

分布式系统课程大作业

Client-Server 架构的集中式

分布式键值存储系统

21307376 曹永皓 系统结构班

一、开发环境

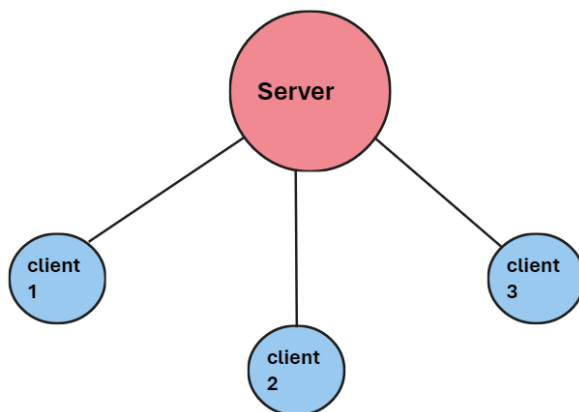
操作系统: Windows10;
编程语言: Python 3.8.10
IDE: VSCode

二、项目描述

本项目实现了一个 Client-Server 架构的集中式的分布式键值存储系统。

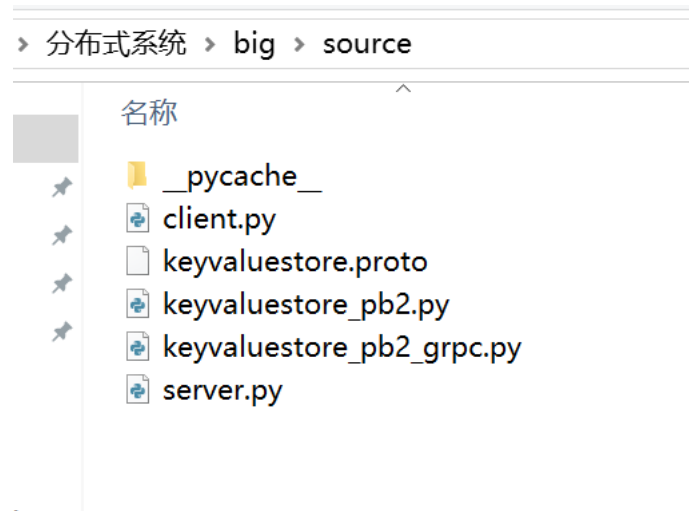
该系统的运作方式是: 一个服务器保持运行状态, 监听来自多个客户端的关于键值存储的操作, 包含“PUT”“GET”“DEL”, 即“添加/修改键值对”, “读取键值对”, “删除键值对”三种基本操作。客户端与服务器端的通信方式为 RPC。并且该系统通过简单的锁机制, 实现了当其中一个客户端节点对某一特定键进行访问操作时, 其余客户端节点对该键值对的访问操作将被阻塞 (中断), 同时一并保证了面向客户的单调写一致性。

测试时的系统架构图:



测试方式：

项目的文件结构如下：



进行测试时由于只有一台设备，故采用开启多个终端窗口的方式模拟多个节点。首先开启一个终端，在项目目录下运行 server.py 文件，以运行服务器节点；随后额外开启 3 个终端，分别运行 client.py 文件，以运行客户端节点。在 3 个客户端上进行关于键值存储的各项操作。

三、项目实施

客户端和服务端节点之间的通信按要求需使用 RPC 机制，这里使用由 google 提供的 gRPC。首先编写一份 .proto 文件，定义 gRPC 的消息和服务，如图所示：

```
syntax = "proto3";

message LockRequest {
    string key = 1;
}

message LockResponse {
    bool lock_acquired = 1;
}

message Request {
    string key = 1;
    string value = 2;
}

message Response {
    string message = 1;
}

service KeyValueStore {
    rpc Put (Request) returns (Response);
    rpc Get (Request) returns (Response);
    rpc Delete (Request) returns (Response);
    rpc AcquireLock (LockRequest) returns (LockResponse);
    rpc ReleaseLock (LockRequest) returns (LockResponse);
}
```

