**分布式键值（key-value）存储系统**

姓名：杨子昂

班级：计科大数据班

学号：21307181

日期：1.9

强烈建议您查看另一份由markdown生成的pdf报告，代码输出和各式控制会好很多

正文：

1. 题目要求

要求

1）必须是分布式的键值存储系统，至少在两个节点或者两个进程中测试；

2) 可以是集中式的也可以是非集中式；

3) 能够完成基本的操作如：PUT、GET、DEL等；

4) 支持多用户同时操作；

5) 至少实现一种面向客户的一致性如单调写；

6) 需要完整的功能测试用例；

7) 涉及到节点通信时须采用RPC机制；

8) 提交源码和报告，压缩后命名方式为：学号\_姓名\_班级

加分项：

1）具备性能优化措施如cache等；

2）具备失效容错方法如：Paxos、Raft等；

3）具备安全防护功能；

4）其他高级功能；

1. 解决思路

在本次实验中我实现了分布式键值系统。具体采用的是CS架构，也就是服务器--客户端架构的形式实现，并且用RPC作为客户端和服务器之间的通信方式。

\*\*远程过程调用（RPC）:\*\*

是一种协议或技术，用于在不同的计算机之间执行代码。在RPC中，一个程序可以执行不在其地址空间的程序代码，就像这些代码是本地的一样。RPC原理可以分为以下几个关键步骤：

1. 过程调用请求 : 客户端程序调用一个执行在远程服务器上的过程（函数）。这通常涉及到对函数的普通调用，但是这个函数并不在同一个本地系统上。

2. 参数传递 : 客户端的RPC库负责收集过程调用的参数，并将其通过网络发送到服务器。这个过程称为“参数打包”或“序列化”。

3. 网络通信 : 客户端的RPC请求通过网络发送给服务器。服务器监听来自客户端的请求。

4. 服务器端处理 : 一旦服务器接收到请求，它将解包或反序列化请求中的参数，并执行本地过程调用。

5. 结果返回 : 执行完成后，服务器会将结果“打包”或“序列化”，然后通过网络发送回客户端。

6. 客户端接收结果 : 客户端的RPC系统接收到来自服务器的响应，解包数据，并将其作为函数调用的结果返回给客户端程序。

RPC的关键在于它使远程过程调用看起来像本地调用一样，从而简化了分布式系统的开发。在本次实验中我选择的是rpyc这个主要应用于python的RPC库，它可以直线客户端和服务器之间的实时连接，并且通过函数映射的方式将服务器端的函数和客户端的函数进行对应和参数传递，从而实现了在客户端远程调用服务器端的函数。

键值存储的后端部分是借助python中的sqlitedict来实现的，是一个轻量化的数据库，对于键值类型的数据有非常高效的操作能力。

1. 实现细节

实验环境：

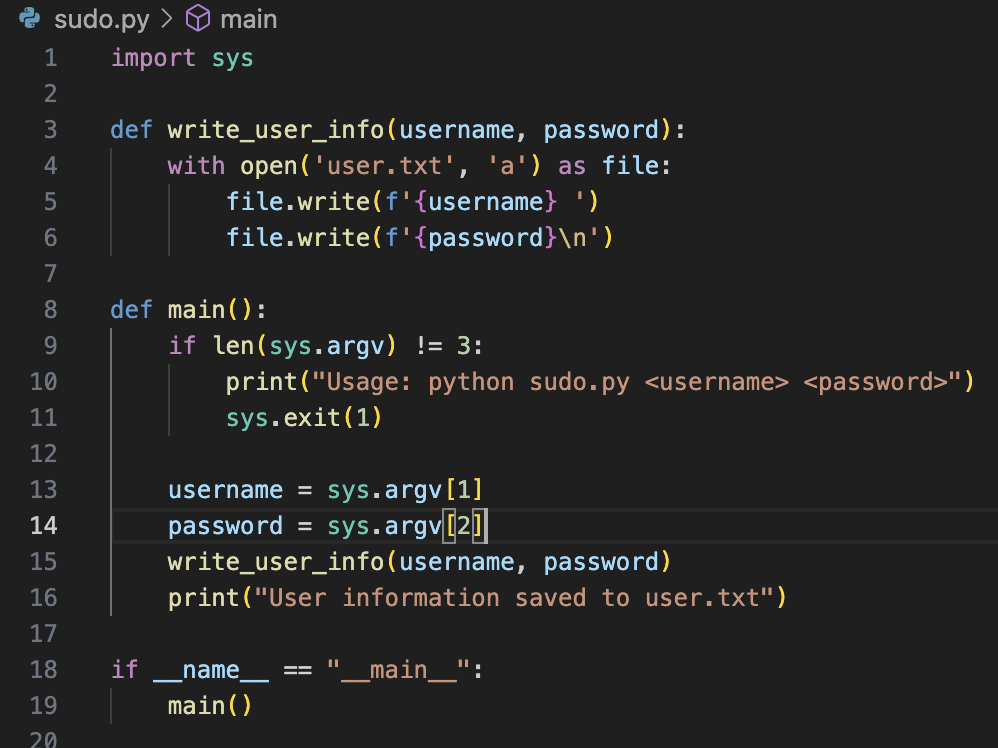
1. python 3.7.16

2. rpyc(RPC通信)

