<https://mp.weixin.qq.com/s/E9mpz7uRM7n0borSzYlvJw>

# 直观讲解一下RPC调用和HTTP调用的区别

很长时间以来都没有怎么好好搞清楚RPC（即Remote Procedure Call，远程过程调用）和HTTP调用的区别，不都是写一个服务然后在客户端调用么？这里请允许我迷之一笑~Naive！

本文简单地介绍一下两种形式的C/S架构，先说一下他们最本质的区别，**就是RPC主要是基于TCP/IP协议的，而HTTP服务主要是基于HTTP协议的**，我们都知道HTTP协议是在传输层协议TCP之上的，所以效率来看的话，RPC当然是要更胜一筹啦！下面来具体说一说RPC服务和HTTP服务。

# **OSI网络七层模型**

在说RPC和HTTP的区别之前，我觉的有必要了解一下OSI的七层网络结构模型（虽然实际应用中基本上都是五层），它可以分为以下几层：（从上到下）

**第一层：应用层。定义了用于在网络中进行通信和传输数据的接口；**

**第二层：表示层。定义不同的系统中数据的传输格式，编码和解码规范等；**

**第三层：会话层。管理用户的会话，控制用户间逻辑连接的建立和中断；**

**第四层：传输层。管理着网络中的端到端的数据传输；**

**第五层：网络层。定义网络设备间如何传输数据；**

**第六层：链路层。将上面的网络层的数据包封装成数据帧，便于物理层传输；**

**第七层：物理层。这一层主要就是传输这些二进制数据。**

实际应用过程中，五层协议结构里面是没有表示层和会话层的。应该说它们和应用层合并了。我们应该将重点放在应用层和传输层这两个层面。因为HTTP是应用层协议，而TCP是传输层协议。好，知道了网络的分层模型以后我们可以更好地理解为什么RPC服务相比HTTP服务要Nice一些！

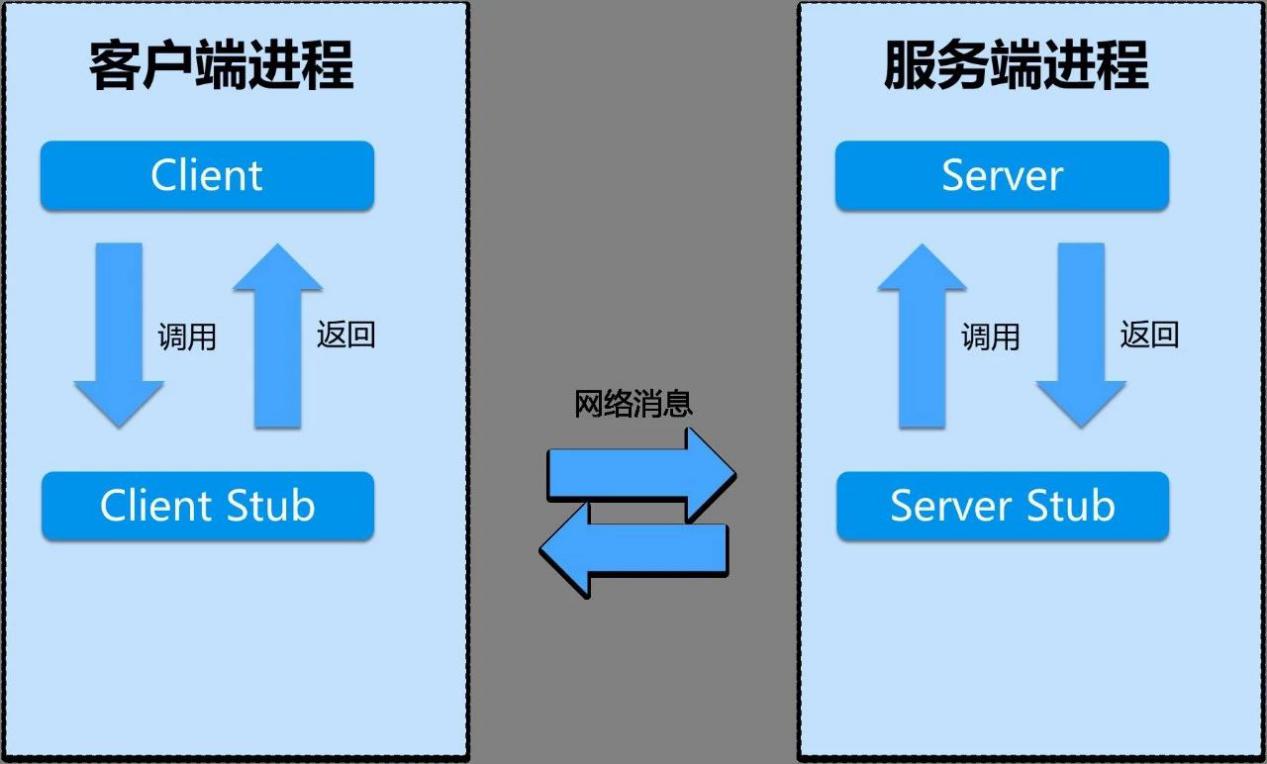
# **RPC服务**

从三个角度来介绍RPC服务：分别是RPC架构，同步异步调用以及流行的RPC框架。

## RPC架构

先说说RPC服务的基本架构吧。允许我可耻地盗一幅图哈~我们可以很清楚地看到，一个完整的RPC架构里面包含了四个核心的组件，分别是Client ,Server,Client Stub以及Server Stub，这个Stub大家可以理解为存根。分别说说这几个组件：

