分布式系统 作业三

陈恩婷 19335015

一、问题描述

使用 protobuf 和 gRPC 远程过程调用的方法实现 Client-Server 系统，Server 提供简单的算数操作如加和等，Client 通过 RPC 向 server 发送请求，Server 返回计算结果。选做功能：Server 能够控制访问请求的数量，以及实现请求超时终止。

二、解决方案

要求的功能：

参考 gRPC 官网上的指南（https://grpc.io/docs/languages/python/quickstart/），先安装好相关的环境， 包括：

Python 3.5 or higher

pip version 9.0.1 or higher

再安装好 gRPC 和 gRPC tools：

```
$ python -m pip install grpcio
$ python -m pip install grpcio-tools
```

接着将 github 上的 gRPC 相关例子 clone 下来：

```
# Clone the repository to get the example code:
$ git clone -b v1.41.0 https://github.com/grpc/grpc
# Navigate to the "hello, world" Python example:
$ cd grpc/examples/python/helloworld
```

找到官网所讲的例子中的 helloworld.protobuf，greeter_client.py 和 greeter_server.py 三个文件并复制到 client-server 文件夹下，就可以开始在官网例子的基础上编写代码了：

helloworld.protobuf

```
// The greeting service definition.
service Greeter {
// Sends a greeting
rpc SayHello (HelloRequest) returns (HelloReply) {}
// Sends another greeting
rpc SayHelloAgain (HelloRequest) returns (HelloReply) {}
// Sends a expression to evaluate
rpc EvaluateExpression (ExpressionRequest)
returns (ExpressionReply) {}
}

// The request message containing the expression to evaluate
message ExpressionRequest {
string expression = 1;
}

// The response message containing the result of the expression
message ExpressionReply {
string result = 1;
}
```

greeter_server.py

```
def EvaluateExpression (self, request, context):
        print("Request arrived, sleeping a bit…")
        time.sleep(10)
        return helloworld_pb2.ExpressionReply(result = str(eval(request.expression)))
```

greeter_client.py

```
def run():
        channel = grpc.insecure_channel('localhost:50051')
        stub = helloworld_pb2_grpc.GreeterStub(channel)
        response = stub.SayHello(helloworld_pb2.HelloRequest(name='you'))
        print("Greeter client received: " + response.message)
        response = stub.SayHelloAgain(helloworld_pb2.HelloRequest(name='you'))
        print("Greeter client received: " + response.message)
        a = input("Plese input the first integer: ")
        b = input("Plese input the second integer: ")
        response = stub.EvaluateExpression(helloworld_pb2.ExpressionRequest(expression= a+
'+' + b))
        print("Greeter client received: " + response.result)
```

Server 能够控制访问请求的数量：

在调用 grpc.server 时加上 maximum_concurrent_rpcs 参数：

```
def serve():

    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10),maximum_concurrent_rpcs
= 1)

    helloworld_pb2_grpc.add_GreeterServicer_to_server(Greeter(), server)
    server.add_insecure_port('[::]:50051')
    server.start()
    server.wait_for_termination()
```

再在 run 函数中加上 try 和 except 用于打印错误信息：

```
        try:
                response = 
stub.EvaluateExpression(helloworld_pb2.ExpressionRequest(expression= a+    '+' + b), 
timeout = 30)
                print("Greeter client received: " + response.result)
        except grpc.RpcError as e:
                print(e.details())
                print("Greeter client received: " + "error")
```

