Distributed Tracing with Jaeger

July 18, 2019

Who Am I?

- **Name** : M Zulfa Achsani
- **Race** : Human
- **Birth** : June 13, 1994
- **Died** : Not specified
- **Role** : Software / Machine Learning Engineer
- **Experience** :
  - 1 year in a corporate as a App Dev
  - 2.5 years in a “green” e-commerce as Backend Engineer for Ads platform
  - Now working at Traveloka
- **Social Media :**
  - **LinkedIn** : https://www.linkedin.com/in/m-zulfa-achsani-72b6b8106/
  - **Github** : github.com/misterciput
  - **Instagram** : @zulfa.achsani

What is Tracing?

- **Cambridge Dictionary:**

  A copy of a drawing or pattern made by drawing over it through a piece of thin, transparent paper

- **Criminology:**

  A subject that aims to determine crime scene activity from trace evidence left at crime scenes

- **Software Engineering:**

  Tracing involves a specialized use of logging to record information about a program's execution

Why we do tracing?

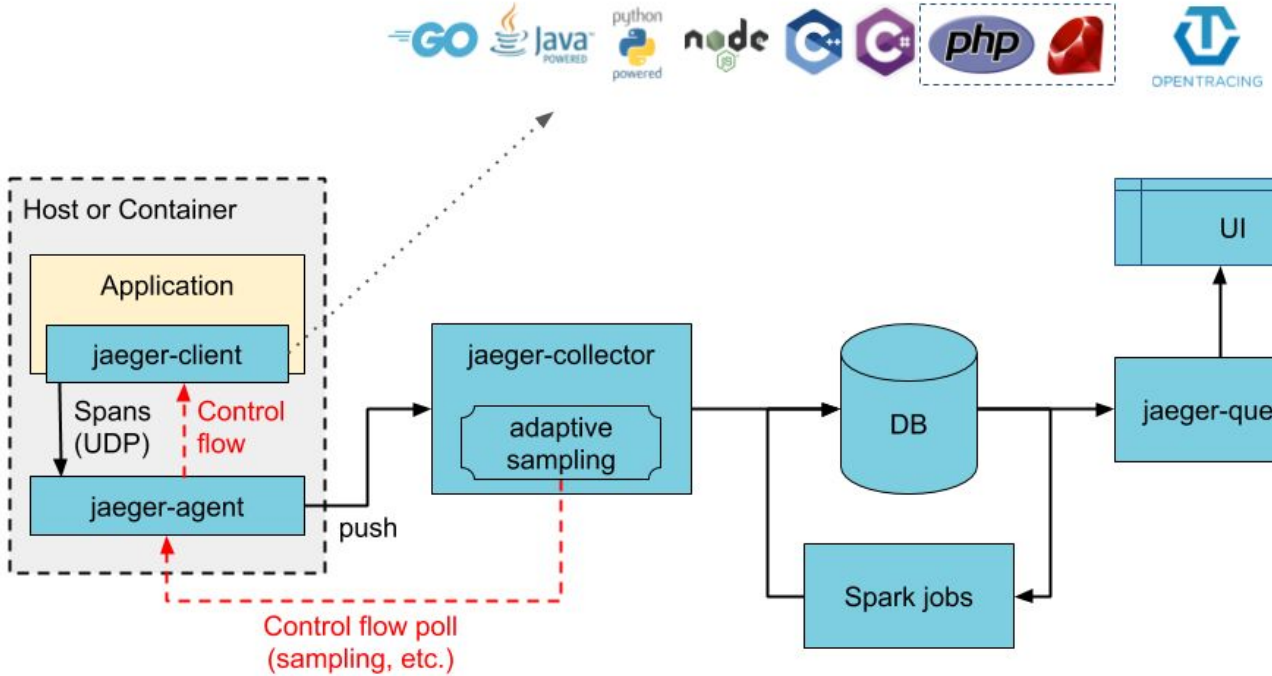
- **Faster Troubleshooting**
- **Better Software Quality**
- **Better Performance**
- **Control system tuning**

How we do tracing?

