Spanning Services

The Practical Guide to Distributed Tracing

Yair Galler, @ygaller

The Starting Point

We already have metrics and logging!

but…

Metrics are aggregates

A metric in Grafana goes crazy as the team stares. Root cause unknown.

Michael Ancher, 1902

Logging lacks context

A developer realizes he has been looking at the wrong set of logs for the last 4 hours.

Auguste Couder, 1817

What is Context?

Contexts

A correlation across service boundaries

Has to be propagated

Logging has no inherent concept of context

Hard to implement context propagation in a multi-language environment

Async flows are Tricky

How Does Tracing Differ?

Provides context propagation
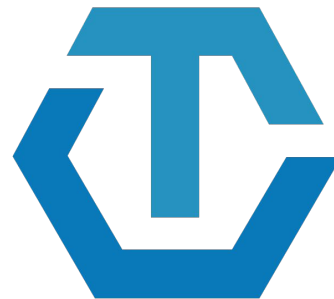
Hierarchical

Explicit causality

Visual flow of time

It's a "distributed stack trace"

Opentracing

An API specification - Not software

Reporting and retention are decoupled

Instrumentation - Many implementations across different technologies

The Span

The basic building block in Opentracing

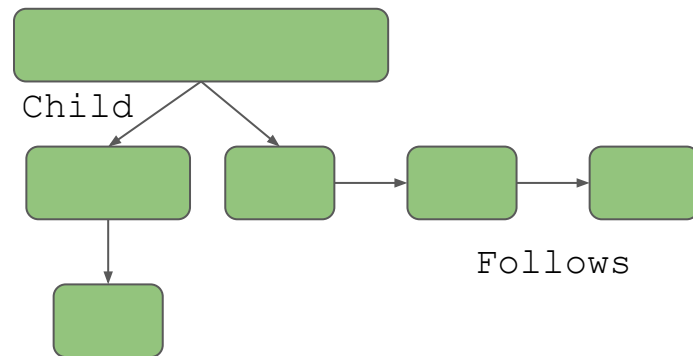Represents an action from start to finish

Tags - Used for filtering
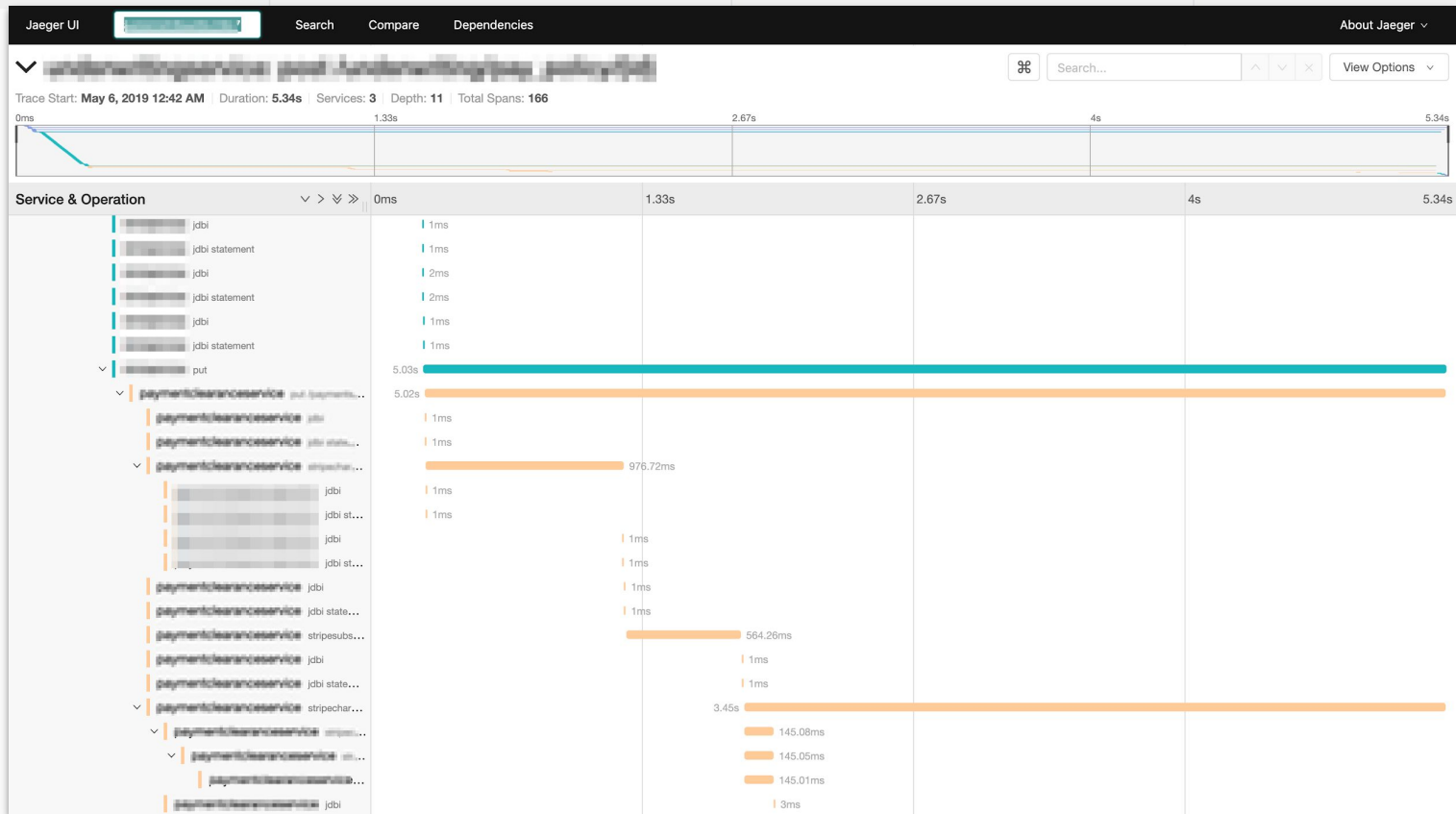
Logs - Internal span debug info

