**Lab5实验报告**

**符永锐 517030910182**

1. **zookeeper环境搭建以及相关配置**
2. **运行环境：**

电脑型号：MacBook Pro (13-inch, 2017, Four Thunderbolt 3 Ports)

处理器：3.1 GHz 双核Intel Core i5

内存：8 GB 2133 MHz LPDDR3

运行系统：MacOS Catalina 10.15.5

运行软件：java 1.8.0\_222，intellij 2019.1.4，maven3.6.0

|  |
| --- |
|  |

1. **下载和配置zookeeper：**

下载的zookeeper版本是3.14.14。按照网上的教程，下载之后进入解压后的zookeeper-3.4.14文件夹,进入conf目录，新建名为“zoo.cfg”的文件，这是zookeeper的配置文件.

tickTime代表服务器与客户端之间交互的基本时间单元（ms）

initLimit 代表zookeeper所能接受的客户端数量

syncLimit代表服务器和客户端之间请求和应答之间的时间间隔

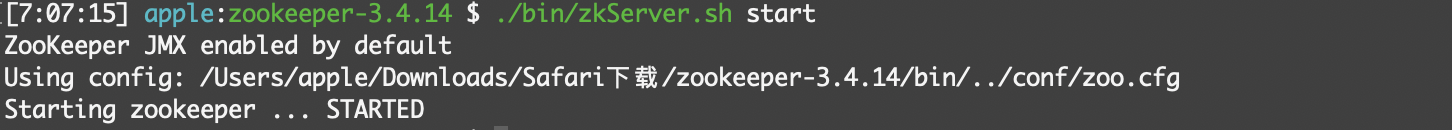
tickTime代表zookeeper中使用的基本时间单位, 毫秒值.

dataDir是数据目录. 可以是任意目录

dataLogDir是 log目录, 同样可以是任意目录. 如果没有设置该参数, 将使用和#dataDir相同的设置.

clientPort是 监听client连接的端口号.

**3、启动zookeeper**

进入解压目录zookeeper-3.4.14，输入 ./bin/zkServer.sh start 指令，得到以下结果：

说明启动zookeeper server进程成功。

**4、然后到intellij的设置。**

新建一个新的maven项目。然后就可以开始码代码啦。

1. **系统实现**

**一）整体架构**

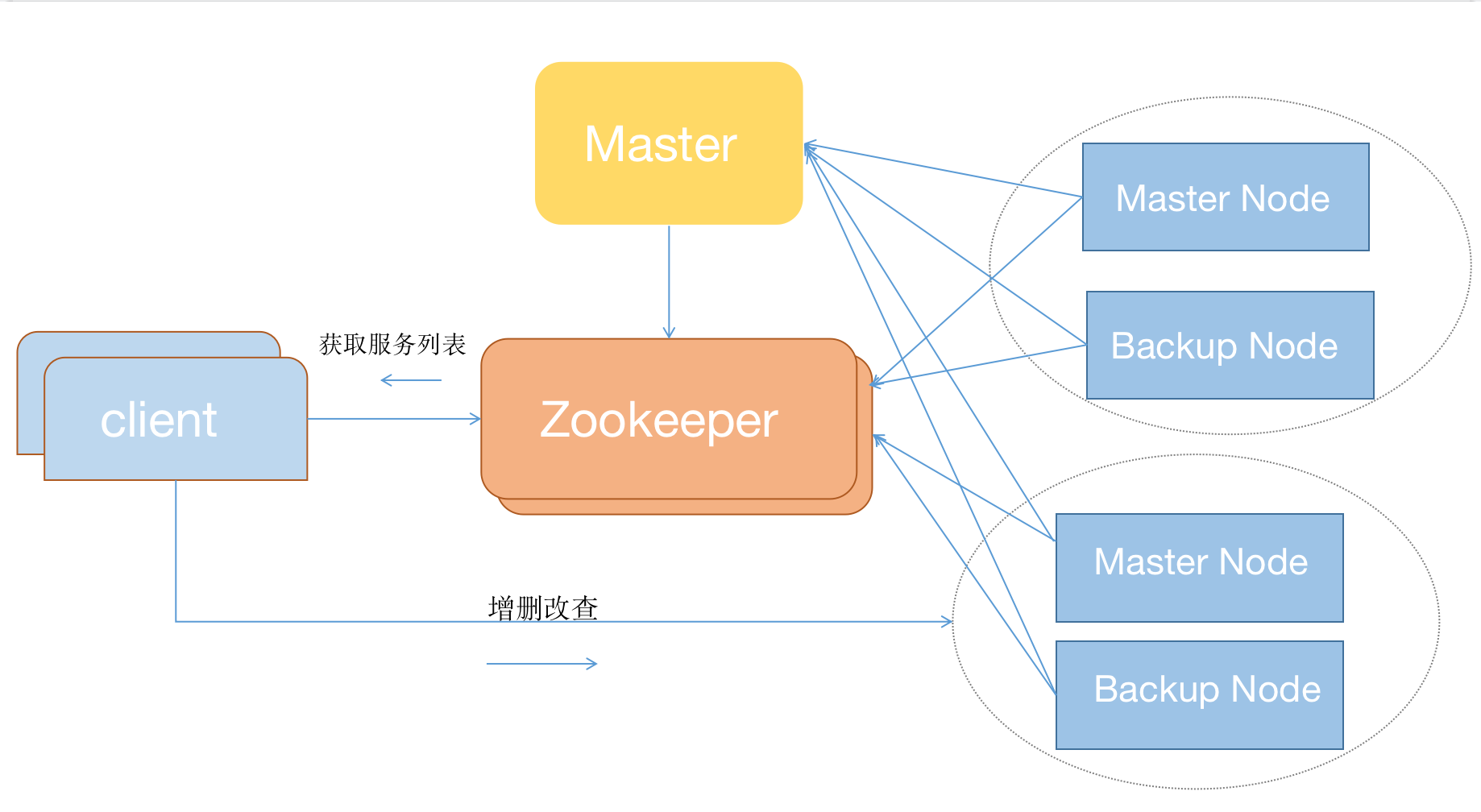
1. **模块组成**：主要由Client、Server、RunMaster三个子模块组成。各个子模块的功能如下：

