PRC原理和使用

RPC（Remote Procedure Call）即远程过程调用，通过名字我们就能看出 RPC 关注的是**远程调用**而非本地调用。

为什么要 RPC ？

因为，**两个不同的服务器上的服务提供的方法不在一个内存空间**，所以，需要通过网络编程才能传递方法调用所需要的参数。并且，方法调用的结果也需要通过网络编程来接收。但是，如果我们自己手动网络编程来实现这个调用过程的话工作量是非常大的，因为，我们需要考虑底层传输方式（TCP还是UDP）、序列化方式等等方面。

RPC 能帮助我们做什么呢？

简单来说，通过 RPC 可以帮助我们调用远程计算机上某个服务的方法，这个过程就像调用本地方法一样简单。并且！我们不需要了解底层网络编程的具体细节。

举个例子：两个不同的服务 A、B 部署在两台不同的机器上，服务 A 如果想要调用服务 B 中的某个方法的话就可以通过 RPC 来做。

一言蔽之：**RPC 的出现就是为了让你调用远程方法像调用本地方法一样简单。**

既有 HTTP ,为啥用 RPC 进行服务调用?

RPC 只是一种设计而已

RPC **只是一种概念、一种设计**，就是为了解决 不同服务之间的调用问题, 它一般会包含有 **传输协议** 和 **序列化协议** 这两个。

但是，HTTP 是一种协议，RPC框架可以使用 HTTP协议作为传输协议或者直接使用TCP作为传输协议，使用不同的协议一般也是为了适应不同的场景。

简单来说成熟的rpc库相对http容器，更多的是封装了“服务发现”，"负载均衡"，“熔断降级”一类面向服务的高级特性。可以这么理解，rpc框架是面向服务的更高级的封装。如果把一个http servlet容器上封装一层服务发现和函数代理调用，那它就已经可以做一个rpc框架了。所以为什么要用rpc调用？因为良好的rpc调用是面向服务的封装，针对服务的可用性和效率等都做了优化。单纯使用http调用则缺少了这些特性。（https://www.zhihu.com/question/41609070?sort=created）

RPC的原理主要用到了动态代理模式，至于HTTP协议，只是传输协议而已。

RPC原理

c/s模式

1. 客户端（服务消费端）：调用远程方法的一端。
2. 客户端 Stub（桩）：这其实就是一代理类。代理类主要做的事情很简单，就是把你调用方法、类、方法参数等信息传递到服务端。
3. 网络传输：网络传输就是你要把你调用的方法的信息比如说参数啊这些东西传输到服务端，然后服务端执行完之后再把返回结果通过网络传输给你传输回来。网络传输的实现方式有很多种比如最近基本的

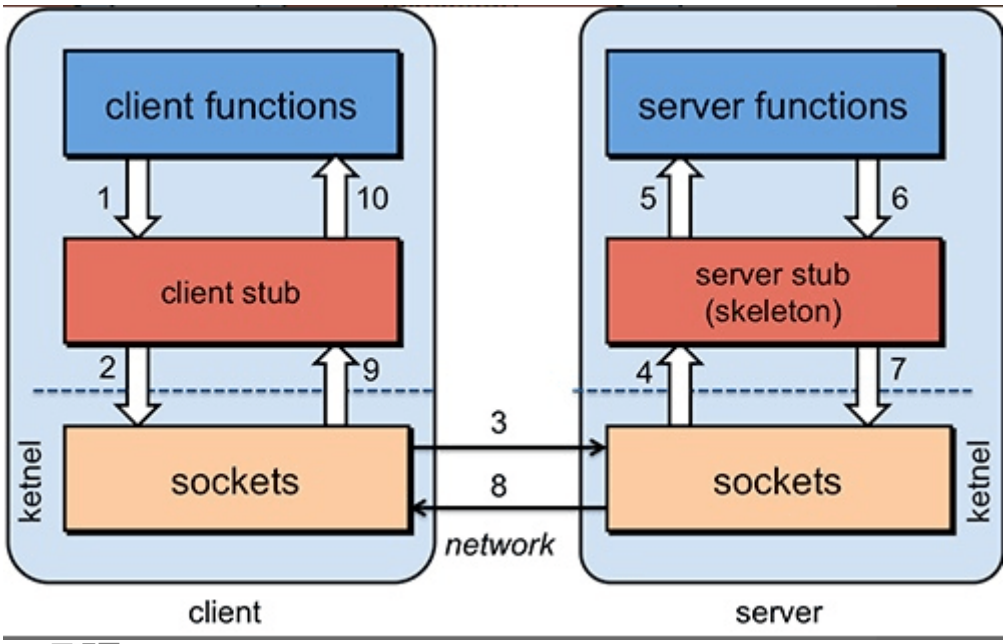
Socket或者性能以及封装更加优秀的 Netty（推荐）。（Netty 是一个基于 JAVA NIO 类库的异步通信框架，它的架构特点是：异步非阻塞、基于事件驱动、高性能、高可靠性和高可定制性。10.Netty应用场景：

