Jaeger Intro

Won Jun Jang (Uber), Prithvi Raj (Uber)
Agenda

● Distributed Tracing introduction
● Demo
● Project introduction
● Roadmap
● Discussion
Distributed Tracing

Concepts and terminology
Transaction Monitoring for Microservices

“The Simple [Inefficient] Thing”

Basic Concurrency

Async Concurrency

Distributed Concurrency
Context Propagation & Distributed Tracing

Edge Service
Unique ID → {context}

TRACE

SPANS

C
D
E
OpenTracing

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

http://opentracing.io
Let’s look at some traces
Jaeger, a Distributed Tracing System

https://jaegertracing.io
Inspired by Google’s Dapper and OpenZipkin
Started at Uber in August 2015
Open sourced in April 2017
Official CNCF project, Sep 2017
Built-in OpenTracing support

https://jaegertracing.io
Technology Stack

- Go backend
- Pluggable storage
  - Cassandra, Elasticsearch, memory, ...
- React/Javascript frontend
- OpenTracing Instrumentation libraries
Community

- 10 full time engineers at Uber and Red Hat
- 60 contributors on GitHub
- Already used by many organizations
  - including Symantec, Red Hat, Base CRM, Massachusetts Open Cloud, Nets, FarmersEdge, GrafanaLabs, Northwestern Mutual, Zenly
Distributed Tracing Systems

- distributed transaction monitoring
- root cause analysis
- distributed context propagation
- performance and latency optimization
- service dependency analysis
Recent Highlights

- Adaptive Sampling
- Asynchronous Ingestion
- Trace Comparisons
- Jaeger Operator
Roadmap

Things we are working on
Post Trace Sampling

- Jaeger currently only supports head based sampling
- Sample all traces and use heuristics to determine “interesting” traces.
Based on Kafka and Apache Flink

Support aggregations and data mining

Examples:
- Pairwise dependencies diagram
- Path-based dependencies diagram
- Latency histograms
Incomplete Span Support

- 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.
Q & A
Open Discussion
Jaeger at CloudNativeCon

- Jaeger Deep Dive
  - Thursday, November 15 • 2:20pm - 2:55pm

- Navigating the Distributed Systems Execution Maze with OpenTracing
  - Wednesday, November 14 • 2:40pm - 3:15pm