Client：模拟用户，可以发增删改查的数据库请求。

Server：用于真正与用户交互数据的模块，是DataNode节点，还分为master和backup两种类型，用户只与master类型的DataNode进行交互。backup类型的DataNode是用来做备份的，当master类型的DataNode挂了之后，backup类型的DataNode会变成master类型的DataNode，以此来提供容错能力。而backup类型的DataNode挂了，对master类型的DataNode则没有什么影响。

RunMaster：用于管理DataNode的元数据；它存储DataNode的zookeeper path，Rmi URL，和DataNode的类型。新建的DataNode的Server都要注册到Master类中。

1. **设计架构图：**



1. **Zookeeper和远程调用框架RMI：**

Server、Master需要注册到Zookeeper，成为一个个Znode，然后通过RMI进行发布，通过RMI来进行交流。client通过RMI与Zookeeper连接，获取服务列表，也就是DataNode的RMI的url列表，然后通过RMI与DataNode进行交流。

1. **具体实现类：**

**1、Client:**一个循环，不断获取远程服务列表。根据key的hash值来决定和哪一个DataNode进行连接。这样可以缓解单一的DataNode的压力。

**DataNode：**一个内存中的数据库，用Map来存key-value，也要用Map来存Master Node的backup Node，存的是RMI的url。这也是业务逻辑的部分，要实现增删改查等业务逻辑。

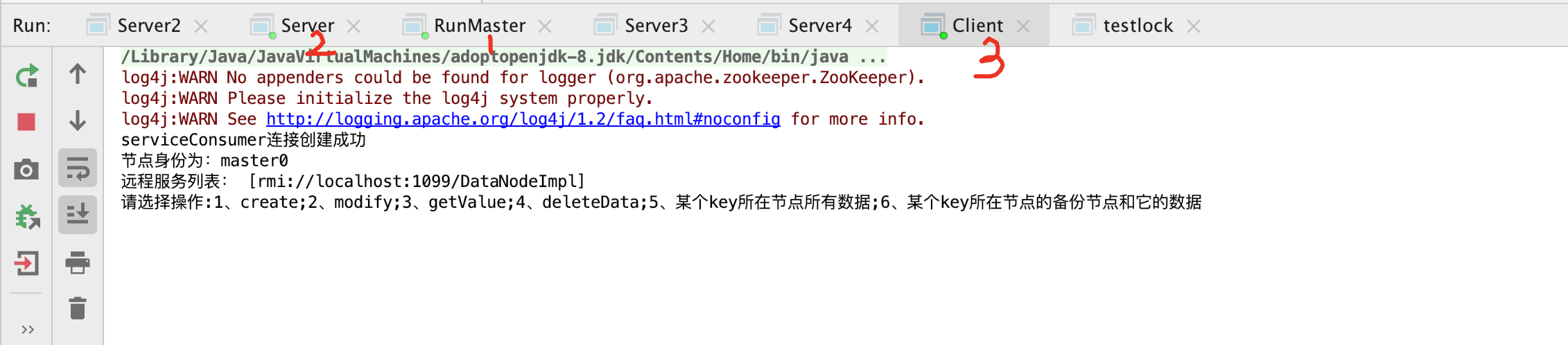
**2、Master：**一个名字系统，用map存节点和节点类型，用list存masterurl、backupurl、Mpath、Bpath。当DataNode类型改变时，要对相应的数据结构进行修正。