Jaeger: open source, end-to-end distributed tracing

- Monitoring and troubleshooting microservices-based distributed systems
- Distributed context propagation
- Native support for Opentracing APIs (span, tags, logs)
- Supported client libraries: Go, Java, Python, NodeJS, C++, C#

Jaeger Architecture

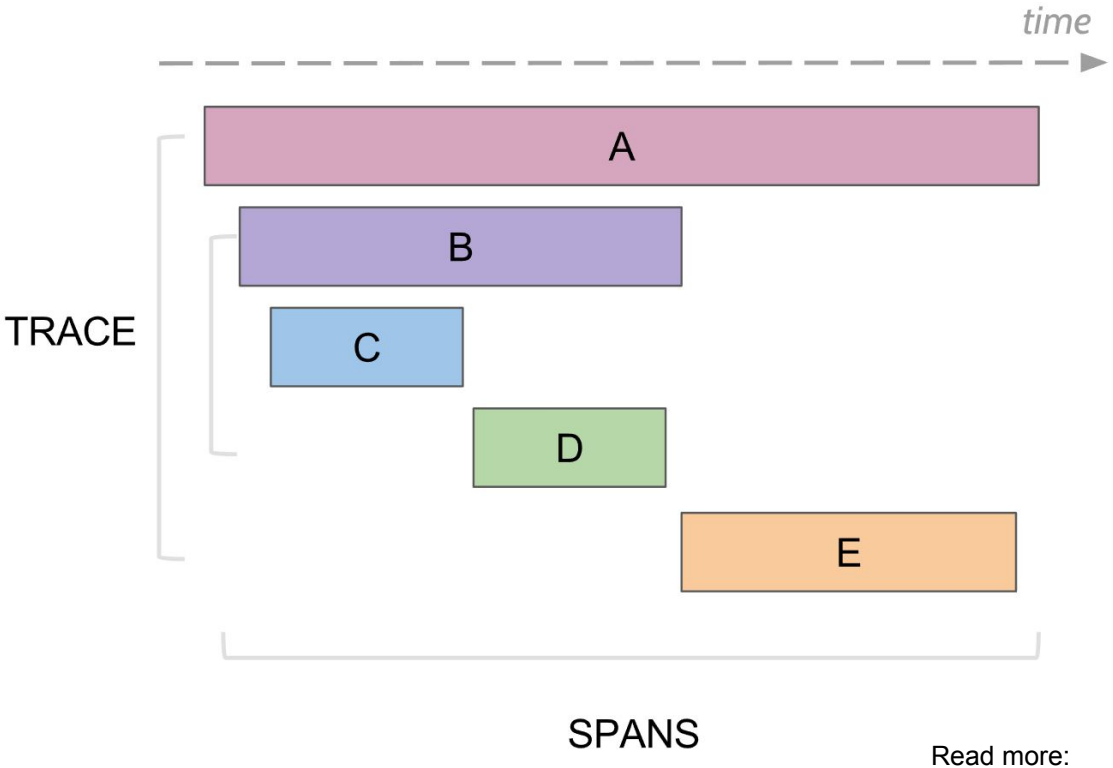


Jaeger Architecture

- JAEGER_SERVICE_NAME
- JAEGER_AGENT_HOST
- JAEGER_AGENT_PORT
- JAEGER_SAMPLER_TYPE
- JAEGER_SAMPLER_PARAM
- JAEGER_REPORTER_MAX_QUEUE_SIZE

```
func getMerchant(ctx context.Context) Merchant {
    span, ctx := opentracing.StartSpanFromContext(ctx, "getMerchant")
}
```

