Microservices Architecture and Programming

Lab Practical and date – Practical 3, 21st August 2020

Name and Roll Number- Het Shah, 17BIT103

Practical Objective- Designing gRPC based micro-service with Ballerina

Steps Involved-

We used python and created a GPRC based microservices- the application will answer the user query regarding the mathematical operations such as addition, subtraction, division, multiplication and square root.

Background

GRPC

In gRPC, a client application can directly call a method on a server application on a different machine as if it were a local object, making it easier for you to create distributed applications and services. As in many RPC systems, gRPC is based around the idea of defining a service, specifying the methods that can be called remotely with their parameters and return types. On the server side, the server implements this interface and runs a gRPC server to handle client calls. On the client side, the client has a stub (referred to as just a client in some languages) that provides the same methods as the server.

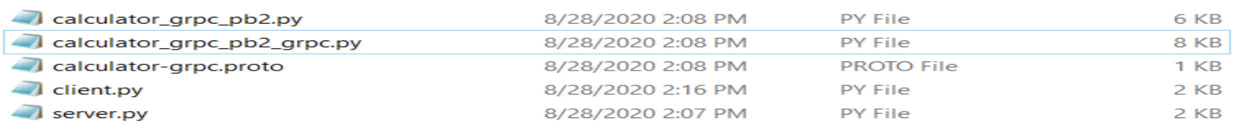
gRPC clients and servers can run and talk to each other in a variety of environments - from servers inside Google to your own desktop - and can be written in any of gRPC's supported languages. So, for example, you can easily create a gRPC server in Java with clients in Go, Python, or Ruby. In addition, the latest Google APIs will have gRPC versions of their interfaces, letting you easily build Google functionality into your applications.

How to Run

```
pip install grpcio
$ pip install grpcio-tools

$ python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. calculator-grpc.proto
```

Use the above commands to compile the proto file (calculator-grpc.proto in this case) and generate the files calculator_grpc_pb2.py and calculator_grpc_pb2_grpc.py automatically



```
calculator_grpc_pb2.py         8/28/2020 2:08 PM    PY File       6 KB
calculator_grpc_pb2_grpc.py    8/28/2020 2:08 PM    PY File       8 KB
calculator-grpc.proto          8/28/2020 2:08 PM    PROTO File    1 KB
client.py                      8/28/2020 2:16 PM    PY File       2 KB
server.py                      8/28/2020 2:07 PM    PY File       2 KB
```

Figure 1 files calculator_grpc_pb2.py and calculator_grpc_pb2_grpc.py are generated automatically

```
$python client.py
```

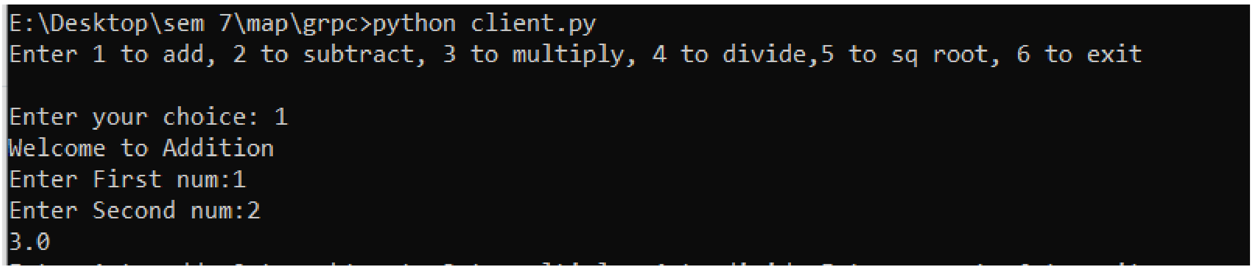
Open new CMD and run the following command- python client.py

```
$python server.py
```

Open new CMD window and run the following command- python server.py

Give inputs from 1 to 6 and the numbers as input to get the desired output

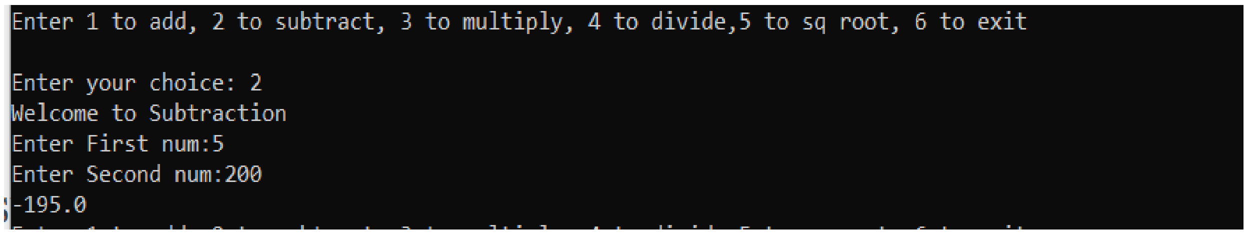
OUTPUT



```
E:\Desktop\sem 7\map\grpc>python client.py
Enter 1 to add, 2 to subtract, 3 to multiply, 4 to divide,5 to sq root, 6 to exit

Enter your choice: 1
Welcome to Addition
Enter First num:1
Enter Second num:2
3.0
```