**3、Server：**注册服务，当DataNode是Master类型的时候，要观察它的backup的变化；当DataNode是backup的时候，要观察它的master的变化。

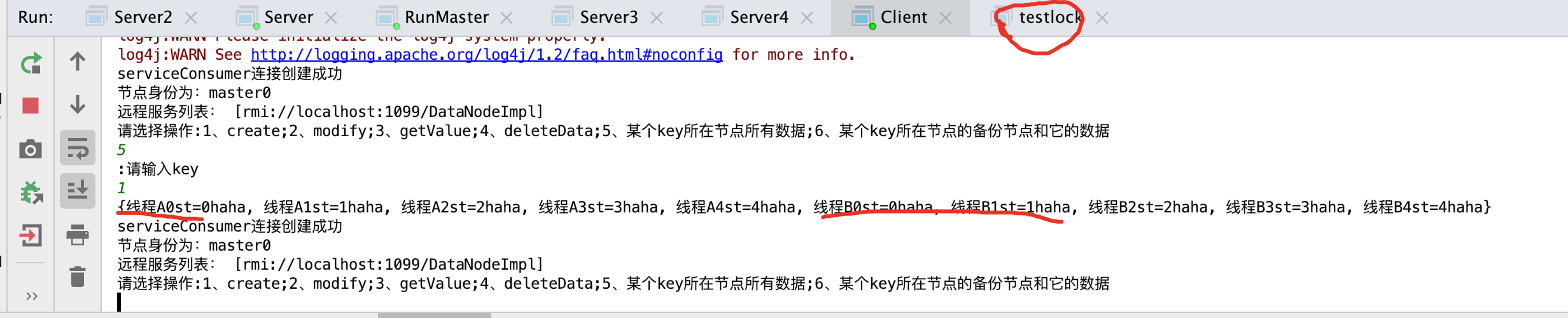
**4、同步：**用synchronized关键字来给create、modify、deleteData的方法加锁；防止多个线程同时修改同一个key-value，以此来支持多个client同时访问数据库。

**三、运行示意图**

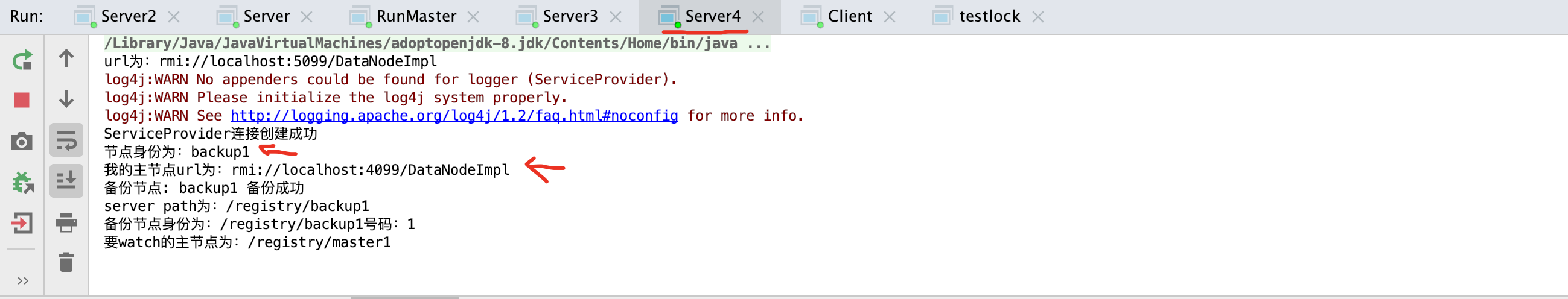
1、依次运行RunMaster、Server、client，建立了基本的数据库环境：

