Distributed tracing with Jaeger

Closing dev-ops gap in performance analysis

Outlines

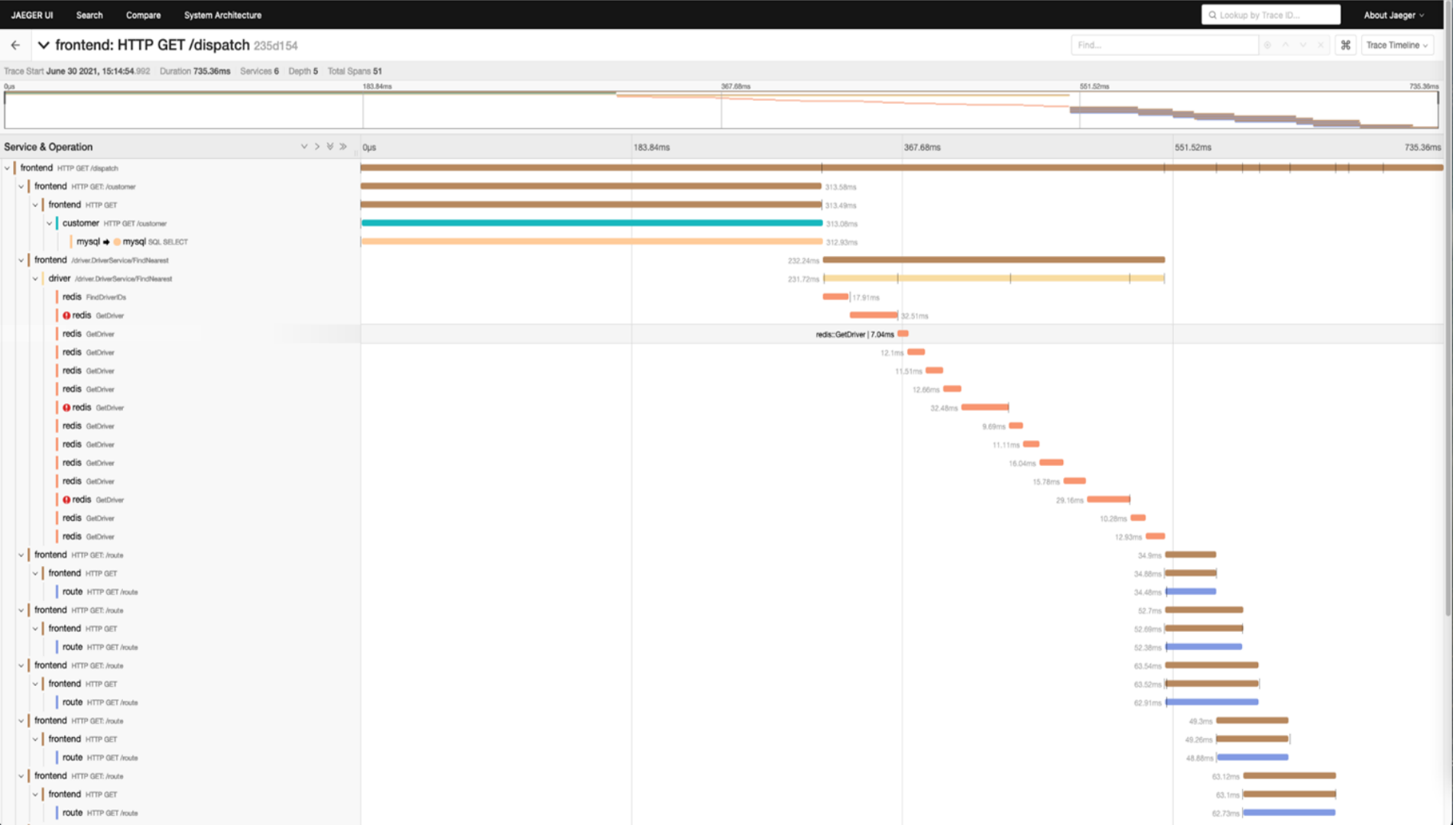
- What is distributed tracing
- Why do we need it
- Opentracing: terms and concepts
- Jaeger implementation
- Live coding demo: write simple apps with distributed tracing libraries

Distributed tracing

- Tracing activity which crosses the barrier between processes
- End-to-end tracing technology, especially for microservices
- Collect details about events from different services
- Put them in the same context (a trace) of request execution flow
- Form the causality relationship between services
- To provide a complete picture of your distributed applications

Why distributed tracing ?

