LAB 24: QUARKUS MONITOR REVIEW

Autor: José Díaz

Github Repo: https://github.com/joedayz/quarkus-bcp-2025.git

Abre el proyecto **21-monitor-review-start.**

Instrucciones

La aplicación de conferencia incluye los siguientes microservicios:

sessions
Lista las sesiones, incluyendo información sobre el ponente de cada sesión.
Este microservicio depende del servicio de ponentes, usa registro en formato JSON y escucha en el puerto **8081**.

speakers
Lista los ponentes. Este microservicio escucha en el puerto **8082**.

dashboard
Una aplicación web para acceder a los endpoints HTTP de los microservicios **sessions** y **speakers**.

Los microservicios **sessions** y **speakers** incluyen todas las extensiones requeridas para este ejercicio.

---

1. **Inicia los microservicios.**
   - Para iniciar los microservicios **sessions** y **speakers**, usa el modo de desarrollo.
   - Para iniciar el dashboard, ejecuta el script ~/D0378/monitor-review/dashboard/serve.py.
2. **Abre el dashboard** en un navegador web y navega a http://localhost:8083.
   En el dashboard, haz clic en **Sessions** para ver la lista de sesiones.

   La página tarda varios segundos en mostrar la lista de sesiones.
   Debes solucionar este problema.

3. **Activa el rastreo (tracing)** en los microservicios **sessions** y **speakers**.
   - Recoge todas las muestras de rastreo, para todas las solicitudes.
   - Envía todas las trazas al colector de Jaeger que esta corriendo localmente en el puerto 14268.
   - Propaga todos los headers b3-* para asegurar full traceabilidad.
   - Se puede formatear los logs del microservicio speaker incluyendo tracing information con el siguiente formato:

```
quarkus.log.console.format=%d{HH:mm:ss} %-5p
traceId=%X{traceId}, spanId=%X{spanId},
parentId=%X{parentId}, sampled=%X{sampled}
[%c{2.}] (%t) %s%e%n
```

4. **Recarga la página de sesiones** en el frontend del dashboard para generar nuevos rastreos.
   Luego, inspecciona los rastreos en **Jaeger UI** para identificar si el problema está en el servicio **sessions** o **speakers**.
   La interfaz de Jaeger está disponible en **http://localhost:16686**.
5. **Aísla el método** que está causando el problema de rendimiento.
   Modifica el microservicio problemático agregando rastreo (tracing) a la clase que provoca la lentitud.
6. **Corrige el código lento** e inspecciona los rastreos en **Jaeger UI** para confirmar que el problema ha sido resuelto.
7. En el microservicio **sessions**, agrega una métrica llamada callsToGetSessions para contar el número de invocaciones recibidas por el endpoint **GET /sessions**.
   Luego, abre la interfaz de **Grafana**, disponible en **http://localhost:3000**, para inspeccionar las métricas.
   Usa **admin** como nombre de usuario y contraseña para iniciar sesión en Grafana.
8. **Cambia la configuración de logs** en el microservicio **sessions** para entornos de desarrollo.
   Aplica la siguiente configuración al perfil de desarrollo:

   - No uses formato JSON para los logs. En su lugar, formatea los logs con el patrón:
     %d %-5p %m - traceId=%X{traceId}, spanId=%X{spanId}%n
   - El nivel de log debe ser **DEBUG** solo para las clases dentro del paquete **com.bcp.training**.

9. **Detén** los microservicios **sessions** y **speakers**, así como el script **serve.py** del dashboard.

SOLUCION

1. Inicia los microservicios

   a. Inicia los microservicios **sessions** y **speakers**, usa el modo desarrollo.

      Sessions levanta en el 8081
      Speakers levanta en el 8082

   b. Inicia el dashboard ejecutando **python3 serve.py**

Dashboard levanta en el 8083

2. Abre el dashboard en el navegador web navegando a http://localhost:8083. En el dashboard clic en **Sessions** para ver la lista de sessions.
   La página toma varios segundos en mostrar la lista de sessions. Tu debes solucionar este problema.

3. Activa el tracing en los microservicios sessions y speakers.

   a. Colleciona todos los trace samples, para todos los requests
   b. Envía todas las trazas al colector de Jaeger que corre localmente en el puerto 14268
   c. Propaga todos los headers b3-* para asegurar full traceabilidad.
   d. Tu deberías formatear los logs de los microservicios speakers para incluir tracing information con el siguiente formato:

```
quarkus.log.console.format=%d{HH:mm:ss} %-5p
traceId=%X{traceId}, spanId=%X{spanId},
parentId=%X{parentId}, sampled=%X{sampled}
[%c{2.}] (%t) %s%e%n
```

   e. Agrega al **speakers/src/main/resources/application.properties** las siguientes propiedades.