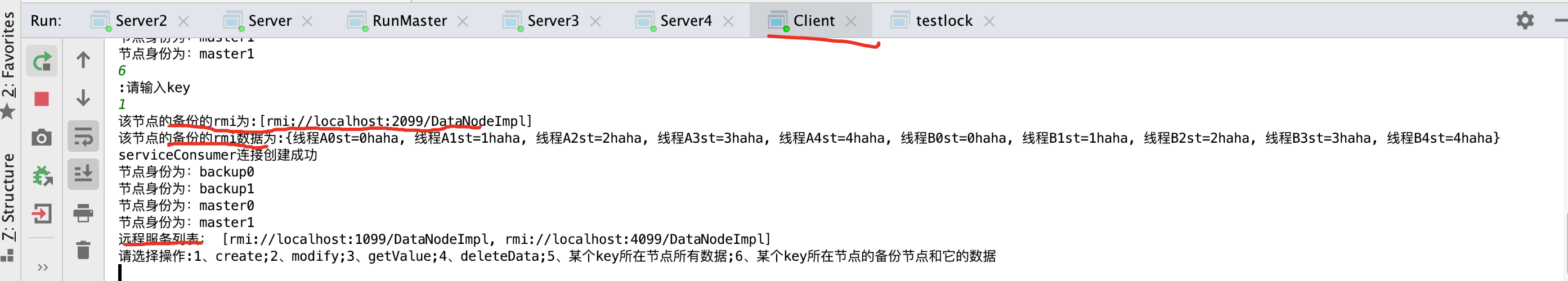


1. 运行testlock，两个线程同时往数据库里加数据，然后client看是否正确。



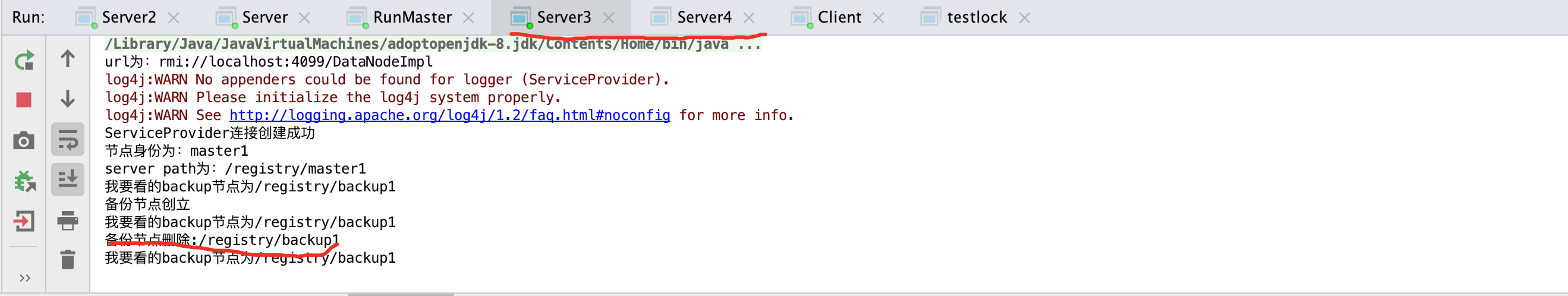
1. 依次运行Server2、Server3、Server4，建立起两个master node,两个backup node，验证master和backup数据是否同步：