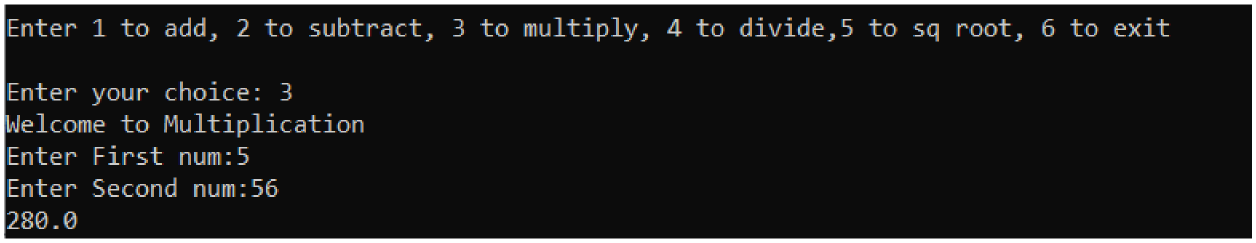
Figure 2 Option 1- Addition, adding 1 and 2 getting result 3



```
Enter 1 to add, 2 to subtract, 3 to multiply, 4 to divide,5 to sq root, 6 to exit

Enter your choice: 2
Welcome to Subtraction
Enter First num:5
Enter Second num:200
-195.0
```

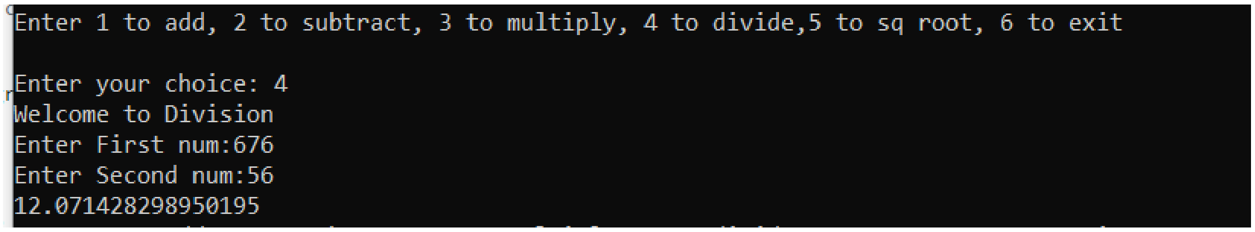
Figure 3 Option 2- subtraction, subtracting 5 and 200 getting result -195



```
Enter 1 to add, 2 to subtract, 3 to multiply, 4 to divide,5 to sq root, 6 to exit

Enter your choice: 3
Welcome to Multiplication
Enter First num:5
Enter Second num:56
280.0
```

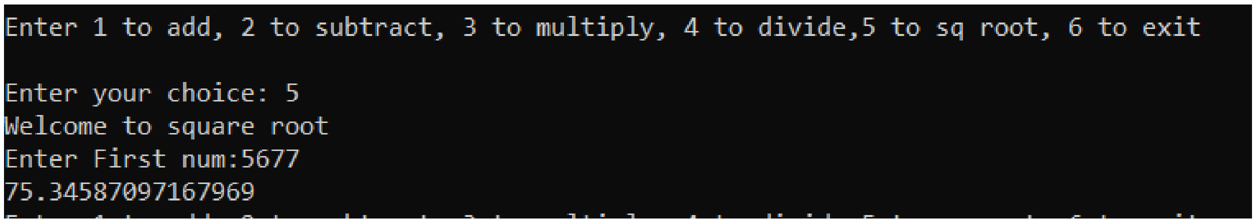
Figure 4 Option 3- multiplication, multiplying 5 and 56 getting result 280



```
Enter 1 to add, 2 to subtract, 3 to multiply, 4 to divide,5 to sq root, 6 to exit

Enter your choice: 4
Welcome to Division
Enter First num:676
Enter Second num:56
12.071428298950195
```