Span context

Relation to other spans - Child or Follows

Properties

Baggage

Child
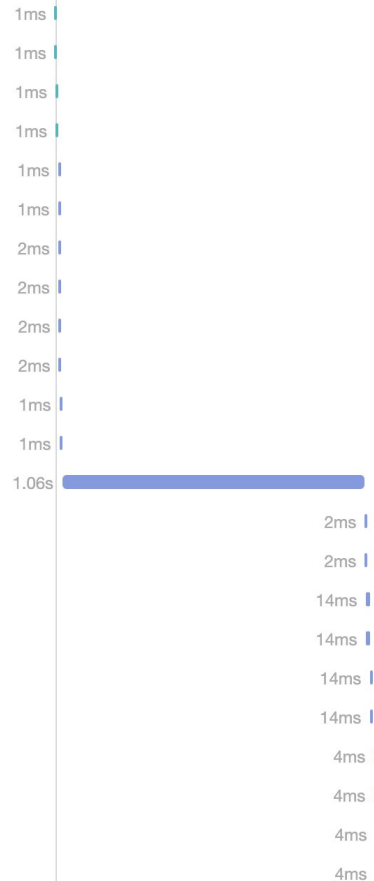
Follows

Use Cases
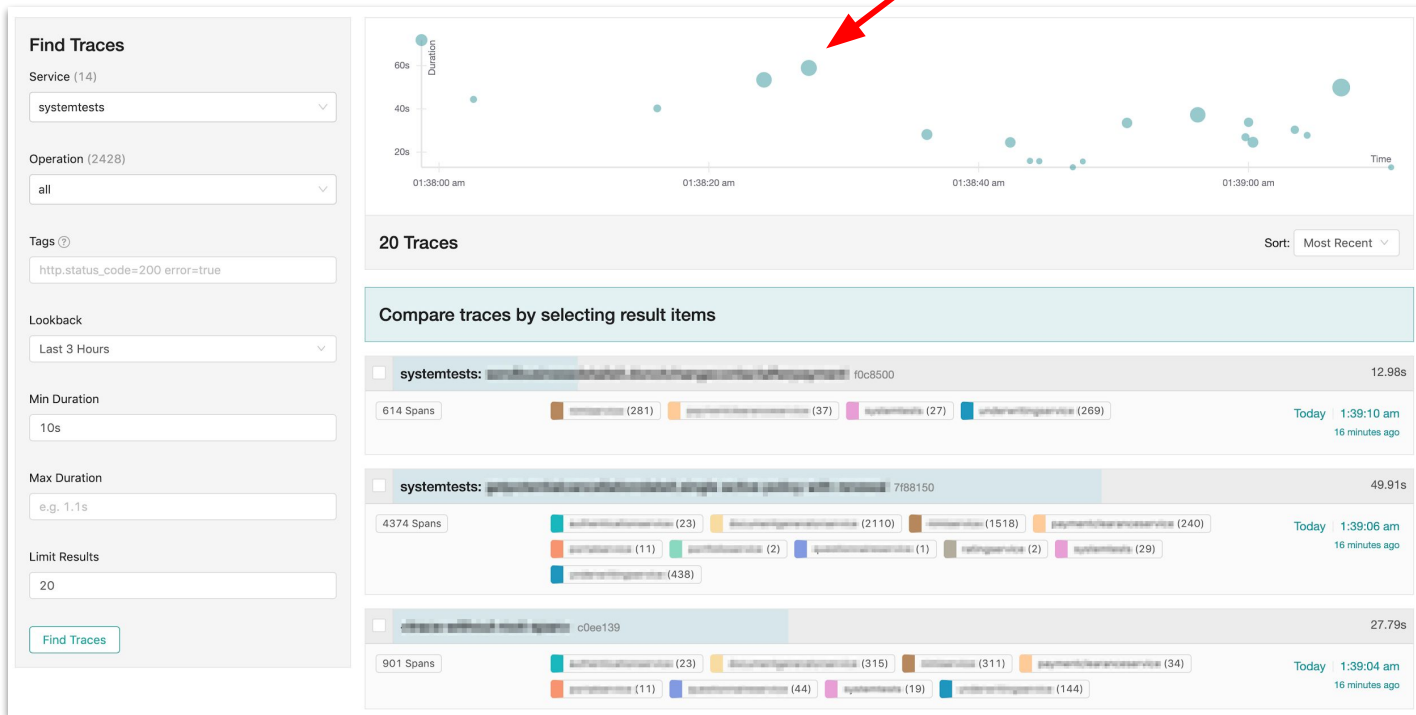
Use Cases - Optimization

Slow Running Segments