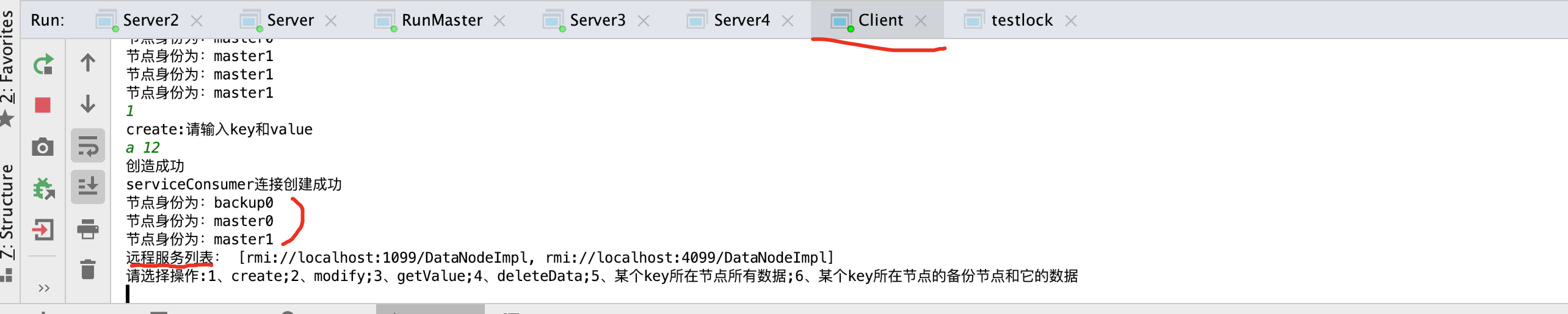




可以看到，backup节点和master节点数据同步，新加入的master节点可以被client观察到。

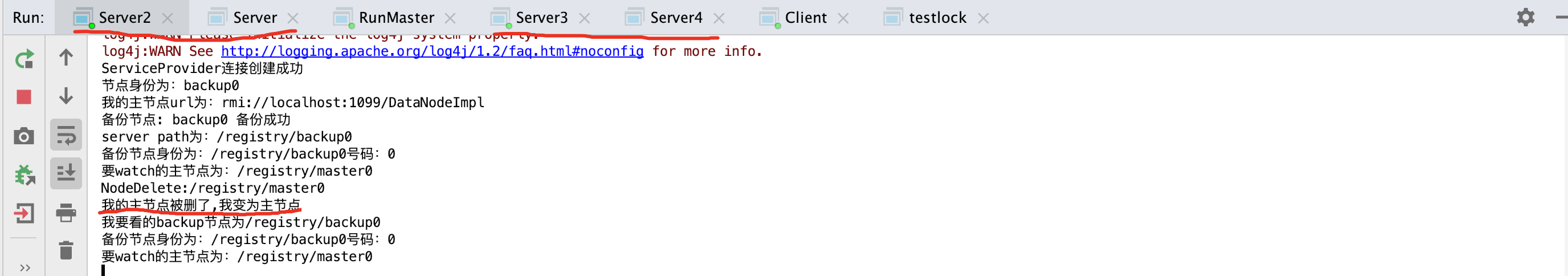
1. 当backup节点挂掉，master功能不受影响。

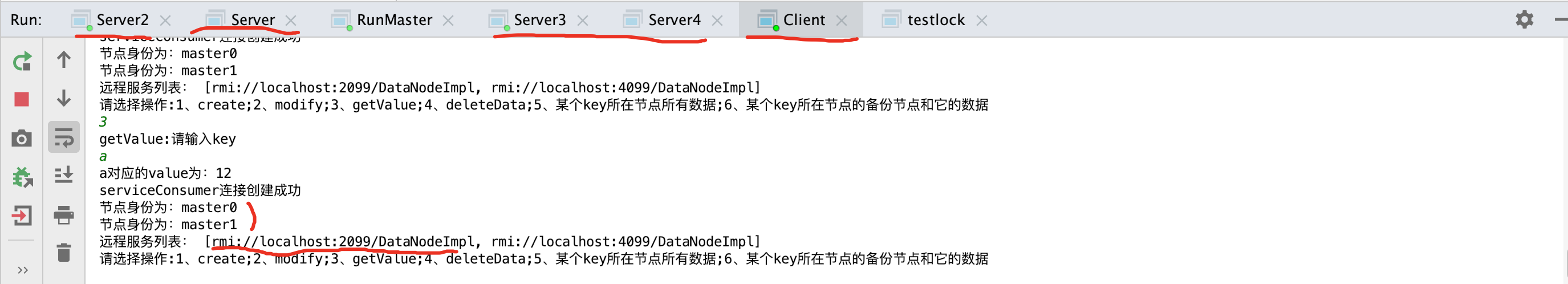




可以看出只是少了一个backup1节点，远程服务列表不受影响。

1. 当master节点挂掉，backup节点会变成master节点，继续给client提供服务。





可以看出原来的端口1099的master节点挂掉之后，端口号为2099的backup节点变为了master节点，而且数据没有丢失。

至此，实验演示完毕。

1. **系统特性**

1、Node：我使用了不同的进程来模拟在不同机器上的节点，有一个master节点，2个master node节点，还有两个 backup node节点。

2、Communication：使用了RMI这一RPC框架。

3、Partion&Scalability：根据key的hash值来决定和哪个DataNode相连，实现了Partion；节点可以动态增加，运行中的client可以发现动态增加的节点，实现了Scalability。

4、operation：PUT、READ、DELETE等操作均支持，还可以查找特定key所在节点的所有值、查找特定key所在节点的backup节点及其所有值。

5、concurrent Data Access：使用了synchronized关键字，可以支持并发的client的请求，包括PUT、READ、DELETE。

6、High Availability：当master DataNode挂掉、backup DataNode挂掉时，都不会影响client的正常访问，实现了High Availability。

**五、总结：**

这是本学期做的最久的一次课程实验了，从0开始布置一个分布式的数据存储系统，需要学习的东西很多，但一步步实现下来也发现了有趣之处，对RMI、zookeeper有了更深的理解，将课堂上所学的东西真正地用于实践，且最后成功地做出了可行的系统，这是一件很有成就感的事！

同时，在实验的过程中遇到了不少的麻烦，和同学、老师的交流让我一步步地攻克难关，总算是完成了课程实验，虽然过程磕磕绊绊，但是最后的收获却十分丰厚！