Monitorama

Distributed Tracing at UBER Scale

Creating a treasure map
for your monitoring data

Yuri Shkuro, UBER Technologies

ABOUT ME

- Software Engineer on the Observability team in NYC
- Working on the open source distributed tracing system Jaeger
- Co-founded the OpenTracing project
- Banking industry survivor

- Github: yurishkuro
- Twitter: @yurishkuro

Would You Like Some Tracing with Your Monitoring?

What does it take to roll it out?

Why Distributed Tracing

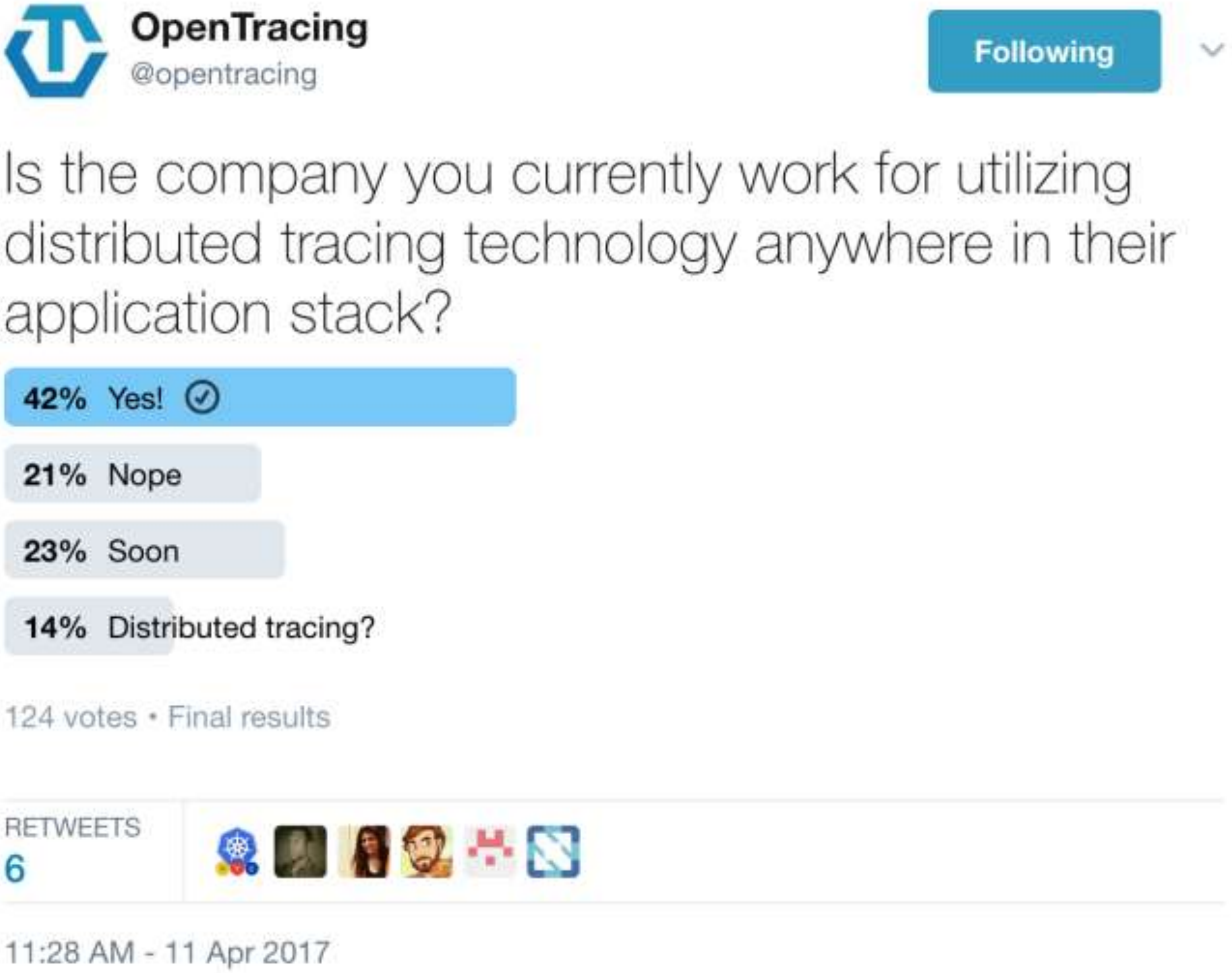
- Distributed transaction monitoring
- Performance / latency optimization
- Root cause analysis
- Service dependency analysis
- Distributed context propagation (“baggage”)

JAEGER, Distributed Tracing

- Open Source
- OpenTracing inside
- In active development
- PRs are welcome
- Zipkin compatible

- github.com/uber/jaeger

Who Thinks Tracing is Awesome?



OpenTracing
@opentracing

Is the company you currently work for utilizing distributed tracing technology anywhere in their application stack?

42%	Yes! ✓
21%	Nope
23%	Soon
14%	Distributed tracing?

124 votes • Final results

RETWEETS
6

11:28 AM - 11 Apr 2017

Lies!

Why Doesn’t Everyone Do Tracing?

