**概要：**

1. Dubbo执行过程
2. 调用端埋点实现(消费者)
3. 接收端埋点实现（提供者）

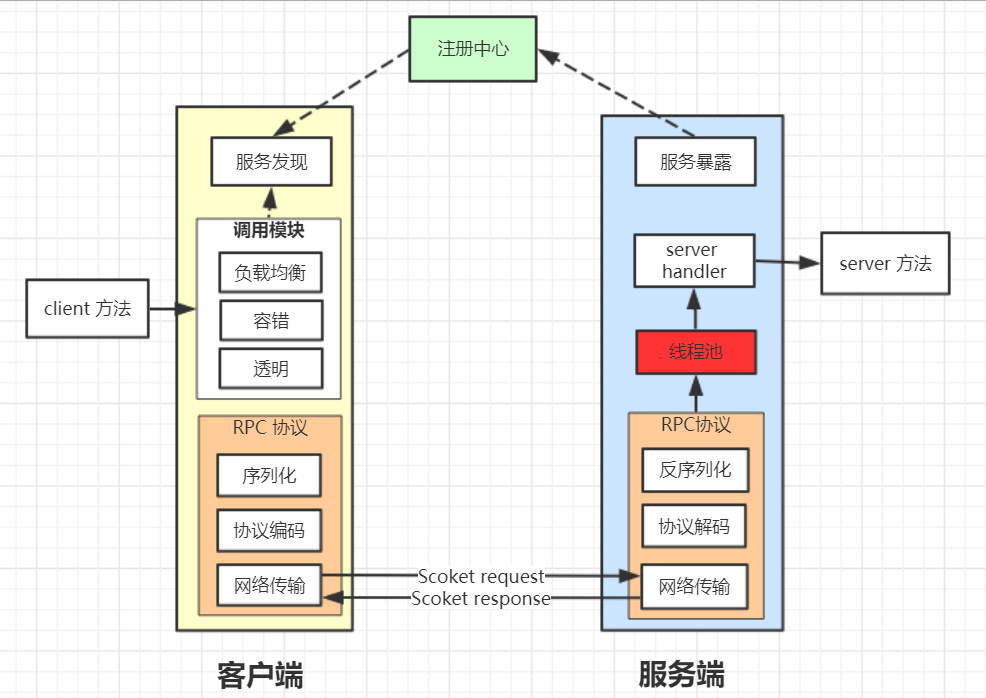
一、Dubbo执行过程

**Dubbo 基本架构：**

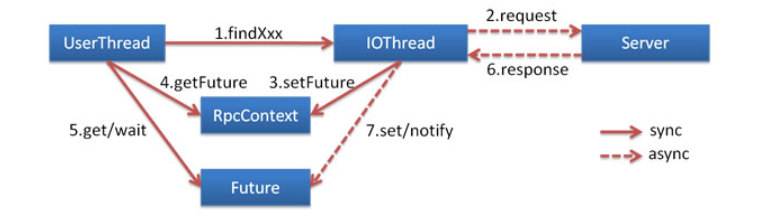


|  |  |
| --- | --- |
| **节点** | **角色说明** |
| Provider | 暴露服务的服务提供方 |
| Consumer | 调用远程服务的服务消费方 |
| Registry | 服务注册与发现的注册中心 |
| Monitor | 统计服务的调用次数和调用时间的监控中心 |
| Container | 服务运行容器 |

**Dubbo调用过程 ：**



**消费者调用过程：**



二、调用端埋点实现

**概要：**

1. 埋点的目的
2. 埋点的位置

**埋点目的：**

1. 捕捉消费者调用信息（远程接口、URL、参数、用时、返回结果、异常）
2. 传递TraceRequest

功能演示:

