

ELEC 477

Kevin Yu: 20203451

Raatik Sharma: 20120770

Ainsley Taylor: 20210012

2024-03-17

Test

For this assignment, you should only check that the put messages are forwarded to the replica servers and the values are changed. You can do this with trace methods.

To test the replicas servers and ensure that the values are changed, I added trace methods (console logs) to the terminal to show what values were received and being stored in the kvPut() method for the replica servers. Afterwards, I would call kvGet() to illustrate that the value was stored in the GDBM file by the replica. In Figure 1, we can see that I have a replica server called “kvserver2” which is the replica server for “kvserver1”. Once the kvput method is called, the primary server forwards the kvput call to its replica server and we have trace methods to showcase what values are received and stored on the replica server as seen in Figure 2. Lastly, in the trace logs in Figure 3, we see the put request from the client to the primary server and then the kvPut method is forwarded to the replica server where the key and value pairs are stored and retrieved respectively.

```
main.cpp 2 x
A2SolutionNetworkKV2NewSemantics > G main.cpp > main(int, char * [])
22  int main(int argc, char *argv[])
23  {
24      std::stringstream ss;
25
26      // start all of the servers first. This will let them get up
27      // and running before the client attempts to communicate
28      std::cout << "Main: *****" << std::endl;
29      std::cout << "Main: starting server" << std::endl;
30
31      std::cout << "Main: starting directory server" << std::endl;
32
33      shared_ptr<SvcDirServer> svcDirServer = make_shared<SvcDirServer>("ServiceDirectory");
34      svcDirServer->setAddress("10.0.0.2");
35      svcDirServer->init();
36      svcDirServer->startServices();
37
38      ServerData::ServerInfo replica1 = {"kvserver2", 5000};
39      ServerData::ServerInfo replica2 = {"kvserver3", 5001};
40      vector<ServerData::ServerInfo> *replicas = new vector<ServerData::ServerInfo>;
41      ;
42      replicas->push_back(replica1);
43      replicas->push_back(replica2);
44
45      ServerData::ServerInfo *primaryServer = new ServerData::ServerInfo{"kvserver1", 5193};
46
47      // Primary Server
48      shared_ptr<KVServer> kvServer1 = make_shared<KVServer>("kvserver1", replicas, primaryServer, 1);
49
50      kvServer1->setAddress("10.0.0.3");
51      kvServer1->setDBMFileName("server1");
52      kvServer1->setSvcDirServer("ServiceDirectory");
53      kvServer1->setSvcName("kv1");
54      kvServer1->setPort(5193);
55      kvServer1->init();
56
57      // Replica Server 1
58      shared_ptr<KVServer> kvServer2 = make_shared<KVServer>("kvserver2", nullptr, primaryServer, 0);
59      kvServer2->setAddress("10.0.0.4");
60      kvServer2->setDBMFileName("server2");
61      kvServer2->setSvcDirServer("ServiceDirectory");
62      kvServer2->setSvcName("replica");
63      kvServer2->setPort(5000);
64      kvServer2->init();
65
66  }
```

Figure 1: Main.cpp file

```

main.cpp  kvservice.cpp 2 X
A2SolutionNetworkKV2NewSemantics > kvservice.cpp > callMethodVersion1(E477KV::kvRequest &, E477KV::kvResponse &)
19 void KVServiceServer::start()
228 }
229
230 void KVServiceServer::callMethodVersion1(E477KV::kvRequest &receivedMsg, E477KV::kvResponse &replyMsg)
231 {
232     if (receivedMsg.has_putargs())
233     {
234         stringstream ss;
235         ss << "KVSERVICE: put message requested" << endl;
236         cerr << ss.str();
237
238         const E477KV::putRequest &preq = receivedMsg.putargs();
239
240         int key = preq.key();
241         string valueAsStr = preq.value();
242
243         bool putRes = kvPut(key, (uint8_t *)valueAsStr.c_str(), valueAsStr.length());
244
245         E477KV::putResponse *presp = replyMsg.mutable_putres();
246         presp->set_status(putRes);
247
248         cout << "\nKVSERVICE: Server Name: " << name << endl;
249         cout << "KVSERVICE: This is the key: " << (int32_t)key << endl;
250         cout << "KVSERVICE: This is the value: " << valueAsStr << endl;
251         cout << "KVSERVICE: This is the value size: " << (uint16_t)valueAsStr.size() << endl;
252         cout << "KVSERVICE: put response is: " << putRes << endl;
253         cout << "KVSERVICE: Is it a Primary server: " << isPrimary << endl;
254
255         // Assuming this is the primary server and the put response succeeded
256         if (isPrimary)
257         {
258
259             bool replicasUpdated = true;
260             for (auto it = replicas->begin(); it != replicas->end(); ++it)
261             {
262                 cout << "KVSERVICE: replica name: " << it->name << endl;
263                 cout << "KVSERVICE: replica port: " << it->portNumber << endl;
264
265                 // Updating all of the replicas
266                 kvPutReplica((int32_t)key, (const uint8_t *)valueAsStr.data(), (uint16_t)valueAsStr.size(), it->name, it->portNumber);
267             }
268         }
269         else
270         {
271             kvGetResult result = kvGet(key);
272             cout << "KVSERVICE REPLICATION GET: Key used --> " << key << endl;
273             cout << "KVSERVICE REPLICATION GET: Status --> " << result.status << endl;
274             cout << "KVSERVICE REPLICATION GET: Value --> " << result.value << endl;
275             cout << "KVSERVICE REPLICATION GET: Value Length --> " << result.vlen << endl;
276
277             cout << "KVSERVICE: END OF REPLICATION\n"
278                 << endl;
279         }
280     }
281     if (receivedMsg.has_getargs())

```

Figure 2: kvservice.cpp

```
client waiting
Client Stub kvclient answer is server kvserver1, port 5193
CLIENT STUB: address of kvserver is: 10.0.0.3

KVCLIENTSTUB: Sending to 10.0.0.3
KVSERVICE: kvserver1.KV_RPC kv server received 33 bytes.
KVSERVICE: put message requested

KVSERVICE: Server Name: kvserver1.KV_RPC
KVSERVICE: This is the key: 25
KVSERVICE: This is the value: This is a test!!
KVSERVICE: This is the value size: 16
KVSERVICE: put response is: 1
KVSERVICE: Is it a Primary server: 1
KVSERVICE: replica name: kvserver2
KVSERVICE: replica port: 5000
KVSERVICE: address of kvserver is: 10.0.0.4

KVSERVICE: Sending to 10.0.0.4
KVSERVICE: Received message from 10.0.0.3:5195
KVSERVICE: The primary server name is: kvserver1
KVSERVICE: The primary server address is: 10.0.0.3
KVSERVICE: The source ip address is coming from: 10.0.0.3
KVSERVICE: kvserver2.KV_RPC kv server received 33 bytes.
KVSERVICE REPLICA: SOURCE ADDRESS IS FROM PRIMARY SERVER!
KVSERVICE: put message requested

KVSERVICE: Server Name: kvserver2.KV_RPC
KVSERVICE: This is the key: 25
KVSERVICE: This is the value: This is a test!!
KVSERVICE: This is the value size: 16
KVSERVICE: put response is: 1
KVSERVICE: Is it a Primary server: 0
KVSERVICE REPLICA GET: Key used --> 25
KVSERVICE REPLICA GET: Status --> 1
KVSERVICE REPLICA GET: Value --> This is a test!!
KVSERVICE REPLICA GET: Value Length --> 16
KVSERVICE: END OF REPLICA
```

Figure 3: Trace Values