Distributed Tracing

- From Theory to Practice -
What is distributed tracing?
Tracing requests across distributed system boundaries
A Simple Use Case
Monolithic Web Process

- Authentication
- User
- Orders
- Items

Web Request
Web Request

Auth Process

Ecommerce Process

User

Orders

Items

New app

Ruby app

User

Orders

Items

Web Request

Auth Process

Ecommerce Process

New app

Ruby app
Web Request

Auth Process

Ecommerce Process

Orders
Recommendaions
Billing

Original Ruby app

Python???
Microservices!

(j/k)
Services

(Micro or otherwise)
<insert container joke>
Why do we need distributed tracing?
Internal services look like external APIs
Web Request  Why is this slow ????

Auth Process

Ecommerce Process

Orders  Recommendations  Billing

Blame data science?
“You can’t tell a coherent macro story about your application by monitoring individual processes”

Ben Seligman
People are bad guessers
How do you tell the story?
Distributed Tracing!
“Distributed tracing commoditizes knowledge”

- Adrian Cole
What’s Stopping You?
Outside Your Language’s Wheelhouse
Domain Specific Vocabulary
Fractured Ecosystem
Theory -> Practice
The Basics
Black Box Tracing
Black Box Tracing
Why might this not work for you?
• Need lots of data
• Delayed results
• Can’t guarantee causality
def my_cool_system
  service_1
  service_2
end

def service_1
  Rails.logger "Service 1"
  execute_async_job
end

def execute_async_job
  Rails.logger "Async Job"
end

def service_2
  Rails.logger "Service 2"
end

Aggregated Log

01-01-2001 01:01:01 Service 1
01-01-2001 01:01:02 Async Job
01-01-2001 01:01:03 Service 2
def my_cool_system
    service_1
    service_2
end

def service_1
    Rails.logger "Service 1"
    execute_async_job
end

def execute_async_job
    sleep 15
    Rails.logger "Async Job"
end

def service_2
    Rails.logger "Service 2"
end

Aggregated Log

01-01-2001 01:01:01 Service 1
01-01-2001 01:01:02 Service 2
01-01-2001 01:01:17 Async Job
White Box Tracing
Metadata Propagation
Realtime Analysis
History Lesson
Dapper
Zipkin
“Distributed Tracing”
“So, you want to trace your distributed system? Key design insights from years of practical experience”

Raja R. Sambasivan, Rodrigo Fonseca, Ilari Shafer, Gregory R. Ganger

Tracing Your Applications
Main Components:

- Tracer
- Transport
- Collector
- Storage
- UI
Tracing Requests
Tracer:
Lives inside your apps, does the tracing
Trace:
The story of a request’s journey through your system
A trace tells this whole story
Span:
Each chapter in that story
A Trace is many Parent - Child Relationships
Directed Acyclic Graph
Annotations:
Gives us richer insights into our spans
<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Start</td>
<td>01:01:01</td>
</tr>
<tr>
<td>Server Receive</td>
<td>01:01:02</td>
</tr>
<tr>
<td>Server Send</td>
<td>01:01:03</td>
</tr>
<tr>
<td>Client Receive</td>
<td>01:01:04</td>
</tr>
</tbody>
</table>
stella cotton | @practice_cactus

Auth

Client
Send

Client
Receive

Server
Receive

Server
Send

Ecommerce
Transporting the Data
Propagates ids in band
Reports out of band
Viewing the Data
All happening inside the “resource allocation & provisioning”
A widening gap here could indicate queueing.
Tracing Incoming Requests
Middleware
def call(env)
    trace do
        @app.call(env)
    end
end

def trace(env, &block)
    # tracing code
end
def call(env)
    trace do
        @app.call(env)
    end
end

def trace(env, &block)
    span = Span.new("authentication", generate_span_id)
    span.record(SERVER_RECV)  # Received a request
    status, headers, body = yield  # Execute the app
    ensure
        span.record(SERVER_SEND)  # Sending back to the client
    end
end

Non-pseudocode version:
# config/initializers/tracing.rb

```ruby
Rails.application.config.middleware.use TracingRackMiddleware, {
  # some configuration
}
```

Use our middleware!
# config/initializers/tracing.rb

```ruby
Rails.application.config.middleware.use TracingRackMiddleware, {
  service_name: "SERVICE_DOMAIN_NAME",
  service_port: 443,
  sample_rate: ENV.fetch("ZIPKIN_SAMPLE_RATE", 0.1).to_f,
  json_api_host: ENV["ZIPKIN_HOST"]
}
```

Sample a portion of requests
Tracing Outgoing Requests
More Middleware!
Faraday
def call(env)
    trace!(env) do |env|
        @app.call(env)  # Execute our http client
    end
end

def trace!(env, &block)
    # some tracing
end
def call(env)
    trace!(env) do |env|
        @app.call(env)
    end
end

def trace!(env, &block)
    env = set_headers(env)
    span = Span.new("external_call", 1234)
    span.record(Trace::Annotation::CLIENT_SEND)
    status, headers, body = yield env
    ensure
        span.record(Trace::Annotation::CLIENT_RECV)
    end
end
Each of these colors represents an instrumented application.
def call(env)
    trace!(env) do |env|
        @app.call(env)
    end
end

def trace!(env, &block)
    env = set_headers(env)
    span = Span.new("external_call", 1234)
    span.record(Trace::Annotation::CLIENT_SEND)  # Client Send
    status, headers, body = yield env
    ensure
        span.record(Trace::Annotation::CLIENT_RECV)  # Client Receive
    end
end
def self.client
    Faraday.new(url: base_url) do |connection|
        connection.use TracingFaradayMiddleware
        connection.adapter Faraday.default_adapter
    end
end

Add our middleware
Buy, Build, or Adopt
Buy?
Lightstep
TraceView...
and more?
Adopt an OSS Solution?
Zipkin
Jaeger
What about Open Tracing?
Standardizes Instrumentation
Where is OpenTracing at today?
Interoperability is Still Messy
“Language Support”
Rinse and Repeat
Build Your Own?
What are other folks doing?
End-to-End Tracing: Adoption and Use Cases

Jonathan Mace, Brown University

• 15 using Zipkin
• 9 using internal solutions
• 1 using other OSS solution
• 1 using paid solution

Jonathan Mace, Brown University
Infra Requirements and Limitations
Dependency matrix of:
- Tracer
- Transport Layer
- Collection Layer
- Storage Layer
Installing a Separate Agent
Authentication
Missing Authentication & Authorization
Client Authorization
Basic auth via htpasswd

https://www.nginx.com/resources/admin-guide/restricting-access-auth-basic/
API Client

Nginx (SSL Termination) $PORT

Basic Authorization?

yes

Zipkin:9411

by Corey Donohoe
Browser Authentication
bit.ly’s Oauth2 proxy

https://github.com/bitly/oauth2_proxy
Browser

Nginx (SSL Termination) $PORT

Authorized Heroku?

no → oauth2_proxy:1480

yes

Zipkin:9411

by Corey Donohoe
Giving people access
Sensitive Data
Custom Instrumentation
What happens when data leaks?
Is Everyone On Board?
Get it on the Roadmap
Open PRs
Evaluating Distributed Tracing Solutions:

- Should you buy, build or adopt?
- What are your infrastructure requirements and limitations?
- How is it authenticated?
- Do you have sensitive data? What will you do if it leaks?
- Is everyone on board?
OMG, this is so much information
Try out Docker Zipkin
Thank you!