* 演示Dubbo 基本信息捕捉
* 演示Dubbo 异常信息捕捉

**调用信息模型表结构：**

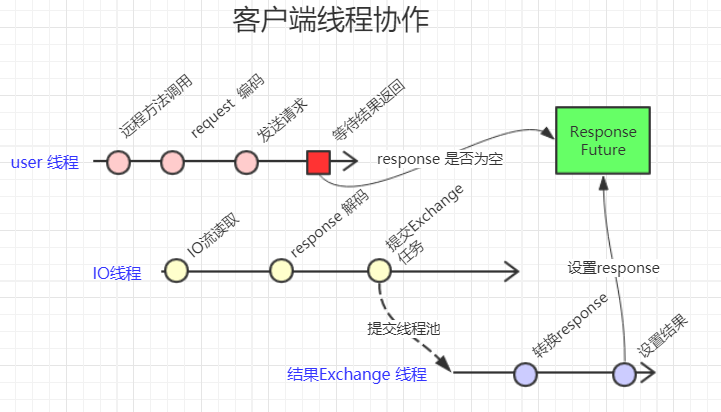
|  |  |  |
| --- | --- | --- |
| **名称** | **类型** | **描述** |
| servicePath | string | 服务路径 |
| serviceName | string | 服务 |
| inParam | json | 返回结果 |
| outParam | json | 返回结果 |
| ErrorMessage | string | 异常信息 |
| ErrorStack | text | 异常堆栈 |
| ResultState | string | 执行状态 |
| beginTime | date | 开始时间 |
| endTime | date | 结束时间 |
| addressIp | string | 远程IP |
| fromIp | string | 调用者IP |

**埋点位置：**

如何才能完整的捕捉到以上信息呢？那么就需要了解Dubbo内部的调用过程：

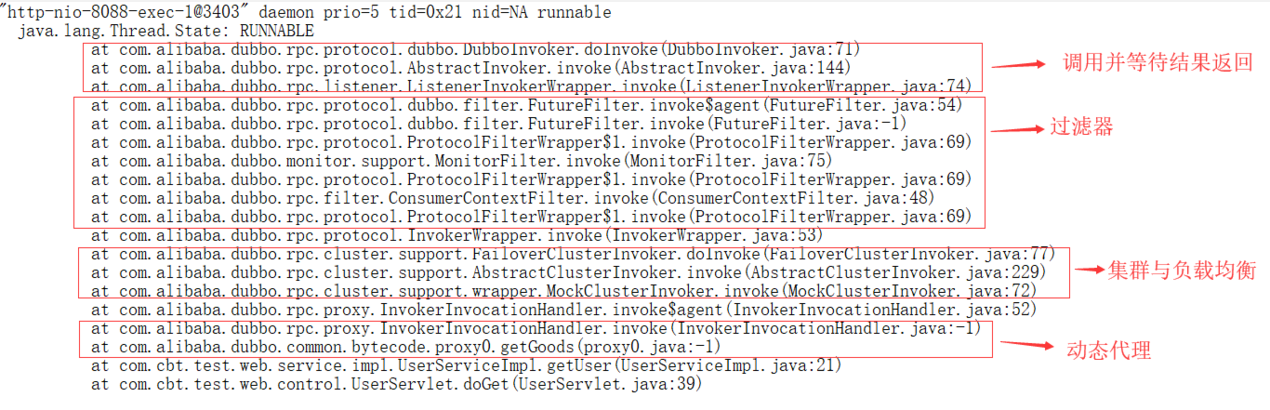
1. 分解调用过程为多个步骤。
2. 这些步骤分别是在哪些协作线程上完成的？
3. 经过了哪些方法？
4. 经过了哪些过滤器？

1、调用过程分解&线程协作：



2、经过了哪些方法&过滤器

* 选择断点位置Debug调试调用过程

消费者调用线程源码分析：

经过对源码的分析，埋点的位置如下:

DubboInvoker.doInvoke()

FutureFilter.invoke()

DubboInvoker.doInvoke() 方法最靠近调用方，异常捕捉范围较大，但是该位置无法通过Attachment 向下传递TraceRequest 参数，所以需要FutureFilter.invoke() 进行补充，其具体分工如下：

|  |  |
| --- | --- |
| DubboInvoker.doInvoke | FutureFilter.invoke |
| 捕获如下信息： 1、开始时间 2、服务路径 3、服务方法 4、输入参数 5、异常信息 6、本地地址 | 1、基于Attachment 向下传递参数 2、异常信息与堆栈 3、返回结果 |

*DubboInvoker.doInvoke拦截源码参见 ：*com.cbt.agent.collects.dubbo.DubboConsumerRpcExceptionMonitorHandle#invokerBefore