实现请求超时终止：

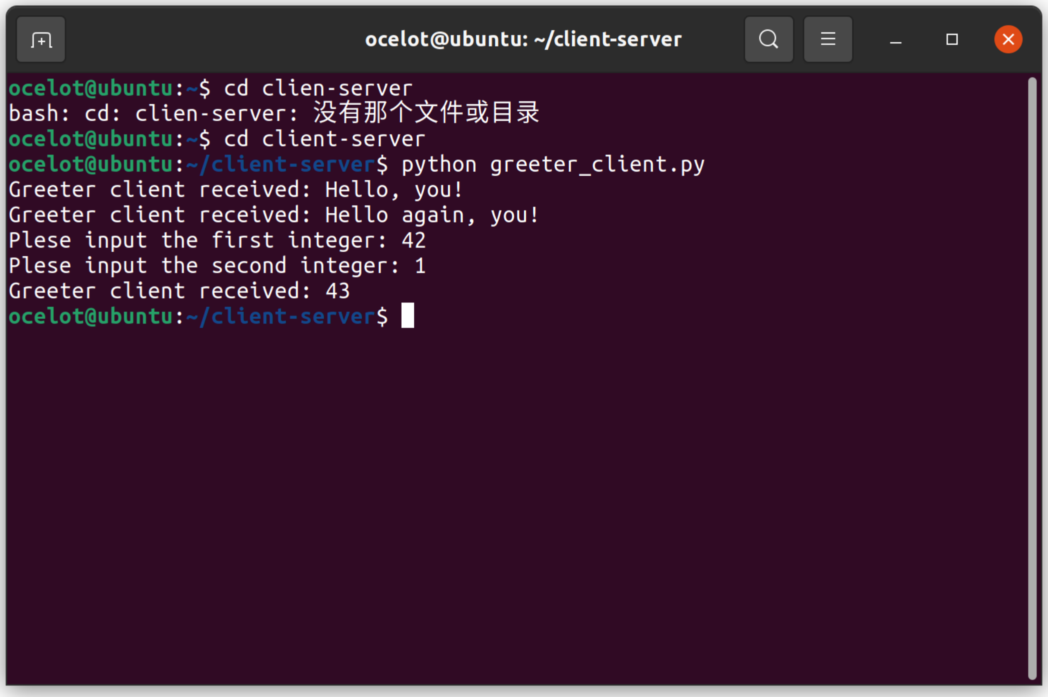
在 greeter_client.py 中加上 timeout 参数，为了能看见超时报错，将 timeout 设为 5 秒（因为之前设定了服务器会在受到请求后 sleep 十秒钟：

```
response = stub.EvaluateExpression(helloworld_pb2.ExpressionRequest(expression=a+'+'+ b), 
timeout = 30)
```

四、实验结果

1. 基础功能

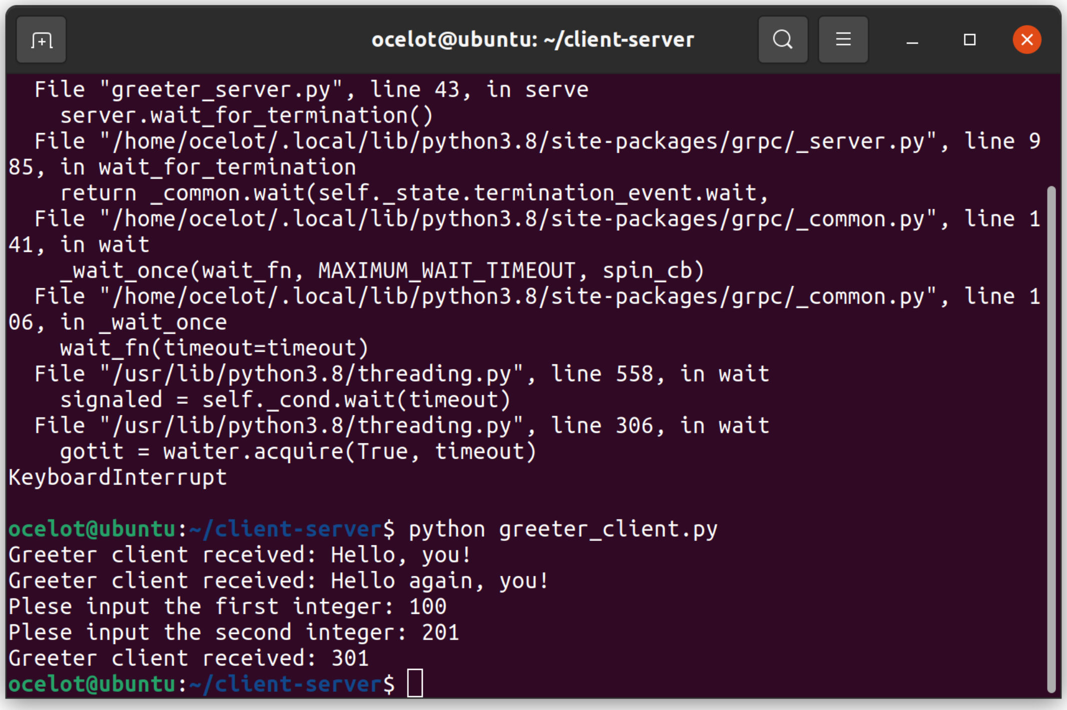
先运行 greeter_server.py，再在另一个窗口运行 greeeter_client.py：



```
ocelot@ubuntu:~$ cd clien-server
bash: cd: clien-server: 没有那个文件或目录
ocelot@ubuntu:~$ cd client-server
ocelot@ubuntu:~/client-server$ python greeter_client.py
Greeter client received: Hello, you!
Greeter client received: Hello again, you!
Plese input the first integer: 42
Plese input the second integer: 1
Greeter client received: 43
ocelot@ubuntu:~/client-server$ 
```