Tracing Instrumentation is

HARD

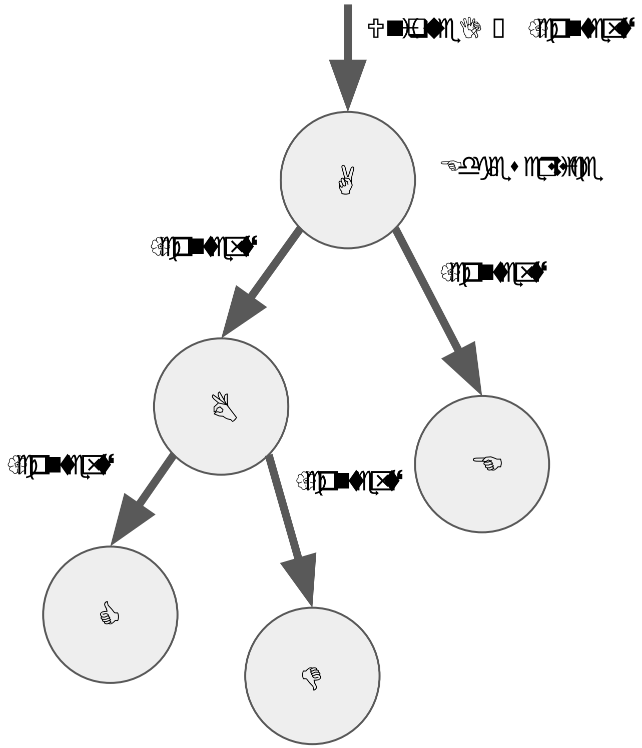
EXPENSIVE

BORING

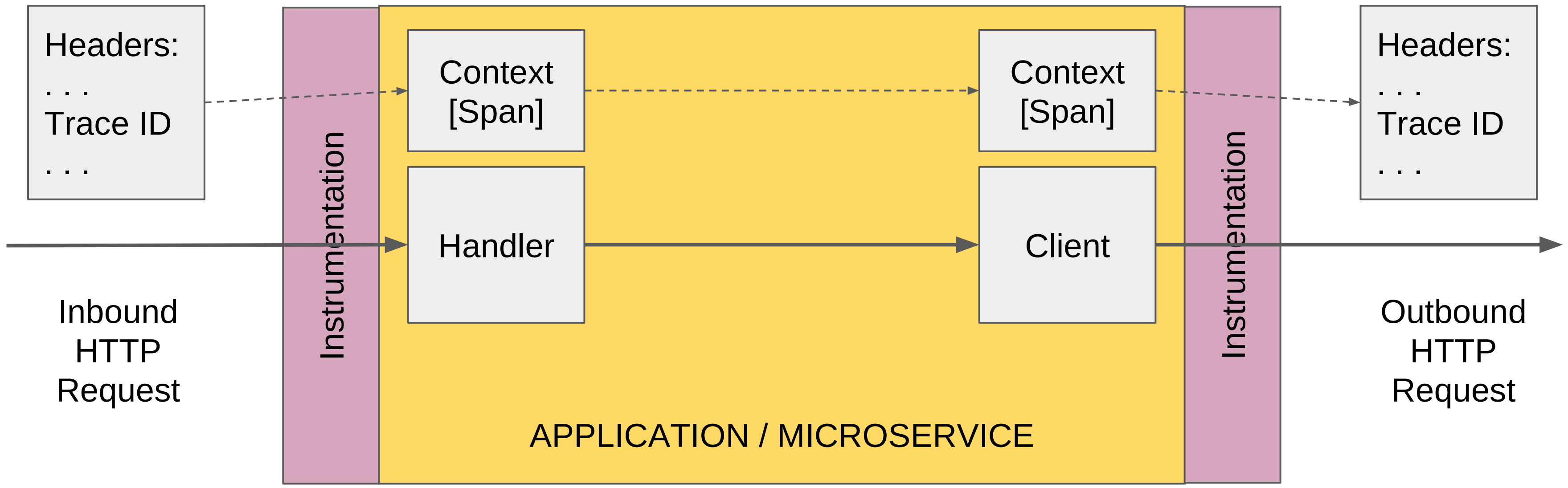
Instrumentation

- Metrics and logging are not new
- Tracing is both new and harder

Context Propagation



Context Propagation



In-Process Context Propagation

Java python

golang node JS

Implicit, via Thread-Locals

Explicit

but: thread pools, futures

It’s Also the Frameworks

- Go: stdlib, gorilla, …
- Java: jaxrs2, okhttp, ApacheHttpClient, …
- Python: Flask, Django, Tornado, urllib2, …
- Node.js – who knows…

OpenTracing to the Rescue

OPENTRACING

CLOUD NATIVE
COMPUTING FOUNDATION

No Help With In-Process Propagation

- Must be done manually
- UBER has 2000-3000 microservices
- Resources of the tracing team are limited

- Developers must instrument their code!

~~BITE~~ MAKE ME!

How do we mobilize the org?