* 客户端（Client），服务的调用方。
* 服务端（Server），真正的服务提供者。
* 客户端存根，存放服务端的地址消息，再将客户端的请求参数打包成网络消息，然后通过网络远程发送给服务方。
* 服务端存根，接收客户端发送过来的消息，将消息解包，并调用本地的方法。



RPC主要是用在大型企业里面，因为大型企业里面系统繁多，业务线复杂，而且效率优势非常重要的一块，这个时候RPC的优势就比较明显了。实际的开发当中是这么做的，项目一般使用maven来管理。

比如我们有一个处理订单的系统服务，先声明它的所有的接口（这里就是具体指Java中的interface），然后将整个项目打包为一个jar包，服务端这边引入这个二方库，然后实现相应的功能，客户端这边也只需要引入这个二方库即可调用了。为什么这么做？主要是为了减少客户端这边的jar包大小，因为每一次打包发布的时候，jar包太多总是会影响效率。另外也是将客户端和服务端解耦，提高代码的可移植性。

## 同步调用与异步调用

什么是同步调用？什么是异步调用？同步调用就是客户端等待调用执行完成并返回结果。异步调用就是客户端不等待调用执行完成返回结果，不过依然可以通过回调函数等接收到返回结果的通知。如果客户端并不关心结果，则可以变成一个单向的调用。

这个过程有点类似于Java中的callable和runnable接口，我们进行异步执行的时候，如果需要知道执行的结果，就可以使用callable接口，并且可以通过Future类获取到异步执行的结果信息。如果不关心执行的结果，直接使用runnable接口就可以了，因为它不返回结果，当然啦，callable也是可以的，我们不去获取Future就可以了。

## 流行的RPC框架

目前流行的开源RPC框架还是比较多的。下面重点介绍三种：

* gRPC是Google最近公布的开源软件，基于最新的HTTP2.0协议，并支持常见的众多编程语言。我们知道HTTP2.0是基于二进制的HTTP协议升级版本，目前各大浏览器都在快马加鞭的加以支持。这个RPC框架是基于HTTP协议实现的，底层使用到了Netty框架的支持。
* Thrift是Facebook的一个开源项目，主要是一个跨语言的服务开发框架。它有一个代码生成器来对它所定义的IDL定义文件自动生成服务代码框架。用户只要在其之前进行二次开发就行，对于底层的RPC通讯等都是透明的。不过这个对于用户来说的话需要学习特定领域语言这个特性，还是有一定成本的。
* Dubbo是阿里集团开源的一个极为出名的RPC框架，在很多互联网公司和企业应用中广泛使用。协议和序列化框架都可以插拔是及其鲜明的特色。同样 的远程接口是基于Java Interface，并且依托于spring框架方便开发。可以方便的打包成单一文件，独立进程运行，和现在的微服务概念一致。  
  偷偷告诉你集团内部已经不怎么使用dubbo啦，现在用的比较多的叫HSF，又名“好舒服”。后面有可能会开源，大家拭目以待。

# **HTTP服务**

其实在很久以前，我对于企业开发的模式一直定性为HTTP接口开发，也就是我们常说的RESTful风格的服务接口。的确，对于在接口不多、系统与系统交互较少的情况下，解决信息孤岛初期常使用的一种通信手段；优点就是简单、直接、开发方便。

利用现成的http协议进行传输。我们记得之前本科实习在公司做后台开发的时候，主要就是进行接口的开发，还要写一大份接口文档，严格地标明输入输出是什么？说清楚每一个接口的请求方法，以及请求参数需要注意的事项等。

比如下面这个例子：  
POST http://www.httpexample.com/restful/buyer/info/shar

接口可能返回一个JSON字符串或者是XML文档。然后客户端再去处理这个返回的信息，从而可以比较快速地进行开发。

但是对于大型企业来说，内部子系统较多、接口非常多的情况下，RPC框架的好处就显示出来了，首先就是长链接，不必每次通信都要像http一样去3次握手什么的，减少了网络开销；其次就是RPC框架一般都有注册中心，有丰富的监控管理；发布、下线接口、动态扩展等，对调用方来说是无感知、统一化的操作。

# **总结**

RPC服务和HTTP服务还是存在很多的不同点的，一般来说，RPC服务主要是针对大型企业的，而HTTP服务主要是针对小企业的，因为RPC效率更高，而HTTP服务开发迭代会更快。

总之，选用什么样的框架不是按照市场上流行什么而决定的，而是要对整个项目进行完整地评估，从而在仔细比较两种开发框架对于整个项目的影响，最后再决定什么才是最适合这个项目的。一定不要为了使用RPC而每个项目都用RPC，而是要因地制宜，具体情况具体分析。