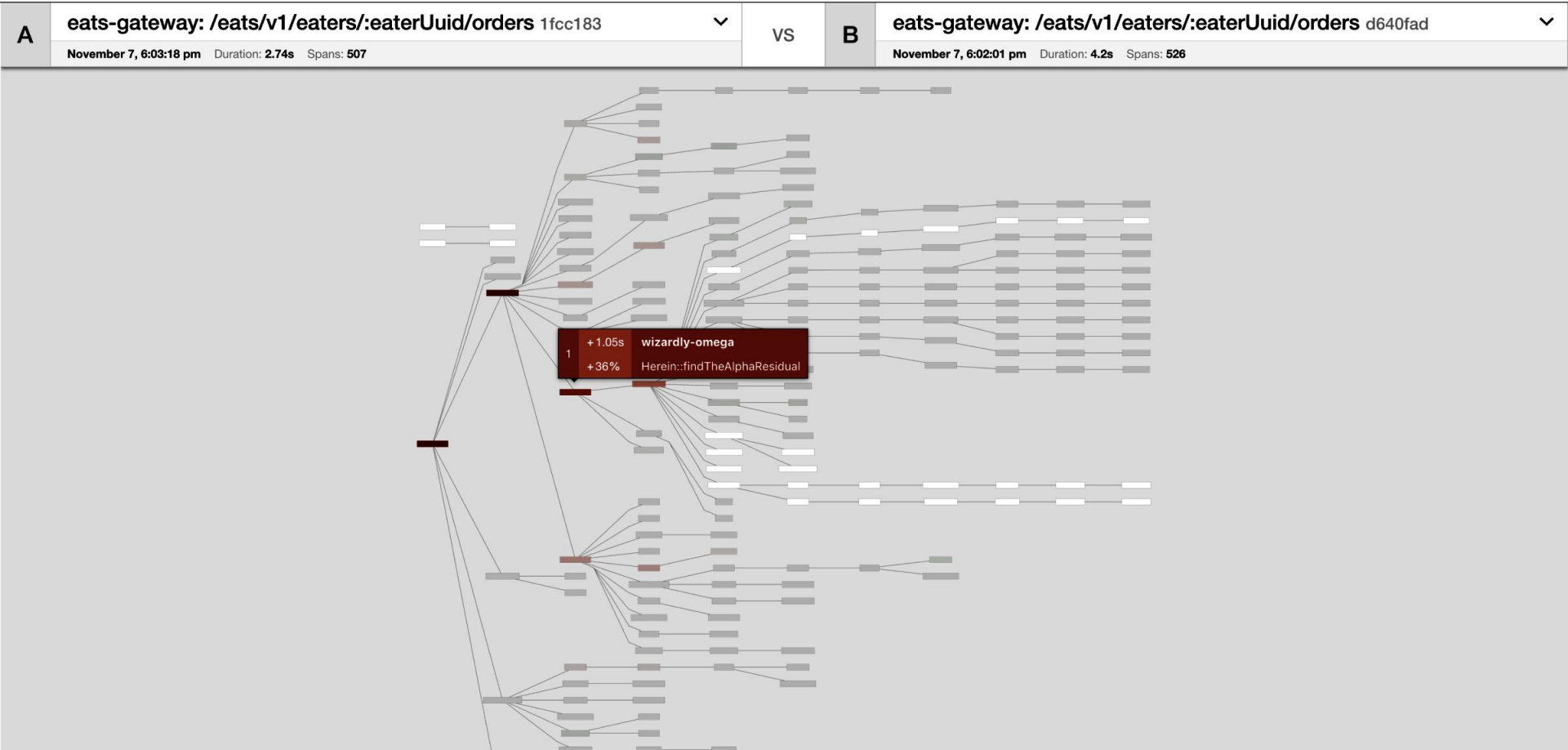
Comparing span durations – Nodes that aren't shared



