Observable microservices with MicroProfile OpenTracing and looking beyond to OpenTelemetry

Pavol Loffay
Senior Software Engineer
Agenda

- Introduction to distributed tracing
- MicroProfile-OpenTracing
- Demo tracing in Quarkus with Jaeger
- OpenTelemetry
Pavol Loffay

- Software Engineer at Red Hat
- Distributed tracing: Jaeger, OpenTracing, OpenTelemetry
- MicroProfile-OpenTracing
Why tracing?
Modern distributed systems are complex.

Loading a page can involve 100s services and multiple nodes.
We want to tell what happened with the request.

Which service or component caused the problem under which conditions.
Metrics - no context, which request caused a spike?

Logs - hard to correlate, parallel requests, multiple hosts.
Monitoring tools must tell stories

Metrics and logs have no context or describe only one instance.

Distributed tracing tells story about the whole the transaction.
Distributed Tracing
Distributed Tracing concepts

![Diagram showing distributed tracing concepts with TraceID, context, and spans representing A, B, C, D, E in a tree structure and timeline.]
Distributed Tracing concepts

In-process context propagation

*) Thread pools, Queues, Futures
MicroProfile-OpenTracing
MicroProfile-OpenTracing

- Tracing API
- Vendor neutral
- No data/wire format

- Uses OpenTracing semantics and API
- Defines additional API
MicroProfile-OpenTracing

Auto instrumentation

- JAX-RS
- MicroProfile Rest Client

```java
@Path("/")
public class Handler {

  @GET
  @Path("/hello")
  public Response hello() {
    Response response = client
        .target("http://localhost:8090/api")
        .request()
        .get();
    String entity = response.readEntity(String.class);
    response.close();
    return Response.ok(entity).build();
  }
}
```
MicroProfile-OpenTracing

Explicit instrumentation

- @Traced
- @Inject

```java
import io.opentracing.Tracer;

@Path("/")
public class Handler {

    @Inject
    private io.opentracing.Tracer tracer;

    @GET
    @Path("/hello")
    @Traced(operationName = "bonjour")
    public Response hello() {
        return Response.ok().build();
    }

    @GET
    @Path("/ping")
    @Traced(false)
    public Response ping() {
        return Response.ok().build();
    }
}
```
**MicroProfile-OpenTracing**

Configuration

- MP Config
  - `mp.opentracing.server.operation-name-provider`
    - GET: `foo.bar.UserHandler.getUser`
    - `/user/{id}`
  - `mp.opentracing.server.skip-pattern`
    - `/health|/foo/bar*`
Demo

Quarkus application

https://github.com/pavolloffay/quarkus-tracing
MicroProfile-OpenTracing

Roadmap

- Operation names for MP Rest Client
- More annotations
- Automatically trace CDI
- MicroProfile-Reactive Messaging
- MicroProfile-Fault tolerance
OpenTelemetry
OpenTelemetry

● “The next major version of OpenTracing and OpenCensus”
● CNCF (Google, Microsoft, LightStep, Dynatrace, DataDog...)

OpenTelemetry
OpenTelemetry

- API, SDK
- Data format and W3C Trace-Context
- Tracing, Metrics, (Logs)
- Named tracers/meters

```
tracer = OpenTelemetry.getTracerFactory().getTracer("io.opentelemetry.contrib.mongodb", "semver:1.0.0");
```

- Agent/Collector
OpenTelemetry in MP

1. New specification
2. Expose OpenTelemetry in MP-OT
3. Hybrid: new project and use OT shim
Start with distributed tracing as early as possible.

Go to MicroProfile forum and vote which proposal you like!
Thank you

Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat  facebook.com/redhatinc
youtube.com/user/RedHatVideos  twitter.com/RedHat