Waiting for tests to complete.

A dream of deployment by Edmund Leighton, 1922
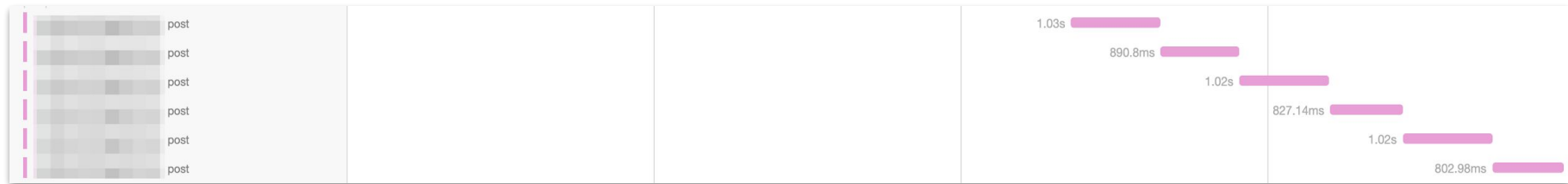
Use Cases - Optimization

Slow Tests

Use Cases - Optimization
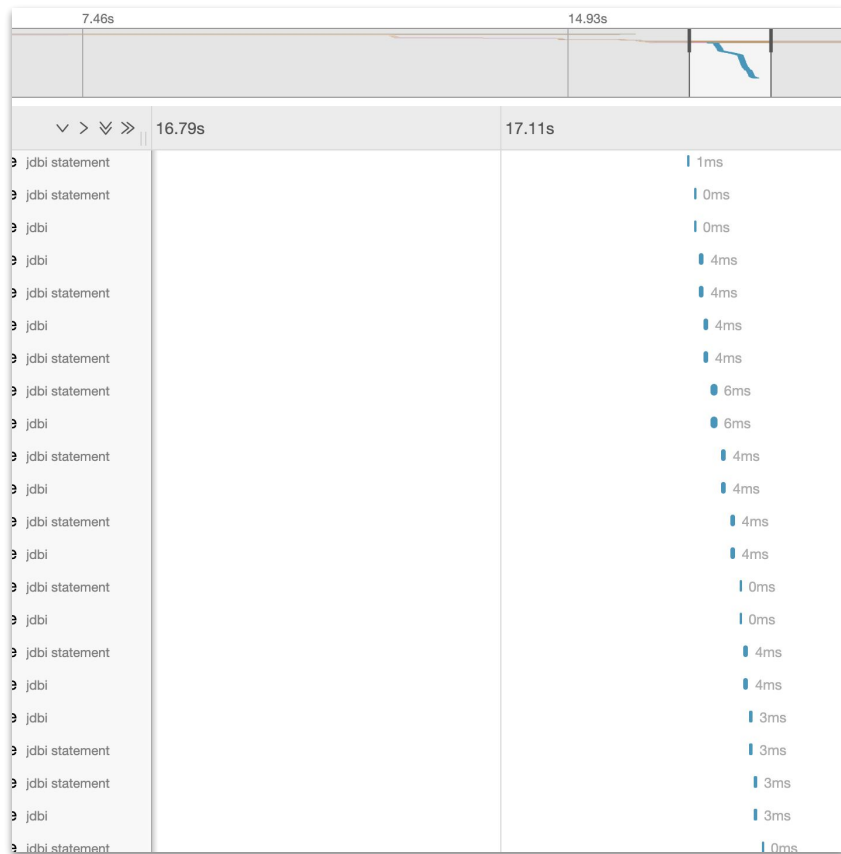
Parallelization

