Distributed Operating System Principles (COP5615)

Project-3

Group Members

JAYACHANDRA GUNTURU **SATYA NAGA AKHILESH IRRINKI**

What is working?

- Implemented Chord protocol - built a ring of nodes, applied SHA-1 hashing algorithm to generate unique m-bit identifier on each node and key and upon key search, we are able to find the location of the key in O(log N) time where N represents the number of nodes.
- Implemented node joins - whenever a new node is joined to the existing chord of nodes, we stabilize the key search by updating the finger tables of all the nodes.
- Implemented hop count tracking – Each actor will keep a track of the number of hops done to find all the keys and then send the sum of all the hops to the master node. Once all the keys are found we print the average number of hops that each node has taken to find the key and then the program terminates.
- For bonus, we implemented a failure model – When a random node fails the chord ring is able to stabilize and successfully run lookup for keys.

What is the largest network you managed to deal with?

- The largest network that we were able to deal with is when:

  Number of Nodes = 30000

  Number of Requests = 100

```
PS D:\Sem-1\DOSP\Projects\Project-3> dotnet fsi project3.fsx 30000 100
Starting Building Chord Network
Done Building Network
Processing requests.......
All requests processed.......
Average Hop count is 45.450116
```

Implementation:

- Using F# and Actor model by AKKA actor library, we have implemented Chord Protocol to efficiently find a key.
- Given the number of nodes N, we generate unique m-bit identifiers using the SHA-1 hash algorithm.
- We place all the nodes in the chord, based on the unique identifiers of each node.
- We populate the finger tables of each node by adding neighbors of the node to the finger table in a way such that each entry into the table succeeds the first node by 2^{i-1} as discussed in the research paper.
- Whenever a new node joins the chord, we generate a unique identifier for it using the SHA-1 algorithm and place it inside the chord according to the value of its identifier.
- Once the node is placed into the ring, we update the predecessors and successors of the new node and its immediate neighbors.
- Then we update the finger table of the new node and all the other nodes in the chord using the same technique of Successor ($n + 2^{i-1}$).
- Finally, the nodes start looking for the keys with the given number of requests.
- First, we pass the key and number of hops which is initialized to zero.
- Then the node checks for the key with its own identifier, if it is equivalent, it returns the number of hops else its checks for the nearest neighbor in its finger table that might have the key and passes the key and increments the hop count by 1.
- The same procedure is iterated for all the nodes till all the keys are found, parallelly the boss actor checks whether all the requests are completed or not alongside keeping track of the hop count of each node.
- Once all the keys are found the boss actor prints the average number of hop count required by a node to find and key and gets terminated.

Execution:

- The .Zip file consists of F# script file named project3.fsx
- Command for executing the file:

dotnet fsi project3.fsx NumberOfNodes NumberOfRequests

Results:

Number of Nodes	Number of Requests	Average number of Hops
100	100	13.260500
1000	100	24.826430
2000	100	28.712645
5000	100	33.017076
10000	100	37.418259
15000	100	40.673629
20000	100	41.923850

Number of Nodes = 100, Number of requests = 100

```
PS D:\Sem-1\DOSP\Projects\Project-3> dotnet fsi project3.fsx 100 100
Starting Building Chord Network
Done Building Network
Processing requests.......
All requests processed.......
Average Hop count is 13.260500
PS D:\Sem-1\DOSP\Projects\Project-3>
```

Number of Nodes = 1000, Number of requests = 100

```
PS D:\Sem-1\DOSP\Projects\Project-3> dotnet fsi project3.fsx 1000 100
Starting Building Chord Network
Done Building Network
Processing requests.......
All requests processed.......
Average Hop count is 24.826430
PS D:\Sem-1\DOSP\Projects\Project-3>
```

Number of Nodes = 2000, Number of requests = 100

```
PS D:\Sem-1\DOSP\Projects\Project-3> dotnet fsi project3.fsx 2000 100
Starting Building Chord Network
Done Building Network
Processing requests.......
All requests processed.......
Average Hop count is 28.712645
PS D:\Sem-1\DOSP\Projects\Project-3>
```

Number of Nodes = 5000, Number of requests = 100

```
PS D:\Sem-1\DOSP\Projects\Project-3> dotnet fsi project3.fsx 5000 100
Starting Building Chord Network
Done Building Network
Processing requests.......
All requests processed.......
Average Hop count is 33.017076
PS D:\Sem-1\DOSP\Projects\Project-3>
```

Number of nodes = 10000, number of requests = 100

```
PS D:\Sem-1\DOSP\Projects\Project-3> dotnet fsi project3.fsx 10000 100
Starting Building Chord Network
Done Building Network
Processing requests.......
All requests processed.......
Average Hop count is 37.418259
PS D:\Sem-1\DOSP\Projects\Project-3>
```

Number of nodes = 15000, Number of requests = 100

```
PS D:\Sem-1\DOSP\Projects\Project-3> dotnet fsi project3.fsx 15000 100
Starting Building Chord Network
Done Building Network
Processing requests.......
All requests processed.......
Average Hop count is 40.673629
PS D:\Sem-1\DOSP\Projects\Project-3>
```

Number of nodes = 20000, Number of requests = 100

```
PS D:\Sem-1\DOSP\Projects\Project-3> dotnet fsi project3.fsx 20000 100
Starting Building Chord Network
Done Building Network
Processing requests.......
All requests processed.......
Average Hop count is 41.923850
PS D:\Sem-1\DOSP\Projects\Project-3>
```