分布式系统作业

第 3 次作业

姓名：施天予

班级：计科 2 班

学号：19335174

一、问题描述

1. 一种可靠的多播服务允许发送方向一组接收方可靠地传递消息。这种服务属于中间件层还是属于更底层的一部分?
2. 描述一下客户端和服务器端之间使用套接字的无连接通信是如何进行的?
3. 当要在基于DHT的系统中解析一个键值时,递归查询的主要缺点是什么?
4. 使用protobuf和gRPC等远程过程调用的方法实现消息订阅(publish-subscribe)系统,该订阅系统能够实现简单的消息传输,还可以控制消息在服务器端存储的时间。

二、解决方案

1. 一种可靠的多播服务允许发送方向一组接收方可靠地传递消息。这种服务属于中间件层还是属于更底层的一部分?

中间件层。

一种可靠的多播服务可以很容易的成为传输层，甚至是网络层的一部分。例如，不可靠的IP多播服务是在网络层实现的。但是，由于这些服务目前尚无法应用，它们通常使用传输层的服务来实现，传输层的服务将它们放在中间件中。如果把可扩展性加以考虑的话，只有充分考虑应用程序的需求时可靠性才能得到保证。用更高、更特殊的网络层来实现这些服务存在一个很大的争议。

2. 描述一下客户端和服务器端之间使用套接字的无连接通信是如何进行的?

1. 客户端和服务端都需要创建一个套接字，但是只有服务器把套接字绑定到本地的端点上。
2. 服务器可以执行一个阻塞的read调用以等待从客户端发送的数据。
3. 在创建套接字之后，客户端仅执行一个阻塞调用以向服务端写数据。关闭连接是没有必要的。

3. 当要在基于DHT的系统中解析一个键值时,递归查询的主要缺点是什么?

如果没有结果返回，那么请求客户端无法发现问题在哪里。在解析键值过程中，可能发生消息丢失或节点失败的情况。

因此动态查询有时更好，客户端可以知道查询的问题在哪部分，并可以求助另一节点。

4. 使用protobuf和gRPC等远程过程调用的方法实现消息订阅(publish-subscribe)系统,该订阅系统能够实现简单的消息传输,还可以控制消息在服务器端存储的时间。

• 环境搭建

操作系统：Ubuntu18.04

编程语言：Python

```
pip install grpcio
pip install protobuf
pip install grpcio-tools
```

• 配置proto文件

`Protobuf` 是一套类似 `Json` 或者 `XML` 的数据传输格式和规范，用于不同应用或进程之间进行通信时使用。通信时所传递的信息是通过 `Protobuf` 定义的 `message` 进行打包，然后编译成二进制的码流再进行传输或者存储。

`service pubsub` 定义了需要编写的服务的名称，其接口为 `rpc pubsubServe (mes2server) returns (mes2client) {}` ，即通信时，客户端向服务器发送消息 `mes2serve` ，服务器返回消息 `mes2client` 给客户端。这两个消息在之后生成的代码中会以结构体的形式保存。

具体配置如下：

```
syntax = "proto3";

package rpc_package;

// 定义服务
service pubsub {
    // 定义服务的接口
    rpc pubsubServe (mes2server) returns (mes2client) {}
}

// 定义上述接口的参数数据类型
message mes2server {
    string mes1 = 1;
}

message mes2client {
    string mes2 = 1;
```

```
}
```

在终端运行以下指令，使用 `gRPC protobuf` 生成工具生成对应语言的库函数：

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

指令执行后，会在当前文件夹下生成文件 `pubsub_pb2.py` 和 `pubsub_pb2_grpc.py` 。文件组织如下：

• 客户端

通过 `grpc.insecure_channel` 可以配置通信的服务器的IP地址和端口。这里设置本机地址和端口 `50074` ：

在循环中调用之前声明好的服务 `pubsubServe` 。消息订阅系统中的本质是服务器能向订阅的客户端统一发送消息的群发功能，服务器发送消息而连接的客户端集体各自接受消息并打印，客户端不需要向服务器发送消息。

使用参数 `timeout=500` ，从而能够**控制消息在服务器端存储的时间。**

```
import logging
import grpc
from pubsub_pb2 import mes2server
from pubsub_pb2_grpc import pubsubStub

def run():
    # 创建通信信道
    with grpc.insecure_channel('localhost:50074') as channel:
        # 客户端通过stub来实现rpc通信
        stub = pubsubStub(channel)
        while 1:
            try:
                mes = stub.pubsubServe(mes2server(mes1='client'),
timeout=500)
                print(mes)
            except KeyboardInterrupt:
```

```
                exit(0)

logging.basicConfig()
run()
```

