

Lab4说明文档

[mit6.5840-lab2B](#)

[两阶段提交设计](#)

[运行方法与实验结果](#)

[基础配置](#)

[运行方法与结果](#)

[执行助教提供的手动测试](#)

[细节设计](#)

[副本节点选择](#)

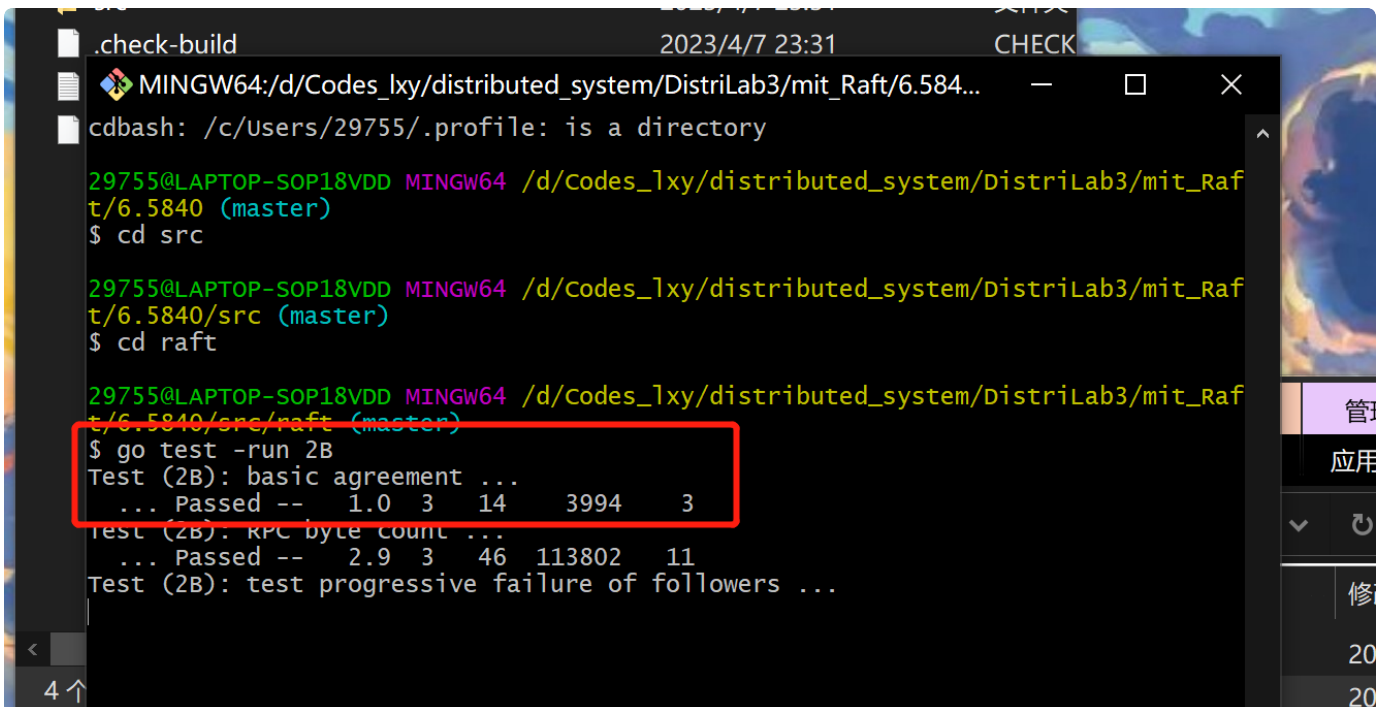
[故障恢复与保持一致性](#)

[实现同时提交只有一个成功](#)