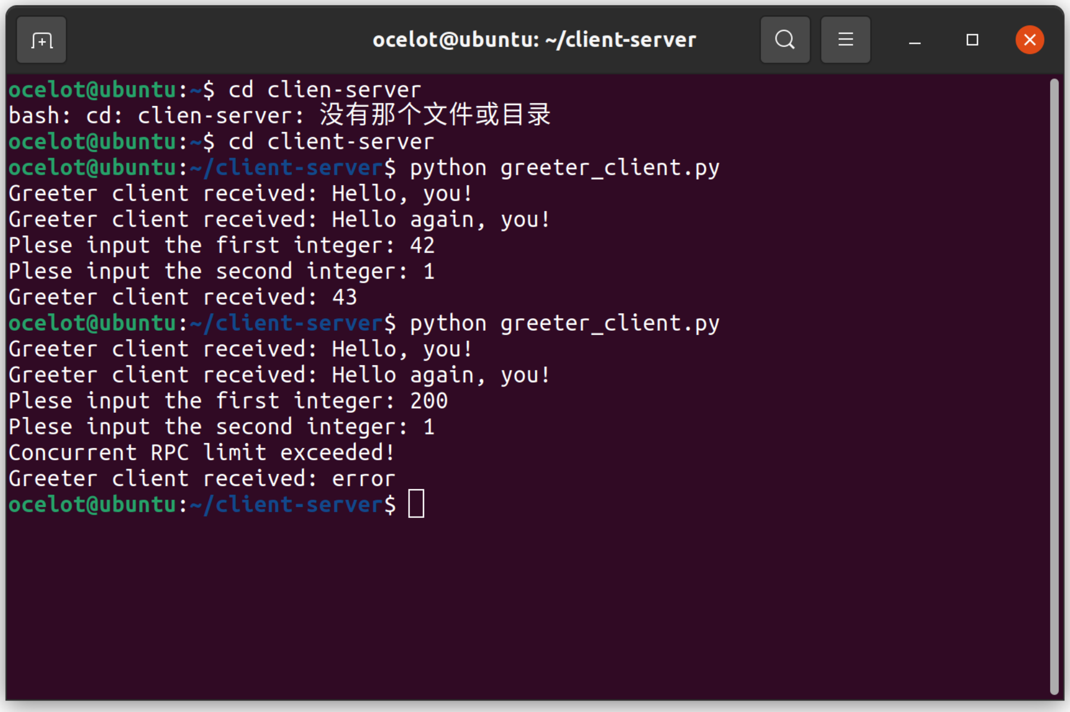
2. 控制访问请求的数量

不关闭 greeter_server.py 的窗口，在两个窗口中先后运行 greeter_client.py，注意时间间隔在 10 秒内：


ocelot@ubuntu: ~/client-server

```
  File "greeter_server.py", line 43, in serve
    server.wait_for_termination()
  File "/home/ocelot/.local/lib/python3.8/site-packages/grpc/_server.py", line 9
85, in wait_for_termination
    return _common.wait(self._state.termination_event.wait,
  File "/home/ocelot/.local/lib/python3.8/site-packages/grpc/_common.py", line 1
41, in wait
    _wait_once(wait_fn, MAXIMUM_WAIT_TIMEOUT, spin_cb)
  File "/home/ocelot/.local/lib/python3.8/site-packages/grpc/_common.py", line 1
06, in _wait_once
    wait_fn(timeout=timeout)
  File "/usr/lib/python3.8/threading.py", line 558, in wait
    signaled = self._cond.wait(timeout)
  File "/usr/lib/python3.8/threading.py", line 306, in wait
    gotit = waiter.acquire(True, timeout)
KeyboardInterrupt

ocelot@ubuntu:~/client-server$ python greeter_client.py
Greeter client received: Hello, you!
Greeter client received: Hello again, you!
Plese input the first integer: 100
Plese input the second integer: 201
Greeter client received: 301
ocelot@ubuntu:~/client-server$
```

