北京邮电大学软件学院

前沿课题讲座心得体会

报告人：赵炜 学号：2011127136

导师：吴国仕

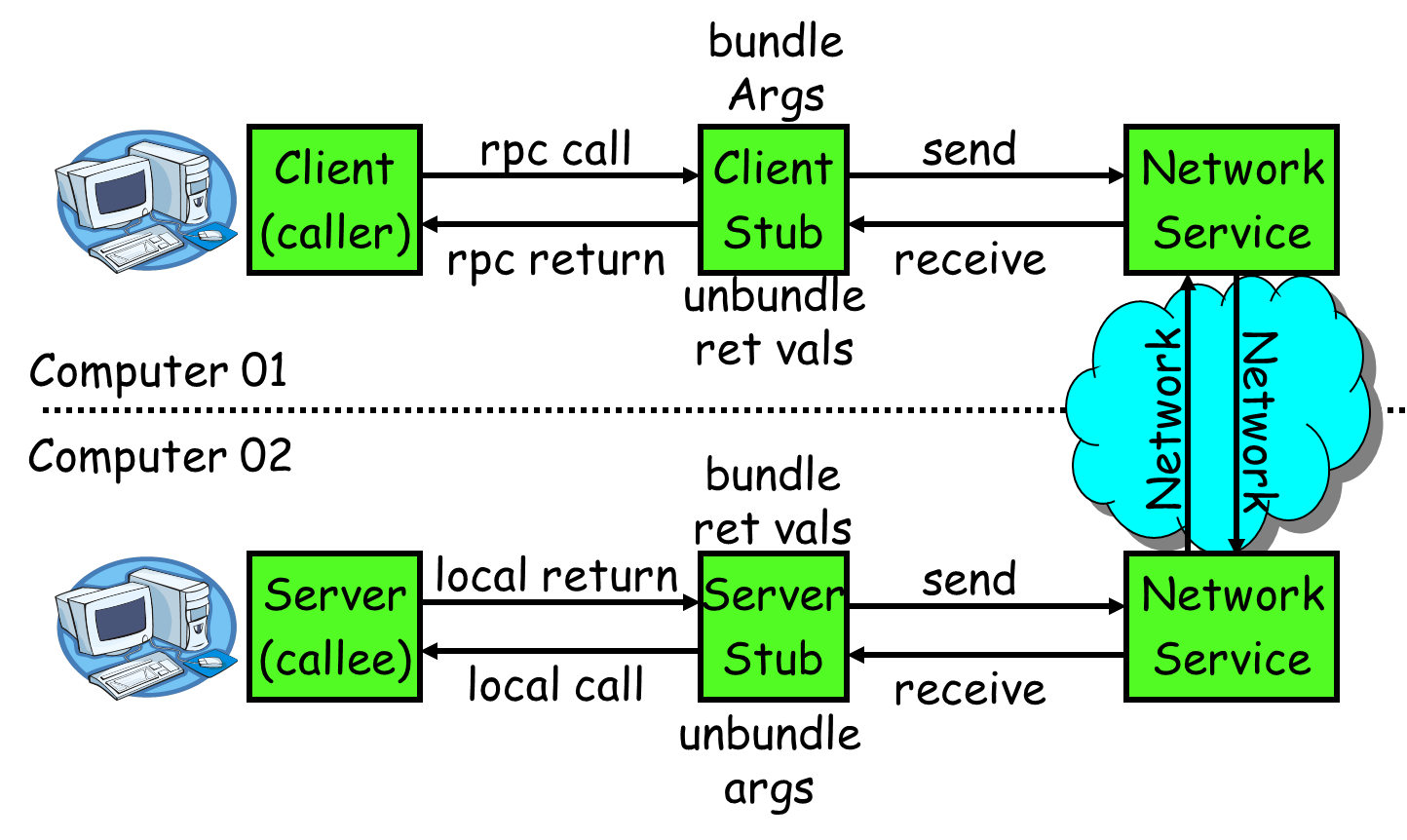
（日期：2016 年 5 月 12 日）

在北京邮电大学软件学院学习期间，我积极参加学校组织的前沿课题讲座和各大企业举办的新技术讲座，还经常通过网络了解最新技术动态。通过这些活动，开阔了视野，增长了知识。下面以现代移动互联网中热门的分布式远程方法调用 技术（RPC）与几个开源的主流技术框架gRPC、Dubbo、Thrift来谈谈个人的学习体会：