再在终端执行指令：

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

以编译生成 keyvaluestore_pb2.py 和 keyvaluestore_pb2_grpc.py 两个文件，用于后续的 RPC 通信实现。（这里已经事先安装好了实现 gRPC 的所需包以及编译 proto 文件的工具）

分别编写服务器端和客户端的代码。

服务器端：

定义一个键值存储服务器类，拥有一个字典成员 datastore，用于存储键值对；一个字典成员 locks，作为缓冲区用于记录客户端访问指定键时所提供的锁。

根据所编写的 keyvaluestore.proto 文件中所定义的服务名称，定义成员函数 Put(),Get(),Del()。这里以函数 Put()为例，解释在服务器端执行键值添加或修改的过程。

```
def Put(self, request, context):
    lock_key = request.key

    if lock_key in self.locks: # 使用锁来保护对存储的访问
        self.datastore[request.key] = request.value
        return keyvaluestore_pb2.Response(message=f'Key "{request.key}" stored successfully.')
    else:
        return keyvaluestore_pb2.Response(message=f'Failed to store key "{request.key}". Lock not acquired.')
```

客户端发送来的请求消息包含了指定的键，以及希望写入的对应的值。服务器检查锁缓冲区中是否含有该键对应的锁，若含有锁，则允许访问，将 datastore 中的指定键对应的值进行添加或修改，完成写入操作，并向客户端返回写入成功的消息；若不含有锁，则不允许访问，向客户端返回写入失败的消息。

Get(),Del()函数也以类似的方式进行实现，具体可查看附带的源码。

```
def serve():
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    keyvaluestore_pb2_grpc.add_KeyValueStoreServicer_to_server(KeyValueStoreServicer(), server)
    server.add_insecure_port('[::]:8000')
    server.start()
    print("Server listening on [::]:8000...")
    server.wait_for_termination()
```

函数 serve()定义了启动服务器的方法，架设通过 gRPC 进行通信的服务器，将端口设为 8000

客户端：

首先连接到服务器

```
# 连接到服务器
channel = grpc.insecure_channel('localhost:8000')
stub = keyvaluestore_pb2_grpc.KeyValueStoreStub(channel)
```

客户端启动后，会进入一个 while True 的死循环中，直到用户输入退出指令以结束客户端的运行。

客户端启动后，可以输入的指令有 put、get、del、quit。

这里同样以 PUT 操作为例，展示在客户端的键值存储操作实现

```
# 在客户端使用键值存储
while True:

    print("Please input the operation you want:")
    operation = input()

    # 键值对添加/修改
    if operation == "PUT" or operation == "put":
        key_input = input("Please input the Key:\n")

        # 请求获取锁
        lock_response = stub.AcquireLock(keyvaluestore_pb2.LockRequest(key=key_input))
        if lock_response.lock_acquired:
            value_input = input("Please input the Value you want to match:\n")
            put_response = stub.Put(keyvaluestore_pb2.Request(key=key_input, value=value_input))
            print(put_response.message)
            lock_release_mes = stub.ReleaseLock(keyvaluestore_pb2.Request(key=key_input)) # 释放分布式锁
        else:
            print("Failed to acquire lock. Another client is currently accessing the key.")
```

首先向服务器端发送申请锁的请求，根据从 stub 获得的发送回来的消息判断锁是否已成功申请。若已经成功申请锁，则输入希望写入的值，提交写入并获得从服务器端返回的写入是否成功的消息；若未能成功申请锁，说明有其他节点（进程）正在访问当前的键，输出占用消息并中断操作。

GET, PUT 操作也以类似的方式实现，重点在输入希望访问的键后向服务器端申请对应的锁。

锁机制的实现：

在服务器端定义有两个函数 AcquireLock()和 ReleaseLock()，对应.proto 文件中定义的服务名。两函数的实现如下：

```
def AcquireLock(self, request, context):
    lock_key = request.key

    lock = True

    if lock_key not in self.locks:
        self.locks[lock_key] = lock # 将锁存储到字典中
        return keyvaluestore_pb2.LockResponse(lock_acquired=True)
    else:
        return keyvaluestore_pb2.LockResponse(lock_acquired=False)

def ReleaseLock(self, request, context):
    lock_key = request.key

    # 释放分布式锁
    if lock_key in self.locks:
        del self.locks[lock_key]

    return keyvaluestore_pb2.LockResponse(lock_acquired=False)
```

