Do Auto and Manual Instrumentation In Apache SkyWalking

Wu Sheng
Apache SkyWalking original creator, PMC, Committer
Microsoft MVP
What am I doing?

- Apache SkyWalking (incubating)
- SkyWalking multiple language agents
- OpenSkywalking community
- CNCF OpenTracing
- W3C Trace Context Spec
- Zipkin <-> SkyWalking integration
- OpenCensus
- Sharding-Sphere
SkyWalking APM

- Why create SkyWalking?

- Where did the name come from?
SkyWalking APM

- Why need auto instrumentation?
- Most commercial APMs are AUTO
SkyWalking APM

- Differences between APM and tracing
How does SkyWalking work?
Why still need manual Instrumentation

- Instrumentation under conditions, to reduce performance payload
- Controlled by developers, rather than APM
- Across thread
  - Thread pool
  - Task assignment
  - Coroutine
  - Goroutine
- Low level transports, TCP
Manual instrumentation APIs

- SkyWalking
  - Manual Instrumentation
    - Library
- OpenTracing
  - Manual Instrumentation
    - APIs
SkyWalking trace toolkit

- Trace any method with only annotation required
- Tag supported

```java
@Trace(operationName = "selectUser")
public void selectUser(String name) {
    ActiveSpan.tag("user.name", name);
    Connection connection = null;
    PreparedStatement preparedStatement = null;
    try {
        connection = dataSource.getConnection();
        preparedStatement = connection.prepareStatement("SELECT * FROM user WHERE name =?");
        preparedStatement.setString(1, name == null ? "" : name);
        preparedStatement.executeQuery();
    }
```
OpenTracing APIs bridge

@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    ActiveSpan span = new SkywalkingTracer().buildSpan("OpentracingCase/doGet").startActive();
    service.testLocalSpan("Test", span);

    // Business codes are here
    // If you want to trace this code block

    span.deactivate();
    PrintWriter printWriter = resp.getWriter();
    printWriter.write("success");
    printWriter.flush();
    printWriter.close();
}
How auto instrumentation works?

- Running with Instrumentation, Pseudocode Only
- Auto-instrumentation mechanism
  - AOP (Aspect Oriented Programming) in bytecode level.
  - Manipulate codes at runtime.
Manual instrumentation on agent

Application codes

- Manual instrumentation APIs
- Business codes
- Apache HttpComponent lib

- Lib weaver codes
- APIs bridge weaver
- Lib weaver codes

- Tracing core context
- Tracing core context
- Tracing core context

Trace segment in single thread, part of distributed trace
APM from trace

- Overview the whole cluster
APM from trace

- Topology detected
APM from trace

- Metrics of every application and its instance
APM from trace

- Application Map
- Application metrics
- Application instances metrics
- JVM metrics
APM from trace

- Metrics of Rest and RPC services
Cool things in 5.x series

- Native metric data support. Traces become optional
- Zipkin -> SkyWalking collector -> visualization
- OpenCensus -> SkyWalking reporter
SkyWalking community

- **GitHub**
  - Java agent, collector, UI: [https://github.com/apache/incubator-skywalking](https://github.com/apache/incubator-skywalking)
  - .NET Core agent: [https://github.com/OpenSkywalking/skywalking-netcore](https://github.com/OpenSkywalking/skywalking-netcore)
  - Node.js server side agent: [https://github.com/OpenSkywalking/skywalking-nodejs](https://github.com/OpenSkywalking/skywalking-nodejs)

- **Gitter:** [https://gitter.im/OpenSkywalking/Lobby](https://gitter.im/OpenSkywalking/Lobby)

- Mail list: dev@skywalking.apache.org
LINUX CON
containercon
CLOUD OPEN
CHINA 中国
THINK OPEN
开放性思维
Skywalking

LF ASIA, LLC