3. sqlitedic(后端数据库)

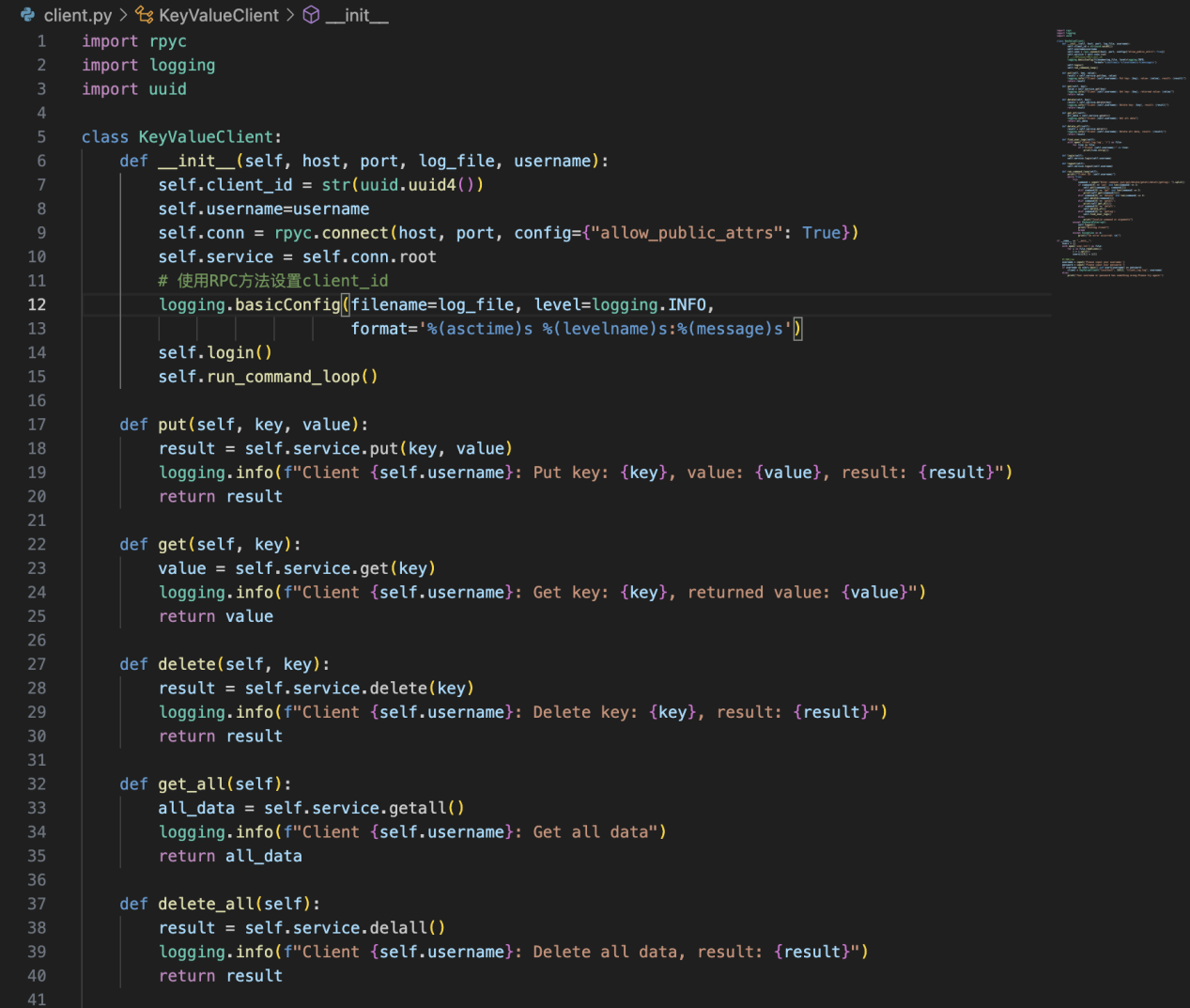
4. logging(⽣成⽇志)

