基于 RPC 的分布式系统实验

李青坪[1]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)
通讯作者: 李青坪, E-mail: lqp19940918@163.com

摘 要: RPC(Remote Procedure Call)，即远程过程调用，是一种通过网络在进程间通信的通信方式，不需要考虑底层网络协议。RPC 采用 C/S 模式，发送请求的是客户端，接受并处理请求的是服务器。传统的过程调用模式只能进行本地调用，无法充分利用网络上的资源，使得主机资源大量浪费。通过 RPC，我们可以充分利用非共享内存的计算资源，可以很方便地实现资源代码共享，提高资源的利用率。本文即利用 RPC，设计一个获取并显示服务器时间的分布式应用，并解决客户端授权以及消息传递过程中的时延问题。

关键词: 远程过程调用;分布式;时延;授权

Distributed File System Experiment Based on RPC

LI Qingping[1]

[1](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China)

Abstract: RPC(Remote Procedure Call), which do not need to consider the underlying network protocol, is a communication mode between processes through network. RPC uses C/S mode, computers sending requests are clients, and computers receiving and handling requests are servers. The traditional process call mode can only be local calls, which can not take full advantage of the resources on the network, making a lot of host resources wasted.Through RPC, we can make full use of non-shared memory computing resources, and can easily make resource code shared, improve resource utilization. This article uses RPC to design a distributed application that obtains and displays server time and solves the problem of client authorization and the delay in the messaging process.

Key words: remote procedure call; distributed; time delay; authorization

1 RPC 简介

RPC 是指远程过程调用，也就是说两台服务器 A， B，一个应用部署在 A 服务器上，想要调用 B 服务器上应用提供的函数/方法，由于不在一个内存空间，不能直接调用，需要通过网络来表达调用的语义和传达调用的数据。

首先，要解决通讯的问题，主要是通过在客户端和服务器之间建立 TCP 连接，远程过程调用的所有交换的数据都在这个连接里传输。连接可以是按需连接，调用结束后就断掉，也可以是长连接，多个远程过程调用共享同一个连接。

第二 ,要解决寻址的问题 ,也就是说， A 服务器上的应用怎么告诉底层的 RPC 框架 ,如何连接到 B 服务器（如主机或 IP 地址）以及特定的端口，方法的名称是什么，这样才能完成调用。比如基于 Web 服务协议栈的 RPC，就要提供一个 endpoint URI，或者是从 UDDI 服务上查找。如果是 RMI 调用的话，还需要一个 RMI Registry 来注册服务的地址。

第三，当 A 服务器上的应用发起远程过程调用时，方法的参数需要通过底层的网络协议如 TCP 传递到 B 服务器[1]，由于网络协议是基于二进制的，内存中的参数的值要序列化成二进制的形式，也就是序列化（ Serialize）或编组（ marshal），通过寻址和传输将序列化的二进制发送给 B 服务器。

第四， B 服务器收到请求后，需要对参数进行反序列化（序列化的逆操作），恢复为内存中的表达方式，然后找到对应的方法（寻址的一部分）进行本地调用，然后得到返回值。

第五，返回值还要发送回服务器 A 上的应用，也要经过序列化的方式发送，服务器 A 接到后，再反序列化，恢复为内存中的表达方式，交给 A 服务器上的应用。

2 系统设计与实现

2.1 系统设计要求

设计一个分布式应用。 客户端负责收集服务器的时间并进行打印， 服务器负责提供时间， 一直处于服务状态， 等待响应客户端。 具体要求如下：

服务器收集自己的时间，并打印在本机屏幕上。

多个客户端负责请求， 得到当前服务器上的时间， 以一秒的速率不同的打印在本地机器上

解决服务器和客户端之间的通信时延

只有授权的客户端才能获取服务器上的时间

服务器可以运行在 windows 以及 linux 系统下

客户端也至少有 windows 和 linux 两个版本

2.2 系统实现

2.2.1 Client 端

首先执行 register 方法，向服务器注册，请求的 url 后缀为”/register“，并将授权密码加入报文中一并发送到服务器，等待服务器响应，服务器注册成功后会返回字符串“ok”，否则返回”wrongpassword“。客户端收到”ok“消息后，开启一个定时器向服务器每秒发送一个获取服务器时间的请求，请求后缀为”/get“，通过 http 包里面的 post 方法发送该请求，如果服务器返回错误，则打印错误；否则，接受服务器发送的时间，计算传输延时后打印该时间。

2.2.2 Server 端

首先初始化一个名为 authorized 的 map[string]int，记录授权后的客户端的 ip 和对应客户端上一次向服务器发送请求到现在的时间 t。如果 t>10，则认为该客户端已经不再向服务器发送请求，则将该客户端从 authorized 中删除。开启一个线程，执行 countTime 方法，用来对 t 进行计数。服务器监听在固定端口，对请求 url 中后缀为”/register“的请求执行授权操作，解析客户端发送的请求报文中提供的授权密码，比较是否正确，如果不正确则向客户端发送”wrongpassword“，否则授权该客户端并发送”ok“；对请求中后缀为”/get“的请求，获取当前时间，并返回给客户端，同时将 t 置为 0。

2.2.3 时延及授权

由于服务器响应客户端请求后，在将时间发送给客户端的过程中，由于网络的传输，存在时延，实际上客户端得到的服务器发送过来的时间并不是服务器此时真正的时间，需要加上传输的时延才是此时服务器真实的时间。可以记录客户端发送请求和收到回复这两个时间点，服务器真实时间=客户端收到服务器发送的时间+(客户端受到回复的时间点-客户端发送请求的时间点)/2。

