POLYTECHNIQUE MONTRÉAL

# Tracing and Performance Analysis of Modern Distributed Applications

Progress Report Meeting - May 10, 2018

DORSAL

Ecole polytechnique de Montréal

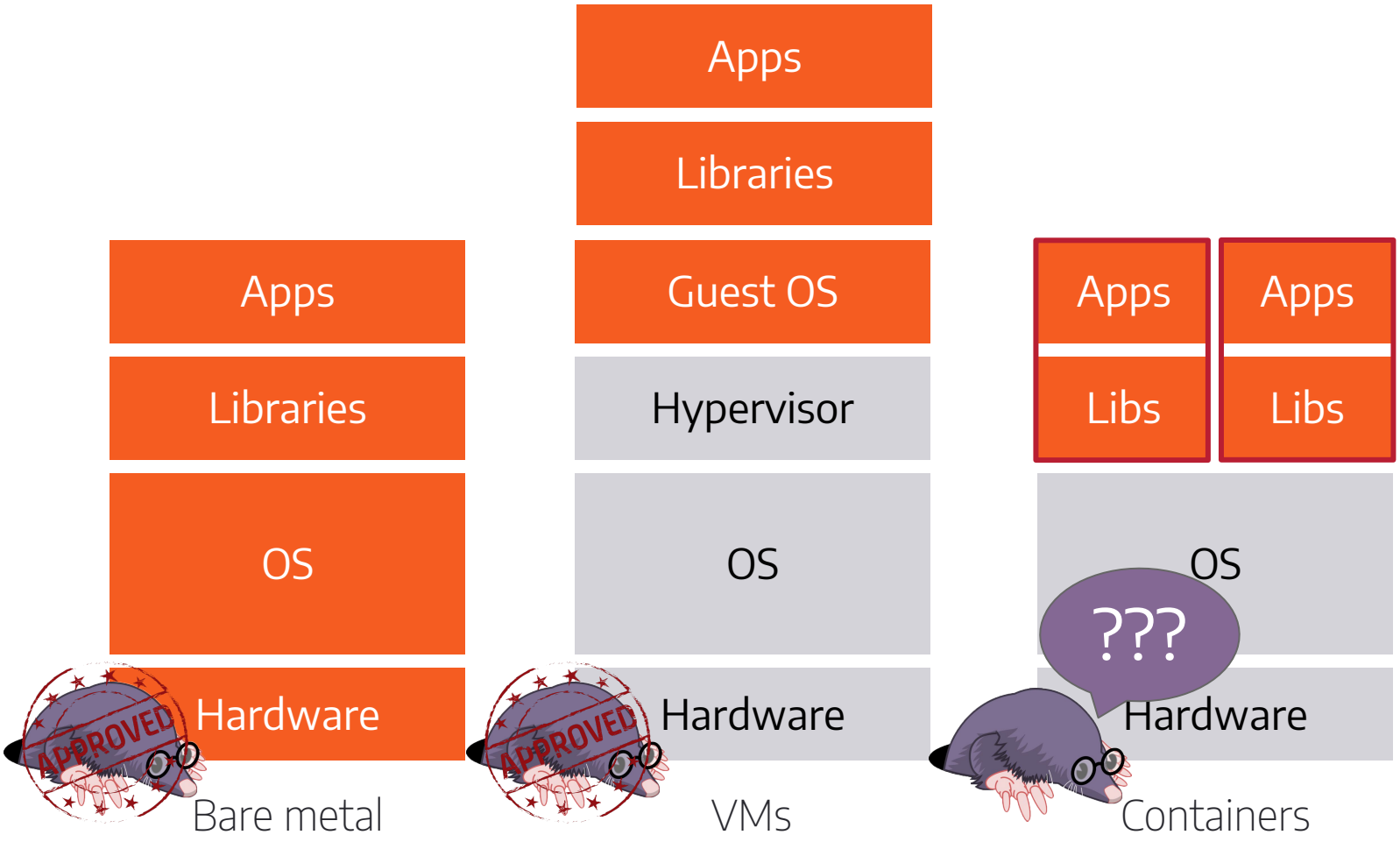
Loïc Gelle

Michel Dagenais

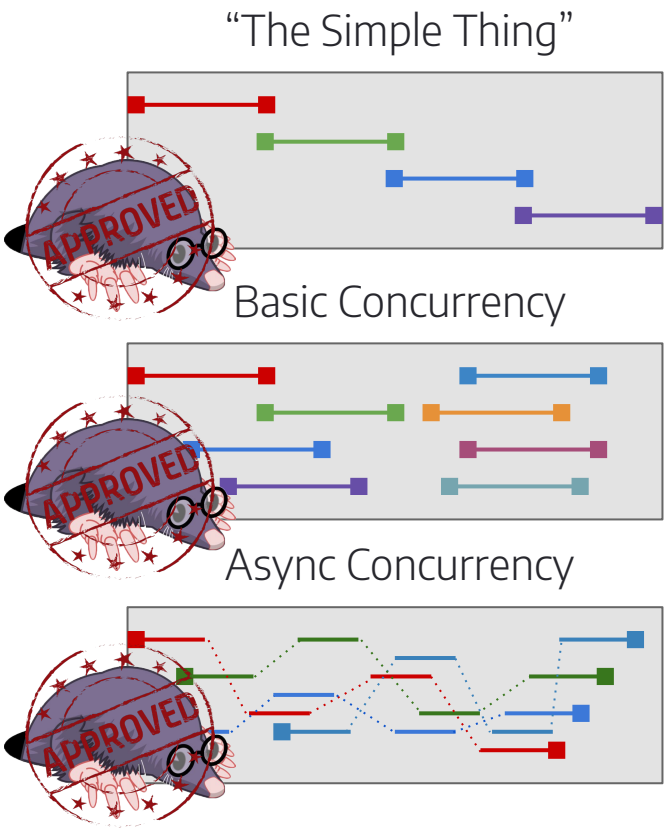
# Context

The evolution of computer systems

# From bare metal to VMs to Containers



# From monoliths to microservices



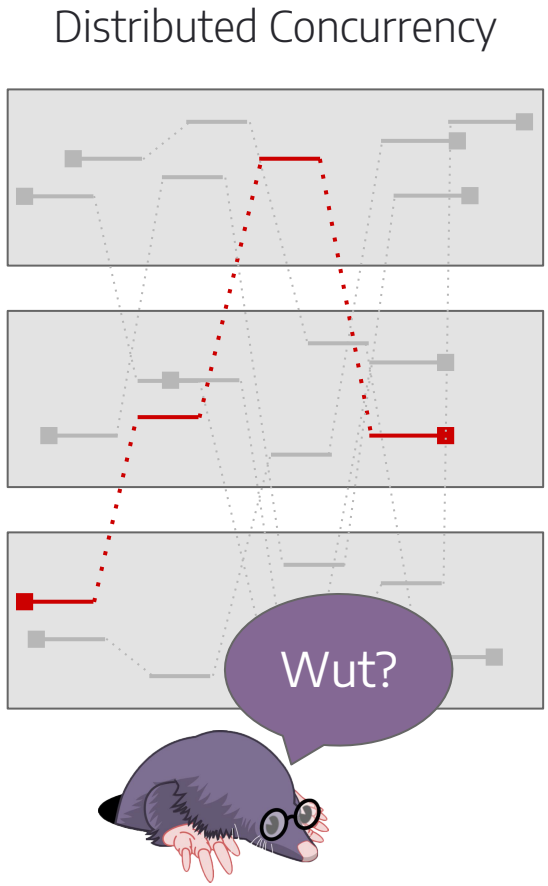