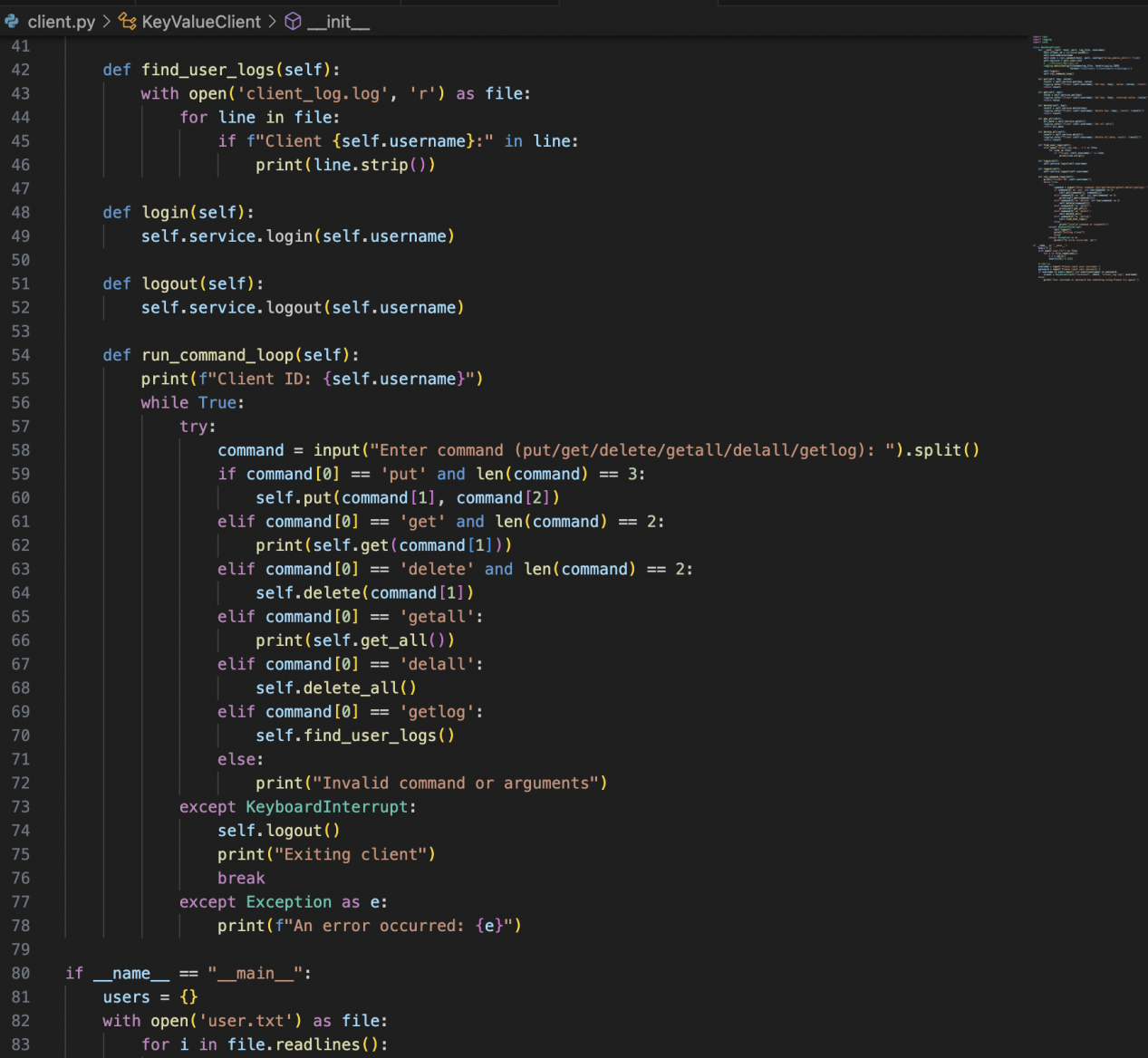
5. uuid(⽣成不重复的uid)



sudo.py是⽤来管理⽤户信息的，主要功能是向其中添加⽤户名 username以及密码

password, 由于这并不是整个键值存储系统的主要部分，故只实现了保证测试中添 加⽤户的功能，并未涉及到增删改查user信息的功能，当然这些实现也⽐较简单 ⽽且并 不是重点。（甚⾄可以⽤键值存储系统来实现⽤户信息管理） 功能是在命 令⾏输⼊ python sudo.py <username> <password>这样格式的⽂字 即可向user.txt ⽂档写⼊新的⽤户名和密码。







KeyValueClient 类:

* init ⽅法: 类的初始化⽅法。它接受主机地址、端⼝、⽇志⽂件路径和⽤户名作为参数。客户端ID通过 uuid.uuid4() ⽣成，确保每个客户端实例有唯⼀的ID。此外，它还建⽴了与远程服务的连接，并初始化了⽇志记录。
* put、get、delete、get\_all 和 delete\_all ⽅法: 这些⽅法分别对应于键值存储的基本操作，它们通过RPC调⽤远程服务的相应⽅法，并记录操作⽇志。
* find\_user\_logs ⽅法: 从⽇志⽂件中检索并打印与当前⽤户相关的⽇志记录。
* login 和 logout ⽅法: 通过RPC调⽤远程服务来处理⽤户登录和注销。
* run\_command\_loop ⽅法: 这是⼀个交互式命令循环，允许⽤户通过命令⾏输⼊执⾏不同的操作。

主程序:

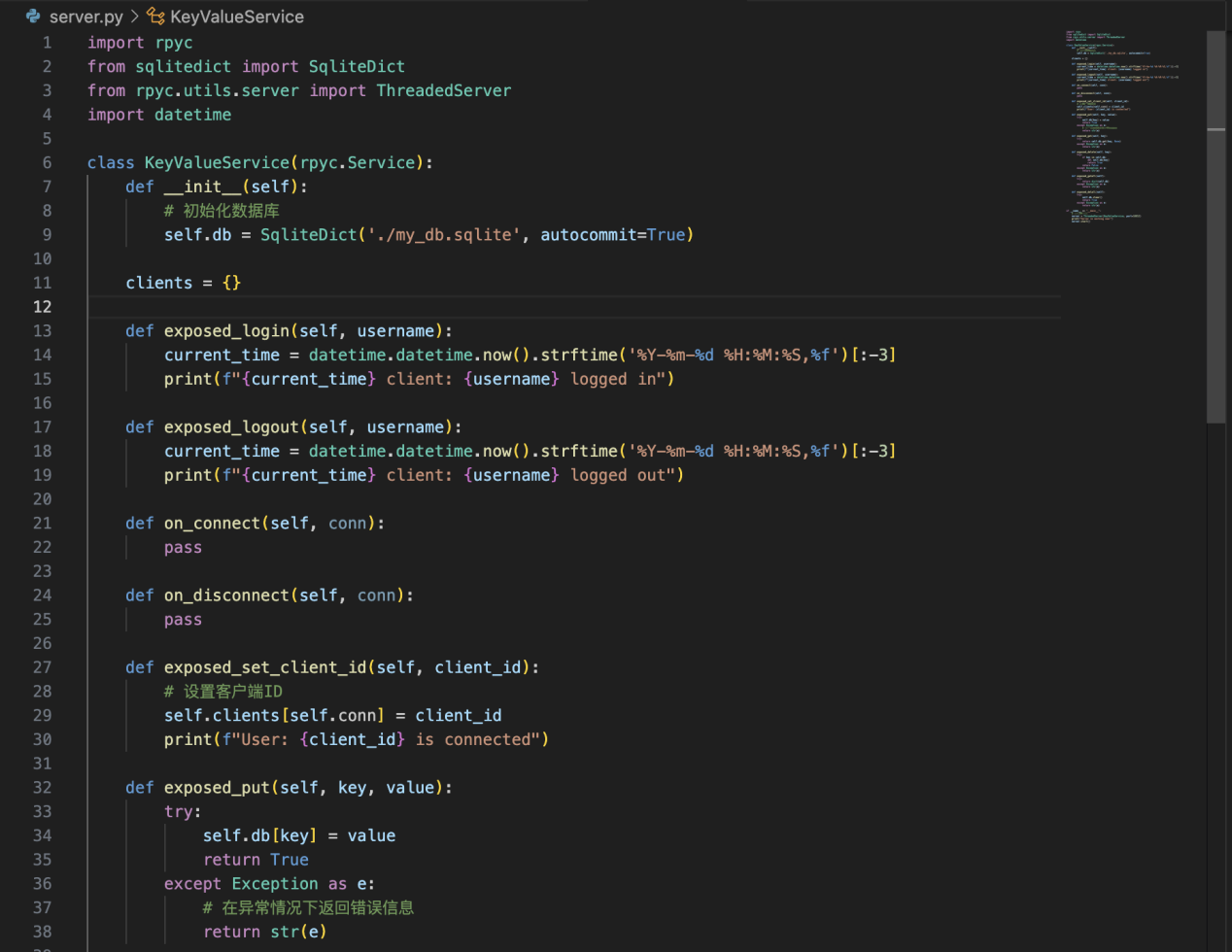
1. 从 user.txt ⽂件中读取⽤户信息，并存储在字典 users 中。

2. 请求⽤户输⼊⽤户名和密码。

3. 验证⽤户名和密码。如果认证通过，则创建 KeyValueClient 实例并连接到服务端；

否则，提示错误信息。

这段代码的核⼼功能是通过RPC技术实现客户端与服务端的通信，使得客户端可以远程执⾏键值存储的操作。它同时也提供了⼀个基本的⽤户认证机制和命令⾏界⾯，使得⽤户可以直接通过命令与远程服务进⾏交互。





**KeyValueService 类:**

* 类继承⾃ rpyc.Service，这是创建RPC服务的基础。
* init\_ ⽅法: 初始化时创建⼀个 SqliteDict 实例，⽤于数据存储。
* exposed\_login 和 exposed\_logout ⽅法: ⽤于记录客户端登录和登出的时间。
* \*\*on\_connect 和 on\_disconnect ⽅法: \*\*这些⽅法可以在客户端连接或断开时执⾏操作，⽬前它们是空的。
* exposed\_set\_client\_id ⽅法: 允许客户端设置其ID，这些ID被存储在 clients 字典中。
* exposed\_put、exposed\_get、exposed\_delete、exposed\_getall 和
* exposed\_delall ⽅法: 这些⽅法提供了键值对存储的基本功能，包括增加、获取、删除单个键值对，获取所有键值对，以及删除所有键值对。