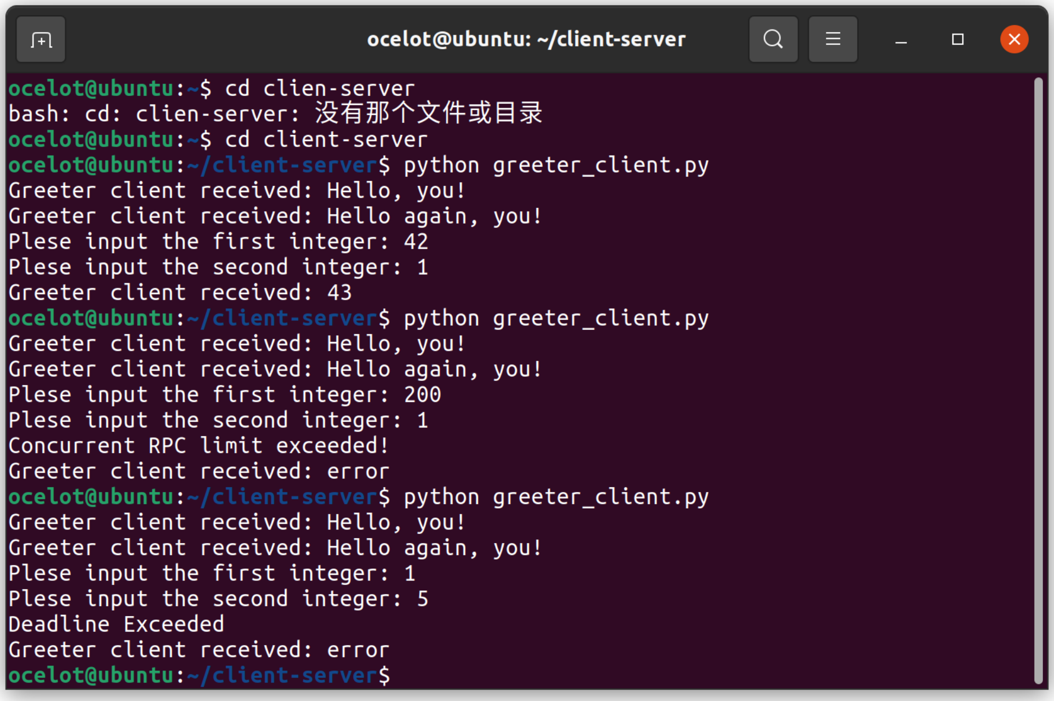

ocelot@ubuntu: ~/client-server

```
ocelot@ubuntu:~$ cd clien-server
bash: cd: clien-server: 没有那个文件或目录
ocelot@ubuntu:~$ cd client-server
ocelot@ubuntu:~/client-server$ python greeter_client.py
Greeter client received: Hello, you!
Greeter client received: Hello again, you!
Plese input the first integer: 42
Plese input the second integer: 1
Greeter client received: 43
ocelot@ubuntu:~/client-server$ python greeter_client.py
Greeter client received: Hello, you!
Greeter client received: Hello again, you!
Plese input the first integer: 200
Plese input the second integer: 1
Concurrent RPC limit exceeded!
Greeter client received: error
ocelot@ubuntu:~/client-server$
```

可以看到第一个 client 成功运行了请求，而第二个由于 server 只允许一个时间一个请求而被拒绝。

3. 请求超时终止

不关闭 server，修改 timeout 参数为 5 秒，再次运行 client：



```
ocelot@ubuntu:~$ cd clien-server
bash: cd: clien-server: 没有那个文件或目录
ocelot@ubuntu:~$ cd client-server
ocelot@ubuntu:~/client-server$ python greeter_client.py
Greeter client received: Hello, you!
Greeter client received: Hello again, you!
Plese input the first integer: 42
Plese input the second integer: 1
Greeter client received: 43
ocelot@ubuntu:~/client-server$ python greeter_client.py
Greeter client received: Hello, you!
Greeter client received: Hello again, you!
Plese input the first integer: 200
Plese input the second integer: 1
Concurrent RPC limit exceeded!
Greeter client received: error
ocelot@ubuntu:~/client-server$ python greeter_client.py
Greeter client received: Hello, you!
Greeter client received: Hello again, you!
Plese input the first integer: 1
Plese input the second integer: 5
Deadline Exceeded
Greeter client received: error
ocelot@ubuntu:~/client-server$
```

五、实验总结

本次实验复习了和 gRPC 相关的知识和应用，在实验中我也学到了一些基础的 gRPC 函数，受益匪浅，对分布式系统也有了更深的理解。