服务器需要针对特定的客户端发送服务器时间，避免未经授权的客户端窃取服务器信息。本次实验采用客户端携带密码的形式进行客户端验证，服务器获取客户端发送的密码，验证通过后对该客户端进行授权。

3 实验结果

3.1 本地测试

服务器端：

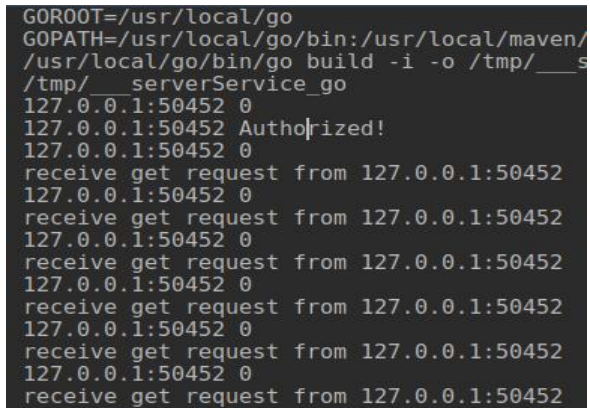

图 1　服务器接受客户端请求并处理

客户端：

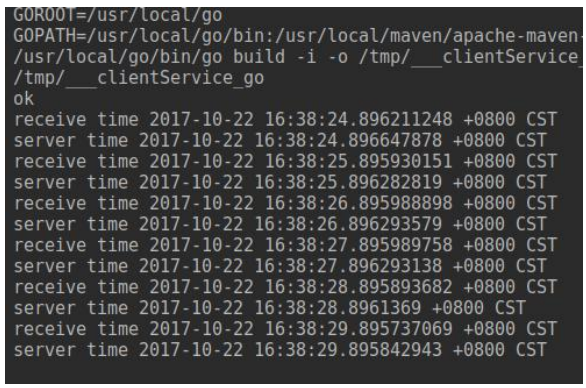

图 2　客户端获取服务器端时间

客户端验证失败：

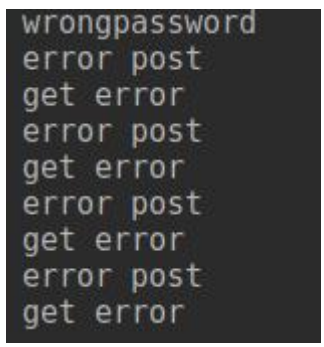

图 3　客户端验证失败

服务器端验证失败时，抛出异常，但程序继续执行。

服务器删除超时客户端：

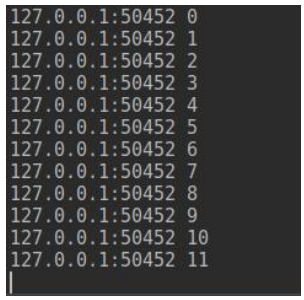

图 4　服务器删除超时客户端

多客户端在 t 时间内连续请求服务器：

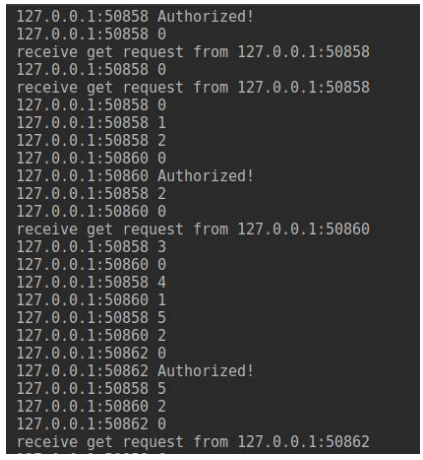

图 5　多客户端在 t 时间内连续请求服务器

3.2　分布式测试

服务器(114.212.86.58)：

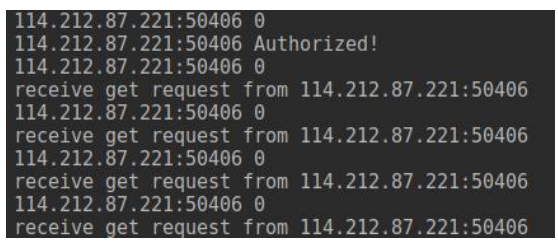

图 6　服务器输出

客户端(114.212.87.221)：

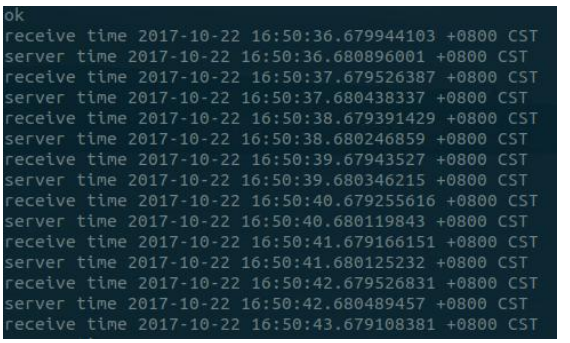

图 7　客户端输出

4　总结与体会

本次实验中，我了解了 RPC 的实现原理，并结合课程所学习的知识，使用 golang 语言实现了简单的 RPC。同时 Golang 语言提供的多线程机制对于解决多线程编程有很大的帮助，代码中一条简单的 go 语句即可实现多线程。从实验可以看出，客户端接受的时间同服务器真实的时间并没有太大的差异，是因为我们的测试都是在同一局域网完成的，因此传输时延不大，但当服务器与客户端不在同一局域网内或者网络比较拥塞时，传输时延会增大。由于硬件条件限制，本文没有做非同一局域网内的测试，但传输时延在公网上肯定会增大。

References:

[1]　http://blog.csdn.net/gaijianwei/article/details/45913473