**主程序:**

1. 创建⼀个 ThreadedServer 实例，指定 KeyValueService 作为服务类，并设置监 听端⼝为 18812。

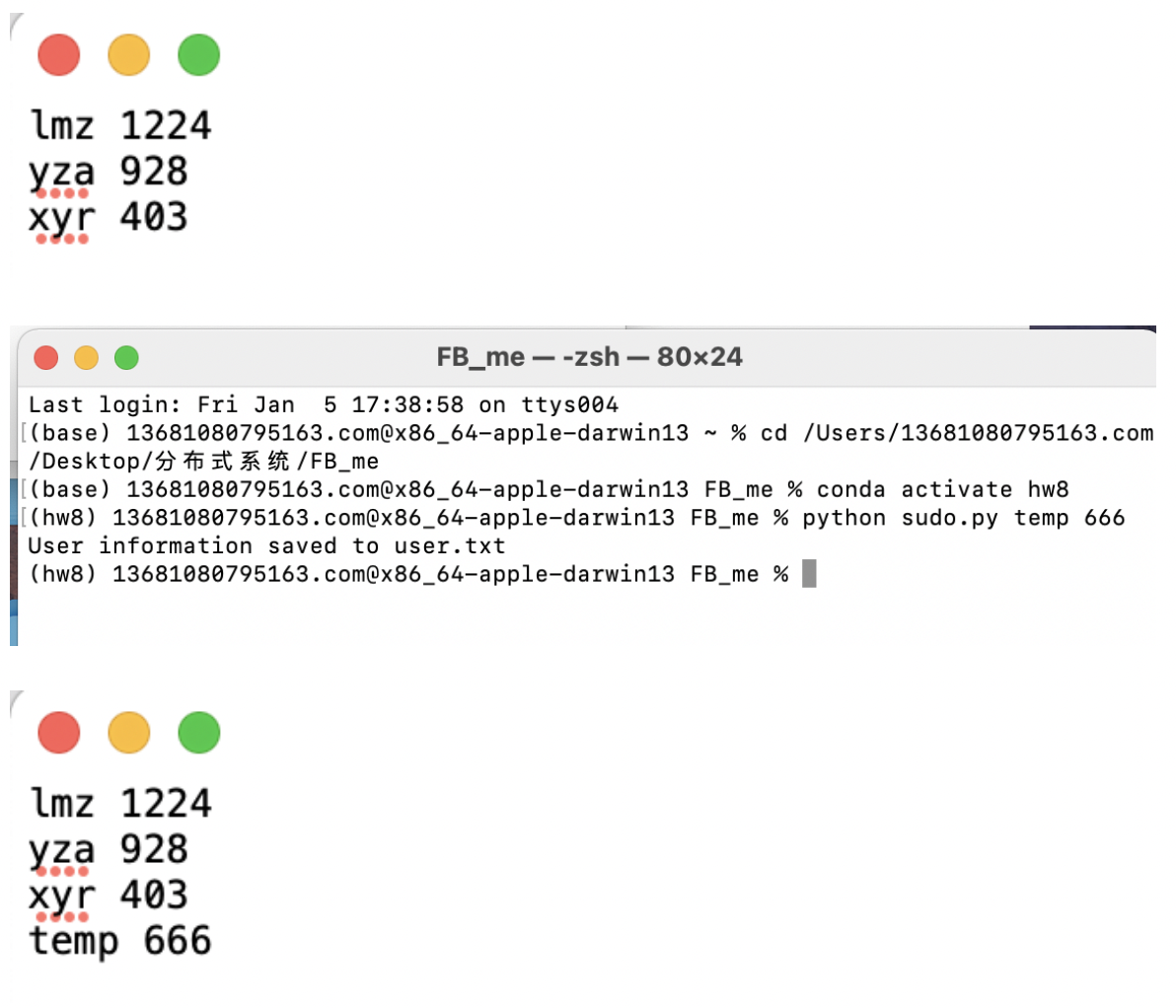
2. 启动服务器，等待客户端连接。 这个程序的主要功能是通过RPC（远程过程调 ⽤）技术实现了⼀个基本的键值对存储服务。客户端可以通过远程调⽤来执⾏ 数据的添加、检索和删除等操作。它利⽤ sqlitedict 库将数据持久化保存在 SQLite数据库中，确保数据的稳定存储和管理。此外，由于服务器采⽤了多线 程处理机制，它能够同时处理多个客户端的连接和请求。这种服务适合于那些 需要键值存储功能的应⽤场景

1. 运行情况

在实验测试环节，我将运行三个client程序和一个server程序以此来模拟多用户的使用场景。

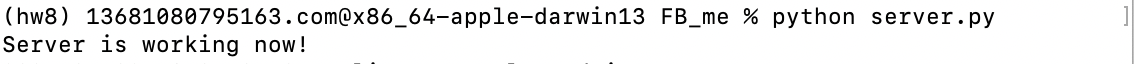
1. 首先是测试sudo.py

这是运行前的user.txt文件的内容，一会儿我将会使用这三个名称和密码去进行 client和server的测试。sudo.py的作用是向其中写入用户名和密码进行注册