[对lab1代码的修改均在注释中以CHANGE标记出，可以搜索CHANGE来查看修改](#)

mit6.5840-lab2B

运行结果如下，发现通过测试



```
.check-build 2023/4/7 23:31 CHECK
MINGW64:/d/Codes_lxy/distributed_system/DistriLab3/mit_Raft/6.5840...
cdbash: /c/Users/29755/.profile: is a directory

29755@LAPTOP-SOP18VDD MINGW64 /d/Codes_lxy/distributed_system/DistriLab3/mit_Raft/6.5840 (master)
$ cd src

29755@LAPTOP-SOP18VDD MINGW64 /d/Codes_lxy/distributed_system/DistriLab3/mit_Raft/6.5840/src (master)
$ cd raft

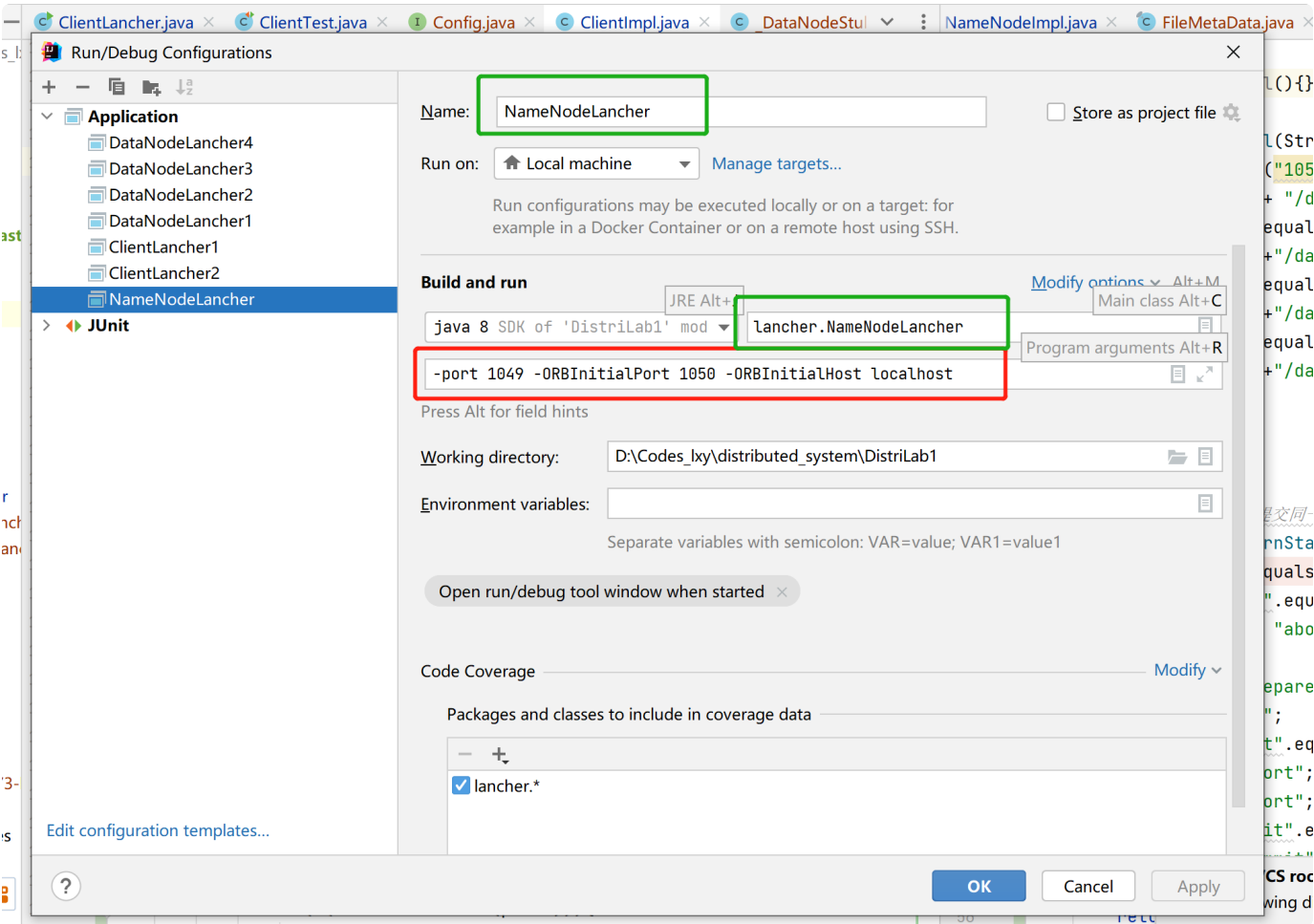
29755@LAPTOP-SOP18VDD MINGW64 /d/Codes_lxy/distributed_system/DistriLab3/mit_Raft/6.5840/src/raft (master)
$ go test -run 2B
Test (2B): basic agreement ...
... Passed -- 1.0 3 14 3994 3
Test (2B): RPC byte count ...
... Passed -- 2.9 3 46 113802 11
Test (2B): test progressive failure of followers ...
```

