Jaeger: The Future with OpenTelemetry and Metrics

Maintainer Talk - Kubecon US 2024

Jonah Kowall [@jkowall]
Pavol Loffay [@ploffay]

Pavol Loffay?

ploffay

Jonah Kowall?

jkowall

Agenda

1. Why Distributed Tracing?
2. Intro to Jaeger + Demo
3. Monitoring with Prometheus + Jaeger
4. Jaeger V2
5. New Key Features + Roadmap for Jaeger
6. Q&A from the audience

Why Distributed Tracing?

Microservices Architectures

63% of enterprises are using Microservices most often managed by different teams.

When things **break or are slow**, how do you find the right team to help resolve the problem?

Benefits of Distributed Tracing

- Root cause analysis
- Build a service map of relationships and dependencies
- Measure and drill into specific user actions
- Collaborate across teams
- Monitor and maintain SLAs

Tracing Tools

- Instrumentation
- Data Collection
- Storage & Analysis & Visualization

Intro to Jaeger

OpenTelemetry Semantics

Trace represents an end-to-end request (and response); made up of single or multiple **Spans**

Span represents work done by a single-service or component with time intervals and associated metadata such as **Tags**

Tags/Attributes contain metadata to help contextualize a span

Relationships in tracing

```
Causal relationships between Spans in a single Trace


        [Span A]  ←←←(the root span)
            |
     +------+------+
     |             |
 [Span B]      [Span C] ←←←(Span C is a `ChildOf` Span A)
     |             |
 [Span D]      +---+-------+
               |           |
           [Span E]    [Span F] >>> [Span G] >>> [Span H]
                                         ↑
                                         ↑
                                         ↑
                         (Span G `FollowsFrom` Span F)
```

```
Temporal relationships between Spans in a single Trace


--|-------|-------|-------|-------|-------|-------|-------|-> time

 [Span A········ ····· ····· ····· ····· ····· ····· ·····]
   [Span B····· ····· ····· ····· ····· ····· ·····]
      [Span D····· ····· ····· ····· ····· ·····]
    [Span C····· ····· ····· ····· ····· ·····]
         [Span E·····]        [Span F··] [Span G··] [Span H··]
```

Measure errors, latency, and other indicators across each span

Jaeger Demo

JAEGER UI | Search | Compare | System Architecture | Monitor | Lookup by Trace ID... | About Jaeger

Search | Upload

Service (9)
frontend

Operation (2)
/dispatch

Tags
http.status_code=200 error=true

Lookback
Last Hour

Max Duration
e.g. 1.2s, 100ms, 500...

Min Duration
e.g. 1.2s, 100ms, 500...

Limit Results
200

Find Traces

200 Traces

Sort: Most Recent | Download Results | Deep Dependency Graph

Compare traces by selecting result items

test-executor: runTest a4849fa — 576ms
15 Spans | 9 Errors | customer (2) | driver (3) | frontend (3) | mysql (1) | redis (2) | test-executor (2) | ui (2) | Today 4:55:42 pm, 8 minutes ago

test-executor: runTest 1166b5b — 652.73ms
23 Spans | 9 Errors | customer (2) | driver (7) | frontend (3) | mysql (1) | redis (6) | test-executor (2) | ui (2) | Today 4:55:42 pm, 8 minutes ago

test-executor: runTest b328ce8 — 570.22ms
15 Spans | 9 Errors | customer (2) | driver (3) | frontend (3) | mysql (1) | redis (2) | test-executor (2) | ui (2) | Today 4:55:41 pm, 8 minutes ago

test-executor: runTest 9c32feb — 634.45ms
21 Spans | 9 Errors | customer (2) | driver (6) | frontend (3) | mysql (1) | redis (5) | test-executor (2) | ui (2) | Today 4:55:41 pm, 8 minutes ago

test-executor: runTest 298961e — 675.08ms
25 Spans | 9 Errors | customer (2) | driver (8) | frontend (3) | mysql (1) | redis (7) | test-executor (2) | ui (2) | Today 4:55:41 pm, 8 minutes ago

test-executor: runTest 2e26ed7 — 597.51ms

Search and drill into specific traces

Relationships between services & components

JAEGER UI | Search | Compare | System Architecture | Monitor | Lookup by Trace ID... | About Jaeger

frontend: /dispatch 6598465

Find... | Trace Timeline | Archive Trace

Trace Start **March 15 2024, 13:14:29**.965 | Duration **1.1s** | Services **6** | Depth **4** | Total Spans **40**

0µs | 273.85ms | 547.7ms | 821.55ms | 1.1s

Service & Operation	0µs	273.85ms	547.7ms	821.55ms	1.1s
frontend /dispatch					
frontend HTTP GET	685.92ms				
customer /customer	684.7ms				
mysql → mysql SQL SELECT	684.35ms				
frontend driver.DriverService/FindNearest	224.91ms				
driver driver.DriverService/FindNearest	223.96ms				
redis-manual FindDriverIDs	23.55ms				
redis-manual GetDriver	33.35ms				
redis-manual GetDriver					

SQL SELECT

Service: **mysql** | Duration: **684.35ms** | Start Time: **1.53ms**

Tags

internal.span.format	`otlp`
peer.service	`mysql`
request	`8470-7`
span.kind	`client`
sql.query	`SELECT * FROM customer WHERE customer_id=567`

Process: host.name = fedora | os.type = linux | otel.library.name = mysql | telemetry.sdk.language = go | telemetry.sdk.name = opentelemetry | telemetry.sdk.version = 1.24.0

Logs (2)

1.56ms: event = Waiting for lock behind 2 transactions | blockers = [8470-5 8470-6] | level = info

419.29ms: event = Acquired lock; 0 transactions waiting behind | level = info | waiters = []

Log timestamps are relative to the start time of the full trace.