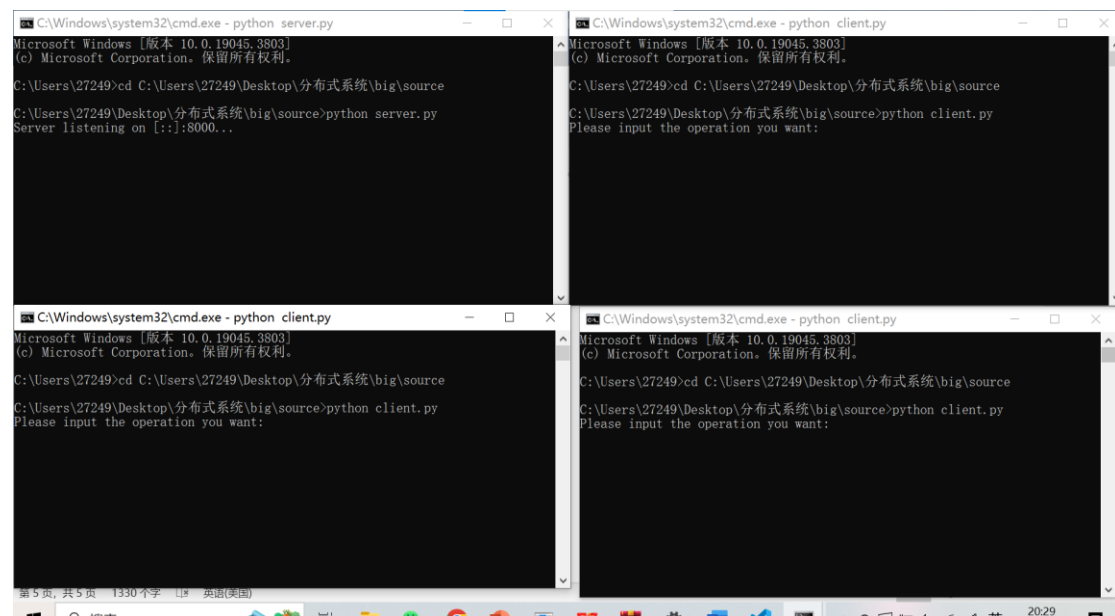
客户端申请锁时，根据占用情况决定是否将锁缓冲区中的对应位置加锁，并返回申请结果消息；客户端申请释放锁时，将对应键位置的锁去除。

