lab03 grpc

常烁晨-521021910369

1、介绍

远程过程调用（RPC）技术允许一个程序可以在不需要了解网络细节的情况下请求另一个位于网络上其他计算机中的程序提供的服务。gRPC作为一个现代化的开源RPC框架，它、在传统RPC的基础上提供了显著的性能改进和更多的语言互操作性。gRPC支持多种编程语言，显著提高了远程调用的性能，并简化了连接系统的构建过程。

2、实验过程

根据.proto文件的分析，它定义了Receiver服务，以及客户端和服务器交换的消息格式。Receiver服务中定义了一个RPC方法receive，该方法接受一个Event消息作为请求，并返回一个Reply消息作为响应。这种请求-响应模式是gRPC通信的基础，其中客户端发送一个请求给服务器，并期待得到一个响应。

Event是一个消息类型，它代表从客户端发往服务器的请求数据。appid是一个字符串字段，用于标识发送事件的应用程序。xwhen是一个32位整数字段，用于记录事件发生的时间。xwho是一个字符串字段，用于指示事件的发起者。xwhat是一个字符串字段，用于描述事件本身。xcontext是一个Struct字段，使用google.protobuf.Struct类型，它允许包含一个灵活的键值对映射，其中可以存储事件的附加上下文信息。

Reply是响应消息类型，它代表服务器处理完请求后返回给客户端的数据。status字段表示操作完成后的状态码，message字段是提供操作结果的额外信息的字符串。

输入以下指令得到对应 python 文件：

```
python3 -m grpc_tools.protoc -I. --python_out=. --
grpc_python_out=. ./receiver.proto
```

`receiver_pb2.py`：包含由`receiver.proto`文件定义的所有消息类型的类。这些类包括`Event`和`Reply`消息类型。

`receiver_pb2_grpc.py`：包含了`receiver.proto`中定义的服务接口的gRPC服务端和客户端代码。客户端将调用服务器上的方法，而服务器将实现其方法。

server.py定义了Receiver类，重写了receive方法，服务器打印接收到的请求，并返回一个Reply消息，通过grpc.server初始化，并指定了并发执行器ThreadPoolExecutor最多10个工作线程。它通过add_ReceiverServicer_to_server方法将定义的Receiver服务添加到服务器中，并监听50051端口。服务器启动后，它将持续运行，直到接收到键盘中断信号（如Ctrl+C），此时服务器关闭。

client.py代码定义了一个gRPC客户端，它连接到在本地运行的gRPC服务器并向其发送一个Event消息。在run函数中，客户端设置了一个到指定地址（localhost:50051）通道，创建了一个ReceiverStub对象，该对象提供了服务定义中指定的方法，允许客户端像调用本地方法一样调用远程方法。之后客户端调用stub.receive方法，向服务器发送一个Event消息。这个消息包含了事件的描述（xwhat）、应用程序标识符（appid）、事件时间（xwhen）、事件发起者（xwho）以及自定义的上下文数据（xcontext）。

3、实验结果：

```
root@miraclecsc-vm:~/lab03# sudo python3 server.py
server start...
request: appid: "fuckgod"
xwhen: 123
xwho: "jerry"
xwhat: "install"
xcontext {
  fields {
    key: "idfa"
    value {
      string_value: "idfa1"
    }
  }
  fields {
    key: "amount"
    value {
      number_value: 123
    }
  }
}

root@miraclecsc-vm:~# cd lab03
root@miraclecsc-vm:~/lab03# sudo python3 client.py
client status: 0 received: Hello, jerry!
root@miraclecsc-vm:~/lab03#
```

可以看到实验完美符合要求。