- 1 request spreads through multiple services
- A need for step-by-step instrumentation over the network
- Context passing and merging
- No-root, language-agnostic implementation across services
- Inspect your services like a browser’s requests
- Service dependencies visualization

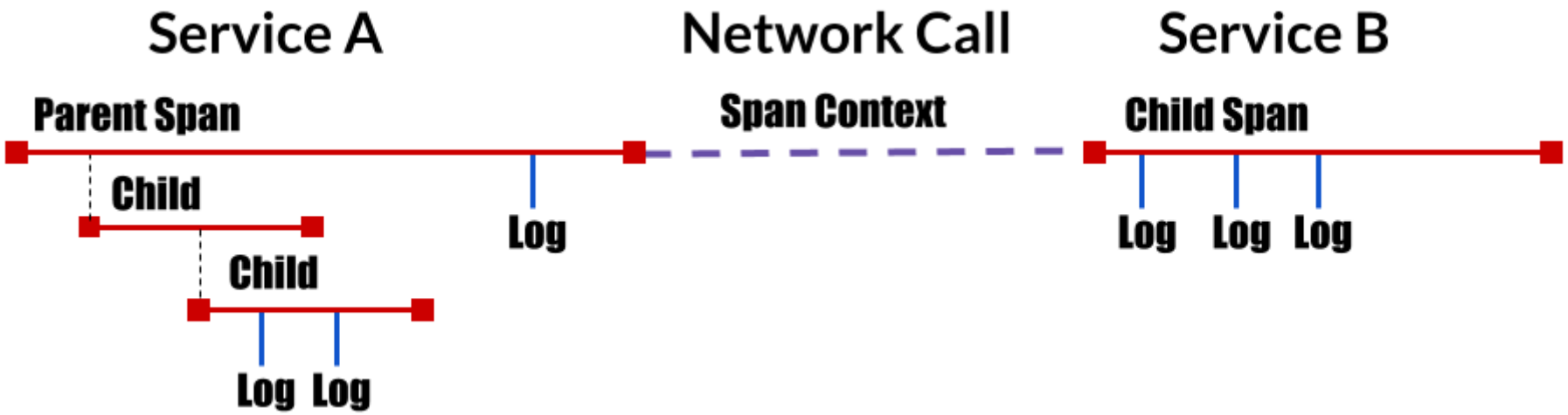

JAEGER UI
Search
Compare
System Architecture
Lookup by Trace ID...
About Jaeger
frontend: HTTP GET /dispatch 235d154
Find...
Trace Timeline
Trace Start June 30 2021, 15:14:54.992 Duration 735.36ms Services 6 Depth 5 Total Spans 51
0µs
183.84ms
367.68ms
551.52ms
735.36ms
Service & Operation
frontend HTTP GET /dispatch
frontend HTTP GET: /customer
frontend HTTP GET
customer HTTP GET /customer
mysql mysql SQL SELECT
frontend /driver.DriverService/FindNearest
driver /driver.DriverService/FindNearest
redis FindDriverIDs
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
frontend HTTP GET: /route
frontend HTTP GET
route HTTP GET /route
frontend HTTP GET: /route
frontend HTTP GET
route HTTP GET /route
frontend HTTP GET: /route
frontend HTTP GET
route HTTP GET /route
frontend HTTP GET: /route
frontend HTTP GET
route HTTP GET /route
frontend HTTP GET: /route
frontend HTTP GET
route HTTP GET /route
313.58ms
313.49ms
313.08ms
312.93ms
232.24ms
231.72ms
17.91ms
32.51ms
redis::GetDriver | 7.04ms
12.1ms
11.51ms
12.66ms
32.48ms
9.69ms
11.11ms
16.04ms
15.78ms
29.16ms
10.28ms
12.93ms
34.9ms
34.88ms
34.48ms
52.7ms
52.69ms
52.38ms
63.54ms
63.52ms
62.91ms
49.3ms
49.26ms
48.88ms
63.12ms
63.1ms
62.73ms

Opentracing

An standardization effort for distributed tracing system, which define:

- tracing instrumentation API:
  - Tracer initialization
  - Inject / Extract
- Wire protocol: format of data sent along side with app’s data (trace context)
  - IDs
  - References
  - Tag, Log , Baggage
- Data protocol: propagation msg format , transport protocol, encodes

Opentracing terms



- A library that enables observability for another library is called an instrumentation library
- API / SDK
- Collector + agent for data gathering
- Backend tracing: for data processing and visualization

