## 简介

rpc的意思就是，api端和server端之间的远程调用的信息不走http协议了，把消息序列化成二进制的，然后让消息变成tcp的二进制的buffer，这样只是tcp层的交互，整个流程会快很多。

thrift主要用于各个服务之间的RPC通信，支持跨语言。thrift是一个典型的CS结构，客户端和服务端可以使用不同的语言开发，thrift通过IDL(Interface Description Language)来关联客户端和服务端。

Idl interface description language 接口定义语言。

所有这些步骤的效果是，将客户过程对客户桩发出的本地调用转换成对服务器过程的本地调用，而客户端和服务器都不会意识到有中间步骤的存在。

## restful和rpc和二进制序列化协议

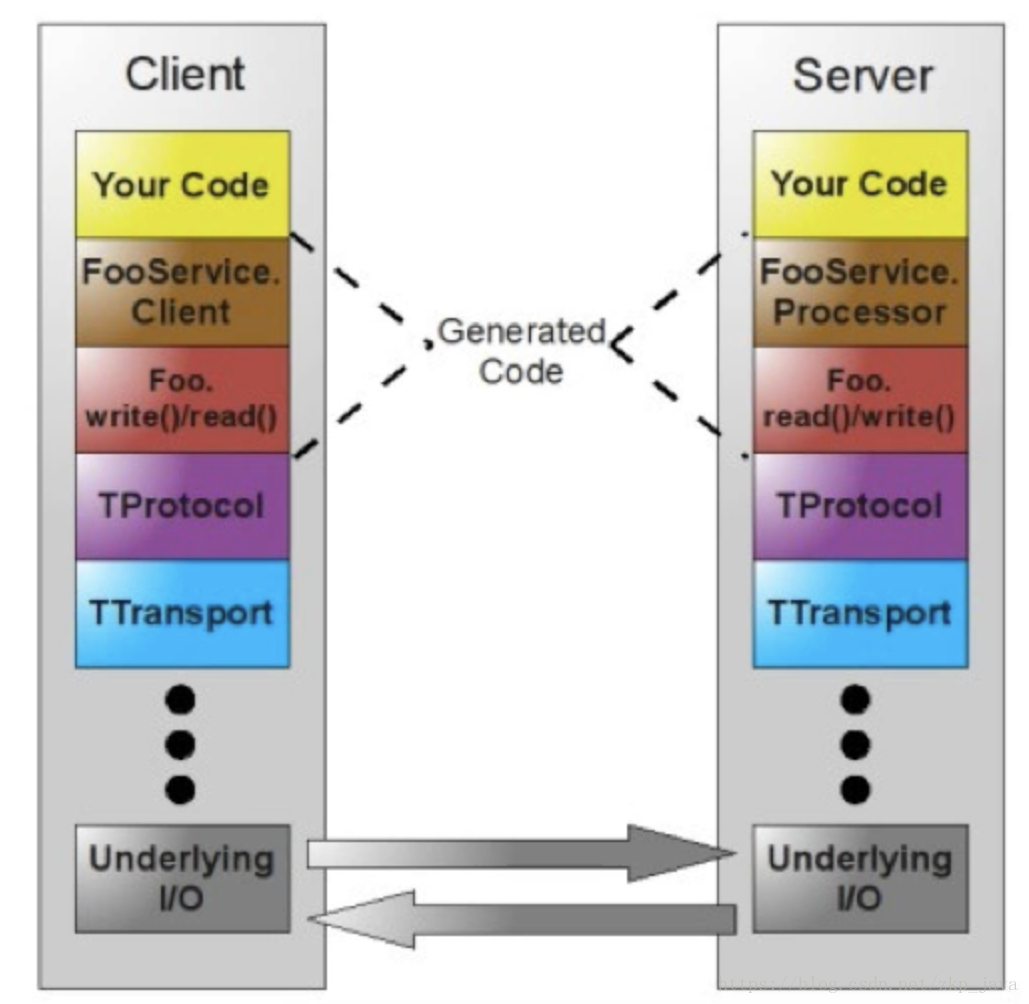
这个时候，你可能会想，既然是调用另一台机器的服务，使用 RESTful API 也可以实现啊，为什么要选择 RPC 呢？我们可以从两个方面对比：

资源粒度。RPC 就像本地方法调用，RESTful API 每一次添加接口都可能需要额外地组织开放接口的数据，这相当于在应用视图中再写了一次方法调用，而且它还需要维护开发接口的资源粒度、权限等；

流量消耗。RESTful API 在应用层使用 HTTP 协议，哪怕使用轻型、高效、传输效率高的 JSON 也会消耗较大的流量，而 RPC 传输既可以使用 TCP 也可以使用 UDP，而且协议一般使用二制度编码，大大降低了数据的大小，减少流量消耗。

对接异构第三方服务时，通常使用 HTPP/RESTful 等公有协议，对于内部的服务调用，应用选择性能更高的二进制私有协议。

## thrift架构



图中Your Code是用户实现的业务逻辑。

接下来的FooService.Client和Foo.write()/read()是thrift根据IDL生成的客户端和服务端的代码，对应于RPC中Client stub和Server stub。

TProtocol 用来对数据进行序列化与反序列化，具体方法包括二进制，JSON 或者 Apache Thrift 定义的格式。

TTransport 提供数据传输功能，使用 Apache Thrift 可以方便地定义一个服务并选择不同的传输协议。这里的传输协议可以选择，可以是tcp协议。

**thirft使用socket进行数据传输，数据以特定的格式发送，接收方进行解析。我们定义好thrift的IDL文件后，就可以使用thrift的编译器来生成双方语言的接口、model，在生成的model以及接口代码中会有解码编码的代码。**

## TTransport和TProtocol

TTransport层

代表thrift的数据传输方式，thrift定义了如下几种常用数据传输方式

TSocket: 阻塞式socket；

TFramedTransport: 以frame为单位进行传输，非阻塞式服务中使用；

TFileTransport: 以文件形式进行传输；

TProtocol层

代表thrift客户端和服务端之间传输数据的协议，通俗来讲就是客户端和服务端之间传输数据的格式(例如json等)，thrift定义了如下几种常见的格式

TBinaryProtocol: 二进制格式；

TCompactProtocol: 压缩格式；

TJSONProtocol: JSON格式；

TSimpleJSONProtocol: 提供只写的JSON协议；

## Server模型和IDL

thrift主要支持以下几种服务模型

TSimpleServer: 简单的单线程服务模型，常用于测试；

TThreadPoolServer: 多线程服务模型，使用标准的阻塞式IO；

TNonBlockingServer: 多线程服务模型，使用非阻塞式IO(需要使用TFramedTransport数据传输方式);

THsHaServer: THsHa引入了线程池去处理，其模型读写任务放到线程池去处理，Half-sync/Half-async处理模式，Half-async是在处理IO事件上(accept/read/write io)，Half-sync用于handler对rpc的同步处理；

Idl 接口定义语言。

## 案例代码

idl文件定义好方法和数据结构体。

namespace go echo

struct EchoReq {

1: string msg;

}

struct EchoRes {

1: string msg;

}

service Echo {

EchoRes echo(1: EchoReq req);

}

thrift --gen java src/thrift/data.thrift

执行命令生成对应语言的代码，

server 主要是设置protol和port，和ip啥的，然后设置好idl定义的接口的生成的接口的实现类，当做消息的handler，也就是逻辑所在，然后run.server()

client端，主要是通过port和protol和server ip啥的，指定出一个client对象，远程调用的时候就

Clietn.rpc的方法就可以了（）；