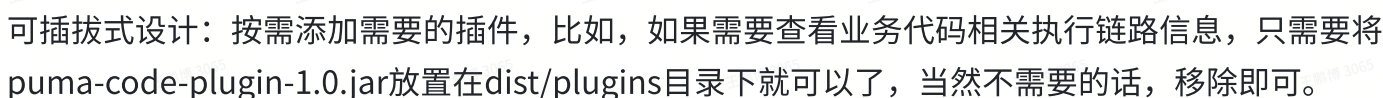


1.可插拔式插件方案设计



上述“加载文件成功”体现的便是可插拔插件设计中按需加载思想。

- @@@ 前缀表示是精准测试 **框架自身日志**
- ### 前缀表示是精准测试 **框架扩展插件日志**，比如：Tomcat/SpringFramework/JDBC插件输出的日志信息

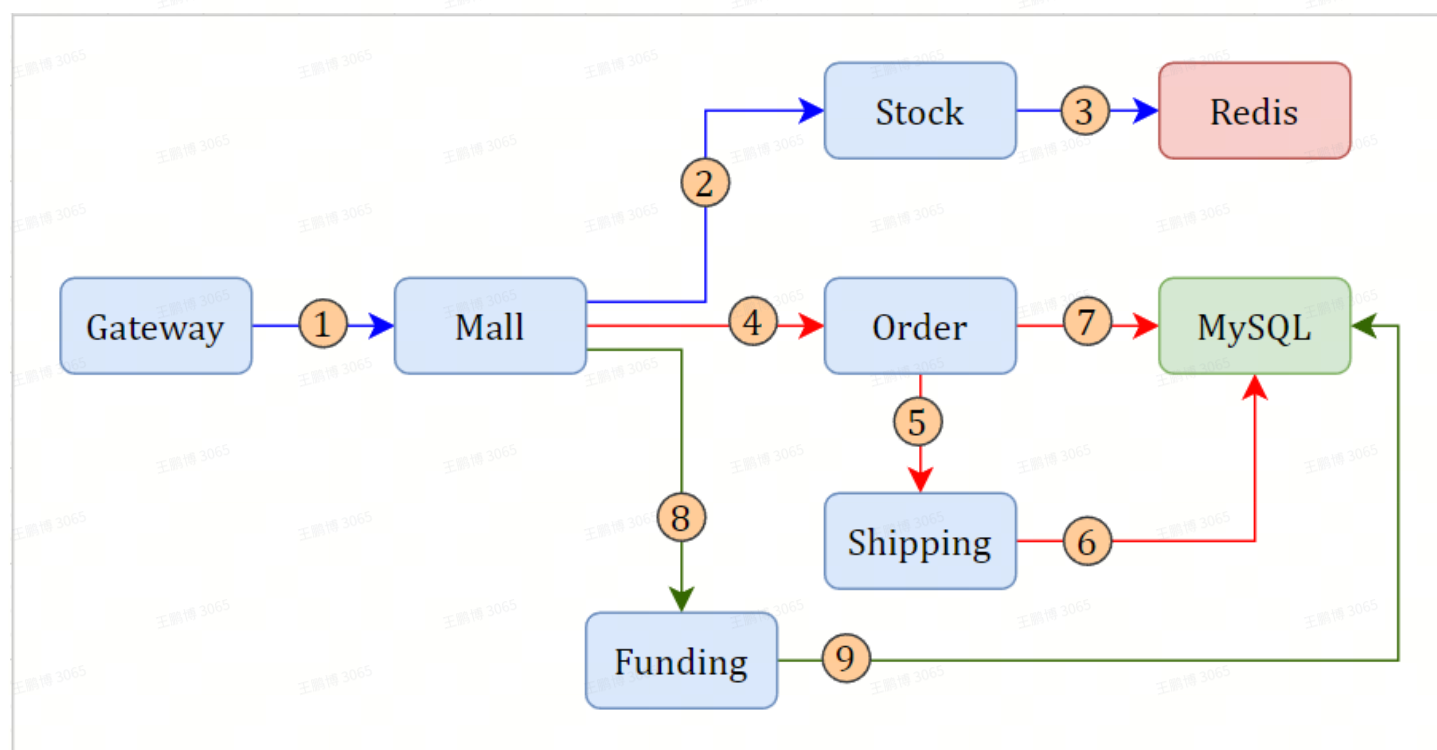
2.链路追踪方案设计

2.1.推荐阅读

- [OpenTracing语义规范](#)
- [OpenTracing语义规范译文](#)
- [谷歌Dapper论文](#)
- [谷歌Dapper论文译文](#)
- [SkyWalking源码](#)
- [SkyWalking官方文档](#)