一致性实现：

在锁机制下，多个客户不能同时访问同一个键。面向客户的单调写一致性被自然地保证了。

四、操作测试

启动终端，运行各自对应的 python 文件，其中左上为服务器端，右上、左下、右下分别为客户端 1、2、3



```
C:\Windows\system32\cmd.exe - python_server.py
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python server.py
Server listening on [::]:8000...

C:\Windows\system32\cmd.exe - python_client.py
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python client.py
Please input the operation you want:

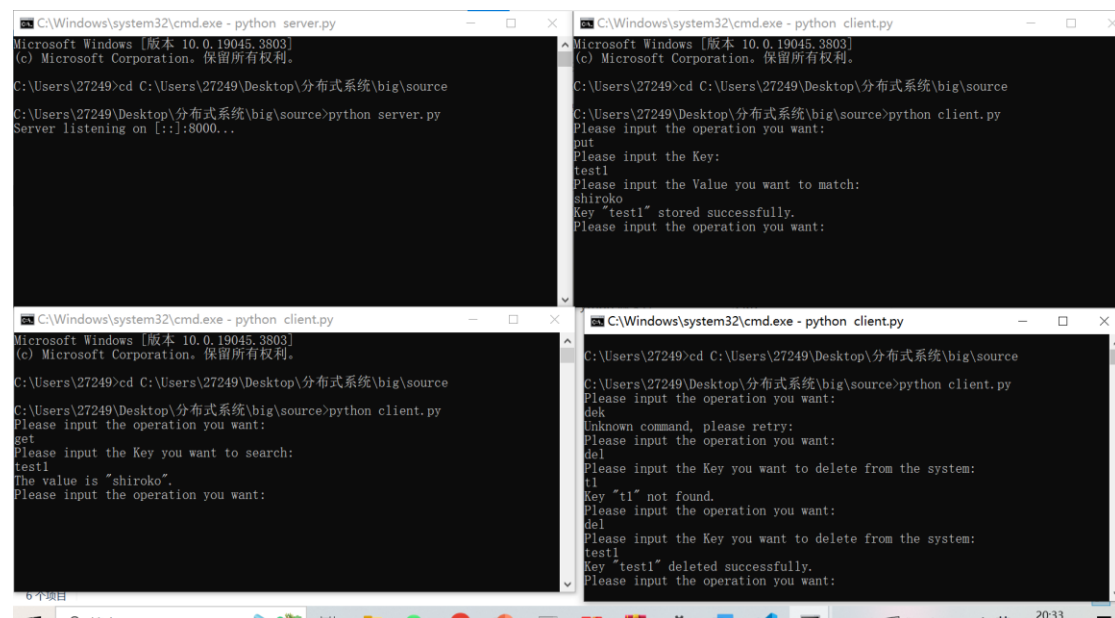
C:\Windows\system32\cmd.exe - python_client.py
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python client.py
Please input the operation you want:

C:\Windows\system32\cmd.exe - python_client.py
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python client.py
Please input the operation you want:
```

在不同的客户端，执行 PUT，GET，DEL 操作



```
C:\Windows\system32\cmd.exe - python_server.py
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python server.py
Server listening on [::]:8000...

C:\Windows\system32\cmd.exe - python_client.py
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python client.py
Please input the operation you want:
put
Please input the Key:
test1
Please input the Value you want to match:
shiroko
Key "test1" stored successfully.
Please input the operation you want:

C:\Windows\system32\cmd.exe - python_client.py
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python client.py
Please input the operation you want:
get
Please input the Key you want to search:
test1
The value is "shiroko".
Please input the operation you want:

C:\Windows\system32\cmd.exe - python_client.py
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python client.py
Please input the operation you want:
del
Unknown command, please retry:
Please input the operation you want:
del
Please input the Key you want to delete from the system:
t1
Key "t1" not found.
Please input the operation you want:
del
Please input the Key you want to delete from the system:
test1
Key "test1" deleted successfully.
Please input the operation you want:
```

所有操作均成功执行。

下面演示锁机制：两个客户端节点（进程）同时访问相同的键

```
C:\Windows\system32\cmd.exe - python server.py
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python server.py
Server listening on [::]:8000...

C:\Windows\system32\cmd.exe - python client.py
test1
Please input the Value you want to match:
shiroko
Key "test1" stored successfully.
Please input the operation you want:
put
Please input the Key:
test2
Please input the Value you want to match:

C:\Windows\system32\cmd.exe - python client.py
C:\Users\27249>cd C:\Users\27249\Desktop\分布式系统\big\source
C:\Users\27249\Desktop\分布式系统\big\source>python client.py
Please input the operation you want:
get
Please input the Key you want to search:
test1
The value is "shiroko".
Please input the operation you want:
put
Please input the Key:
test2
Please input the Value you want to match:
mika
Key "test2" stored successfully.
Please input the operation you want:

C:\Windows\system32\cmd.exe - python client.py
Unknown command, please retry:
Please input the operation you want:
del
Please input the Key you want to delete from the system:
t1
Key "t1" not found.
Please input the operation you want:
del
Please input the Key you want to delete from the system:
test1
Key "test1" deleted successfully.
Please input the operation you want:
put
Please input the Key:
test2
Failed to acquire lock. Another client is currently accessing the key.
Please input the operation you want:
```

可以看到在客户端 1 进行 PUT 操作时，客户端 3 想要访问相同的键，但过程被中断。