

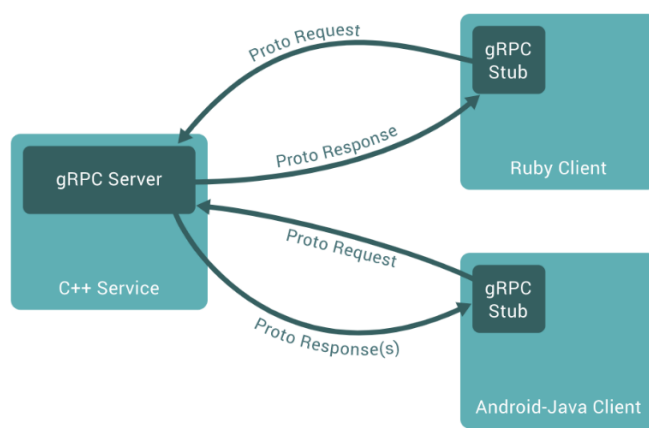
分布式系统作业三

远程过程调用

数据科学与计算机学院 17大数据与人工智能

17341015 陈鸿峰

问题 1. 远程过程调用(*RPC*)用将网络编程变得非常简单。根据所学的*RPC*相关原理，实现客户端-服务器通信，并进行简单的计算如数据库查询、算术计算、数据挖掘、深度学习推导等。



gRPC框架

- 要求:

1. 采用*gRPC*
2. 采用*Protobuf*作为*C-S*数据传输格式
3. 服务器端采用线程池，支持并发
4. 支持至少两种的计算服务如简单的算术运算+数据挖掘算法 (*K-means*、*KNN*等)
5. 编程语言不做要求

- 建议: *gRPC*和*Protobuf*在配置环境时可能有些复杂, 对于有些编程语言如*C++*、*Go*等会有一些挑战, *Python*问题会少一些。

- 关于*gRPC*的相关例子: <https://github.com/grpc/grpc/tree/master/examples>

一、实验内容

在本次实验中我使用*gRPC*主要实现了两个功能: **Python解释器**、**MySQL查询器**。客户端设计得与真实的解释器类似, 功能也与真实的解释器相似, 使得用户即使在本机上没有安装

部分Python环境（如numpy/Tensorflow等）或者没有安装MySQL数据库也能够通过远程客户端实现计算、查询等功能。

Python解释器主要采用Python的反射机制(reflection)，即eval和exec。在客户端方面将用户需要运行的字符串通过gRPC传递给服务器端，服务器端直接对字符串进行运算，并以字符串形式返回给客户端。

而MySQL同样采用类似的机制，客户端通过gRPC将用户输入以字符流的形式传输，服务器端收到后调用mysql.connector与数据库建立起链接，并通过MySQL提供的API execute语句进行查询，最后将结果返回客户端。

Protobuf文件定义如下(pyinterpreter.proto)，这里提供两个服务Evaluation和Query，并且共用了字符流格式EvalRequest和EvalReply。

```
syntax = "proto3";

package pyinterpreter;

service Evaluation {
  rpc eval (EvalRequest) returns (EvalReply) {}
}

service Query {
  rpc query (EvalRequest) returns (EvalReply) {}
}

message EvalRequest {
  string msg = 1;
}

message EvalReply {
  string msg = 1;
}
```

定义好后需要使用下列指令对其进行编译，生成对应了类文件，供服务器端和客户端进行调用。

```
python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. pyinterpreter.
↪ proto
```

服务器端(server.py)主要有以下几个注意点：

- Evaluation为Python的解释器部分，采用了异常处理方式进行编写。先尝试对用户输入的语句进行直接估值(eval)，并将结果返回。如果不行则尝试直接执行(exec)，这里包括赋值等操作，是没有返回结果的。但这里有一个变量作用域和反射的问题。首先由于Evaluation只

是一个类，其中的函数作用域是局部的，因此调用`exec`生成的用户变量也是局部的，用户很可能没法再次访问；另一方面，直接采用反射的方法进行操作其实是非常危险的，对于一些安全性要求比较高的应用不应该这么做。这两个是目前我的程序中存在的缺陷，但不会影响正常使用。