Use Cases - Optimization

Network Latency

Use Cases - Optimization

Call Redundancy

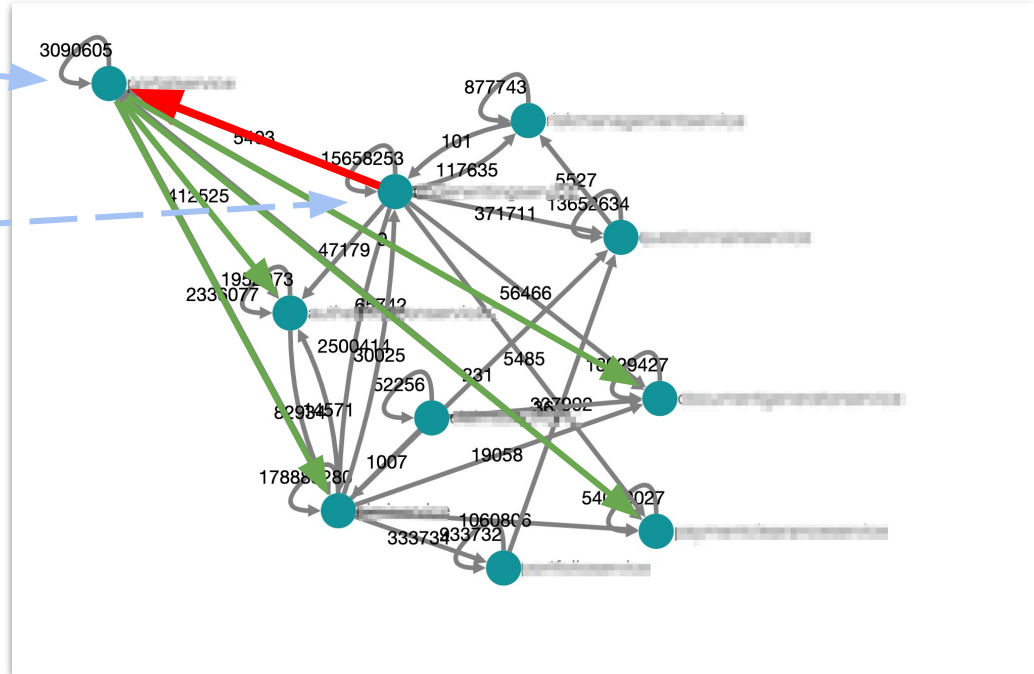Multiple calls can be batched into one

Use Cases - Architecture

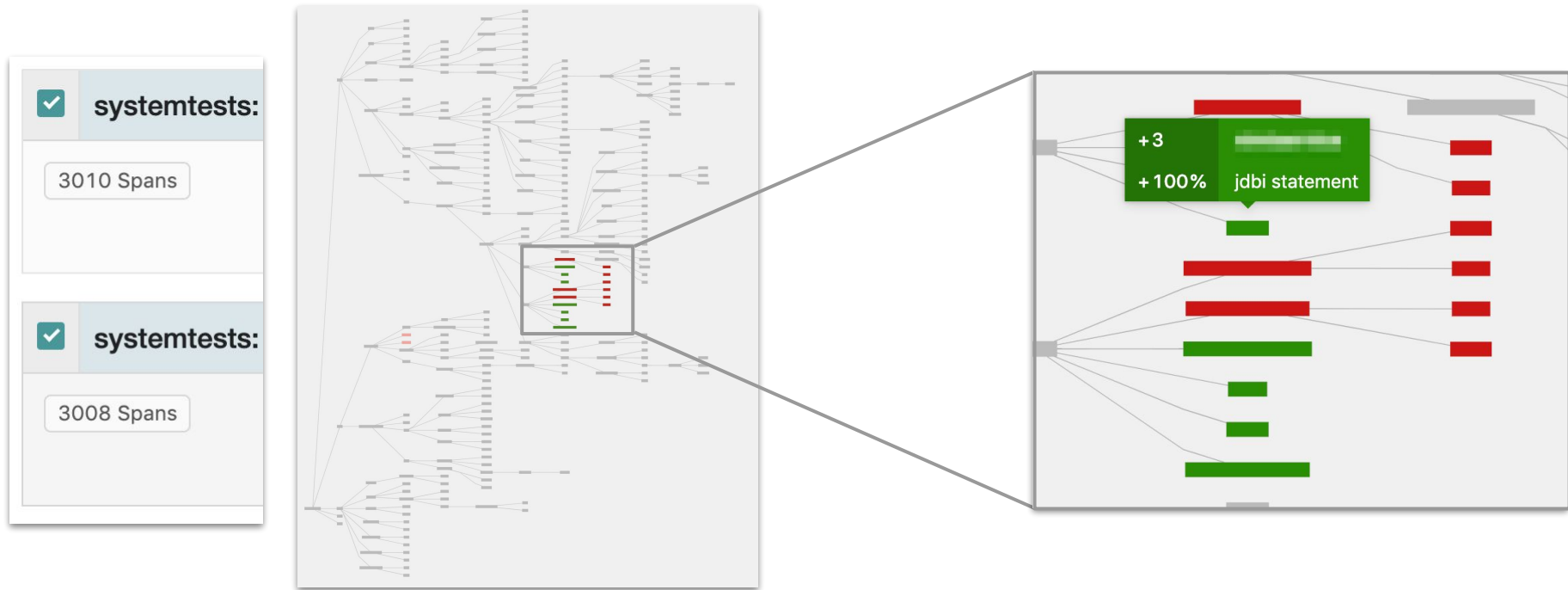## Service