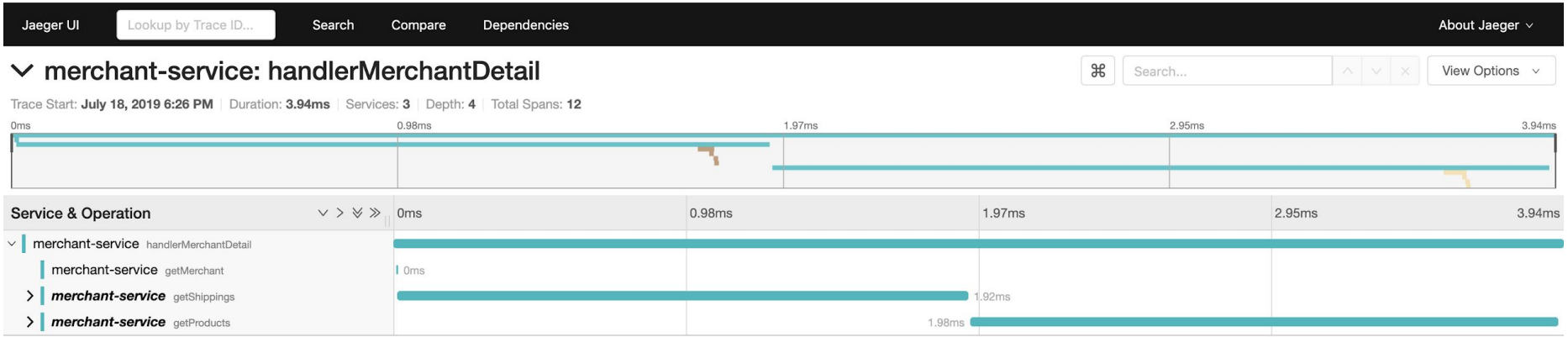
OpenTracing API



Read more:

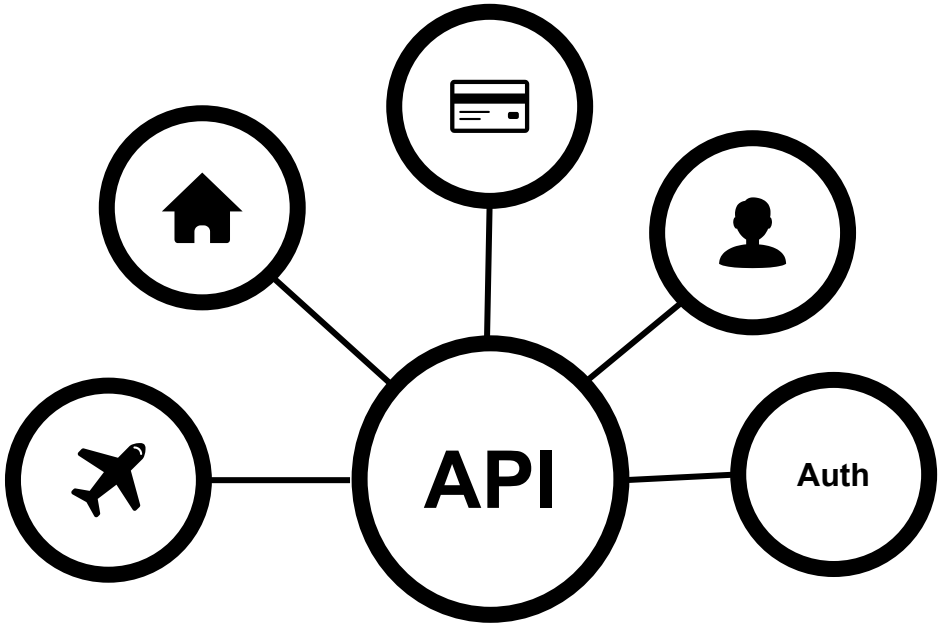
https://opentracing.io/specification/

Jaeger Trace

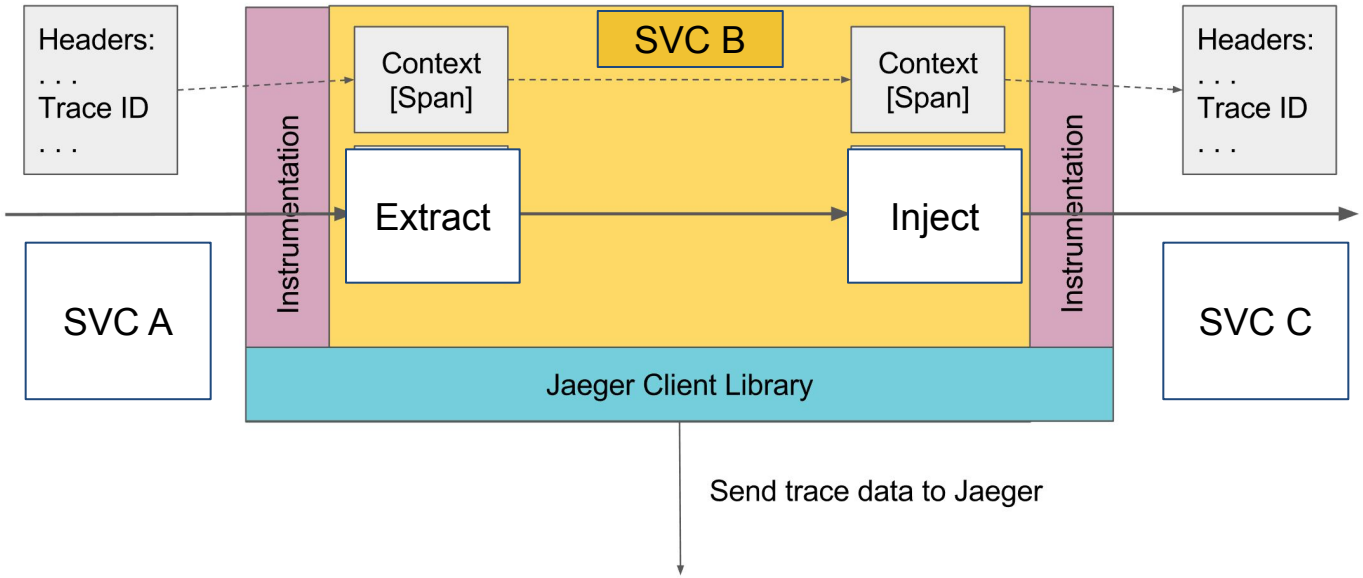


Distributed Tracing

Distributed tracing, also called distributed request tracing, is a method used to profile and monitor applications, especially those built using a microservices architecture. Distributed tracing helps pinpoint where failures occur and what causes poor performance.



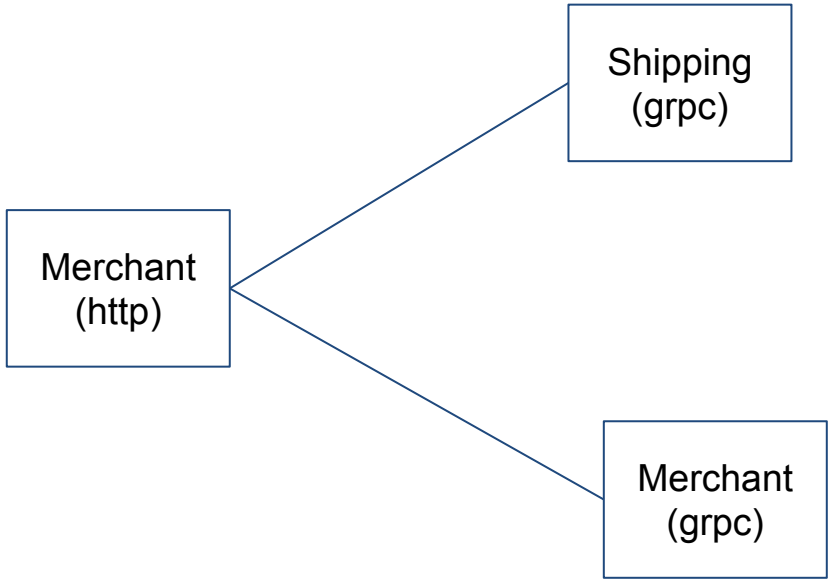
Distributed Tracing



Distributed Tracing

- Injection
- Extraction
- Carrier
  - Text Map
  - HTTP Headers
  - Binary

Example Use Case



https://github.com/misterciput/distributed-tracing-example

Conclussion

- Jaeger and Opentracing is a new way of debugging
- Able to trace end-to-end microservice
- Easy to implement

Question?