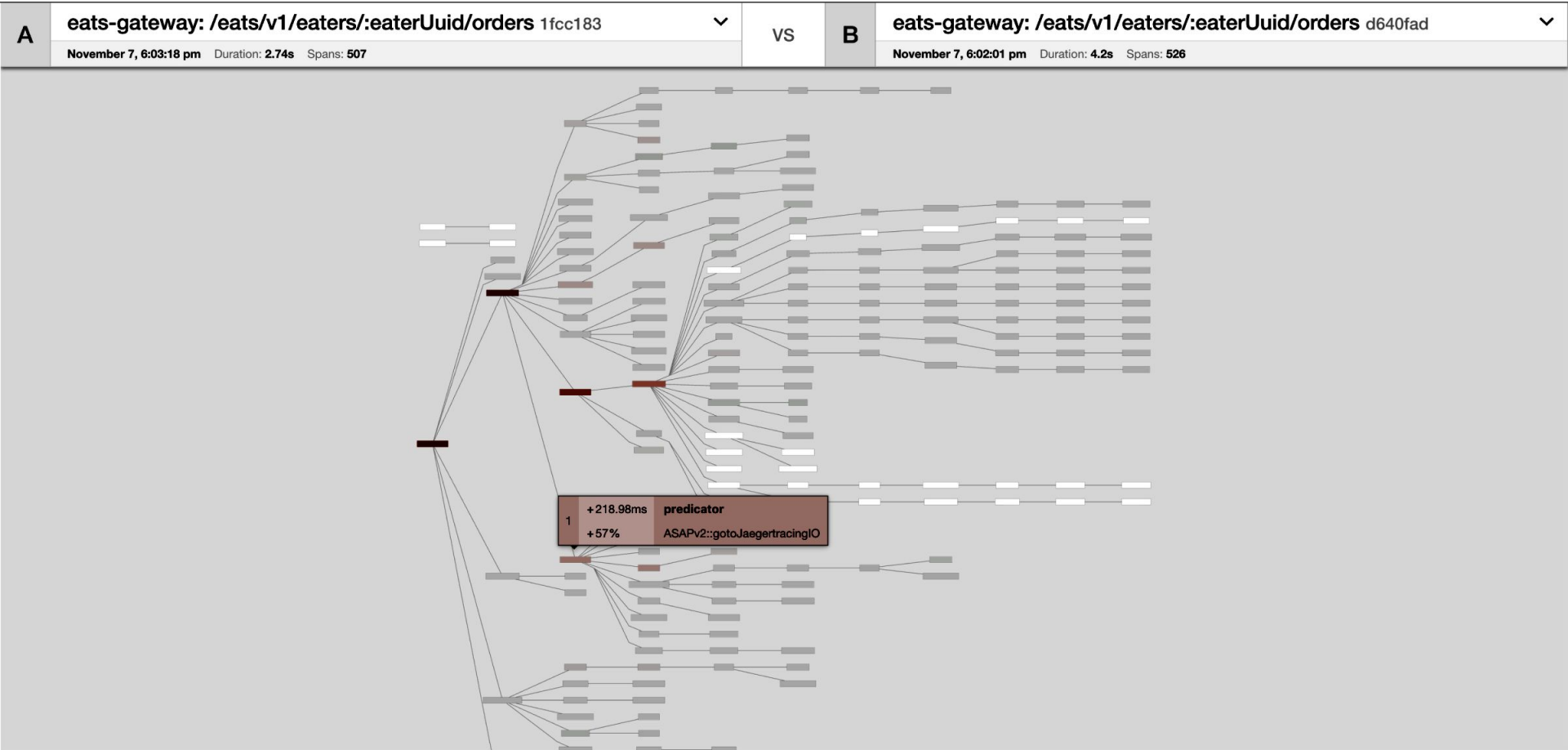
Comparing span durations – Follow the slower nodes



Comparing span durations – Coming soon...



Comparing span durations – Coming soon...



Learn More

Website: jaegertracing.io/

Blog: medium.com/jaegertracing

Getting in Touch

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

Q & A

Open Discussion

Common Questions

- More resources: Mastering Distributed Tracing Book by Yuri

  https://www.shkuro.com/books/2019-mastering-distributed-tracing/

- Overhead of tracing

  https://medium.com/@soria.gaby/what-is-the-cost-of-doing-instrumentation-aae5844d673f

- Sampling: Tail sampling in OpenCensus, Adaptive sampling in Jaeger
- Message bus / Async / Long traces
  - Varies — OpenTracing supports via follows from
  - Visualization can be hard - Jaeger UI is primarily for RPC view