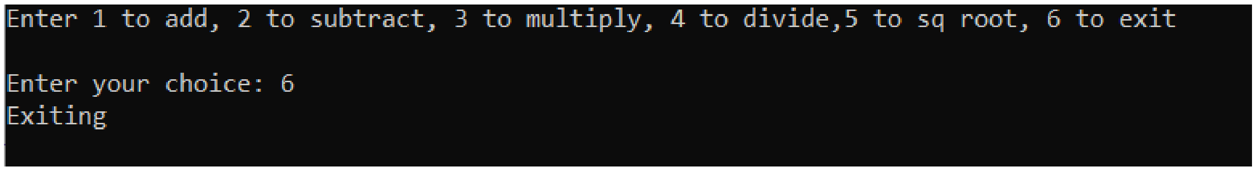
Figure 5 Option 4- division, dividing 676 and 56 getting result 12.07



```
Enter 1 to add, 2 to subtract, 3 to multiply, 4 to divide,5 to sq root, 6 to exit

Enter your choice: 5
Welcome to square root
Enter First num:5677
75.34587097167969
```

Figure 6 option 5- square root, sq root of 5677 is 75.34



```
Enter 1 to add, 2 to subtract, 3 to multiply, 4 to divide,5 to sq root, 6 to exit

Enter your choice: 6
Exiting
```

Figure 7 option 6, exit the program

Conclusion

In this practical we learned how to make GRPC calls to a sever using python using a protobuf file and using python for hosting the server and making the calls to find the answer of mathematical operations such as addition, subtraction, multiplication, division and square root by giving input from 1 to 5.

client.py

```python

import grpc
import calculator_grpc_pb2
import calculator_grpc_pb2_grpc
def connect():
    channel = grpc.insecure_channel('localhost:9999')
    client = calculator_grpc_pb2_grpc.apiStub(channel)
    while True:
        print('Enter 1 to add,'
              ' 2 to subtract,'
              ' 3 to multiply,'
              ' 4 to divide,5 to sq root, 6 to exit')
        print()
        n = input('Enter your choice: ')
        if n == '1':
            print('Welcome to Addition')
            x = int(input('Enter First num:'))
            y = int(input('Enter Second num:'))
            # raise pybreaker.CircuitBreakerError
            res = client.add(calculator_grpc_pb2.twoNums(numOne=x, numTwo=y))
            print(res.num)
        elif n == '2':
            print('Welcome to Subtraction')
            x = int(input('Enter First num:'))
            y = int(input('Enter Second num:'))
            res = client.sub(calculator_grpc_pb2.twoNums(numOne=x, numTwo=y))
            # tme.sleep(5)
            print(res.num)
        elif n == '3':
            print('Welcome to Multiplication')
            x = int(input('Enter First num:'))
            y = int(input('Enter Second num:'))
            res = client.mul(calculator_grpc_pb2.twoNums(numOne=x, numTwo=y))
            print(res.num)
        elif n == '4':
            print('Welcome to Division')
            x = int(input('Enter First num:'))
            y = int(input('Enter Second num:'))
            res = client.div(calculator_grpc_pb2.twoNums(numOne=x, numTwo=y))
            print(res.num)
        elif n == '5':
            print('Welcome to square root')
            x = int(input('Enter First num:'))
            res = client.sq(calculator_grpc_pb2.twoNums(numOne=x))
            print(res.num)
        elif n == '6':
            print('Exiting')
            exit()
        else:
            print('Invalid Input,')

connect()
```

server.py

```python
import grpc
import time
from concurrent import futures
import calculator_grpc_pb2
import calculator_grpc_pb2_grpc


def serve():
    grpc_server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    calculator_grpc_pb2_grpc.add_apiServicer_to_server(CalculatorServicer(), grpc_server)
    grpc_server.add_insecure_port('[::]:9999')
    grpc_server.start()
    while True:
        time.sleep(860000)


class CalculatorServicer(calculator_grpc_pb2_grpc.apiServicer):

    def add(self, request, context):
        return calculator_grpc_pb2.num(num=(request.numOne+request .numTwo))

    def sub(self, request, context):
        return calculator_grpc_pb2.num(num=(request.numOne-request .numTwo))

    def mul(self, request, context):
        return calculator_grpc_pb2.num(num=(request.numOne*request .numTwo))

    def div(self, request, context):
        return calculator_grpc_pb2.num(num=(request.numOne/request .numTwo))

    def sq(self, request, context):
        return calculator_grpc_pb2.num(num=(request.num ** 0.5))

if __name__ == '__main__':
    serve()

```

proto file for calculator

```
syntax= "proto3";
package main;

message num {
    float num = 1;
}

message twoNums{
    float numOne = 1;
    float numTwo = 2;
}

service api {
    rpc add (twoNums) returns (num);
    rpc sub (twoNums) returns (num);
    rpc mul (twoNums) returns (num);
    rpc div (twoNums) returns (num);
    rpc sq  (num) returns (num);
}



```