CLOUD NATIVE
COMPUTING FOUNDATION

Jaeger
SIG Deep Dive Session

Pavol Loffay (Red Hat), Yuri Shkuro (Uber)

CloudNativeCon NA, Austin, Dec-7-2017

Agenda

- Demo
- Architecture, data model
- Configuration
- Zipkin drop-in replacement
- Roadmap
  - Path based dependency diagrams
  - Adaptive sampling
- Project governance, public meetings, contributions
- Discussion

CLOUD NATIVE
COMPUTING FOUNDATION

Let's look at some traces

demo time: http://bit.do/jaeger-hotrod

Distributed Tracing Systems

distributed transaction monitoring

performance and latency optimization

root cause analysis

service dependency analysis

distributed context propagation

CLOUD NATIVE
COMPUTING FOUNDATION

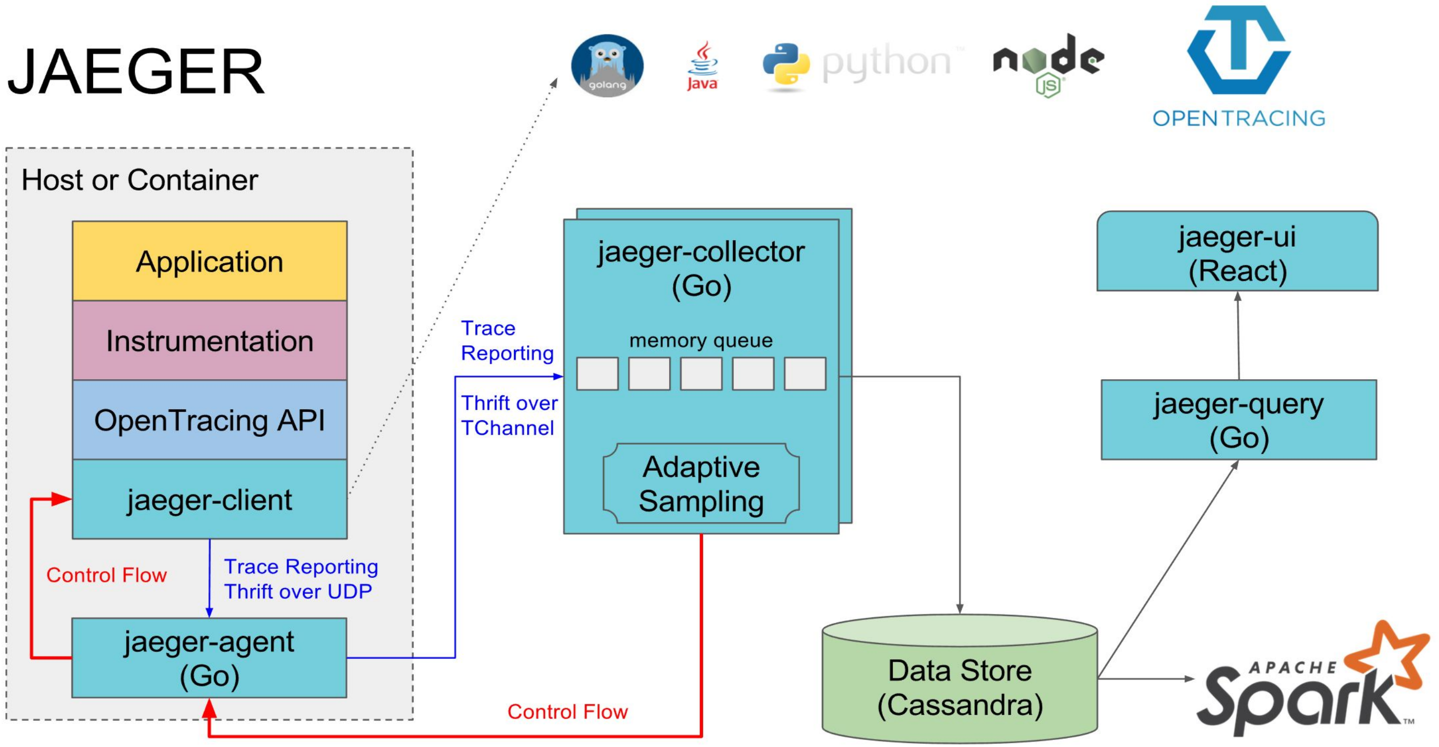
Under the hood

Architecture, etc.