Illustration courtesy of Ben Sigelman

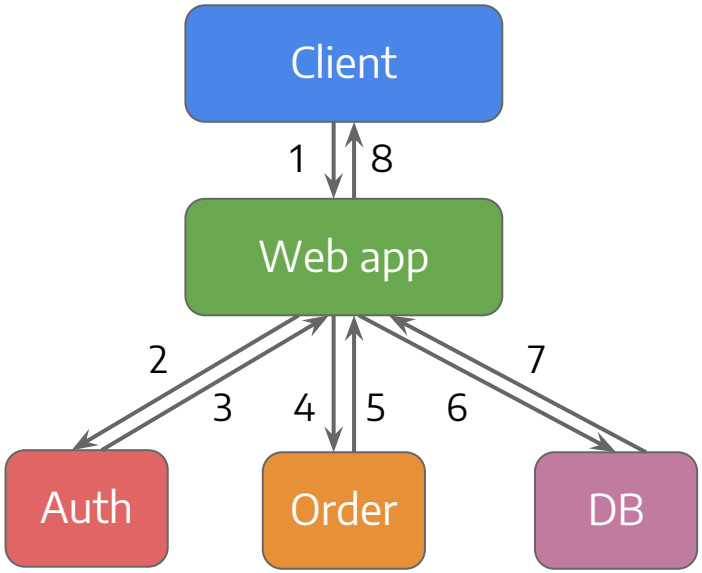

Distributed tracing to the rescue

# Key facts about OpenTracing

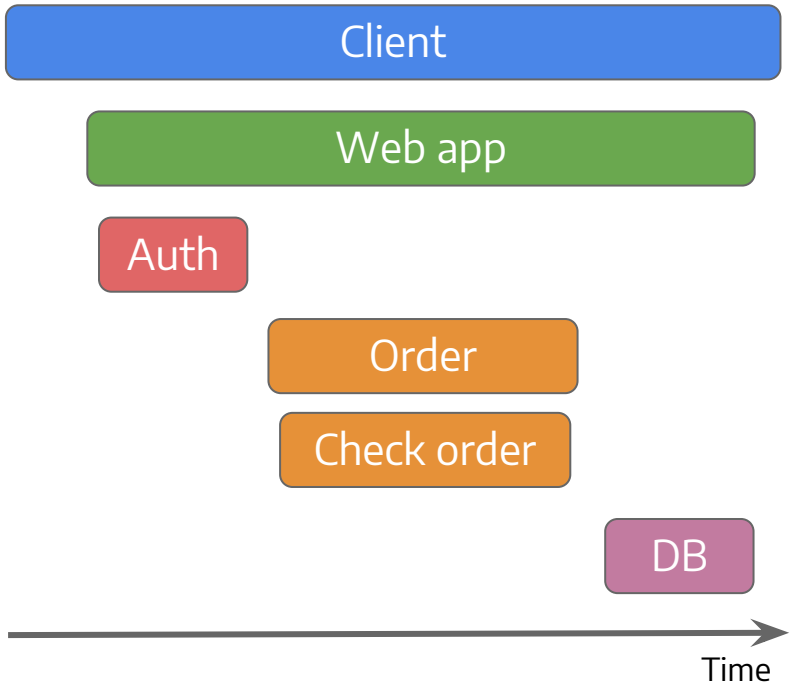
- An open-source **specification for distributed tracing**
- A **vendor-neutral API** for instrumenting libraries
  - API available for **popular languages** like Java, Go, C++, Python…
  - Lots of **libraries** like gRPC, NodeJS… are instrumented
- Many tracers (Jaeger, OpenZipkin, LightStep…) implement the OpenTracing specification
  - OpenTracing **leaves implementation details** to the tracers
  - Each tracer has **different purposes and analyses / UI**

# Describing complex transactions

OpenTracing focuses on describing **tasks** instead of events.