一．RPC远程方法调用技术

在校期间本人都写过不少程序，大多数程序都是集成在一个工程中的本地调用模式。这些程序的特点是服务消费方和服务提供方是本地调用关系。而一旦踏入公司尤其是大型互联网公司就会发现，公司的系统都由成千上万大大小小的服务组成，各服务部署在不同的机器上，由不同的团队负责。这时就会遇到两个问题：1）要搭建一个新服务，免不了需要依赖其它的服务，而现在其它的服务都在远端，怎么调用？2）其它团队要使用我们的服务，我们的服务该怎么发布以便他人调用？

由于各服务部署在不同机器，服务间的调用免不了网络通信过程，服务消费方每调用一个服务都要写一坨网络通信相关的代码，不仅复杂而且极易出错。如果有一种方式能让我们像调用本地服务一样调用远程服务，而让调用者对网络通信这些细节透明，那么将大大提高生产力，比如服务消费方在执行helloWorldService.sayHello(“test”)时，实质上调用的是远端的服务。这种方式其实就是RPC（Remote Procedure Call Protocol），在各大互联网公司中被广泛使用，如阿里巴巴的hsf、Dubbo（开源）、Facebook的Thrift（开源）、Google gRPC（开源）、Twitter的finagle等。要让网络通信细节对使用者透明，我们自然需要对通信细节进行封装，我们先看下一个RPC调用的流程：



1）服务消费方（client）调用以本地调用方式调用服务；

2）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；

3）client stub找到服务地址，并将消息发送到服务端；

4）server stub收到消息后进行解码；

5）server stub根据解码结果调用本地的服务；

6）本地服务执行并将结果返回给server stub；

7）server stub将返回结果打包成消息并发送至消费方；

8）client stub接收到消息，并进行解码；

9）服务消费方得到最终结果。

RPC的目标就是要2~8这些步骤都封装起来，让用户对这些细节透明。在开源社区有很多优秀的RPC框架可以使用，下面我们就来对比几个现代的几个常见的主流RPC服务框架：Dubbo、Motan、Thrift

二．gRPC

gRPC是一个高性能、通用的开源RPC框架，其由Google主要面向移动应用开发并基于HTTP/2协议标准而设计，基于ProtoBuf(Protocol Buffers)序列化协议开发，且支持众多开发语言。gRPC提供了一种简单的方法来精确地定义服务和为iOS、Android和后台支持服务自动生成可靠性很强的客户端功能库。客户端充分利用高级流和链接功能，从而有助于节省带宽、降低的TCP链接次数、节省CPU使用、和电池寿命。

gRPC具有以下重要特征：

强大的IDL特性，gRPC使用ProtoBuf来定义服务，ProtoBuf是由Google开发的一种数据序列化协议（类似于XML、JSON、hessian）。ProtoBuf能够将数据进行序列化，并广泛应用在数据存储、通信协议等方面。不过，当前gRPC仅支持 Protobuf ，且不支持在浏览器中使用。由于gRPC的设计能够支持支持多种数据格式，所以读者能够很容易实现对其他数据格式（如XML、JSON等）的支持。

定义服务的示例代码如下：

message HelloRequest {

string greeting = 1;

}

message HelloResponse {

string reply = 1;

}

service HelloService {

rpc SayHello(HelloRequest) returns (HelloResponse);

}

支持多种语言

gRPC支持多种语言，并能够基于语言自动生成客户端和服务端功能库。目前，在GitHub上已提供了C版本grpc、Java版本grpc-java 和 Go版本grpc-go，其它语言的版本正在积极开发中，其中 grpc支持C、C++、Node.js、Python、Ruby、Objective-C、PHP和C#等语言，grpc-java已经支持Android开发。

基于HTTP/2标准设计

由于gRPC基于HTTP/2标准设计，所以相对于其他RPC框架，gRPC带来了更多强大功能，如双向流、头部压缩、多复用请求等。这些功能给移动设备带来重大益处，如节省带宽、降低TCP链接次数、节省CPU使用和延长电池寿命等。同时，gRPC还能够提高了云端服务和Web应用的性能。gRPC既能够在客户端应用，也能够在服务器端应用，从而以透明的方式实现客户端和服务器端的通信和简化通信系统的构建。

gRPC应用的范围：低延迟、高扩展性、分布式的系统；同云服务器进行通信的移动应用客户端；设计语言独立、高效、精确的新协议；便于各方面扩展的分层设计，如认证、负载均衡、日志记录、监控等。

三．Dubbbo

Dubbo 是阿里巴巴公司开源的一个高性能优秀的服务框架。Dubbo致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。简单的说，Dubbo就是个服务框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有Dubbo这样的分布式服务框架的需求，并且本质上是个服务调用的东东，说白了就是个远程服务调用的分布式框架（告别Web Service模式中的WSdl，以服务者与消费者的方式在Dubbo上注册）其核心部分包含:1. 远程通讯: 提供对多种基于长连接的NIO框架抽象封装，包括多种线程模型，序列化，以及“请求-响应”模式的信息交换方式。2. 集群容错: 提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持。3. 自动发现: 基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器。Dubbo的整体框架如下：



