## RPC框架设计文档

#### 一、设计原理

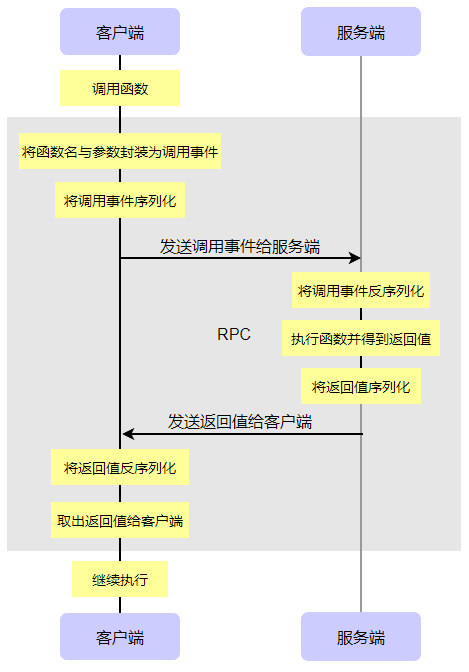
RPC 的全称是 Remote Procedure Call，是一种进程间通信方式。它允许程序调用另一个地址空间（通常是共享网络的另一台机器上）的过程或函数，而不用程序员显式编码这个远程调用的细节。简言之，RPC使得程序能够像访问本地系统资源一样，去访问远端系统资源。

为了达到这样的效果，需要几个部分：

1.客户端与服务端通过某种方式（一般就是网络socket）进行通信。

2.客户端把调用函数的事件（比如调用函数的名称，函数参数）进行序列化后发送给服务端；服务端将客户端的调用事件反序列化，并执行相应的函数，将返回值进行序列化后再发送给客户端。客户端将消息反序列化得到最终的结果。

下面是这个过程的时序图：



#### 二、概要设计

1. 任务描述

编写一个简易RPC框架，即远程调用框架，使用该RPC实现一个简单的远程调用，1个服务端上包含求和sum和字母大写upper函数，不少于2个客户端从服务端上调用函数，支持输入基本数据类型：int, float, string，然后得到函数结果。开发语言不限。

1. 开发环境以及工具

本次实验使用python 3.8进行开发。

RPC的网络通信部分使用基于TCP的socket进行连接。

RPC的序列化工具使用pickle，它能够实现任意对象与文本之间的相互转化，也可以实现任意对象与二进制之间的相互转化。

#### 三、设计实现

根据上面的时序图，总体上分为客户端实现部分和服务端实现部分，两个部分有共同部分，比如网络通信、序列化等，也有各自不同的部分。

###### 网络通信部分

网络通信部分使用基于TCP的socket。这一部分主要由ClientTcp和ServerTcp两个py文件进行实现。

1.在客户端中主要完成了通信连接，发送数据，处理接收到的数据，关闭连接等工作。具体如下。

（1）通信连接。首先创建socket对象，然后利用内置函数connect与输入的ip地址和端口号进行连接同时将socket连接设置为阻塞状态。

（2）发送消息。利用socket内置函数send将数据发送。

（3）处理接收到的数据。调用socket内置函数recv读取服务器返回的内容，将接收到的数据打包为一个个的事件，进行封装。

（4）关闭连接。利用socket内置函数close将网络通信关闭。

2.在服务端主要完成了通信连接，发送数据，监听网络事件，处理连接请求，处理接收到的数据，关闭连接等工作。具体如下。

（1）通信连接。首先创建socket对象，使用内置函数bind关联 socket 到指定的网络接口（IP 地址）和端口号，将socket连接设置为阻塞状态。

（2）发送消息。利用socket内置函数send将数据发送给连接id对应的连接，确保数据发送给正确的客户端。

（3）监听网络事件。采用select模块来监听网络事件，如果服务端未收到任何的网络消息会一直阻塞；如果监听到有客户端连接服务器，就需要处理连接请求；如果收到的是数据，客户端套接字被触发，利用套接字处理接收到的数据。

（4）处理连接请求。使用socket内置函数accept阻塞并等待传入连接。当一个客户端连接时，它将返回一个新的 socket 对象，对象中有表示当前连接的 conn 和一个由主机、端口号组成的 IPv4/v6 连接的元组，用这个 socket 对象和客户端进行通信。在这里我们对连接的信息进行了打印，方便后续多客户端操作时了解连接信息。

（5）处理接收到的数据。调用socket内置函数recv读取服务器返回的内容，将接收到的数据打包为一个个的事件，进行封装。如果收到的数据为空，需要将socket断开，并打印离开信息。

（6）关闭连接。利用socket内置函数close将网络通信关闭。

###### 打包消息部分

这一部分主要完成数据的序列化和反序列化工作，由Task这一py文件进行完成。

在Task类中利用pickle序列化工具实现了对调用函数信息（包括函数名，函数参数等）的序列化和反序列化。

在TaskRespond类中利用pickle序列化工具实现了对函数返回值的序列化和反序列化。

###### 线程

这一部分主要完成多线程操作，对于多个客户端连接服务端，我们采用新开现成的方式，由Thread这一py文件进行完成。

使用threading模块，使用Thread函数创建线程，使用内置函数start开启线程活动。同时使用Event函数创建事件对象，利用内置函数wait阻塞线程直到内部标志为True，或者发生超时事件。

###### 客户端服务端实现

前面大部分工作都完成了，这里主要时调用前面实现的方法，完成整个客户端和服务端的工作流程。

服务端的主要任务是：

1. 创建一个指定类对象；

2. 监听网络事件；

3. 解析调用事件，调用指定类中的对应函数；

4. 将结果发回客户端。

客户端的主要任务是：

1. 将类中的非私有函数提取出来作为rpc函数；

2. 当有函数调用事件时，连接服务端；

3. 创建一个task并发送给对应的服务端；

4. 得到返回值。

###### 测试实现

这一部分只要是对整个RPC框架进行测试，测试文件包括三个，ServerTest代表服务端，ClientTest1代表客户端1，ClientTest2代表客户端2。

在服务端中在Function类中实现了sum和upper两个函数，并完成网络连接操作。

在客户端1中，完成网络连接以及调用服务端的upper函数，求distribute的大写。

在客户端2中，完成网络连接以及调用服务端的sum函数，求8.8和4.45的和。

#### 四、结果展示

如下图所示是本次实验的结果。第一张图片是服务端，可以看到有两个线程进行，以及他们的离开信息。第二张图片是客户端1，可以看到得到了distribute的大写DISTRIBUTE。第三张图片是客户端2，可以看到得到了8.8+4.45的结果13.25。实验结果与实验任务要求相同。