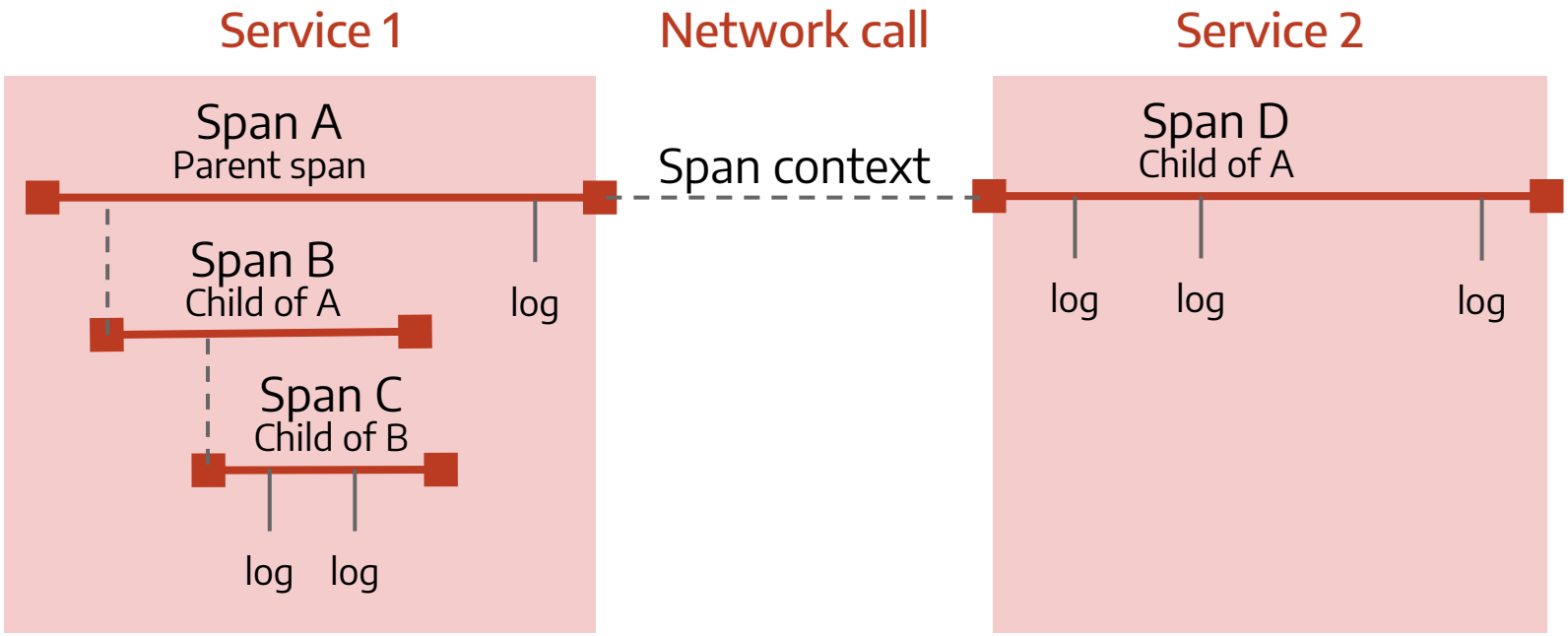



What the transaction looks like




What the trace looks like

# Key concepts in OpenTracing




- A **span** has a name, a start, a duration, tags and attached logs.
- The **span context** identifies the trace; it is injected into requests.
- A **trace** is the recording of the whole transaction using the above!

# Enough theory

Let's see how a distributed trace looks like using **Jaeger**.

Use your laptop / cell phone / connected watch to go to

**secretaire.dorsal.polymtl.ca:8081**

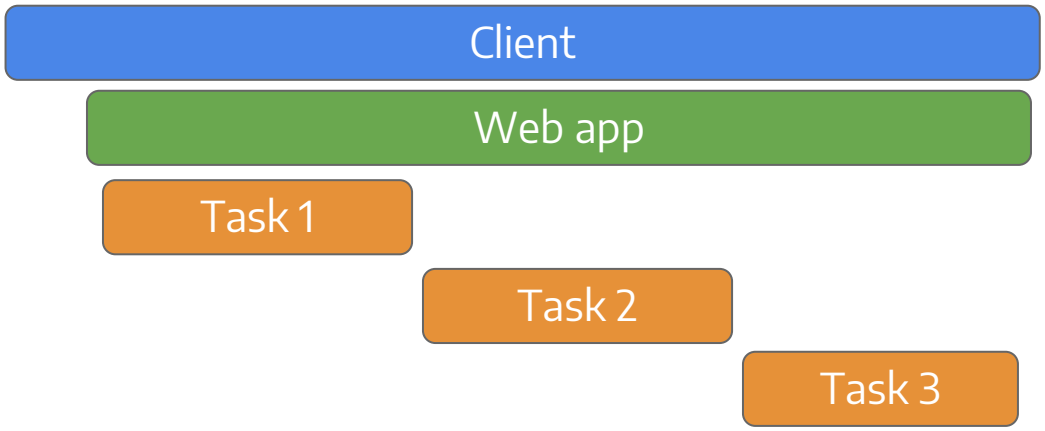
and have fun clicking everywhere.