- `SQLQuery`为MySQL的解释器部分。首先需要在服务器端建立起与MySQL数据库的连接，这里采用了yaml进行数据库登录信息的存储，可以有效避免敏感信息泄露，并且具有高度的灵活性。建立连接后对其`cursor`进行`execute`操作，就与普通的数据库查询相似了。
- 这里采用了Python的线程池`ThreadPoolExecutor`实现并发操作，在后面的实验中可以看到一个服务器端可以同时连接多个客户端，各个服务互不干扰正常工作。

```
from concurrent import futures
import logging
import yaml

import grpc

import numpy as np
import mysql.connector
import pyinterpreter_pb2
import pyinterpreter_pb2_grpc

class Evaluation(pyinterpreter_pb2_grpc.EvaluationServicer):

    def __init__(self):
        print("Created Python interpreter!")

    def eval(self, request, context):
        try:
            msg = str(eval(request.msg))
        except:
            exec(request.msg)
            msg = ""
        return pyinterpreter_pb2.EvalReply(msg=msg)

class SQLQuery(pyinterpreter_pb2_grpc.QueryServicer):

    def __init__(self):
        config = yaml.load(open('config.yaml'))
        print("Connected MySQL '{}'@'{}'!".format(config["user"], config["host"]))
        self.db = mysql.connector.connect(
            host=config["host"],
            user=config["user"],
```

```

        passwd=config["passwd"]
    )
    self.cursor = self.db.cursor()

def query(self,request,context):
    self.cursor.execute("use school;")
    self.cursor.execute(request.msg)
    return pyinterpreter_pb2.EvalReply(msg=str(self.cursor.fetchall()))

def serve():
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    pyinterpreter_pb2_grpc.add_EvaluationServicer_to_server(Evaluation(),server)
    pyinterpreter_pb2_grpc.add_QueryServicer_to_server(SQLQuery(),server)
    server.add_insecure_port('[::]:50051')
    server.start()
    server.wait_for_termination()

if __name__ == '__main__':
    logging.basicConfig()
    serve()

```

Python解释器客户端(client.py)如下，通过调用生成的gRPC类文件，可以实现传输的功能。同时通过简单的循环输入输出操作，伪造出了一个解释器界面。

```

import logging

import grpc

import pyinterpreter_pb2
import pyinterpreter_pb2_grpc

def run():
    with grpc.insecure_channel('localhost:50051') as channel:
        stub = pyinterpreter_pb2_grpc.EvaluationStub(channel)
        print("gRPC-based Python interpreter")
        while True:
            print(">>> ",end="")
            message = input()
            response = stub.eval(pyinterpreter_pb2.EvalRequest(msg=message))
            if response.msg != "":
                print(response.msg)

if __name__ == '__main__':
    logging.basicConfig()
    run()

```

MySQL远程客户端(clientdb.py)与上述类似。

```
import logging

import grpc

from decimal import Decimal
import pyinterpreter_pb2
import pyinterpreter_pb2_grpc

def run():
    with grpc.insecure_channel('localhost:50051') as channel:
        stub = pyinterpreter_pb2_grpc.QueryStub(channel)
        print("gRPC-based MySQL interpreter")
        while True:
            print("mysql> ", end="")
            message = input()
            response = stub.query(pyinterpreter_pb2.EvalRequest(msg=message))
            if response.msg != "":
                result = eval(response.msg)
                for item in result:
                    print(item)

if __name__ == '__main__':
    logging.basicConfig()
    run()
```

二、实验结果

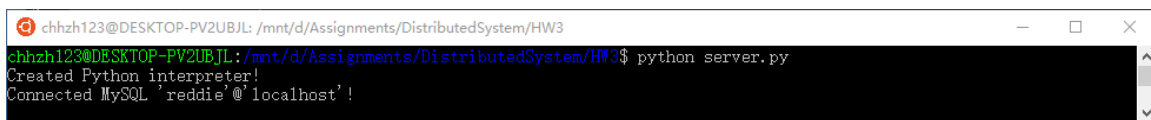
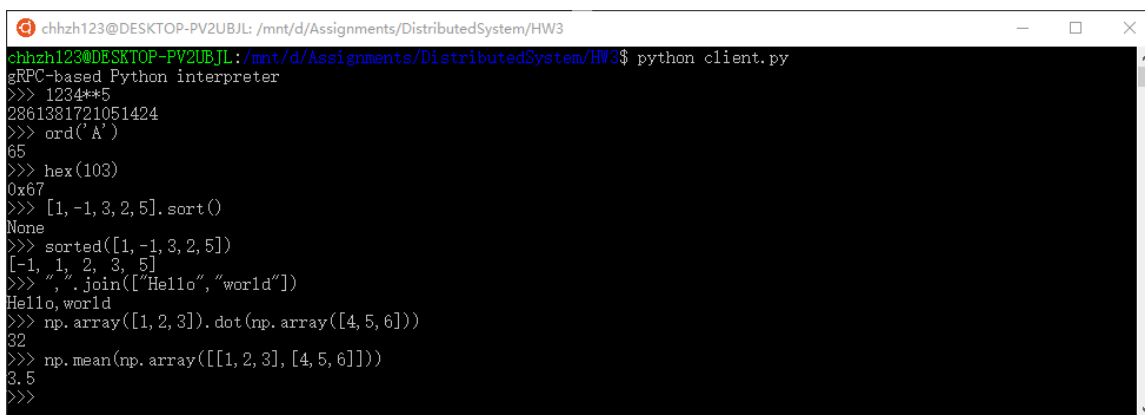