可以看到成功将新的⽤户名temp和密码666写⼊user.txt

2. 接下来我们⽤前三个⽤户名和密码登录client端，并且连到同⼀个server上

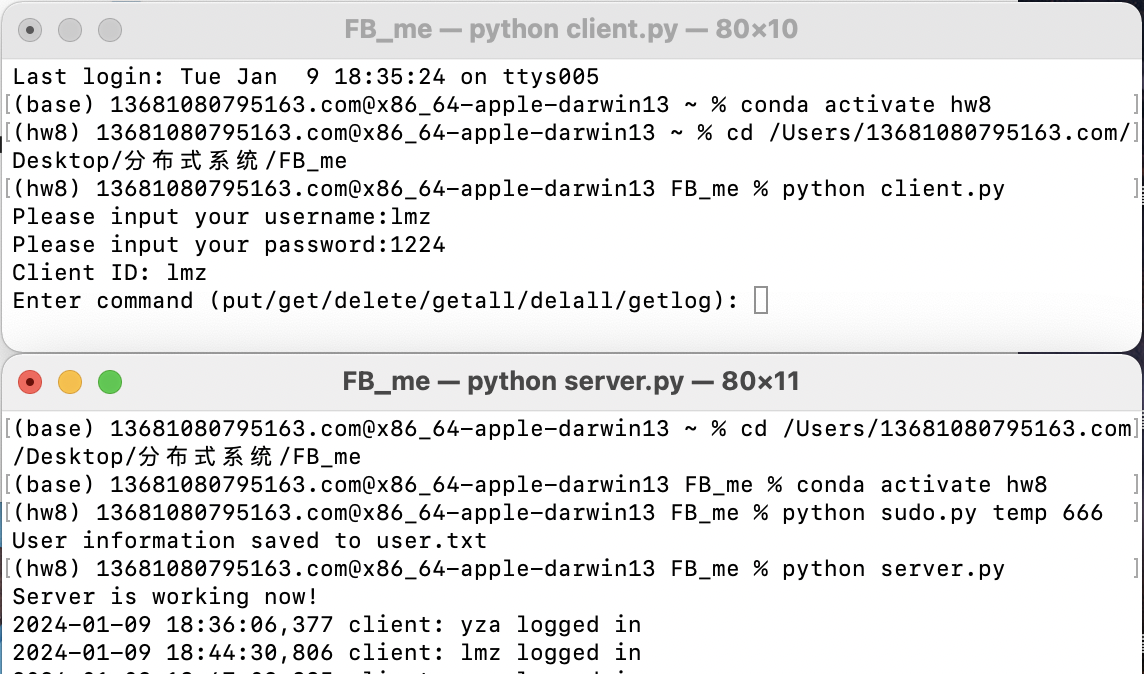


可以看到server已经在运⾏，然后我们登陆三个客户端并连接到服务器



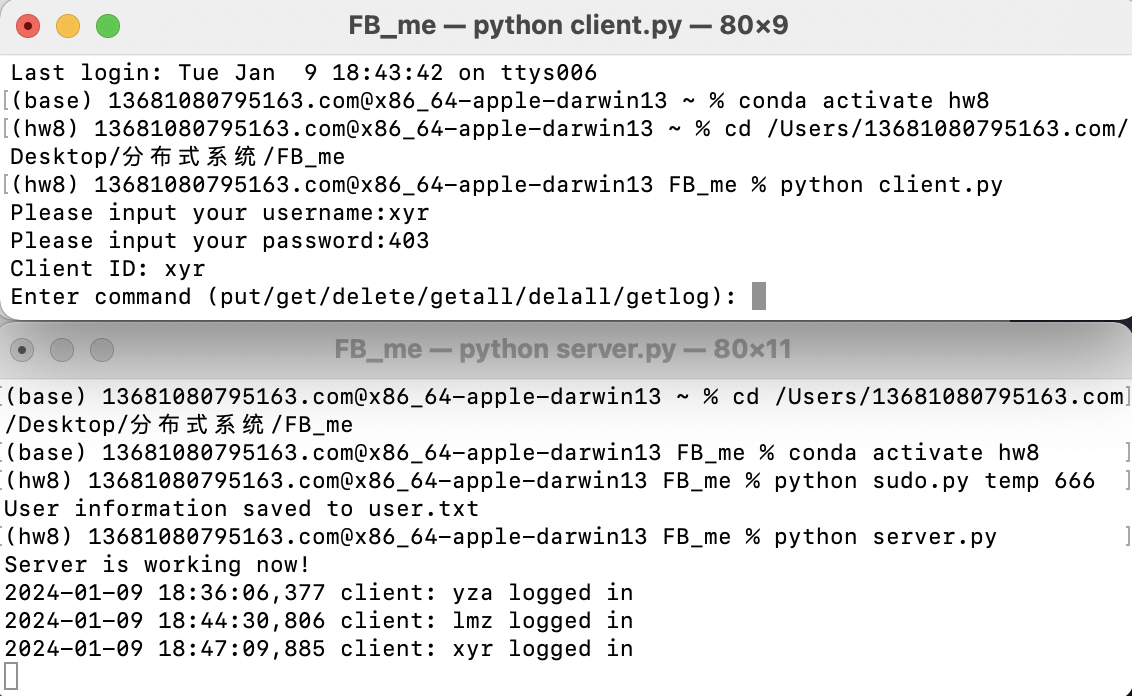
这是⽤户yza登陆的界⾯，可以看到客户端登录成功后在服务器端有⼀个登录成功

的提示。



这是⽤户lmz登陆的界⾯，可以看到客户端登录成功后在服务器端有⼀个登录成功

的提示。