SpanID: 2ff6aaa091dcb492

Look at a trace in timeline view

JAEGER UI | Search | Compare | System Architecture | Monitor | Lookup by Trace ID... | About Jaeger

test-executor: runTest 7660fb2

Find... | Trace Flamegraph

Trace Start **October 9 2024, 16:55:23.124** | Duration **880.47ms** | Services **8** | Depth **9** | Total Spans **53**

Search...

Head first | Reset View | Collapse nodes above | Table | Both | Flamegraph

Location	Self	Total
frontend: HTTP GET	1186.40 ms	1230.11 ms
test-executor: runTest	501.62 ms	880.47 ms
mysql: /sql_select	24.75 ms	24.75 ms
ui: /	21.74 ms	377.60 ms
customer: /customer	18.44 ms	43.71 ms
test-executor: HTTP GET	1.25 ms	378.85 ms
ui: HTTP GET	0.91 ms	355.86 ms
customer: HTTP GET	0.52 ms	25.27 ms
total	0.00 ms	880.47 ms
frontend: /dispatch	0.00 ms	354.96 ms

Pyroscope

Or as a flamegraph

JAEGER UI | Search | Compare | System Architecture | Monitor | Lookup by Trace ID... | About Jaeger

A

frontend: /dispatch cc9a9f4

- **Today, 1:14:29 pm**
- Duration: **970.16ms**
- Spans: **39**

VS

B

frontend: /dispatch 6598465

- **Today, 1:14:29 pm**
- Duration: **1.1s**
- Spans: **40**

Find...

Monitoring with Prometheus + Jaeger

Tracing and Monitoring

- Moving Jaeger from “distributed tracing” towards APM
  - Traces / Events
  - Metrics
- Additional use cases
  - Operational Monitoring
  - Operational Alerting
  - Change Planning

Jaeger Natively emits Prometheus Metrics

Remote write or scrape Prometheus metrics from Jaeger directly to your Prometheus compatible metrics backend.

Application level metrics to measure and alert on performance and usage:

- Rate
- Errors
- Duration

Jaeger Metrics Data Flow

Prometheus handles all metrics use cases (monitor, alert, plan)

Can also export to other metric systems with Otel exporters and SpanMetrics Connector

SpanMetrics Connector built into Jaeger v2

```
connectors:
  spanmetrics:
    exemplars:
      enabled: true

service:
  pipelines:
    traces:
      receivers: [otlp]
      exporters: [otlp/traces, spanmetrics]
    metrics:
      receivers: [spanmetrics]
      exporters: [otlphttp/metrics] #export data to Prometheus
```

Visualize app performance R.E.D metrics

The “monitor” page in Jaeger to provides the status and health of transactions and application components

Deeper analysis can be done with Perces or Grafana.

Jaeger Monitor Demo

NOW AVAILABLE!

Jaeger v2

based on OpenTelemetry collector

Jaeger V2 Goals

- Adapt Jaeger to OpenTelemetry by moving the collector into the core
- New storage APIs for Jaeger v2.x
- Allow for backwards compatibility to Jaeger v1.x storage
- Native OLTP support for storage

- Single binary for all components
- Configuration is via configuration file not CLI flags (similar to OpenTelemetry)

Jaeger v2 Binary Architecture

V2 Configuration

```yaml
service:
  extensions: [jaeger_storage, jaeger_query, remote_sampling]
  pipelines:
    traces:
      receivers: [otlp, jaeger, zipkin]
      processors: [batch, adaptive_sampling]
      exporters: [jaeger_storage_exporter]

extensions:
  jaeger_query:
    storage:
      traces: my_memory_store

  jaeger_storage:
    backends:
      my_memory_store:
        memory:
          max_Traces: 10000

exporters:
  jaeger_storage_exporter:
    trace_storage: memstore
```

Jaeger v2 deployment architecture

Jaeger v2 deployment architecture with queuing

New Key Features Since Kubecon NA + Roadmap

New Features

- Jaeger V2 based on OpenTelemetry collector
  - OpenTelemetry-native data pipeline
  - Full native Kafka support
- Support for ElasticSearch v8.x
- Jaeger V3 query API exposes stable V1 OTLP
- Sampling store support for Badger, ElasticSearch, OpenSearch
- UI: Zoom in/out controls in graphs
- UI: SPM add span kind selector
- UI: ability to upload OLTP
- UI: Critical path visualization in trace view

Roadmap

- Helm Chart for Jaeger v2
- Kubernetes operator for Jaeger v2
- Support ClickHouse as official storage backend
- Upgrading UI to use OpenTelemetry data natively
- Upgrading UI to normalize dependency views

And more useful and interesting capabilities coming in the future

Q&A

Resources

jaegertracing.io/docs

monthly community call and Notes
CNCF Slack #jaeger : https://slack.cncf.io

@jaegertracing

medium.com/jaegertracing

Leave feedback on this session