9、Skywalking的埋点-Trace的基本概念

rock_fish 关注
2021.05.23 17:07:57 字数 1,470 阅读 429

开始之前先仔细阅读skywalking创始人吴晟的一些文章资料：

- opentracing文档中文版吴晟
- 增强高度分布式和大规模应用系统的拓扑自动检测

TraceSegment

skywalking中关于 `trace` 的一些概念，较opentracing来说是进行了一些扩展，比如其核心 `TraceSegment` 表示一类span的聚合。
我们这样来理解：在微服务架构中，一个请求基本都会涉及跨进程（以及跨线程）的操作，例如， RPC 调用、通过 MQ 异步执行等这类操作就需要涉及到多个服务的多个线程，`TraceSegment` 就记录了一个请求在一个线程中的执行流程。当将该请求所关联的全部 `TraceSegment` 串起来，就能得到该请求的完整 `Trace` ，总结来说即是：

- 一个 `trace` 由多个 `tracesegment` 构成
- 一个 `Tracesegemnt` 记录了一个请求在一个线程中的执行流程
- 一个 `TraceSegment` 内包含一个 `Span` 集合

`TraceSegment` 的核心字段结构如下：

- `traceSegmentId` （ID 类型）：通过 `GlobalIdGenerator` 生成，是 `TraceSegment` 的全局唯一标识。
- `ref` （`TraceSegmentRef` 类型）：它指向父 `TraceSegment` 。在 RPC 调用、HTTP 请求等跨进程调用中，一个 `TraceSegment` 最多只有一个父 `TraceSegment` ，但是在一个 `Consumer` 批量消费 MQ 消息时，同一批内的消息可能来自不同的 `Producer` ，这就会导致 `Consumer` 线程对应的 `TraceSegment` 有多个父 `TraceSegment` 了，系统只保留第一个父 `TraceSegment` ,早期版本是保留了全部 。
- `relatedGlobalTraceId` （`DistributedTraceIds` 类型）：记录当前 `TraceSegment` 所属 `Trace` 的 Trace ID，批处理场景下也只保留第一个。
- `spans` （`List<AbstractTracingSpan>` 类型）：当前 `TraceSegment` 包含的所有 `Span` 。
- `ignore` （`boolean` 类型）： `ignore` 字段表示当前 `TraceSegment` 是否被忽略。主要是为了忽略一些问题 `TraceSegment` （据说是对只包含一个 `Span` 的 `Trace` 进行采样收集）。
- `isSizeLimited` （boolean 类型）：每个 `TraceSegment` 中 `Span` 的个数是有上限的（默认值为 300，可动态配置），超过上限之后，就不再添加 `Span` 了；这是一个内存保护措施。

Span

skywalking中 `Span` 分为 2大类， `RemoteSpan` 和 `LocalSpan` ，其中 `RemoteSpan` 又分为 `EntrySpan` 和 `ExitSpan` ：

- `EntrySpan` ：当请求进入服务时，会创建 `EntrySpan` 类型的 `Span` ，它也是 `TraceSegment` 中的第一个 `Span` 。例如，HTTP 服务、RPC 服务、MQ-Consumer 等入口服务的插件在接[illegible]求时都会创建相应的 `EntrySpan` 。

rock_fish 关注
总资产6

21、Skywalking的埋点-Agent动态采样控制
阅读 219

22、skywalking的Trace数据协议
阅读 12

skywalking的日常维护1：com.netflix.zuul.exception.ZuulExcept
阅读 41

推荐阅读

RPC框架（1 - 实现服务端注册一个服务）
阅读 308

RPC的底层原理
阅读 1,429

[Spring MVC]HandlerMapping的初始化
阅读 266

Java2021高频面试题(含答案，适合面试前冲刺一下快速记忆)
阅读 3,530

（seata源码研究）JAVA代码实现eureka的服务注册、服务下线、服...
阅读 880

Span 。

- LocalSpan ： 它是在本地方法调用时可能创建的 Span 类型，处于 EntrySpan 和 ExitSpan 之间，应用程序中可通过@Trace 注解标注在方法上创建一个 LocalSpan 。