Source of the demo app: *Take OpenTracing for a HotROD ride*, Yuri Shkuro
https://medium.com/opentracing/take-opentracing-for-a-hotrod-ride-f6e3141f7941

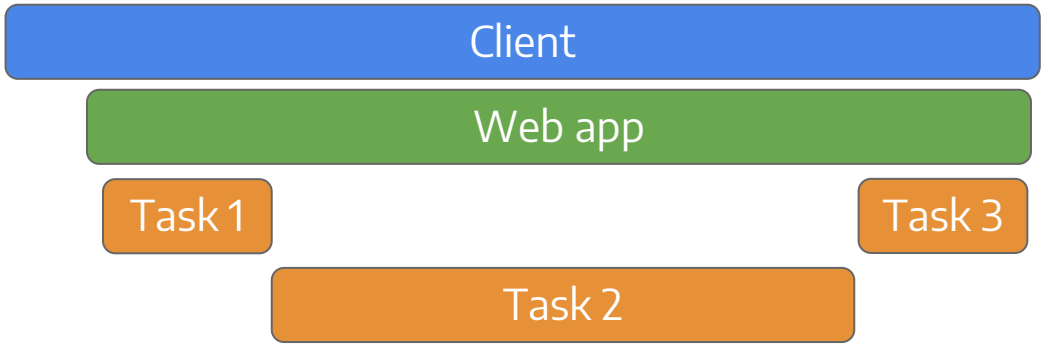
# New investigations

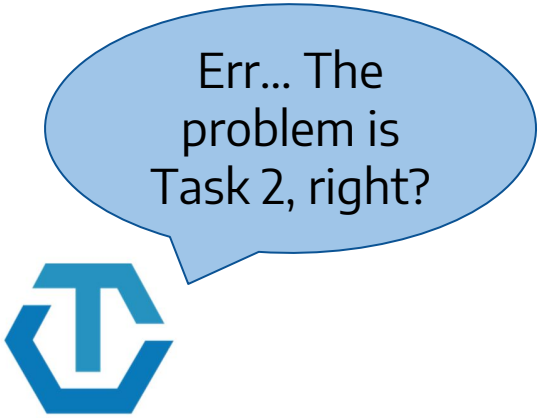
Objectives and future work

# Where does OpenTracing fail?



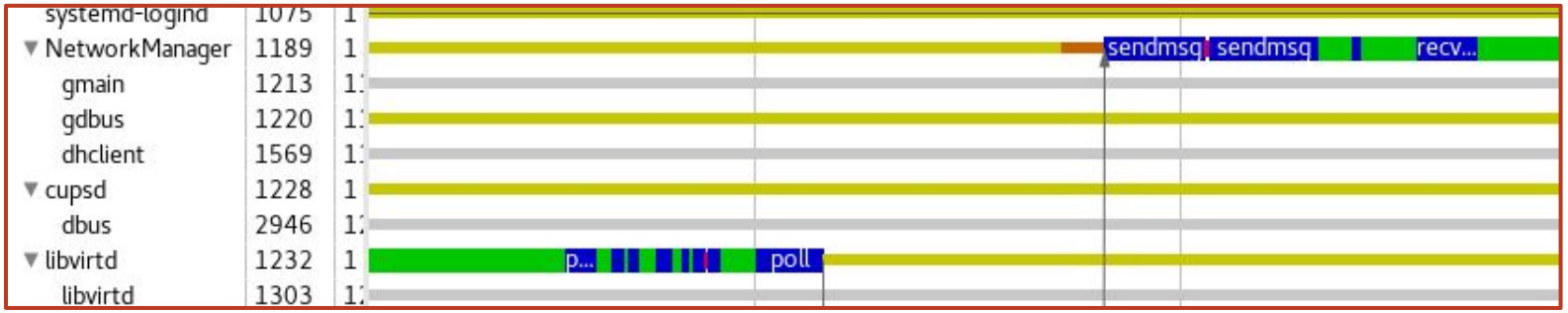




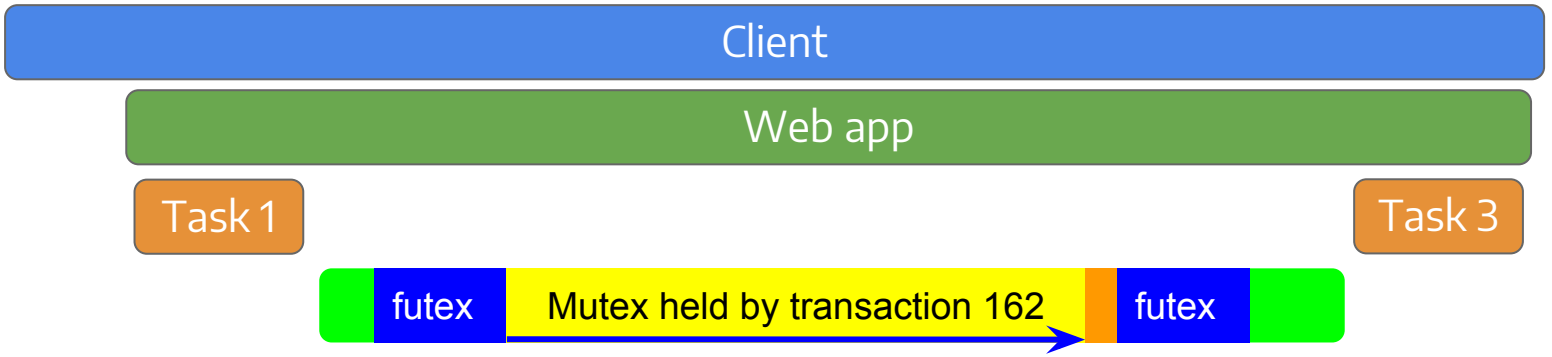
# What do we want?

- A solution for debugging complex problems
  - Mutex or I/O or network contention
  - Other subtle bottlenecks
- But... it does exist, right?

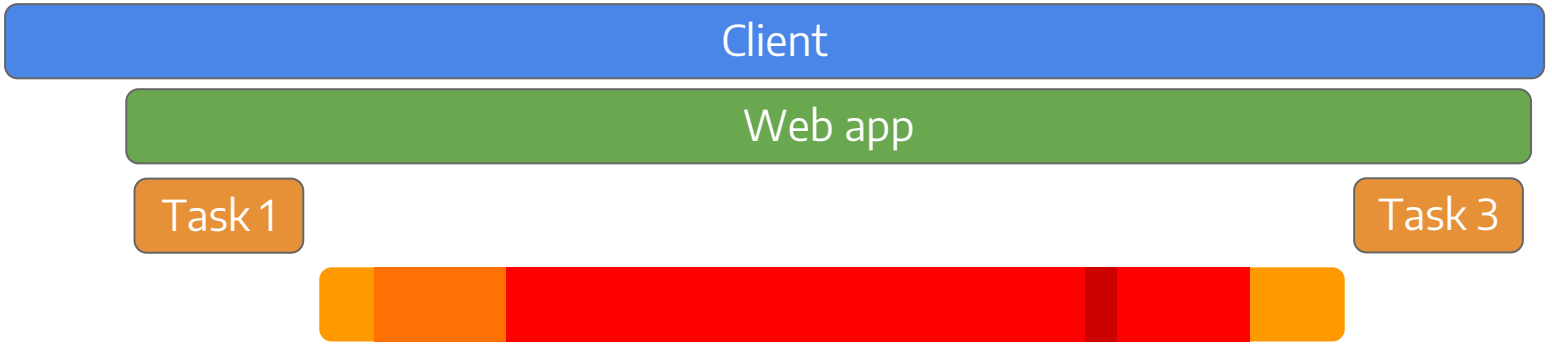


# The best of both worlds

Mutex contention analysis




Density of I/O events



# The long and winding road...

- Integrating LTTng traces into OpenTracing is not easy
  - Concept of spans vs. concepts of events
  - LTTng says **threads**, OpenTracing says **tasks**...
  - How to **synchronize** precisely the traces?
- Our tools do not fully **support containers**
  - Track containers creation and destruction (WIP)
  - Capture events from within containers

# Summarizing the objectives

- Develop **container-aware tracing** using LTTng
- **Joint analysis** of LTTng and OpenTracing traces
- Design **specific analyses** for distributed transactions
  - We can use the TraceCompass backend!
- Propose and implement a **workflow** that would integrate well with the OpenTracing ecosystem

POLYTECHNIQUE MONTRÉAL

# Thank you!

Questions, ideas, remarks?



loic.gelle@polymtl.ca


Github: @loicgelle