~~Three pillars~~ Triangle of observability

- Traces
- Metrics
- Logs

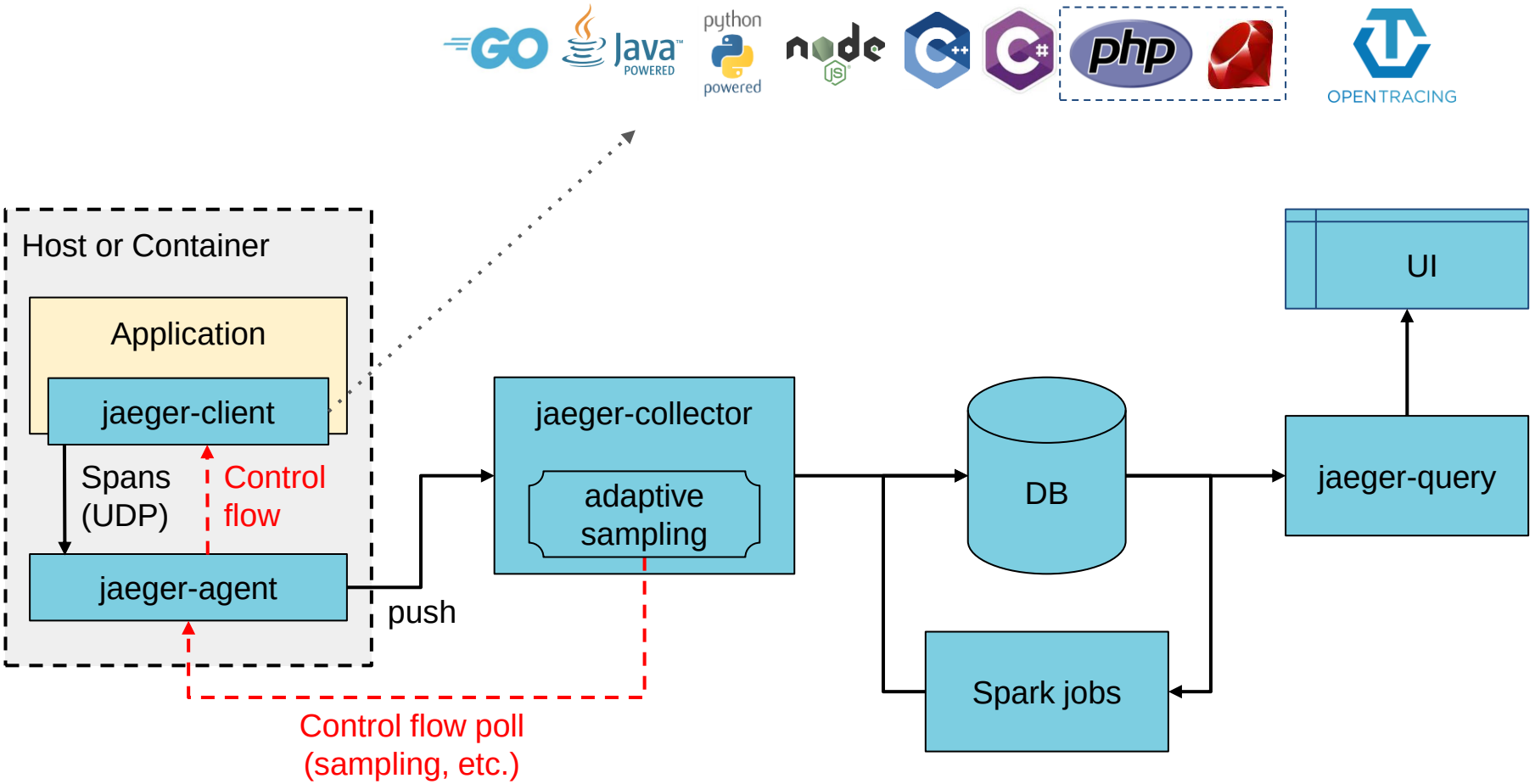
API vs SDK

- If you’re developing a library or some other component that is intended to be consumed by a runnable binary, then you would only take a dependency on the API.
- If your artifact is a standalone process or service, then you would take a dependency on the API and the SDK.

Technology Stack

- Go backend
- Pluggable storage
  - Cassandra, Elasticsearch, memory, ...
- React/Javascript frontend
- OpenTracing Instrumentation libraries
- Integration with Kafka, Apache Flink

OPENTRACING


Host or Container
Application
jaeger-client
Spans
(UDP)
Control
flow
jaeger-agent
push
jaeger-collector
adaptive
sampling
DB
UI
jaeger-query
Spark jobs
Control flow poll
(sampling, etc.)
OPENTRACING

Ecosystem and alternatives

Distributed backend tracing solution:

- Origin: dapper
- Opensource: zipkin, skywalking
- Enterprise solutions: lightstep, stackdriver, aws x-ray

