PathTracer client http-api 1. req 8. res 7.更新PathTrace 2. 初始化PathTrace server 1 3. req 6. res 4. 更新PathTrace 5. 更新PathTrace server 2 Request 初始化 PathTrace,存入 req中 逻辑处理 异常发生, 更新req中的PathTrace 第三方服务调用, 更新PathTrace 信息,必要时将exception信息更新 并传递给第三方服务 到PathTrace中 PathTrace request 开始请求处理,用request带 来的PathTrace初始化当前的 PathTrace, 并更新 PathTrace信息 逻辑处理完毕, 并返回逻 PathTrace 辑处理, 更新PathTrace 返回逻辑处理 Reponse 将PathTrace写入 第三方日志系统 Response

【开始一个请求处理】

pathTracer:

【闭路】

生统一的回路,并从起始服务处处理response。

begin

-req-start

service 1

特点:

相互嵌套。

service 1

{ // [string] 当前trace的标识ID,每一个path trace都有独一的ID。 id // [string] 标识当前pathtrace的类型。 type // [string] 开始处理请求的时间戳。 date // [string] 客户端发起请求的api路由。 router pre_path // [array] 在开路的path tracer中, pre_path保存前一路的栈信息。 // [array] 用于记录每一次操作的行为。 path type: closed(default) I open // 如果在服务调用过程中,存在不用返回的业务。在调用点将type置为open, 成为开路。那么,closed为闭路。 pre_path: [[...], [...], ...] // 仅当前path trace为开路时才有值。 path: {sign: "begin", service_name: "service_1", exception: "", extra: ""}, {sign: "req-start", service_name: "service_1", router: "", extra: ""}, service_name: "service_2", exception: "", extra: ""}, {sign: "start", {sign: "finish", service_name: "service_2", exception: "", extra: ""}, {sign: "req-finish", service_name: "service_1",router: "", extra: ""}, service_name: "service_1", exception: "", extra: ""} {sign: "end", sign: "begin I end I req-start I req-finish I start I finish", // 表示符号。表示链路中的节点类型。 service_name: "service_1", // 当前行为所处的服务的值。(每个服务都应该有一个唯一的值) router: "/v3/api/xxx", // 记录跨服务的api exception: "", // 当前服务发生异常的信息。 extra: "" // 放一些额外的信息。

闭路path trace能非常完整的收集到整个链路的服务调用栈信息。

service 2

start

service 3

是指服务从收到请求开始,无论中间需要依赖多少次其他服务(或嵌套多少次其他服务),均会产

闭路/开路

```
-req-start-
                                                                       start
                                                                       finish
                                         req-finish-
                                  finish
     -req-finish-
 end
                    end: [service 1]
              req-finish: [to -> service 2]
                   finish: [service 2]
              req-finish: [to -> service 3]
                   finish: [service 3]
stack
                    start: [service 3]
               req-start: [to -> service 3]
                    start: [service 2]
               req-start: [to -> service 2]
                   begin: [service 1]
```

```
【开路】
是指服务从收到请求开始,中间调用了第三方微服务,并且不需要第三方服务的返回。从而,服务
的链路和第三方服务执行的链路形成了两条独立的pathtrace。
```

service 2

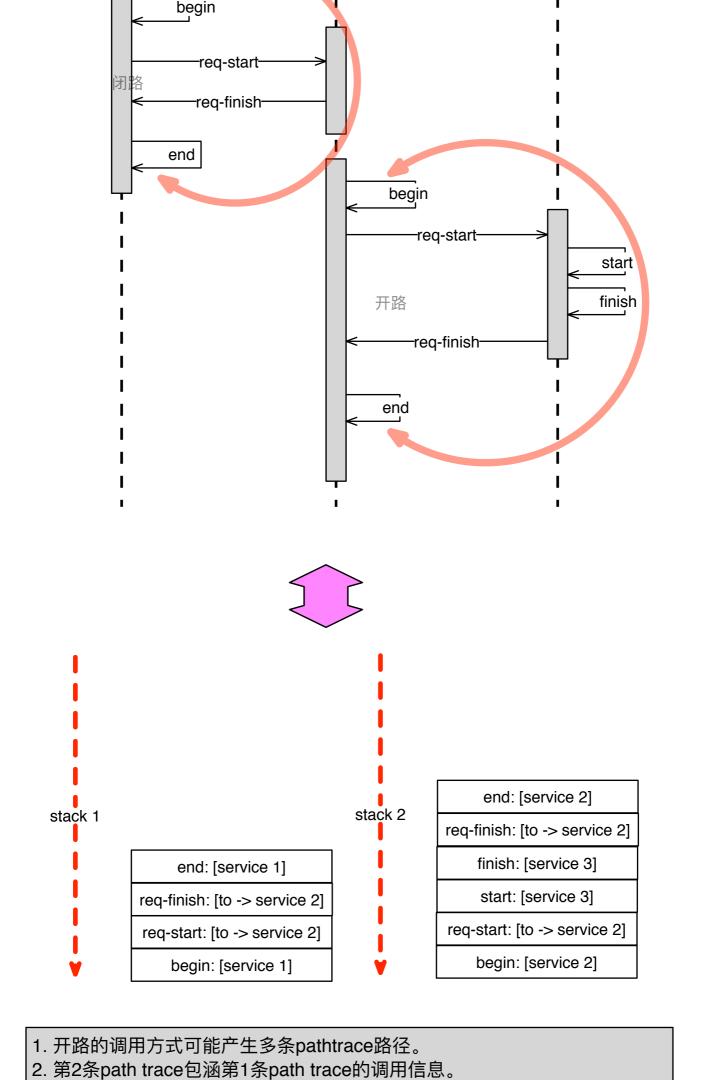
service 3

1. 一条path trace只有一对begin - end。begin代表最开始,end代表结束。

4. 一条path trace可以包含多条start - finish和 regstart —regfinish,而且可以

3. regstart - regfinish 代表当前服务的逻辑需要依赖其他服务。

2. start - end 代表服务处理当前收到的请求。



3. 多条path trace之间通过 同一个id 来相互关联。

4. 其他同闭路的特点。