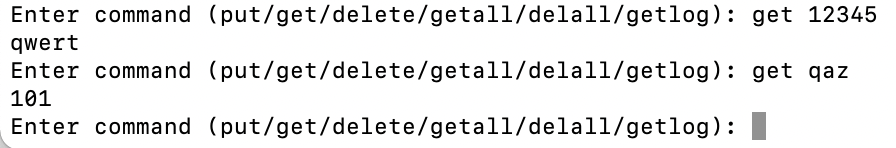
这是⽤户xyr登陆的界⾯，可以看到客户端登录成功后在服务器端有⼀个登录成功

的提示。

3. 接下来我们测试功能是否正常，我将在yza客户端上进⾏测试，在输⼊键值阶段结 束后我会在lmz和xyr客户端进⾏查看，验证整体的⼀致性

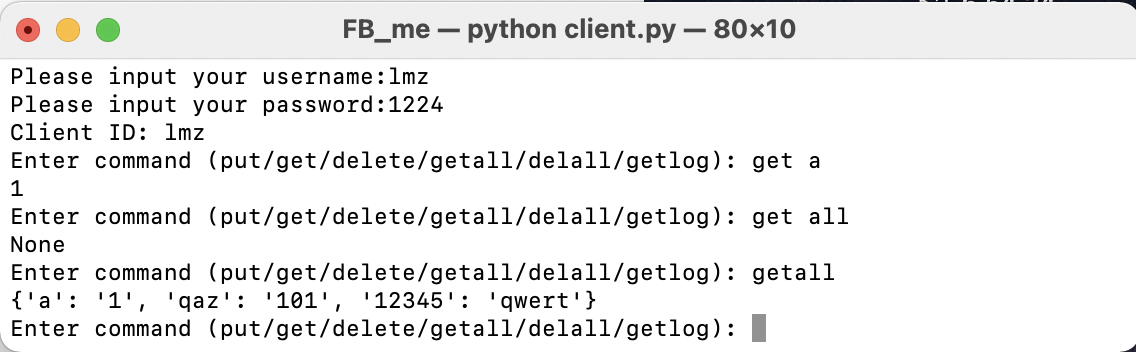


在这张图上我⾸先查看已有键值 getall并全部清除 delall，然后我测试了三次 put功能，分别是单个字符，字符串，和数字搭配字符。并调⽤getall⼀次性 查 看了所有键值。当然我们也可以使⽤ get查看单个键值对。