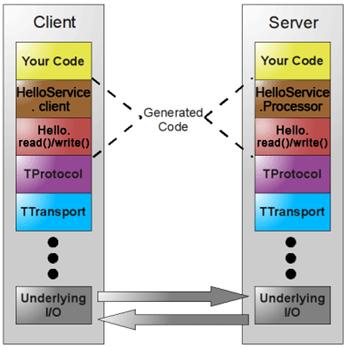
图中的节点角色为：Provider: 暴露服务的服务提供方。Consumer: 调用远程服务的服务消费方。Registry: 服务注册与发现的注册中心。Monitor: 统计服务的调用次调和调用时间的监控中心。Container: 服务运行容器。

图中的调用关系为：0. 服务容器负责启动，加载，运行服务提供者。1. 服务提供者在启动时，向注册中心注册自己提供的服务。2. 服务消费者在启动时，向注册中心订阅自己所需的服务。3. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

四．Thrift

Apache Thrift 是 Facebook 实现的一种高效的、支持多种编程语言的远程服务调用的框架和二进制通讯协议。主要用来定义和创建分布式的远程服务。Thrift的优点有：相比类似于SOAP的协议，基于二进行格式的数据传输更加高效、快速；引入库代码少，没有编码框架，没有配置文件；Thrift的IDL对数据定义与各语言之间的翻译很自然，映射的很好；应用层与序列化层的通讯格式分离，独立修改；支持协议的向下兼容，定义良好的协议是可以向之前的版本兼容的。Thrift它采用接口描述语言定义并创建服务，支持可扩展的跨语言服务开发，所包含的代码生成引擎可以在多种语言中，如 C++，Java，Python，PHP，Ruby，Erlang，Perl，Haskell，C#，Cocoa，Smalltalk 等创建高效的、无缝的服务。Thrift还可以使用Java的NIO模式，极大的提高了服务整体的性能和吞吐能力。

Thrift 架构包含一个完整的堆栈结构用于构建客户端和服务器端。下图描绘了 Thrift 的整体架构。



如图所示，图中黄色部分是用户实现的业务逻辑，褐色部分是根据 Thrift 定义的服务接口描述文件生成的客户端和服务器端代码框架，红色部分是根据 Thrift 文件生成代码实现数据的读写操作。红色部分以下是 Thrift 的传输体系、协议以及底层 I/O 通信，使用 Thrift 可以很方便的定义一个服务并且选择不同的传输协议和传输层而不用重新生成代码。Thrift 服务器包含用于绑定协议和传输层的基础架构，它提供阻塞、非阻塞、单线程和多线程的模式运行在服务器上，可以配合服务器 / 容器一起运行，可以和现有的 J2EE 服务器 /Web 容器无缝的结合。

五．总结体会

国内互联网的用户已经达到了亿级别，所以在设计一个互联网服务的时候性能成为了我们设计的第一优先考虑。上述的三个框架在目前的一些性能对比测试中，Thrift因为其成熟的异步I/O事件模型和二进制的数据序列化机制暂时领先于其他两个框架。但是Dubbo框架提供功能却更为全面，Dubbo框架包含了完整的集群负载均衡方案与集群高可用的监控方案，对于开发者来说加入的门槛更低，开发效率更高。Google的gRPC是一个比较新的RPC框架，这个框架基于未来的HTTP/2标准设计，为使用者带来的一些更先进的特性，如双向流、头部压缩、多复用请求等。gPRC这些新的特性对移动设备应用的场景更有优势，在移动互联网迅猛发展的今天gRPC的前景要更好。

这三个框架，在对互联网云服务的设计中各有优劣，并不存在绝对的好快。Dubbo由于其集成度较高，对这个框架的二次构建成本也相对较高。gPRC技术新颖，但是目前社区还没有其稳定的版本，没有在生产环境中得到过验证。Thrift相对要成熟，框架的体量相对Dubbo要小很多。其负载和高可用的特性可以插入其他的组件来保证，所以设计Thrift的完整技术架构的解决方案对架构师的要求要高。好在现在开源的社区已经有了很多的基于Thrift的完整技术架构的解决方案供我们参考。

以上是对目前先进的RPC框架的体会，重点是描述了自己在前沿讲座中听到的，见到的内容，通过讲座和自己体会的总结对这些新技术的了解更有帮助。