1.分布式开源框架中dubbo、Zookeeper，RocketMQ底层rpc通讯使用就是netty。

2.游戏开发中，底层使用netty通讯。）

4. 服务端 Stub（桩）：这个桩就不是代理类了。我觉得理解为桩实际不太好，大家注意一下就好。这里的服务端 Stub 实际指的就是接收到客户端执行方法的请求后，去指定对应的方法然后返回结果给客户端的类。
5. 服务端（服务提供端）：提供远程方法的一端。

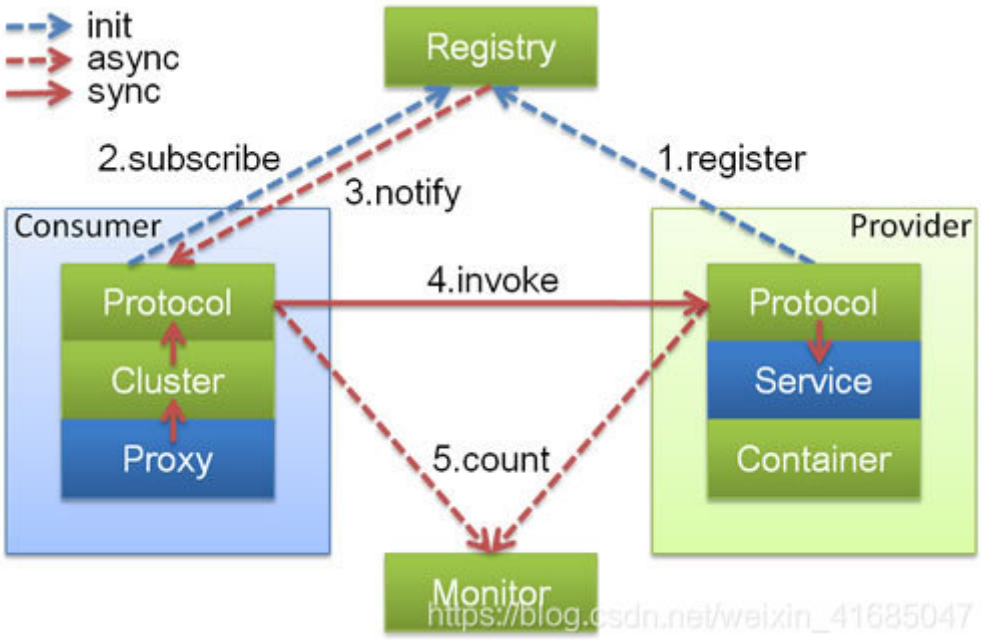
具体原理图如下，后面我会串起来将整个RPC的过程给大家说一下。

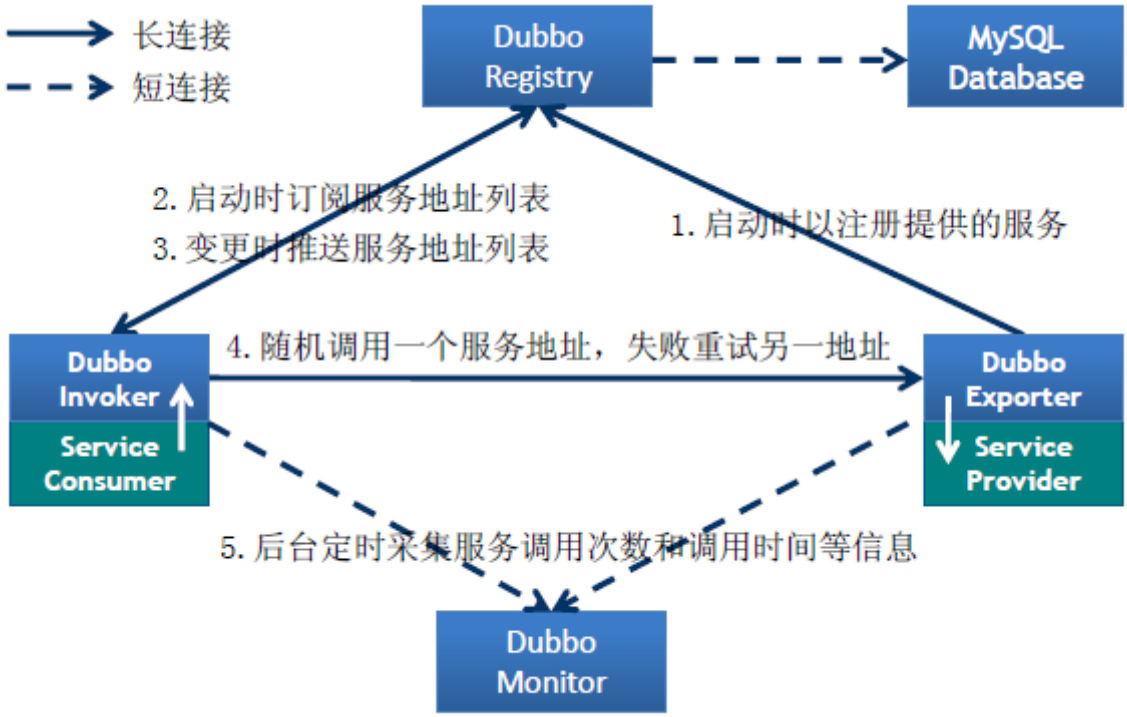


RPC原理图

1. 服务消费端（client）以本地调用的方式调用远程服务；
2. 客户端 Stub（client stub）接收到调用后负责将方法、参数等组装成能够进行网络传输的消息体（序列化）：`RpcRequest`；
3. 客户端 Stub（client stub）找到远程服务的地址，并将消息发送到服务提供端；
4. 服务端 Stub（桩）收到消息将消息反序列化为Java对象: `RpcRequest`；
5. 服务端 Stub（桩）根据`RpcRequest`中的类、方法、方法参数等信息调用本地的方法；
6. 服务端 Stub（桩）得到方法执行结果并将组装成能够进行网络传输的消息体：`RpcResponse`（序列化）发送至消费方；