• 服务器

在 `serve` 中，首先创建服务器实例， `ThreadPoolExecutor` 创建线程池， `max_workers=2` 表示只有2个线程，即最多有2个客户端与服务器相连。将对应的任务处理函数添加到 `rpc server` 中，然后设置IP和端口，开放服务器。

消息订阅系统相当于服务器将消息广播给连接的客户端的能力。我准备实现的基本功能是服务器能够随时输入一个字符串，输入完成后所有的客户端都能收到该字符串。每个服务器-客户端的连接由一个子线程完成（主线程一直处于循环等待中，不参与服务器与客户端的信息交互）。

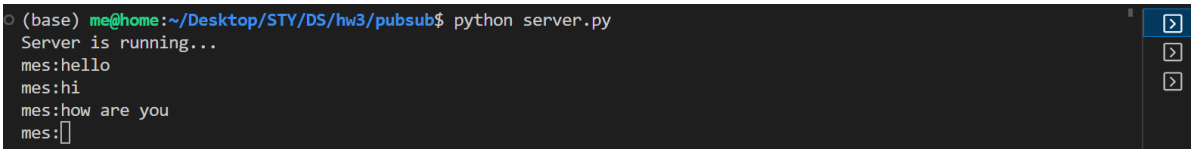
在实现消息订阅服务的函数 `pubsubServe` 中， `threadLock` 为实现线程互斥机制的锁，之后通过锁来控制对消息的输入和线程的阻塞等待输入。 `n` 为标志位， `n==1` 表示消息已经被输入，否则 `n==0` ，消息还没有输入，需要一个线程发起消息输入的命令，其他所有线程阻塞等待。等 `n==1` 时，输入完成，所有线程将输入的消息发给对应的客户端并且将 `n` 改为0。

```
import grpc
import logging
import time
import threading
from concurrent.futures import ThreadPoolExecutor
from pubsub_pb2_grpc import add_pubsubServicer_to_server, pubsubServicer
from pubsub_pb2 import mes2client

class PubsubServer():
    # 实现pubsub.proto中定义的接口
    def __init__(self):
        self.threadLock = threading.Lock()
        self.n = 0
        self.mes = "default"

    def pubsubServe(self, request, context):
        if self.n == 0:
            self.threadLock.acquire()  # 线程锁
            self.n += 1
            self.mes = input('mes:')
            self.threadLock.release()  # 释放锁
        self.threadLock.acquire()  # 线程锁
        self.n = 0
        self.threadLock.release()  # 释放锁
        return mes2client(mes2=self.mes)

def serve():
    # 通过thread pool来并发处理server的任务
    server = grpc.server(ThreadPoolExecutor(max_workers=2))
```

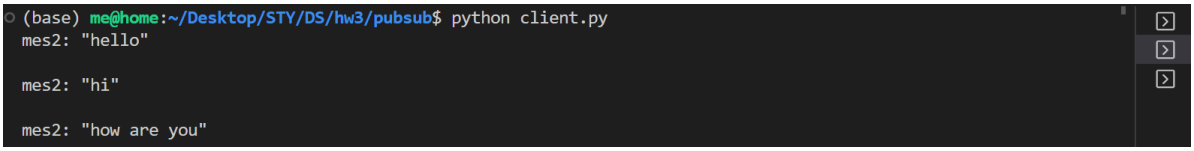
```python
        # 将对应的任务处理函数添加到rpc server中
        add_pubsubServicer_to_server(PubsubServer(), server)
        # 设置IP地址和端口
        server.add_insecure_port('[::]:50074')
        print("Server is running...")
        server.start()
        try:
            while True:
                time.sleep(60 * 60 * 24)
        except KeyboardInterrupt:
            server.stop(0)

    logging.basicConfig()
    serve()
```

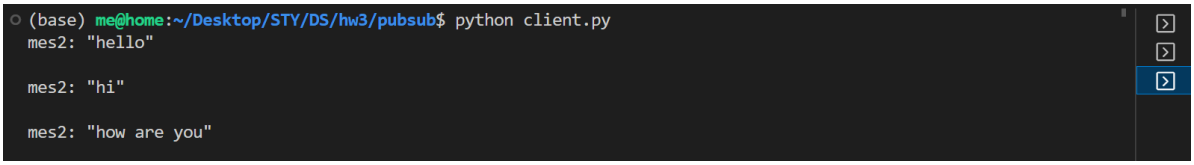
三、实验结果

启动服务器，等待客户端连接。客户端连接后，输入消息...



消息被转发到同时运行的2个客户端





四、遇到的问题及解决方法

在实现消息订阅服务的函数 `pubsubServe` 时，未指明消息赋值的变量，导致如下报错：

```
TypeError: No positional arguments allowed
```

解决方法：指明消息赋值变量为mes2，即发送给客户端的string变量。

```
    return mes2client(mes2=self.mes)
```