两阶段提交设计

运行方法与实验结果

基础配置

在idea中运行，需要手动添加7个application，以NameNode为例，应该进行如下配置



需要配置application的Name、运行的主类、传入的参数

下面列出各个application所需要的参数

application的Name	运行的主类	传入的参数
ClientLancher1	lancher.ClientLancher	-port 1049 -ORBInitialPort 1050 -ORBInitialHost localhost

ClientLancher2	lancher.ClientLancher	-port 1049 -ORBInitialPort 1050 -ORBInitialHost localhost
NameNodeLancher	lancher.NameNodeLancher	-port 1049 -ORBInitialPort 1050 -ORBInitialHost localhost
DataNodeLancher1	lancher.DataNodeLancher	-port 1051 -ORBInitialPort 1052 -ORBInitialHost localhost
DataNodeLancher2	lancher.DataNodeLancher	-port 1053 -ORBInitialPort 1054 -ORBInitialHost localhost
DataNodeLancher3	lancher.DataNodeLancher	-port 1055 -ORBInitialPort 1056 -ORBInitialHost localhost
DataNodeLancher4	lancher.DataNodeLancher	-port 1057 -ORBInitialPort 1058 -ORBInitialHost localhost

运行方法与结果

在idea下面打开五个终端，开放java-corba服务，分别对应NameNode与4个DataNode所需要的端口，如下，对应配置阶段的传入参数的端口



```

> orb.db
> out
src
30
31
34
35
36
// get object reference from the servant

Terminal: Local (3) × Local × Local (2) × Local (4) × Local (5) × + ▼
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

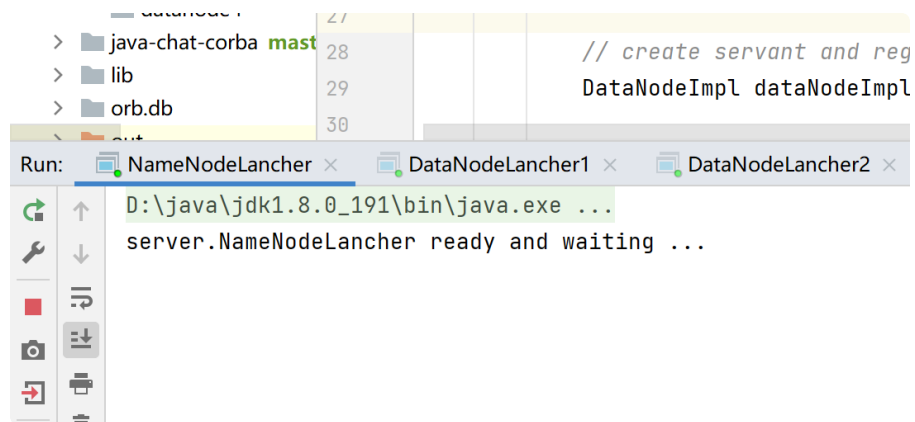
尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS D:\Codes_lxy\distributed_system\DistriLab1> orbd -port 1055 -ORBInitialPort 1056 -ORBInitialHost localhost

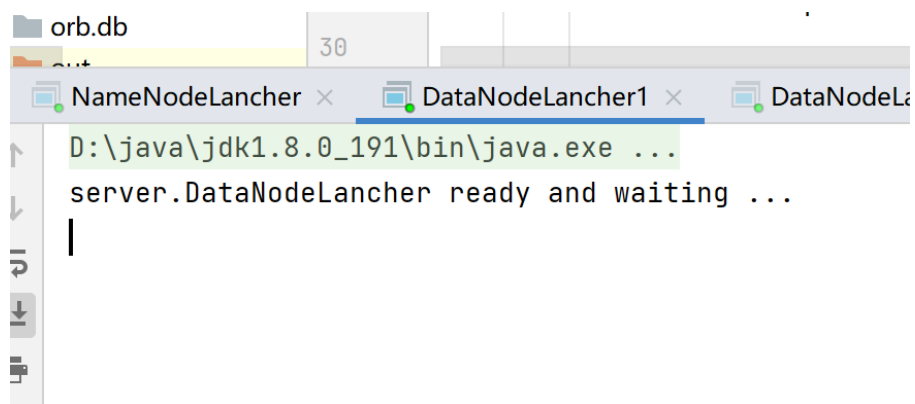
```