TracingContext

每个 TraceSegment 都绑定一个 TracingContext 上下文对象，记录了 TraceSegment 的上下文信息。提供的功能有：

- 管理 TraceSegment 生命周期
- 创建 Span 比如三个创建Span的方法 #createEntrySpan 、 #createLocalSpan 方法、#createExitSpan
- 跨进程传播上下文
- 跨线程传播上下文

跨进程传播

ContextCarrier 见名知意，是 Context 的搬运工（Carrier），负责在**进程**之间搬运 Context 的一些基本信息，将夸进程调用链 连接起来。

看下其成员的作用：

- traceId （String 类型）： 它记录了当前 Trace ID。
- traceSegmentId （ID 类型）： 从 Client 端看，它记录了 Client 中 `TraceSegment ID；从 Server 端看，记录的是父 TraceSegment 的 ID。
- spanId （int 类型）： 从 Client 端看，它记录了当前 ExitSpan 的 ID；从 Server 端看，记录的是父 Span 的 ID。
- parentService ： 它记录的是 Client 的服务描述信息
- parentServiceInstance ： 它记录的是 Client 服务实例的描述信息
- parentEndpoint （String 类型）： 它记录了 Client 端的EndpointName
- addressUsedAtClient （String 类型）： 它记录了 client端请求Server 端的地址(ip:port, hostname:port) 。
- extensionContext （ExtensionContext 类型）： 作为扩展上下文 ， 其内包含一些可选上下文，以增强某些场景中的分析。
- correlationContext （CorrelationContext 类型）： 作为容器，用于放置用户自定义的Context 信息。

跨进程传播 Context 上下文信息的核心流程大致为：

- 远程调用的 Client 端会调用 inject(ContextCarrier) 方法，将当前 TracingContext 中记录的 Trace 上下文信息填充到传入的 ContextCarrier 对象。
- 后续 Client 端的插件会将 ContextCarrier 对象序列化成字符串并将其作为附加信息添加到请

对象中取出 Context 上下文信息，填充到当前 `TracingContext` （以及 `TraceSegmentRef` ）中。

对于Dubbo组件来说，其 `ContextCarrier` 的传播过程如下图所示:

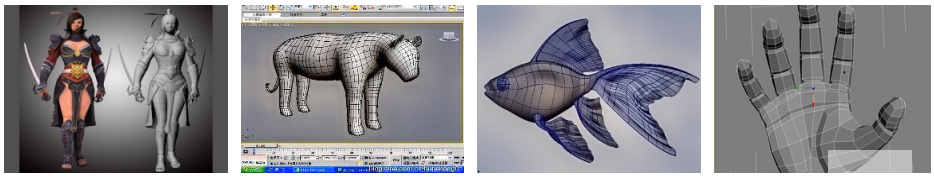
image.png

序列化之后的 `ContextCarrier` 字符串会利用 `attachments` 的机制放到 `RpcContext` 中，在服务端从 `attachments` 中取出反序列化后填充到当前 `TraceContext` 中。

跨线程转播

跨线程转播，是在同一个进程中，不同的线程之间传递，这个传递过程不需要序列化，遵循以下步骤实现：

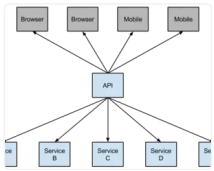
- 调用 `ContextManager#capture` 方法获取 `ContextSnapshot` 对象
- 把这个 `ContextSnapshot` 对象传递给子线程
- 在子线程中调用 `ContextManager#continued(ContextSnapshot snapshot)` 方法



千淘萬漉 阅读 2,178 评论 0 赞 6

Spring Cloud

Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智...



卡卡罗2017 阅读 120,800 评论 16 赞 134

中型车销量排行榜前十名

Linux内核学习1——基本概念

linux源码下载地址：http://www.kernel.org 《深入理解Linux内核》读书笔记 1.Lin...

小可哥哥V 阅读 194 评论 0 赞 0

一个网站访问从输入URL到页面加载的过程丨WEB前端开发 前端开发者

前端开发者丨http请求 https:www.rokub.com 前言见解有限， 如有描述不当之处， 请帮忙指出，...

麋鹿_720a 阅读 9,498 评论 11 赞 30

Python进程、线程、回调与协程 总结笔记 适合新手明确基本概念

怎样让python在现代的机器上运行的更快，充分利用多个核心，有效地实现并行、并发一直是人们的追求方向。 GIL ...

treelake 阅读 6,611 评论 2 赞 75