Traveling Salesman Problem

2017 edition

They Must Want Your Product

or Sticks and Carrots

Recap: Why Distributed Tracing

- Distributed transaction monitoring
- Performance / latency optimization
- Root cause analysis
- Service dependency analysis
- Distributed context propagation ("baggage")

Service Dependency Analysis

- Explain to us what we just built
- Who are my dependencies
- Workflow analysis
- Where is all this traffic coming from?
- Service tiers

Baggage

- Tenancy, test or production
  - Set at the top
  - Used at the storage layer, prod or test DB
- Authentication tokens
  - Signed user or service identity
  - Checked at multiple levels

Sticks and Carrots

- Get other teams build features on top
  - Performance team
  - Capacity & cost accounting
  - Baggage
- More carrots
- Eventually they become sticks (peer pressure)

Each Organization is Different

Find what works best

How to Measure Adoption?

Measure everything

Does Service X Report Traces?

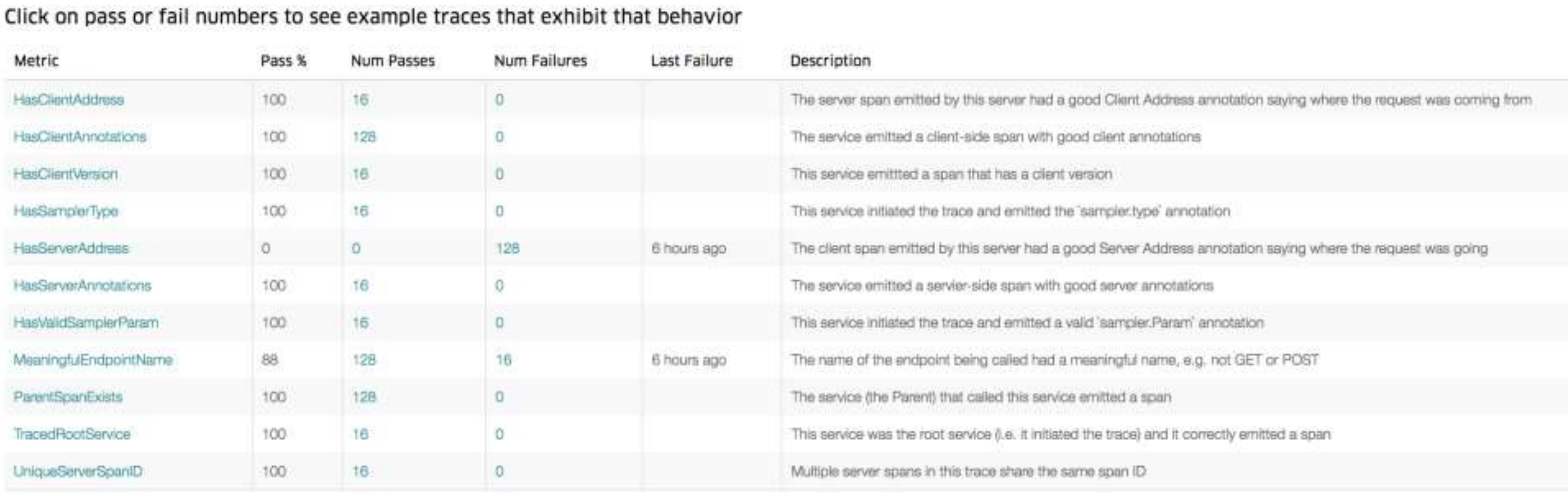
- Daily aggregation job
- Auto-book tickets
- Build a dashboard

- Pass/Fail: too easy to pass

Trace Quality Score

- Inspect traces
  - See a caller, but no spans
- Join with other data
  - Routing logs
- Auto-book tickets (carefully, not for everyone)
  - With detailed report

Trace Quality Metrics by Service



Click on pass or fail numbers to see example traces that exhibit that behavior

Metric	Pass %	Num Passes	Num Failures	Last Failure	Description
HasClientAddress	100	16	0		The server span emitted by this server had a good Client Address annotation saying where the request was coming from
HasClientAnnotations	100	128	0		The service emitted a client-side span with good client annotations
HasClientVersion	100	16	0		This service emittted a span that has a client version
HasSamplerType	100	16	0		This service initiated the trace and emitted the `sampler.type' annotation
HasServerAddress	0	0	128	6 hours ago	The client span emitted by this server had a good Server Address annotation saying where the request was going
HasServerAnnotations	100	16	0		The service emitted a server-side span with good server annotations
HasValidSamplerParam	100	16	0		This service initiated the trace and emitted a valid `sampler.Param' annotation
MeaningfulEndpointName	88	128	16	6 hours ago	The name of the endpoint being called had a meaningful name, e.g. not GET or POST
ParentSpanExists	100	128	0		The service (the Parent) that called this service emitted a span
TracedRootService	100	16	0		This service was the root service (i.e. it initiated the trace) and it correctly emitted a span
UniqueServerSpanID	100	16	0		Multiple server spans in this trace share the same span ID

Thank You

- Jaeger
  - https://github.com/uber/jaeger
  - Blog: Evolving Distributed Tracing at UBER
  - Blog: Take OpenTracing for a HotROD Ride
- OpenTracing: http://opentracing.io/
- We are hiring
- @yurishkuro