然后运行上面的7个application，注意要先运行NameNodeLancher再运行ClientLancher

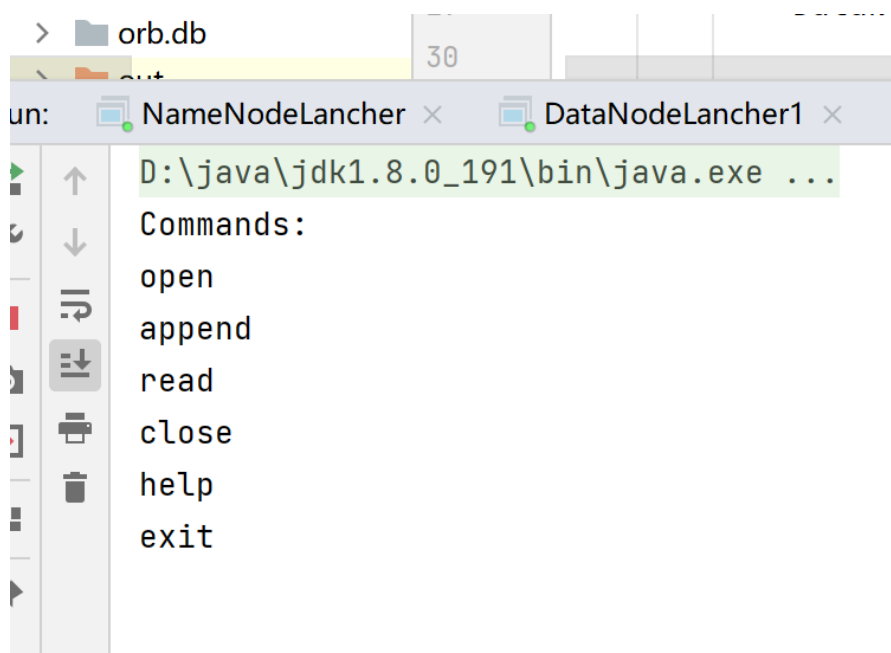
各个运行结果如下



```
27  
28 // create servant and reg  
29 DataNodeImpl dataNodeImpl  
30  
Run: NameNodeLancher x DataNodeLancher1 x DataNodeLancher2 x  
D:\java\jdk1.8.0_191\bin\java.exe ...  
server.NameNodeLancher ready and waiting ...
```



```
30  
Run: NameNodeLancher x DataNodeLancher1 x DataNodeLancher2 x  
D:\java\jdk1.8.0_191\bin\java.exe ...  
server.DataNodeLancher ready and waiting ...  
|
```



```
> orb.db  
30  
Run: NameNodeLancher x DataNodeLancher1 x  
D:\java\jdk1.8.0_191\bin\java.exe ...  
Commands:  
open  
append  
read  
close  
help  
exit
```