*FutureFilter.invoke拦截源码参见 ：*

*com.cbt.agent.collects.dubbo.DubboConsumerMonitorHandle#invokerBefore*

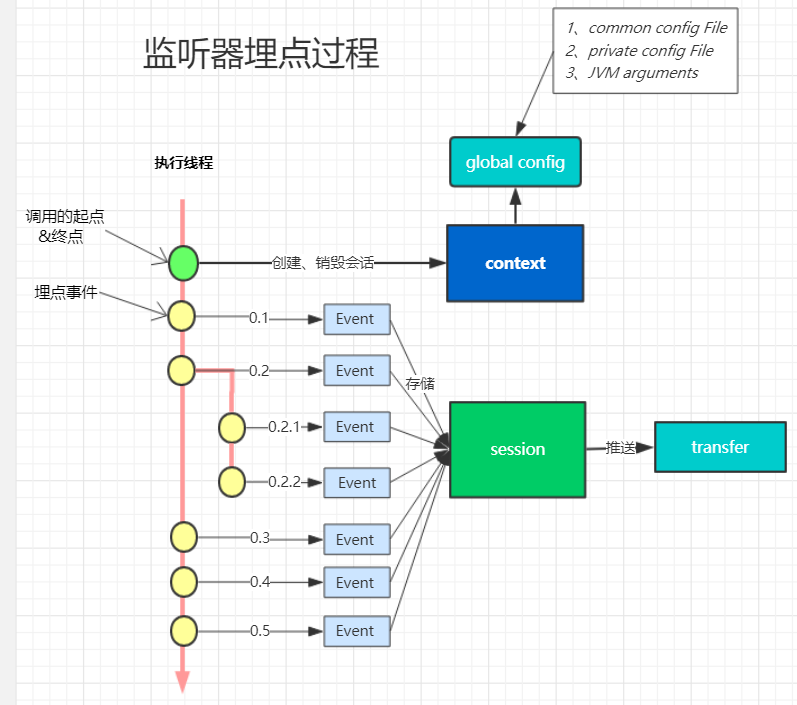
三、接收端埋点实现

**知识点：**

1. 埋点目的
2. 埋点位置
3. 具体实现

**埋点目的**

接收TraceRequest信息 ，并创建会话

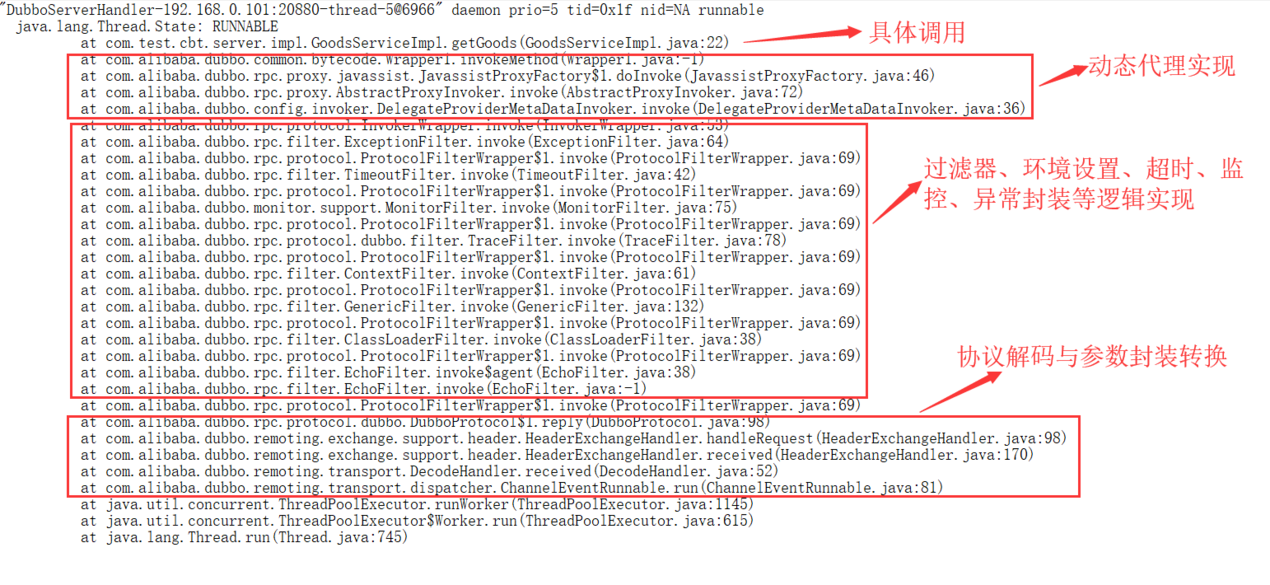


**埋点位置：**

相对调用广方接收方埋点目的较简单，但同样需分析源码找准埋点位置：

* 选择断点位置Debug调试调用过程

**提供者处理线程分析**



经分析埋点位置选在离实际调用方法较远的EchoFilter过滤器理由是捕捉的信息更全面。

**具体会话开启过程：**

1. 基于Attachment获取TraceId、ParentId、TraceProperties。
2. 封装TraceRequest ，并此为参数开启会话。
3. 在调用结束时关闭会话。

*具体源码参见：com.cbt.agent.collects.dubbo.DubboProviderMonitorHandle#invokerBefore*