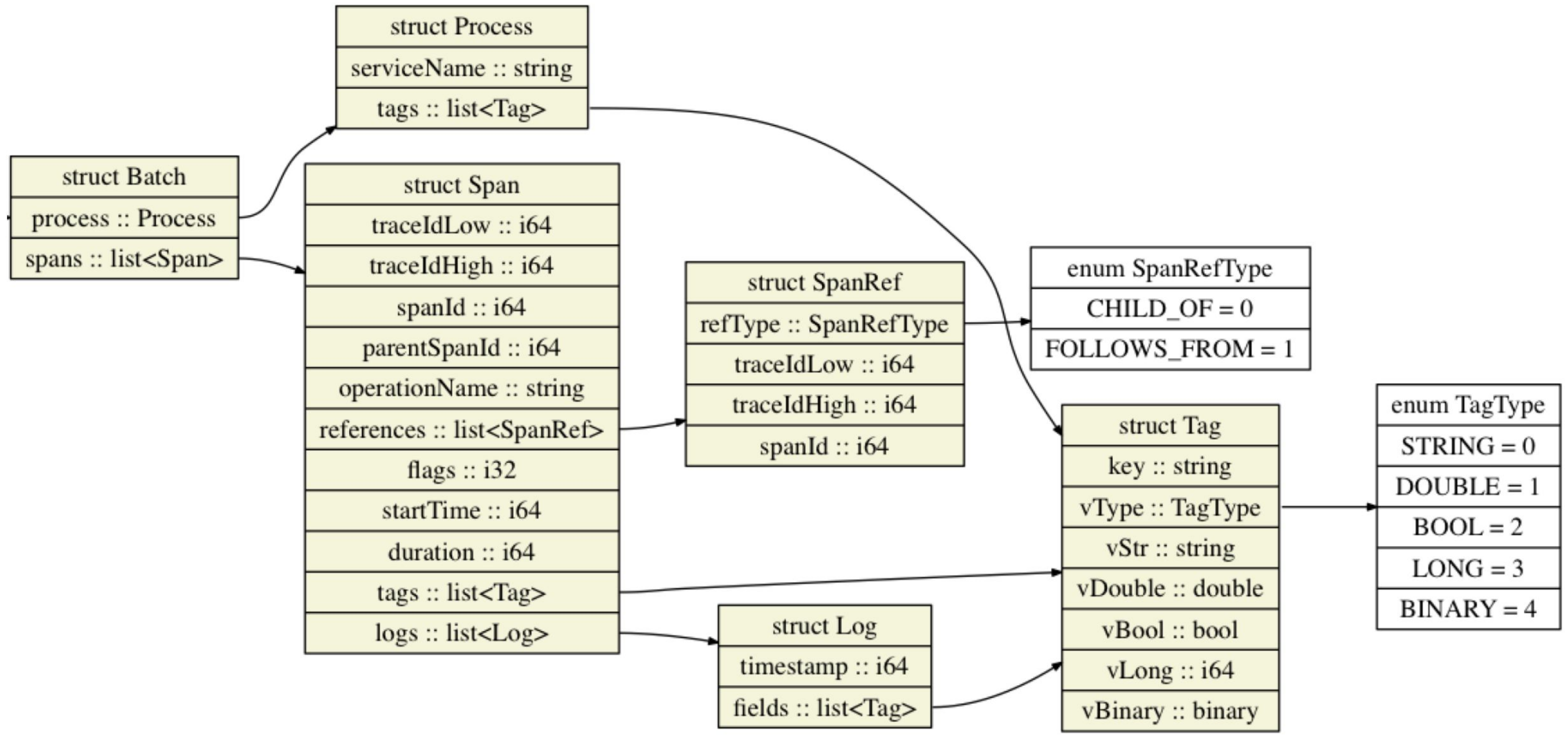
Technology Stack

- Go backend
- Pluggable storage
  - Cassandra, Elasticsearch, memory, ...
- React/Javascript frontend
- OpenTracing Instrumentation libraries

Architecture



Data model


struct Process
serviceName :: string
tags :: list<Tag>
struct Batch
process :: Process
spans :: list<Span>
struct Span
traceIdLow :: i64
traceIdHigh :: i64
spanId :: i64
parentSpanId :: i64
operationName :: string
references :: list<SpanRef>
flags :: i32
startTime :: i64
duration :: i64
tags :: list<Tag>
logs :: list<Log>
struct SpanRef
refType :: SpanRefType
traceIdLow :: i64
traceIdHigh :: i64
spanId :: i64
enum SpanRefType
CHILD_OF = 0
FOLLOWS_FROM = 1
struct Tag
key :: string
vType :: TagType
vStr :: string
vDouble :: double
vBool :: bool
vLong :: i64
vBinary :: binary
enum TagType
STRING = 0
DOUBLE = 1
BOOL = 2
LONG = 3
BINARY = 4
struct Log
timestamp :: i64
fields :: list<Tag>

Zipkin drop-in replacement

Collector:

- JSON v1/v2 and Thrift over HTTP
- Kafka transport not supported

Clients:

- B3 propagation
- Jaeger clients in Zipkin environment

Configuration

- Viper/Cobra
  - `Flag: --span-storage.type=memory`
  - `Env:  SPAN_STORAGE_TYPE=memory`
- k8s ConfigMaps
- UI configuration
  - Control the top menu
  - Google Analytics token
  - Other parameters in the future

Performance Settings

- Collector
  - `--collector.num-workers`
  - `--collector.queue-size`
- Agent
  - `--processor.jaeger-compact.server-queue-size`
  - `--processor.jaeger-compact.workers`

Monitoring

- Metrics
  - `--metrics-backend`
    - `prometheus (default), expvar`
  - `--metrics-http-route`
    - `/metrics (default)`
- Scraping Endpoints
  - Query service - API port 16686
  - Collector - HTTP API port 14268
  - Agent - sampler port 5778

CLOUD NATIVE
COMPUTING FOUNDATION

Roadmap

Things we are working on

Adaptive Sampling

- APIs have endpoints with different QPS
- Service owners do not know the full impact of sampling probability

Adaptive Sampling is per service + endpoint, decided by Jaeger backend based on traffic

Data Pipeline

- Based on Kafka and Apache Flink
- Support aggregations and data mining
- Examples:
  - Pairwise dependencies diagram
  - Path-based dependencies diagram
  - Latency histograms

CLOUD NATIVE
COMPUTING FOUNDATION

Project & Community

Contributors are welcome

Contributing

- make, glide
- 100% test coverage
- Agree to the Certificate of Origin
- Sign commits (-s)
- Backend, client libraries, k8s, data pipeline

Community

- GitHub: https://github.com/jaegertracing
- Chat: https://gitter.im/jaegertracing/
- Mailing List - jaeger-tracing@googlegroups.com
- Blog: https://medium.com/jaegertracing
- Twitter: https://twitter.com/JaegerTracing
- Bi-Weekly Community Meetings

CLOUD NATIVE
COMPUTING FOUNDATION

Q & A

Open Discussion