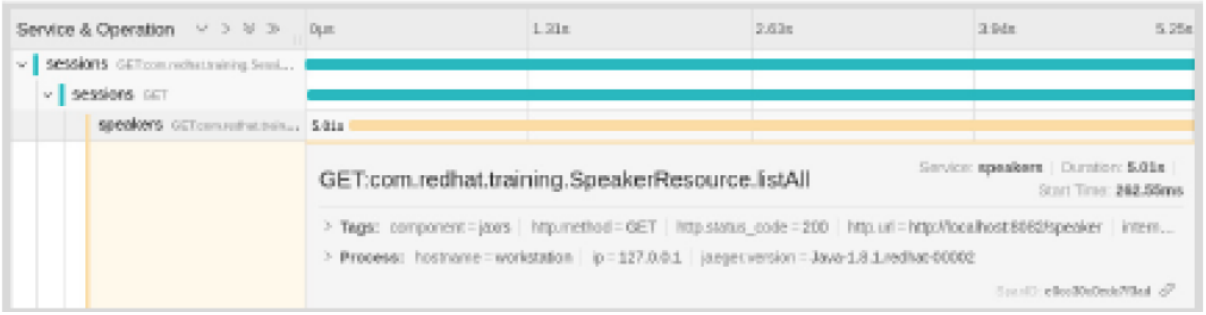
```
# Enable Tracing
quarkus.otel.service.name = speakers
quarkus.otel.traces.sampler=traceidratio
quarkus.otel.traces.sampler.arg=1

quarkus.log.console.format=%d{HH:mm:ss} %-
5p traceId=%X{traceId}, spanId=%X{spanId},
parentId=%X{parentId}, sampled=%X{sampled}
[%c{2.}] (%t) %s%e%n
quarkus.otel.exporter.otlp.traces.endpoint
=http://localhost:4317
```

   f. Agrega a **sessions/src/main/resources/application.properties** las siguientes propiedades.

```
quarkus.otel.service.name = sessions
quarkus.otel.traces.sampler=traceidratio
quarkus.otel.traces.sampler.arg=1
quarkus.otel.exporter.otlp.traces.endpoint=http://
localhost:4317
```

4. Recarga la pagina de sessions en el dashboard para generar nuevas trazas. Luego, inspecciona las trazas en la consola web de Jaeger para identificar si el problema es el servicio sessions o speakers. La Jaeger UI esta en http://localhost:16686.

- Retorna al dashboard y recargar la página Sessions para generar trazas.
- Ve a http://localhost:16686 y abre la Jaeger UI
- En la consola web de Jaeger, selecciona el servicio **sessions** del campo **Service** en el panel Search. Clic en **Find Traces**.
- Clic las primeras sessions: **GET:com.bcp.training.SessionResource.getAllSessions** para ver los detalles de traza.
- En la traza, verificar que el **GET:com.bcp.training.SpeakerResource.listAll** span esta tomando varios segundos en atender el request.

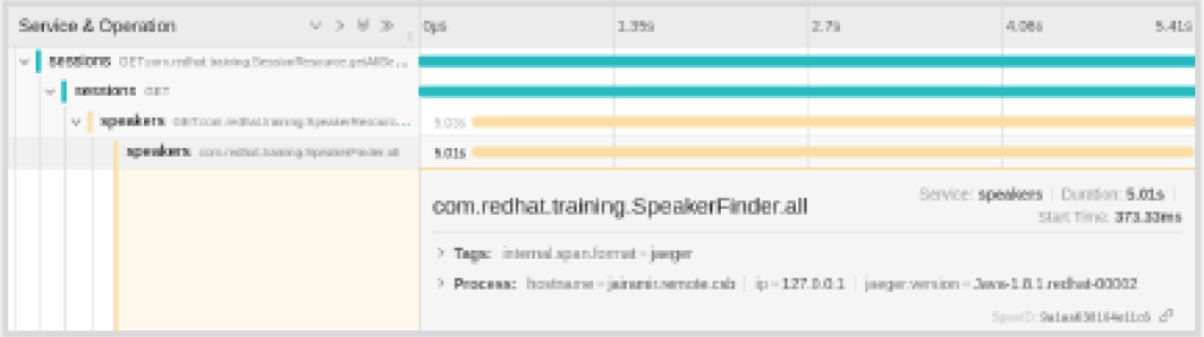


5. Aislar el metodo que esta causando problemas de rendimiento. Modifica el servicio problemático agregando tracing a la clase que causa la lentitud.

a. El **SpeakerResource#listAll** maneja todos los requests del **GET /speaker** endpoint del servicio **speaker**. Este método usa el **com.bcp.training.SpeakerFinder** bean, el cual obtiene los speakers de una base de datos. Agregar la anotación **@WithSpan** a esta clase.

```
@WithSpan
public List<Speaker> all() {
    Log.info( "Retrieving all speakers from
database" );…
}
```

b. Retorna al dashboard en el navegador y carga la pagina de **Sessions** para generar nuevas trazas.
c. Retorna a la consola Jaeger y clic en **Search** en la barra top de navegación.
d. En el panel **Search**, clic **Find Traces** para refrescar la lista de trazas.
e. Clic en el mas reciente trace de la lista.
f. Verificar que el **SpeakerFinder#all** esta causando cuellos de botella. Este método toma varios segundos en cargar la lista de speakers.

