Report 5: Homework Report chordy

Marcus Samuelsson

October 10, 2023

1 Introduction

This report covers the implementation of a distributed hash table(**DHT**) based on the Chord scheme. As the name implies a **DHT** is a hash table that is distributed between different nodes. A hash table is a structure that can map keys to values. Chord is one type of distributed hash table, in which the nodes are arranged in a ring structure. Each node has a unique Id and knows the previous(**Predecessor**) and next(**Successor**) node in the ring. Chord handles how keys are assigned and how values are retrieved.

2 Main problems and solutions

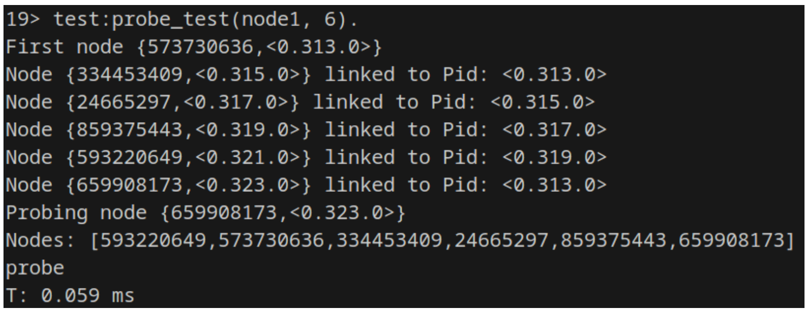
The first and biggest issue was with the **between** function in the **key** module. This function was able to handle every case of a node not having the successor set to itself, but not that specific case. This caused multiple functions to not work and the probe used to check if the ring connected only return the first node when it was called, or it would get stuck in an endless loop. This was an easy fix by just adding the scenario of the **To** and **From** being equal in the **between** function, which made it all work as it should.

Another issue which was also a problem in the previous task was when the function in node1 where copied over and changes where missed. The issue around this was that the **notify** function now returned an object {Predecessor, Store} but in node one the **Store** was not part of that. The fix was just making the call in the **node** function retrieve the correct variables.

The last issue was that the **add** and **lookup** function added to **node2** was first added to the **storage** module. The reason for this is not exactly clear, however the fix was to move it into **node2**.

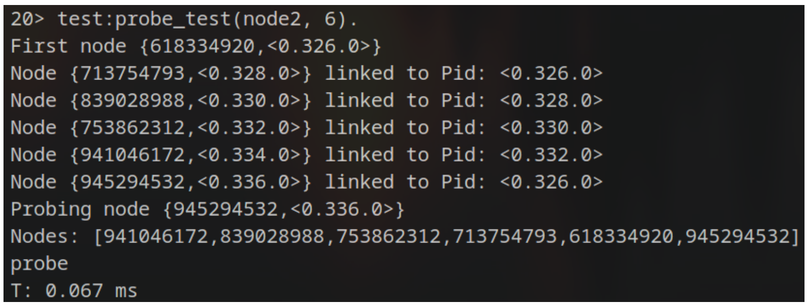
3 Evaluation

Two tests where created to evaluate this task. The first test test to see if the nodes in the hash tables are connected in a ring as specified. This is done through creating a specified number of nodes, which is connected in a circle with the node created before set as the peer of the new node, until the last where it connects to the first again. The program then sleeps to let the automatic stabilization to run. Then the probe is called on the last created node, which will go through the ring and print out the time and the all the. The result for node1 and node2 are bellow(See Figure 1 and Figure 2).



```
19> test:probe_test(node1, 6).
First node {573730636,<0.313.0>}
Node {334453409,<0.315.0>} linked to Pid: <0.313.0>
Node {24665297,<0.317.0>} linked to Pid: <0.315.0>
Node {859375443,<0.319.0>} linked to Pid: <0.317.0>
Node {593220649,<0.321.0>} linked to Pid: <0.319.0>
Node {659908173,<0.323.0>} linked to Pid: <0.313.0>
Probing node {659908173,<0.323.0>}
Nodes: [593220649,573730636,334453409,24665297,859375443,659908173]
probe
T: 0.059 ms
```

Figure 1: Probe test on module **node1** with 6 nodes



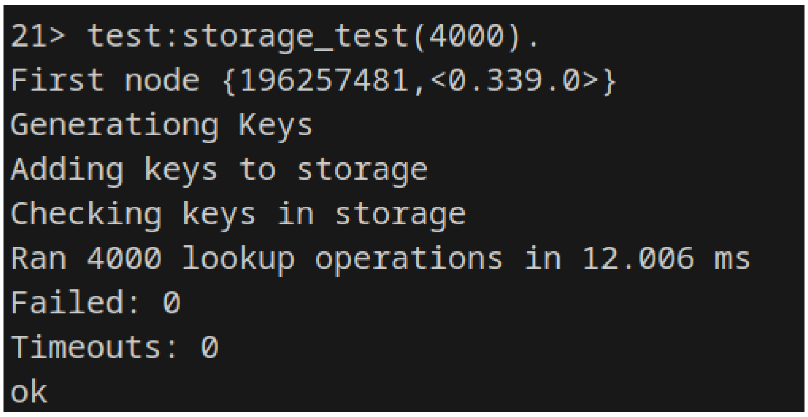
```
20> test:probe_test(node2, 6).
First node {618334920,<0.326.0>}
Node {713754793,<0.328.0>} linked to Pid: <0.326.0>
Node {839028988,<0.330.0>} linked to Pid: <0.328.0>
Node {753862312,<0.332.0>} linked to Pid: <0.330.0>
Node {941046172,<0.334.0>} linked to Pid: <0.332.0>
Node {945294532,<0.336.0>} linked to Pid: <0.326.0>
Probing node {945294532,<0.336.0>}
Nodes: [941046172,839028988,753862312,713754793,618334920,945294532]
probe
T: 0.067 ms
```

Figure 2: Probe test on module **node2** with 6 nodes

The second test is a test on the storage and looking up in the memory of a node. This test creates a node and assigns a specified number of keys to the nodes storage. This is done through generating a list of keys, then calling the add function on the node to add each key to the node and lastly every key is looked up on the node. This is done through a check which calls each key and counts how many timeouts and error that are returned as a result. The result can be seen bellow(See Figure 3).

4 Conclusions

This task was a bit difficult to grasp due to having to understand chord before being able to implement it. Eventually it became clearer and with the guidelines given in the instruction, the implementation eventually came together. Another difficulty after being able to implement the program was to set up tests to see how everything worked. Lastly, this task gave a good understanding of chord and how it is used in distributed systems



```
21> test:storage_test(4000).
First node {196257481,<0.339.0>}
Generationg Keys
Adding keys to storage
Checking keys in storage
Ran 4000 lookup operations in 12.006 ms
Failed: 0
Timeouts: 0
ok
```

Figure 3: Test on storage