Boundary Design Flaws

Stateless edge service

Central backend service

Use Cases - Debugging

Run Comparisons

Major Implementations

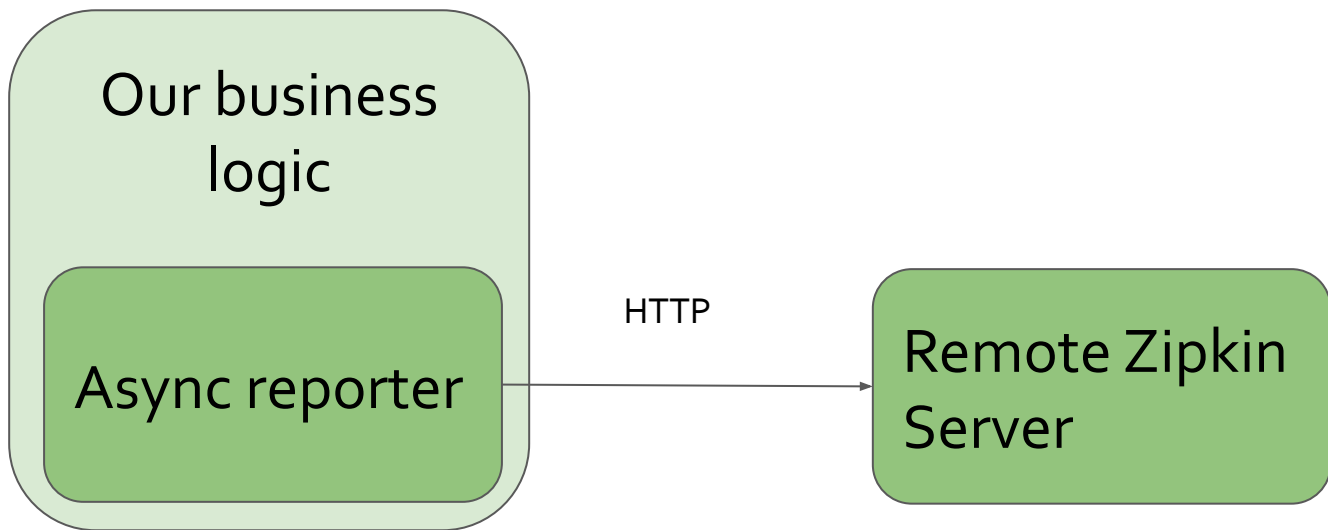Zipkin (Twitter → OpenZipkin Foundation)

