Better JVM Observability With No Code Changes

Tyler Benson
Observability Veteran, ServiceNow
In distributed systems, **observability** is the ability to collect data about program execution, internal states of modules, and the communication between components.

https://en.wikipedia.org/wiki/Observability_(software)
OpenTelemetry

A collection of standards, conventions, tools, APIs, and SDKs for observability

- Tracing
- Metrics
- Logging
- More in progress
java \-javaagent:path/to/opentelemetry-javaagent.jar \
-jar my-app.jar
Application

- Class A
- Class B
- Class C
- Class D

opentelemetry-javaagent.jar
Supported libraries, frameworks, application servers, and JVMs (oh my!)

https://github.com/open-telemetry/opentelemetry-java-instrumentation/blob/main/docs/supported-libraries.md
Aspect Oriented Programming

Advice and pointcuts
public void attendSREcon() {
    System.out.println("Hello Singapore!");
}

AOP - Advice

advice passportCheck() {
    if (!hasPassport()) {
        throw new IllegalAccessException();
    }
}

AOP - Join Point or Pointcut

pointcut before(attendSREcon()) {
    passportCheck();
}

public void attendSREcon() {
    if (!hasPassport()) {
        throw new IllegalAccessException();
    }
    System.out.println("Hello Singapore!");
}
Generic Method Advice

• Method Name
• Method execution duration
• Thrown exception details
• Environmental Attributes
• Parent Span
Defining the Pointcut

Annotation:
- Requires code change
- Evolves with the class
- Preferred if edits are ok

@WithSpan

Configuration
- No code change needed
- Can break if class/method name is changed without updating config

System property:
otel.instrumentation.methods.include

Environment variable:
OTEL_INSTRUMENTATION_METHODS_INCLUDE

my.package.MyClass1[method1,method2]
;my.package.MyClass2[method3]
Introducing: Flashlight

https://unsplash.com/photos/GewrE-mkk4
Available on Github

https://github.com/lightstep/flashlight-java

Latest release download:
https://github.com/lightstep/flashlight-java/releases/latest/download/flashlight.jar
spring-petclinic
Heuristics

- Methods with locking/synchronization
Heuristics

- Methods with locking/synchronization
- Methods with networking client calls (http, database, etc)
Heuristics

- Methods with locking/synchronization
- Methods with networking client calls (http, database, etc)
- Large/complex methods
  - Many method calls
  - High branching logic
Heuristics

- Methods with locking/synchronization
- Methods with networking client calls (http, database, etc)
- Large/complex methods
  - Many method calls
  - High branching logic
- What other generic heuristics can you think of?

Come talk to me after with your ideas.
Ideal Candidates

What to consider when instrumenting methods

- Span value diminishes with over-use. Span limits per trace should be in the hundreds, not thousands.
Ideal Candidates

What to consider when instrumenting methods

- Target distinct methods in business specific classes. (Avoid overlap with javaagent)
Ideal Candidates

What to consider when instrumenting methods

• Focus on filling large gaps with useful detail that will help clarify application execution flow.
Ideal Candidates

What to consider when instrumenting methods

- Identify methods that help differentiate uncommon or important code flows
Ideal Candidates

What to consider when instrumenting methods

- Add Span attributes for important business classifications using
  - @SpanAttribute
  - Manual instrumentation
  - OTel Javaagent Extension
Ideal Candidates

What to consider when instrumenting methods

- Valuable Business Attributes
  - Account name/id
  - User name/id
  - Product plan level (free vs enterprise)
  - Feature Flags State
Avoid Instrumenting

- Don’t duplicate instrumentation already available.
- Don’t instrument everything… be conservative depending on risk tolerance and validation procedures.
- Especially don’t instrument hot methods in a tight loop.
- (Consider performance testing.)
Next Level: OpenTelemetry Java agent Extension

- Customize OpenTelemetry Java agent settings programmatically
- Provide custom samplers, exporters, propagators, etc
- Add additional custom instrumentation

Next Level: AOP Frameworks

- Spring AOP
- AspectJ
- Byte Buddy (used internally by OTel Javaagent)
Next Level: Complementary tools

- Async Profiler
- Java Flight Recorder
- eBPF
Q&A

https://github.com/lightstep/flashlight-java