Distributed Tracing
Understand how your components work together
About me

José Carlos Chávez

- Software Engineer at Typeform focused on the aggregate of responses services.
- Zipkin core team and open source contributor for Observability projects.
Distributed Systems
Distributed systems

A collection of independent components appears to its users as a single coherent system.

Characteristics:

- Concurrency
- No global clock
- Independent failures
Distributed systems
API Proxy

GET /media/e5k2

500 Internal Error

DB1

Images service

DB2

Videos service

DB3

Auth service

DB4

500 Internal Error

TCP error (2003)

Distributed systems: Understanding failures
Distributed systems: Understanding failures

I AM HERE! First floor distributor is clogged!
We do have that, it is called logs!
Logs & Concurrency

[24/Oct/2017 13:50:07 +0000] “GET /media HTTP/1.1” 200 ... **0/13548”
[24/Oct/2017 13:50:07 +0000] “GET /media HTTP/1.1” 200 ... **0/23948”
[24/Oct/2017 13:50:08 +0000] “GET /media HTTP/1.1” 200 ... **0/12396”
[24/Oct/2017 13:50:07 +0000] “GET /videos HTTP/1.1” 200 ... **0/23748”
[24/Oct/2017 13:50:07 +0000] “GET /images HTTP/1.1” 200 ... **0/23248”
[24/Oct/2017 13:50:07 +0000] “GET /auth HTTP/1.1” 200 ... **0/26548”
[24/Oct/2017 13:50:07 +0000] “POST /media HTTP/1.1” 200 ... **0/13148”
[24/Oct/2017 13:50:07 +0000] “GET /media HTTP/1.1” 200 ... **0/2588”
[24/Oct/2017 13:50:07 +0000] “GET /auth HTTP/1.1” 500 ... **0/3248”
[24/Oct/2017 13:50:07 +0000] “POST /media HTTP/1.1” 200 ... **0/23548”
[24/Oct/2017 13:50:07 +0000] “GET /images HTTP/1.1” 200 ... **0/22598”
[24/Oct/2017 13:50:07 +0000] “GET /videos HTTP/1.1” 200 ... **0/13948”
...
Distributed systems: Understanding failures

First floor distributor is clogged!
Distributed Tracing to unclog your pipes
Distributed tracing

API Proxy

Media API

[1508410442] no cache for resource, retrieving from DBc

Images

error

Auth

Videos

TraceID d52d38b69b0fb15efa

Error 500
Distributed Tracing: What answers do I get?

- What services did a request pass through?
- What occurred in each service for a given request?
- Where did the error happen?
- Where are the bottlenecks?
- What is the critical path for a request?
- Who should I page?
Distributed Tracing & friends

Credits: Peter Bourgon
Benefits of Distributed Tracing

- (almost) Immediate feedback
- System insight, clarifies non trivial interactions
- Visibility to critical paths and dependencies
- Understand latencies
- Request scoped, not request’s lifecycle scoped.
Trace’s Anatomy

- A **trace** shows an **execution path** through a distributed system.
- A **span** in the trace represents a logical **unit of work** (with a start and end).
- A **context** includes information that should be **propagated across services**.
- **Tags** and **logs** (optional) add complementary information to spans.
Elements of distributed tracing

Credits: Nic Munroe
Leg 1: Inbound propagation

When your service process a request or consume a message.

TraceID: fAf3oXL6DS
SpanID: dZ0xH1Ba1A
...
Leg 2: Outbound propagation

When your service makes an outbound call to another service

```
Media API

http/get

GET /videos

TraceID: fAf3oXL6DS
ParentID: dZ0xHIBa1A
SpanID: y74fr5udj

Video service
```
Leg 3: In process propagation

When performing an operation inside the service

```
GET /images

Media API

Images service

redis.Get

mysql.Query

Cache service
```
Distributed tracing

API Proxy

Media API

[1508410442] no cache for resource, retrieving from DBc

Images

error

TraceID d52d38b69b0fb15efa

Auth

Videos

Time

500
Any overhead?

For users:
- Observability tools are meant to be unintrusive
- **Sampling** reduces overhead
- *(Don’t)* trace every single operation

For developers:
- Not all libraries are ready to plug instruments
- Instrumentation can be delegated to **common frameworks**
Introducing Apache Zipkin
Apache Zipkin

Based on BigBrotherBird (B3) and inspired on Google Dapper (2010). It was open sourced by Twitter (2012) and joined Apache Incubator on September 2018.