Jaeger (Uber → CNCF Foundation)

Haystack (Expedia)

Commercial solutions - Lightstep, New Relic, Datadog, Honeycomb.io
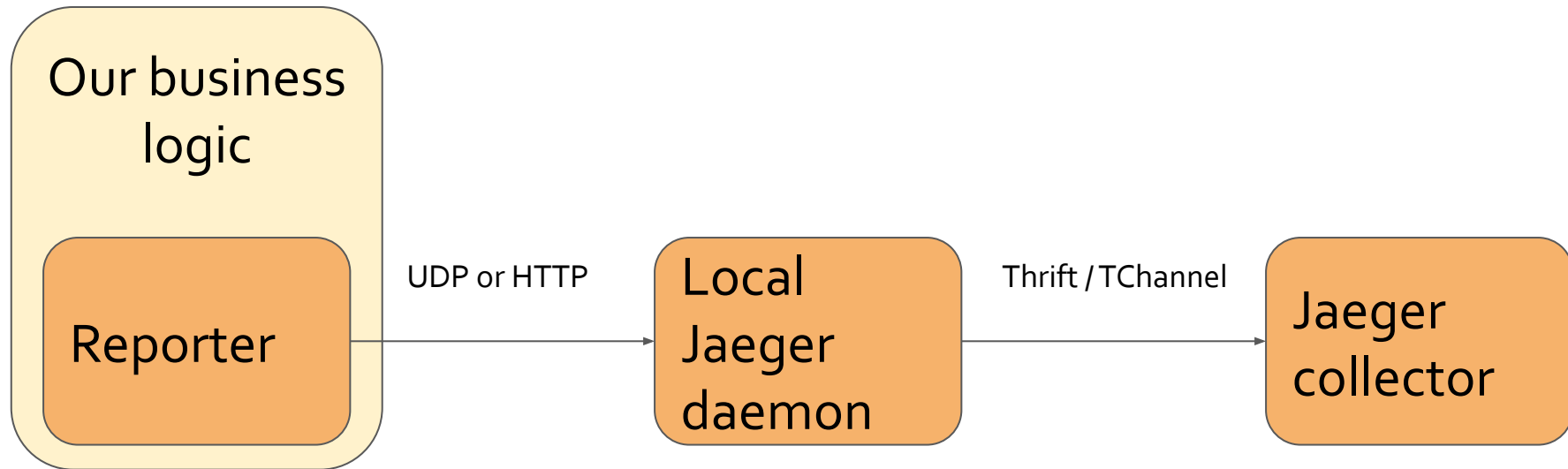
Under The Hood

Reporting with Zipkin

Context propagation using `X-B3-*` headers

Under The Hood

Reporting with Jaeger

Our business logic

Reporter → UDP or HTTP → Local Jaeger daemon → Thrift / TChannel → Jaeger collector

The Tracing Cookbook

Standard Instrumentation

Incoming HTTP calls - Supported by DropWizard & Spring

HTTP Clients - Register a tracer

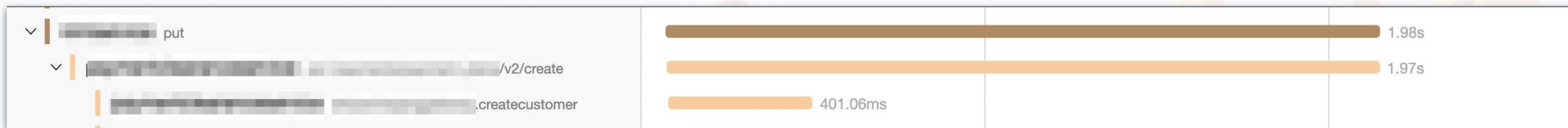So what else?