执行助教提供的手动测试

为了模仿同时写，在client1端设置sleep（3000）来代表此时client1还在写

```
> datanode3      214      try {
> datanode4      215      connectDataNode(norts):
Run: NameNodeLancher x DataNodeLancher4 x ClientLancher1 x ClientLancher2 x
D:\java\jdk1.8.0_191\bin\java.exe ...
Commands:
open
append
read
close
help
exit
open test.txt w
INFO: test.txt fd: 4096
append 4096 hello world
run append
client append for datanode start...
INFO: write done
close 4096
INFO: fd 4096closed
open test.txt rw
INFO: test.txt fd: 4097
read 4097
hello world
```

杀死DataNode只剩一个副本,

```
> datanode3      215      connectDataNode(norts):
Run: NameNodeLancher x DataNodeLancher4 x ClientLancher1 x ClientLancher2 x
D:\java\jdk1.8.0_191\bin\java.exe ...
server.DataNodeLancher ready and waiting ...
DataNode的数据如下:
Block0
来自于文件: test.txt 的第 0 个block
|
```

虽然中间会报错（由于Javacorba无法连接问题报的错），但最终正确读取数据

```
INFO: test.txt fd: 4097
```

```
read 4097
```

```
hello world
```

```
read 4097
```

```
... 10 more
```

```
hello world
```

由于不能获得所有的副本，写入失败

```
hello world
```

```
append 4097 +++ hello world
```

```
run append
```

```
client append for datanode start...
```

```
五月 23, 2023 11:43:25 上午 com.sun.corba.se.impl.transport.Socket
```

```
警告: "IOP00410201: (COMM_FAILURE) Connection failure: socketType
```


```
COMM_FAILURE: Connection failure: socketType: java.net.SocketException: Connection reset
```

```
... 10 more
```

```
not all datanodes are prepared, exit...
```

```
INFO: write error
```

恢复杀死的DataNode，执行写操作



```
not all datanodes are prepared, exit...
INFO: write error
append 4097 +++ hello world
run append
client append for datanode start...
INFO: write done
```

client1、client2读取文件内容结果如下

client1:

```
not all datanodes are prepared, exit...
INFO: write error
append 4097 +++ hello world
run append
client append for datanode start...
INFO: write done
read 4097
hello world+++ hello world
```

client2:



```
read
close
help
exit
open test.txt rw
INFO: test.txt fd: 4096
read 4096
hello world+++ hello world
|
```

同时执行写操作

client1:

```
read 4097
hello world+++ hello world
append 4097 client1
run append
client append for datanode start...
INFO: write done
read 4097
hello world+++ hello worldclient1
|
```

client2:

```
exit
open test.txt rw|
INFO: test.txt fd: 4096
read 4096
hello world+++ hello world
append 4096 client2
run append
client append for datanode start...
not all datanodes are prepared, exit...
INFO: write error
read 4096
hello world+++ hello worldclient1
```

可见只有一个可以写成功

细节设计

副本节点选择

每次都从block数目最少的DataNode开始，按照顺序循环分配（轮巡）DataNode来分配副本节点，这样能够在一定程度上保证分配block的均衡性。

故障恢复与保持一致性

当读取文件时，会依次循环访问文件元数据中每一个block所在的DataNode（有四个副本），若前面的DataNode故障则读取后面的副本DataNode，直到成功读取，退出循环。从而实现故障恢复。

当写文件时，若写到一半时，有DataNode出现故障，则通知其他DataNode与NameNode撤销本次修改，这个是通过存储原始数据实现的。

实现同时提交只有一个成功

当有两个client发送写请求时，即两个client同时修改一个DataNode，要实现同时只有一个能修改成功是通过在DataNode设置一个变量status来记录当前DataNode是否正在被修改，当DataNode接收到第一个client的prepare或者commit信号时，status会对应设置为prepare或commit，当第二个client发送prepare请求时，由于当前status是prepare或者commit，则返回abort，通知第二个client当前不能修改，修改失败。只有当第一个 client写请求结束或者DataNode接收到abort信号时，status设置为abort，表明当前DataNode为空闲状态，其他的client可以修改。

对lab1代码的修改均在注释中以CHANGE标记出，可以搜索CHANGE来查看修改