2.2.重要概念

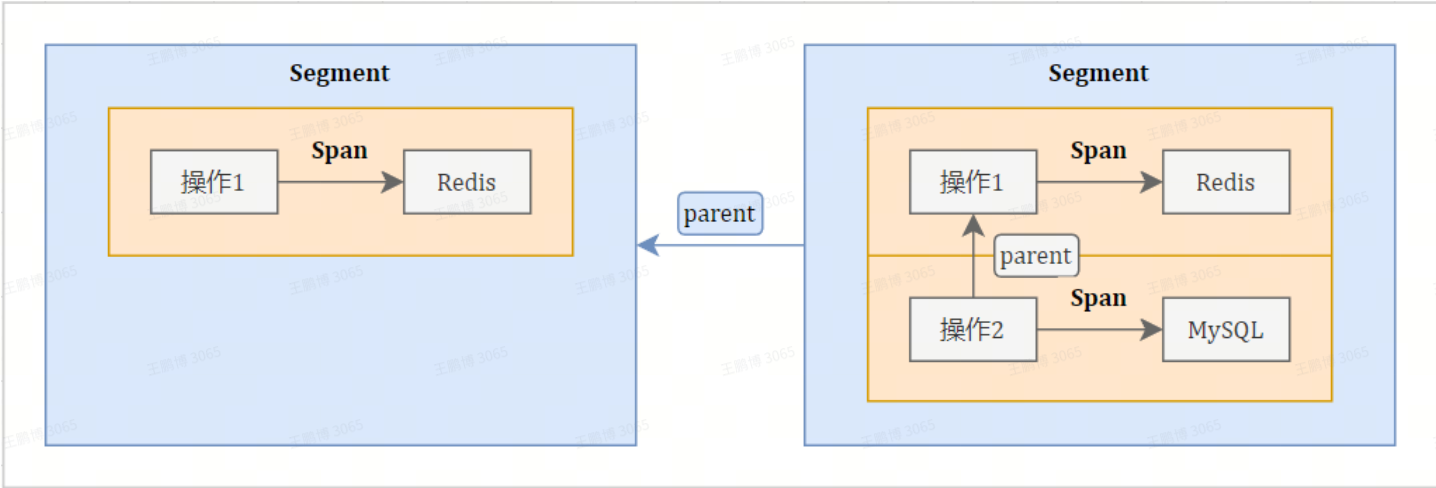
精准测试重点参考开源优秀APM框架SkyWalking对于OpenTracing规范的实现，所以需要深入理解Trace、Segment、Span三个概念。



上图是一个下单接口的链路样例，在链路中首先要理解的概念是Segment，Segment表示一个JVM进程内的所有操作。上图中有6个Segment。Gateway Segment是Mall Segment的parent，通过parent关系就可以把多个Segment按顺序拼起来组装成一个链路。

注意：上述Segment表示的是逻辑概念，物理层面会存在多个Segment，物理层面的单个Segment表示一个JVM进程内一个线程中所有操作集合。

注意：目前虽然不考虑跨进程链路追踪，但是设计上是相通的，且预留了跨进程链路追踪能力的！



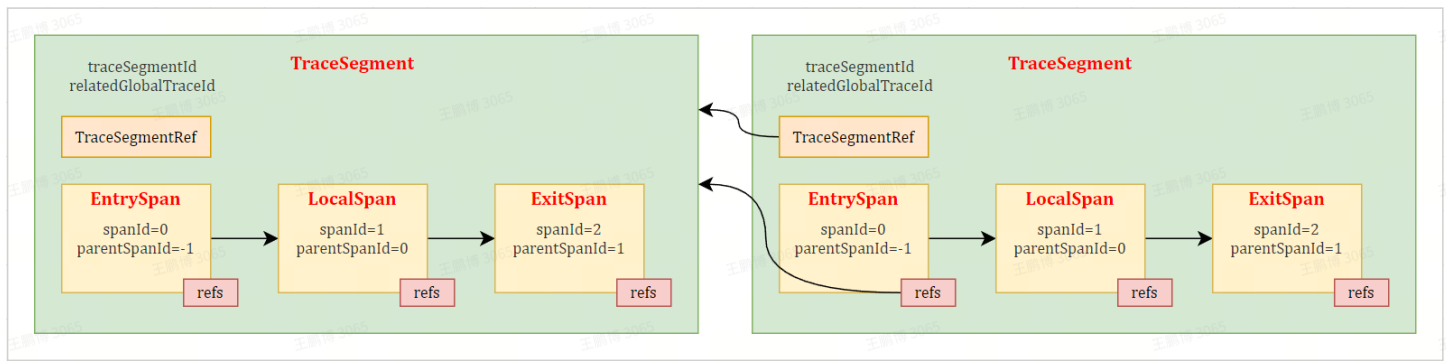
一个Segment里可能发生多个操作，比如操作1是查Redis，操作2是查MySQL，这就是两个Span，Span表示一个具体的操作。Span之间也是基于parent的关系构建起来的，而Segment是Span的容器。