图 1: 服务器端, 新建服务时会产生提示

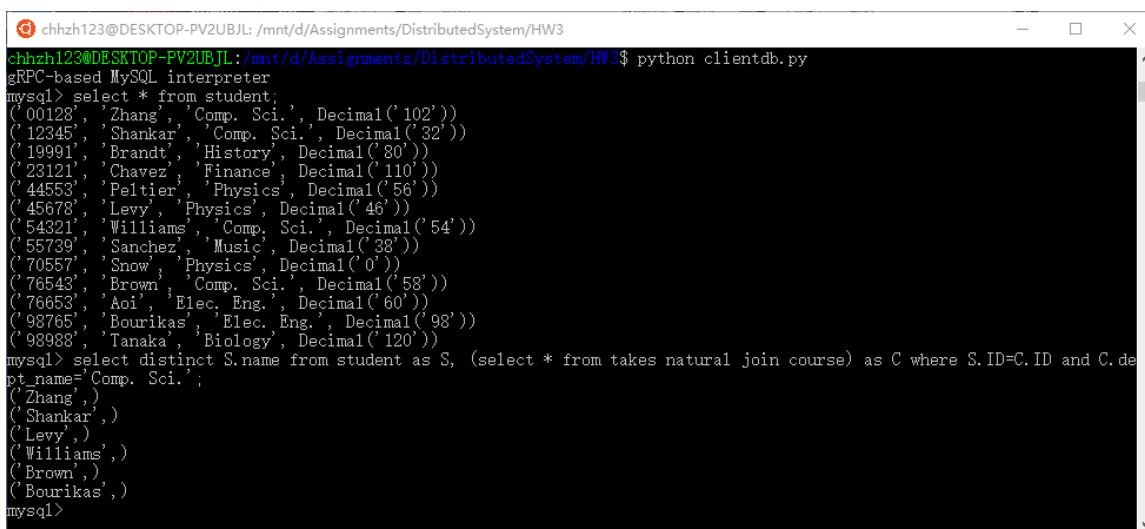


```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/DistributedSystem/HW3
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/DistributedSystem/HW3$ python client.py
gRPC-based Python interpreter
>>> 1234**5
2861381721051424
>>> ord('A')
65
>>> hex(103)
0x67
>>> [1, -1, 3, 2, 5].sort()
None
>>> sorted([1, -1, 3, 2, 5])
[-1, 1, 2, 3, 5]
>>> ", ".join(["Hello", "world"])
Hello, world
>>> np.array([1, 2, 3]).dot(np.array([4, 5, 6]))
32
>>> np.mean(np.array([[1, 2, 3], [4, 5, 6]]))
3.5
>>>

```

图 2: 伪Python解释器客户端，可以实现常见的计算查询服务，并可调用numpy包进行矩阵运算



```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/DistributedSystem/HW3
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/DistributedSystem/HW3$ python clientdb.py
gRPC-based MySQL interpreter
mysql> select * from student;
('00128', 'Zhang', 'Comp. Sci.', Decimal('102'))
('12345', 'Shankar', 'Comp. Sci.', Decimal('32'))
('19991', 'Brandt', 'History', Decimal('80'))
('23121', 'Chavez', 'Finance', Decimal('110'))
('44553', 'Peltier', 'Physics', Decimal('56'))
('45678', 'Levy', 'Physics', Decimal('46'))
('54321', 'Williams', 'Comp. Sci.', Decimal('54'))
('55739', 'Sanchez', 'Music', Decimal('33'))
('70557', 'Snow', 'Physics', Decimal('0'))
('76543', 'Brown', 'Comp. Sci.', Decimal('58'))
('76653', 'Aoi', 'Elec. Eng.', Decimal('60'))
('98765', 'Bourikas', 'Elec. Eng.', Decimal('98'))
('98988', 'Tanaka', 'Biology', Decimal('120'))
mysql> select distinct S.name from student as S, (select * from takes natural join course) as C where S.ID=C.ID and C.de
pt_name= Comp. Sci.;
('Zhang',)
('Shankar',)
('Levy',)
('Williams',)
('Brown',)
('Bourikas',)
mysql>

```

图 3: 伪MySQL客户端，可以正常实现复杂数据查询功能，并将查询记录逐行进行显示（这里采用了本学期数据库的school scheme，可用附件中database的文件夹中的sql程序进行生成）

遇到的问题及解决方法都已经在前文阐述。注意要运行我的程序，需要先配置好Python的MySQL连接器，并且按照如下格式配置好yaml文件，才可正常运行。

```

host: // your MySQL host name
user: // your MySQL user name
passwd: // password to login MySQL

```