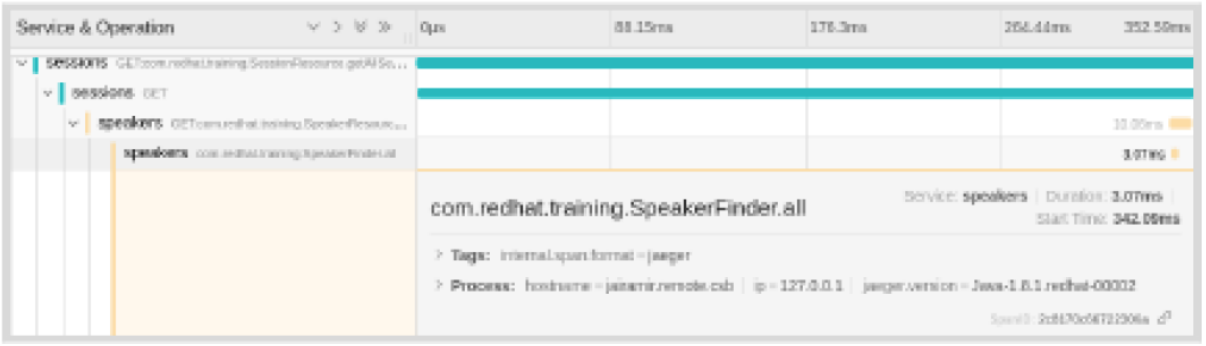


6. Repara el código lento e inspecciona las trazas en el Jaeger UI para confirmar que el problema ha sido solucionado.

- En el método SpeakerFinder#all, elimina o comenta el código que causa el problema.

```
@WithSpan
public List<Speaker> all() {
    Log.info( "Retrieving all speakers from
database" );

    //runSlowAndRedundantOperation();

    return Speaker.listAll();
}
```

- Retorna al dashboard frontend en el navegador y carga la página Sessions para generar nuevas trazas.
- Retorna a la consola Jaeger y clic en Search en la barra top de navegación.
- En el panel Search, clic en Find Traces para refrescar la lista de trazas.
- Clic en el mas reciente trace de la lista.
- Verifica que el método SpeakerFionder#all toma unos cuantos milisegundos para completar el request.



7. En el microservicio sessions, agrega una métrica llamada callsToGetSessions para contar el número de invocaciones recibidas por el endpoint GET /sessions. Luego, abre el Grafana UI que esta disponible en http://localhost:3000, para revisar las métricas. Usa admin como username y password para log in a Grafana.

- Abre el **com.bcp.training.SessionResource** y agrega el **@Counted** al método **getAllSessions**.

```
@GET
@Counted(value = "callsToGetSessions")
public Collection<Session> getAllSessions() throws
Exception {
    return sessionStore.findAll();
}
```

- Retorna al dashboard en el navegador y carga la página Sessions para generar valores para la nueva métrica.
- Abre el navegador y ve a http://localhost:3000/dashboards.
- Ingres **admin** como el username y **admin** como el password, luego clic en **Log in**.
- Clic **Skip** para omitir el cambio de password de la cuenta.
- Clic en el directorio **conference**, y luego clic en **DO378 Conference Metrics Dashboard**.
- Observa el dashboard que colecciona las métricas agregadas a la aplicación.

8. Cambiar la configuración de logging en el microservicio sessions para entornos de desarrollo. Aplica la siguiente configuración al profile desarrollo.

- No uses formatting JSON. En su lugar formatea los logs y habilita DEBUG solo para el paquete com.bcp.training.

```
# Configure Dev Logging
%dev.quarkus.log.console.json=false
%dev.quarkus.log.console.format = %d %-5p %m -
traceId=%X{traceId}, spanId=%X{spanId}%n
%dev.quarkus.log.category."com.bcp.training".level
= DEBUG
```

- Retorna al dashboard en el navegador y recarga la pagina Sessions.
- Revisa los logs del microservicio sessions. La consola debe mostrar mensajes de debug y los logs usan el formato personalizado.

```
...output omitted...
2023-02-01 14:27:48,405 INFO  Finding all sessions - traceId=..., spanId=...
2023-02-01 14:27:48,406 DEBUG Gathering extended speaker information ... -
 traceId=..., spanId=...
2023-02-01 14:27:48,419 DEBUG Added speakers information to session list -
 traceId=..., spanId=...
```

9. Para los microservicios sessions y speakers, y luego el script serve.py.

Felicitaciones. Hemos terminado el laboratorio.

Jose