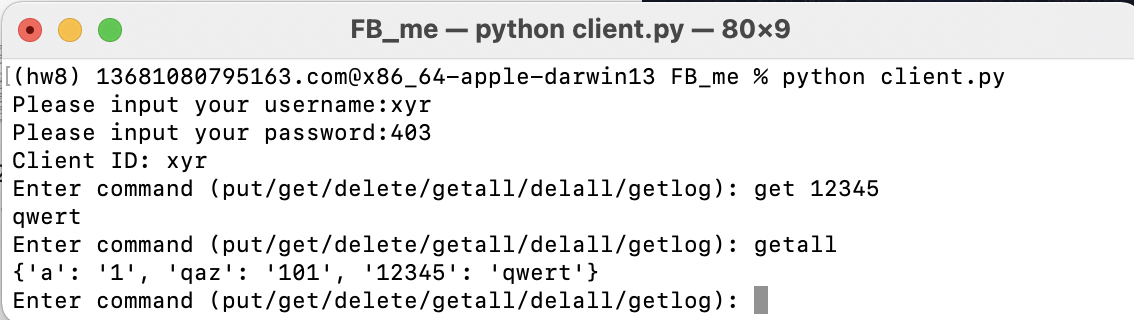


4. 检查多客户端⼀致性，我们将从lmz客户端和xyr客户端来查看yza客户端对数据的更改是否全局可⻅

lmz客户端



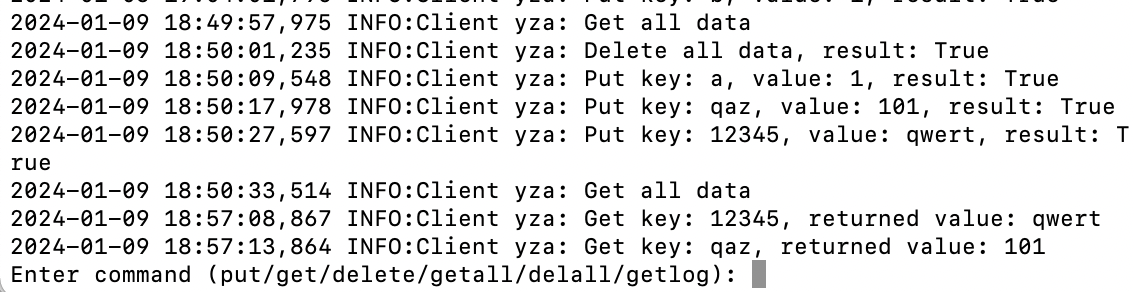
Xyr客户端



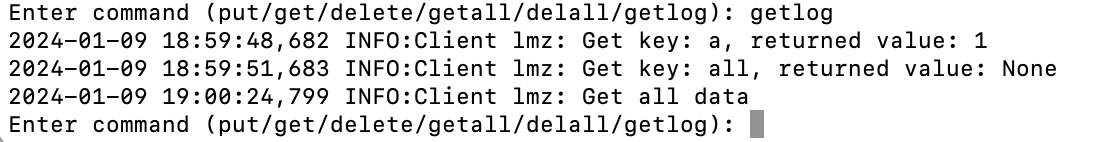
通过上⾯的检查可以证明yza客户端的操作是保证了⼀致性的。

1. ⽇志功能：我在上⾯的基本操作之上实现了⽇志功能，可以记录全局的⽇志操 作，并且每个客户端只能查看⾃⼰的⽇志记录，⽆法查看别⼈的⽇志记录。

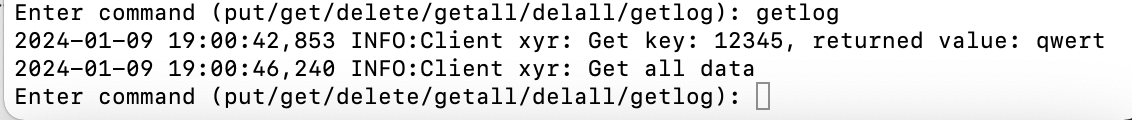
qaz客户端的⽇志记录



Lmz客户端的⽇志记录



Xyr客户端的⽇志记录

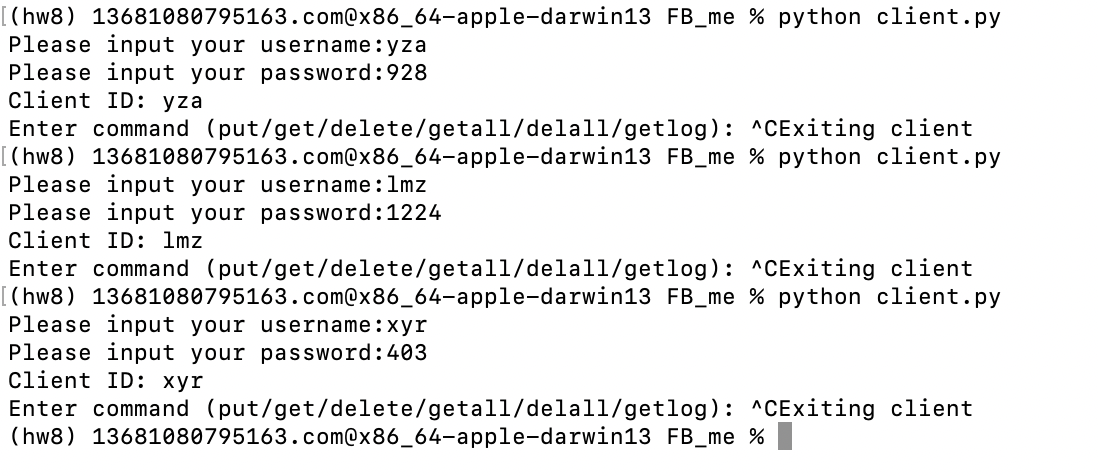


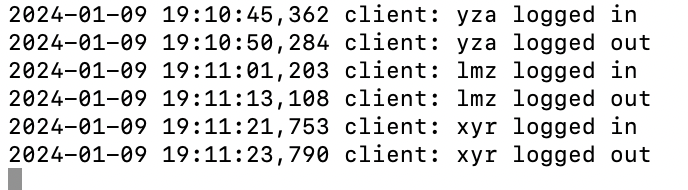
1. 剩余功能的统⼀测试（delete delall等）



delete（单个删除）以及delall（全部删除）功能测试正常

1. 客户端登⼊和登出的服务器端提示 模拟三个⽤户的登⼊登出，以及在服务器端的结果





总结

完成⼀个基于RPC的分布式键值存储系统是⼀次极富挑战性和教育意义的经历。在这个过程中，我深⼊理解了远程过程调⽤的⼯作原理，包括数据序列化、⽹络通信和服务发现等关键技术。⾯对⽹络延迟、数据⼀致性和故障恢复等问题，我学习并实践了诸如RPC等解决⽅案，这不仅增强了我的技术能⼒，也锻炼了我的问题解决和创新思维。此外，通过实际应⽤理论知识，我对分布式系统有了更深刻的理解，同时也认识到了持续学习在技术领域的重要性。总的来说，这是⼀次让我技术和思维都得到提升的宝贵经历