多个Segment连接起来就组成了一个Trace，每个Trace都有一个全局唯一的ID。



注意：所谓的Trace是将所有Segment串起来之后表示一个逻辑概念~

2.3.完整模型



说明：

1. 只有每个TraceSegment的第一个EntrySpan中的refs指向父TraceSegment，且第一个TraceSegment的第一个EntrySpan中的refs为空。
2. 每个TraceSegment的LocalSpan和ExistSpan的refs均为空。
3. 一个TraceSegment可以有多个父TraceSegment，所以refs为多个。
4. 在一个TraceSegment中只能存在一个EntrySpan，后面的EntrySpan会复用前面EntrySpan，并会覆盖掉前一个EntrySpan设置的属性。
5. EntrySpan记录的信息永远是最靠近业务侧的信息。

2.4.数据处理

1. 简单采集数据日志打印（简单）
2. 可视化的采集数据链路日志打印（稍复杂）
3. 采集数据RPC/HTTP发送至服务端存储，服务端提供相关接口，前端做可视化（复杂，最终方案）
 - a. 关于这部分内容参见服务端采集数据存储部分。

3.复杂技术点

关于JavaAgent的技术难题：

- 基于ByteBuddy可插拔式插件如何设计？
- 包冲突依赖如何解决？
- 如何打通父类加载器与子类加载器类信息互相访问？
- 第三方依赖包类如何增强？
- JDK提供的类如何增强？
- 插件配置如何设计？
- 插件日志如何与被检测应用整合？

- 基于ByteBuddy的插件与Spring整合时问题（因为我们的项目基本都是基于SpringBoot，所以是绕不开的问题）
 - SpringBoot类加载机制？IDEA启动时与Java -jar方式启动时类加载机制是不同的~
 - 与Spring整合时@Autowired注入问题？

展开上述任意问题，都会挖掘很深的知识点，这里先做记录，后续有空再做整理~

当然这里也提前说明下，要搞懂上述问题需要掌握一下知识点：

- JavaAgent基础知识
- JVM类加载机制
- SpringBoot类加载机制
- ByteBuddy字节码增强框架
- 设计模式
- maven-shade-plugin与maven-run-plugin打包插件
- 核心插件源码知识点
 - JDK源码
 - Tomcat源码
 - Spring源码
 - MySQL驱动源码

4.TODO

☐ 日志文件输出整合（**优先级高**）

- 问题和困难点：因为JavaAgent的特殊性，存在多版本类加载问题，不同日志组件兼容问题，需要自定义实现日志工具类

☐ 支持JDK 1.8以上版本（优先级低）

☐ 支持由BootstrapClassLoader加载的类插桩（**优先级高**）

- 比如基于JDK原生URLConnection相关工具，比如业务端发送一个HTTP请求需要采集该请求信息需要进行插桩，目前是不支持的

☐ 支持相同组件版本自动识别插桩（比如自动识别Spring Framework 4.x和5.x）

- 比如项目A使用的4.x版本而项目B使用的5.x版本，假如需要插桩的实现不同，目前还不支持版本识别插桩

☒ 支持跨进程链路追踪

- 框架层面已经预留了相关设计

☐ 支持定时任务入口的链路追踪

- 当前仅支持调用HTTP接口生成追踪链路
- 同步程序中存在较多定时任务

☒ 支持插件配置化

- 问题：比如业务代码链路采集插件，可能会存在不同项目采集的包路径是不相同的，这就需要配置化解决

Original Confluence page attachments

Name	Size	Created by	Created on	Labels	Comments
 image-2023-12-20_9-23-... 45.50KB	45.50 KB	王鹏博	2023-12-27T17:39:55.000+08:00		
 image-2023-12-19_10-49-... 21.71KB	21.71 KB	王鹏博	2023-12-27T17:39:54.000+08:00		
 image-2023-11-24_9-44-... 172.40KB	172.40 KB	王鹏博	2023-12-27T17:39:54.000+08:00		
 image-2023-12-20_9-18-... 32.55KB	32.55 KB	王鹏博	2023-12-27T17:39:54.000+08:00		
 image-2023-12-20_9-21-... 24.64KB	24.64 KB	王鹏博	2023-12-27T17:39:54.000+08:00		