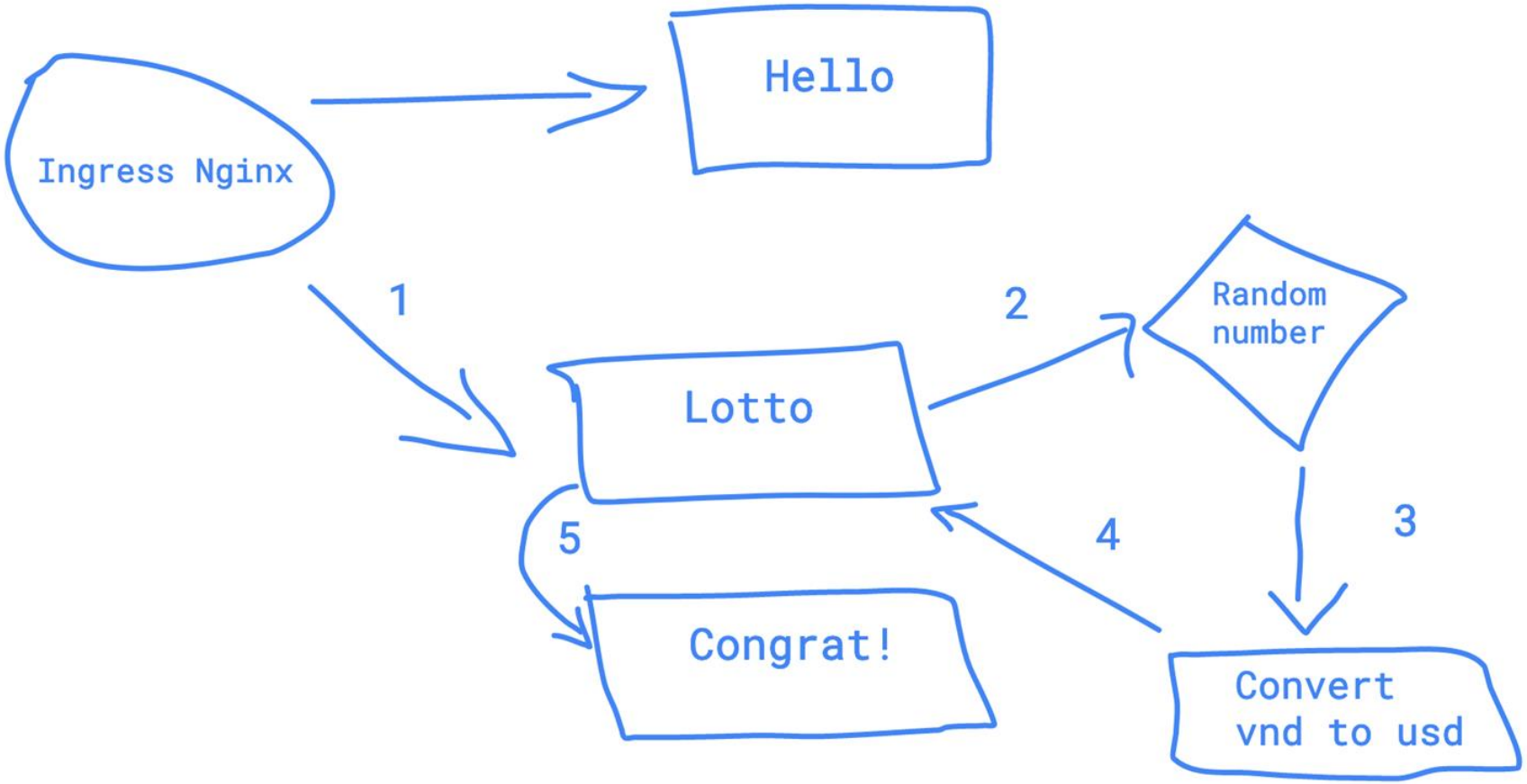
Distributed tracing API / SDK/ specification:

- Opentracing
- Opentelemetry
- Opencencus

Vendors

- https://opentelemetry.io/vendors/

Live demo


Hello
Ingress Nginx
1
2
Random number
Lotto
5
4
3
Congrat!
Convert vnd to usd

Write simple apps armed with distributed tracing

- Prepare trace initialization config
- Start a tracer
- Start a span
- Passing span inside process manually
- Use active span to automate context passing
- Tag, log, baggage
- Context propagation between services

Challenges

- Need to update codes
- Convincing developers is hard
- Performance of o11y instrumentation libraries and infrastructure
- Mature but fast changing community

Culture shift required

- Strong dev/ops collaboration
- Observability and performance mindset

Future plans

- High throughput with kafka
- Smart adaptive sampling mechanism
- Integration with existing metrics (gr, prom) to provide “context-rich” paging msg

References

- https://opentracing.io/docs/
- https://www.jaegertracing.io/
- https://www.shkuro.com/books/2019-mastering-distributed-tracing/
- https://kubernetes.github.io/ingress-nginx/user-guide/third-party-addons/opentracing/
- https://github.com/opentracing/opentracing-python
- https://opentelemetry.io/docs/concepts/
- https://medium.com/jaegertracing/jaeger-and-opentelemetry-1846f701d9f2