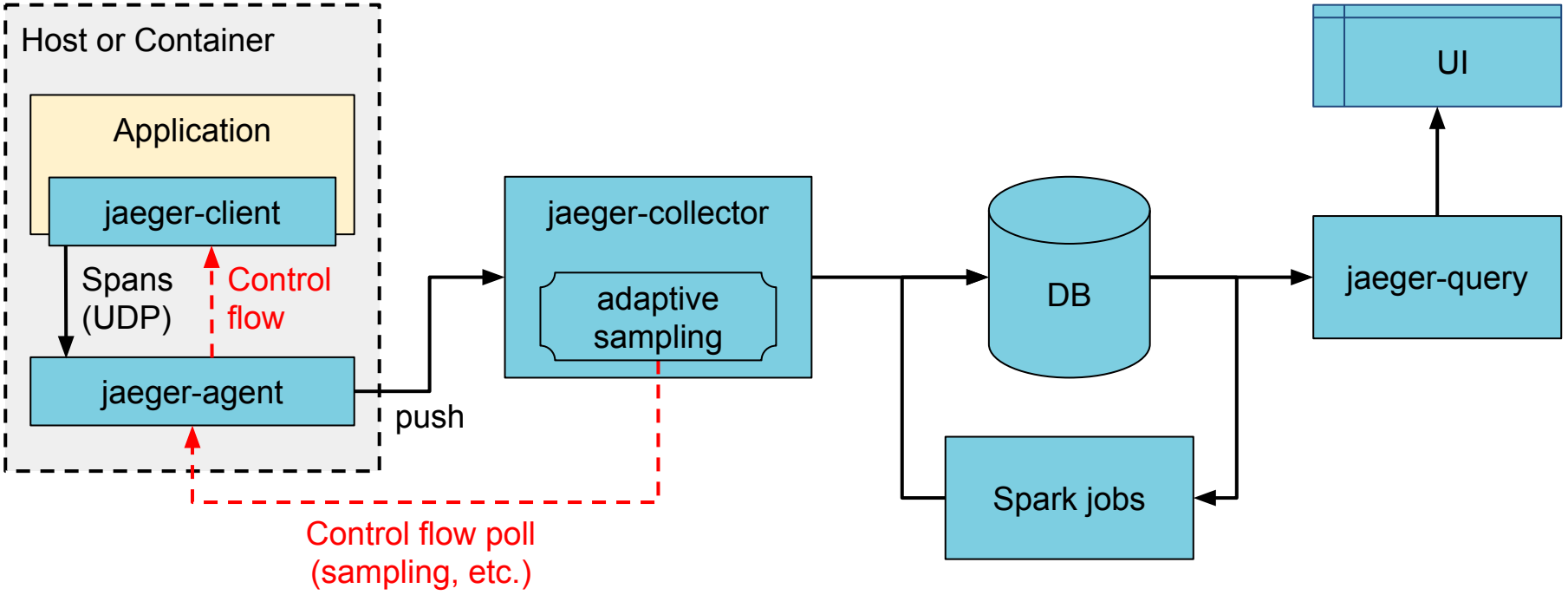
Appendix

Additional Topics and Slides

Architecture Changes

Asynchronous Ingestion

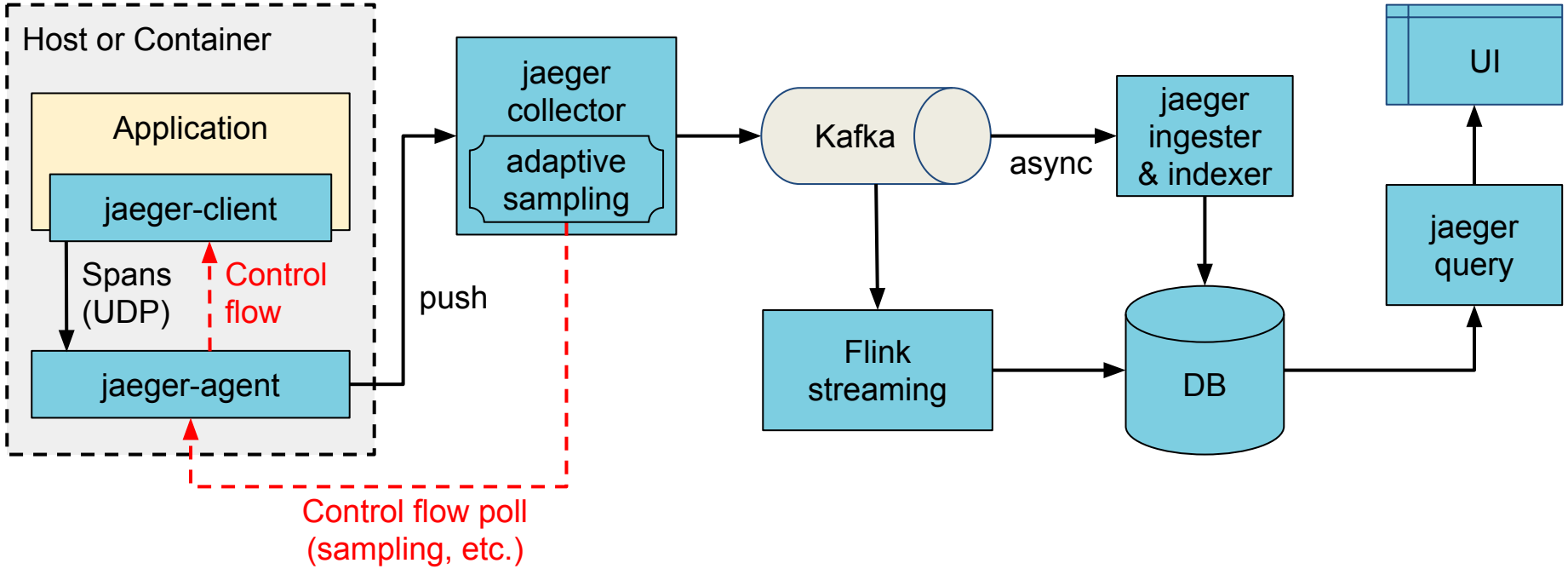
Architecture 2017: Push



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`

Architecture now: Push+Async+Streaming



Protobuf & gRPC

Enabling roadmap

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

Zipkin Compatibility

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

Roadmap

http://bit.do/jaeger-roadmap

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 dependencies diagram
  - Path-based dependencies diagram
  - Latency histograms

Storage plugins

- Based on gRPC/Protobuf work
- PRs in progress for proof of concept
- Community support for different storage backends

Partial Spans (community driven)

- Add ability to store/retrieve partial spans
- Use case:
  - Certain workflows are hours long. Unfortunately spans are only emitted once after it's Finished(). "Root span" is missing until the complete workflow is finished.