The Tracing Cookbook

Extras - Method Tracing

Using annotations & interceptors

Better tracing resolution
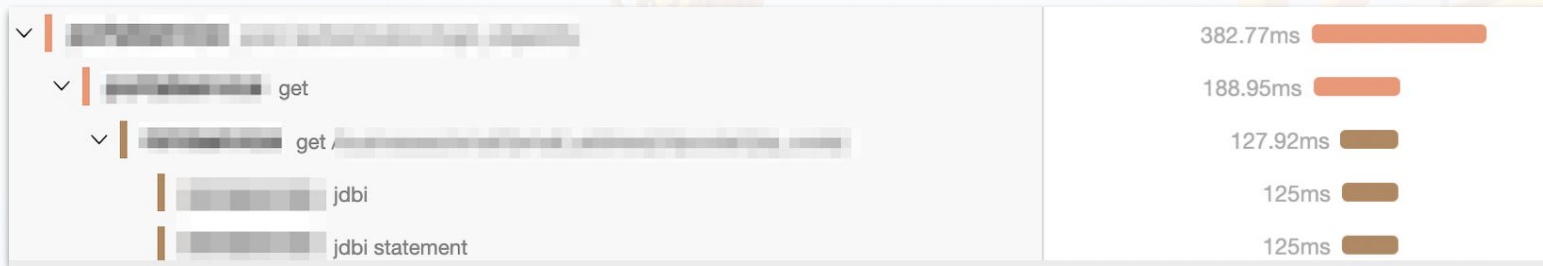
When you cannot trace calls from third party libraries

```
@Traced
open fun createCustomer
```

˅ ▓▓▓▓▓ put	████████████████████████████	1.98s
˅ ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓/v2/create	████████████████████████████	1.97s
▓▓▓▓▓▓▓▓▓▓▓▓▓▓ ▓▓▓▓▓▓.createcustomer	███████ 401.06ms	

The Tracing Cookbook

Extras - JDBI Tracing

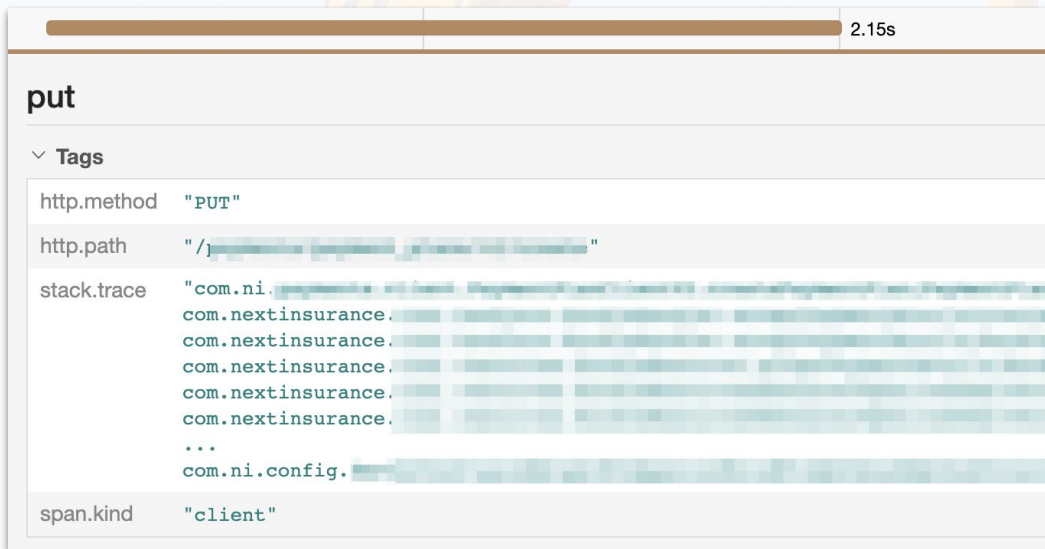Invaluable for catching slow queries!

* We used opentracing-jdbi3

The Tracing Cookbook

Extras - Stack Traces

HTTP calls can come from various paths

The Tracing Cookbook

Setting Up The System - What Worked for us

Zipkin Client

Jaeger Server - Multiple instances for high availability

Terraform for auto-deploy to AWS

Data stored on AWS Elasticsearch Service

Tips & Best Practices

You are going to invest time - Plan ahead and consider:

Lambdas

Kubernetes / containers in general

Get buy-in from all the teams or the relay breaks

Ensure system runs well when tracing server is down

Downsampling

Don't trace everything!

Worried he might be responsible…

Knows it wasn't him thanks to tracing!

Conducting a post-mortem: How was the customer able to use a closed feature?

Jan de Bray, 1675

Worried he might be responsible...

Questions? @ygaller