- Mature tracing model emerged from users’ needs.
- Used by large companies like Netflix, SoundCloud and Yelp but also not too big ones.
- Strong community:
  - @zipkinproject
  - gitter.im/openzipkin
Zipkin: architecture
Zipkin: traces

Service Name
all

Span Name
all

Lookback
1 hour

Annotations Query
e.g. "http.path=/foo/bar/ and cluster=foo and cache.miss"

Duration (µs) >= 10

Limit
10

Sort
Newest First

Showing: 10 of 10
Services: all

331.522ms 14 spans
all 0%
api-proxy x4 280ms auth-service x2 64ms images-service x2 93ms media-api x3 188ms tracking-service x1 44ms videos-service x2 87ms

276.697ms 12 spans
all 0%
api-proxy x3 276ms auth-service x2 67ms images-service x2 99ms media-api x3 189ms videos-service x2 70ms
Zipkin: traces
Zipkin: traces
Zipkin: trace overview
Zipkin: tags and logs

<table>
<thead>
<tr>
<th>Date Time</th>
<th>Relative Time</th>
<th>Annotation</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.3.2018, 13:03:01</td>
<td></td>
<td>Server Receive</td>
<td>127.0.0.1:9423 (api-proxy)</td>
</tr>
<tr>
<td>20.3.2018, 13:03:02</td>
<td>262.361ms</td>
<td>Server Send</td>
<td>127.0.0.1:9423 (api-proxy)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>http.method</td>
<td>GET</td>
</tr>
<tr>
<td>http.status_code</td>
<td>200</td>
</tr>
<tr>
<td>http.url</td>
<td>/media/ZeOBC7</td>
</tr>
<tr>
<td>peer.hostname</td>
<td>ip-171-13-08-112.ec2.internal</td>
</tr>
<tr>
<td>request_id</td>
<td>78f13d22c2b5cd9c45971237ecd23c18</td>
</tr>
</tbody>
</table>
Zipkin: traces with errors
# Zipkin: traces for async operations

![Zipkin Traces Diagram]

- **Duration:** 322.627ms
- **Services:** 8
- **Depth:** 8
- **Total Spans:** 14

## Services

<table>
<thead>
<tr>
<th>Service</th>
<th>Time</th>
<th>Method</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>api-proxy</td>
<td>269.227ms</td>
<td>get</td>
<td>/media/{userid}</td>
</tr>
<tr>
<td>api-proxy</td>
<td>75.274ms</td>
<td>head</td>
<td></td>
</tr>
<tr>
<td>auth-service</td>
<td>64.266ms</td>
<td>head /auth/{oid}</td>
<td></td>
</tr>
<tr>
<td>auth-service</td>
<td>58.094ms</td>
<td>select</td>
<td></td>
</tr>
<tr>
<td>media-api</td>
<td>176.914ms</td>
<td>get</td>
<td>/media/{userid}</td>
</tr>
<tr>
<td>media-api</td>
<td>97.764ms</td>
<td>get</td>
<td></td>
</tr>
<tr>
<td>images-service</td>
<td>90.332ms</td>
<td>get /images/{userid}</td>
<td></td>
</tr>
<tr>
<td>images-service</td>
<td>84.246ms</td>
<td>select</td>
<td></td>
</tr>
<tr>
<td>media-api</td>
<td>73.760ms</td>
<td>get</td>
<td></td>
</tr>
<tr>
<td>videos-service</td>
<td>65.518ms</td>
<td>get /videos/{userid}</td>
<td></td>
</tr>
<tr>
<td>videos-service</td>
<td>57.950ms</td>
<td>select</td>
<td></td>
</tr>
<tr>
<td>api-proxy</td>
<td>45.942ms</td>
<td>post</td>
<td>/tracking/{req}</td>
</tr>
<tr>
<td>tracking-service</td>
<td>37.174ms</td>
<td>post</td>
<td></td>
</tr>
</tbody>
</table>

---
Zipkin: dependency graph
Zipkin: dependency graph
Q&As

twitter.com/jcchavezs