7. 客户端 Stub（client stub）接收到消息并将消息反序列化为Java对象:`RpcResponse`，这样也就得到了最终结果。over!

dubbo 的原理：





注册中心

基础流程

服务启动时，将自身的网络地址等信息注册到注册中心，注册中心记录服务注册数据。服务消费者从注册中心获取服务提供者的地址，并通过地址和基于特定的方式调用服务提供者的接口。各个服务与注册中心使用一定机制通信。如果注册中心与服务长时间无法通信，就会注销该实例，这也称为服务下线，当服务重新连接之后，会基于一定的策略在线上线。服务地址相关信息发生变化时，会重新注册到注册中心。这样，服务消费者就无需手工维护提供者的相关配置。**核心功能**

通过上面的基本流程，不难发现一个注册中心需要具备哪些核心功能：

服务发现 服务发现是指服务在启动后，注册到注册中心，服务方提供自身的元数据，比如IP地址、端口、运行状况指标的Uri 、主页地址等信息。

服务记录 记录注册中心的服务的信息，例如服务名称、IP地址、端口等。服务消费方基于查询获取可用的服务实例列表。

动态管理服务 注册中心基于特定的机制定时测试已注册的服务，例如：默认的情况下会每隔30秒发送一次心跳来进行服务续约。通过服务续约来告知Server该Client仍然可用。正常情况下，如果Server在90 秒内没有收到Client 的心跳，Server会将Client 实例从注册列表中删除。