Distributed Operating System Principles (COP5615): Project - 3

Submitted By:
Santosh Maturi, UFID: 4533-9141
Shashank Kumar Hayyal, UFID: 5142-0992

Problem Statement:

This project aims to demonstrate the utility of F# by implementing it using the actor model, the Chord protocol, and a primary object access service.

Given a key, the key/data pair is stored at the node to which the keymaps. Chord adapts efficiently as nodes join and leave the system: the communication cost and state scale logarithmically with the number of Chord nodes.

Overview:

The chord is a distributed key-value network that associates keys with their associated nodes. Almost certainly, the hash function balances the load. When an Nth node joins (or departs) the network, only O(1/N) of the keys are moved.

Each node in an N-node network stores information about O(log N) other nodes. Chord enhances the scalability of consistent hashing by removing the requirement that each node is aware of all other nodes. Because this information is spread, a Chord node requires only a limited amount of "route" knowledge about other nodes.

System Prerequisites:

- Installed the.NET SDK
- Multicore System
- Installed the F# language server for.NET
- If you're using Visual Studio Code, use the Ionide extension for F Sharp.

How to Run?
The zip file contains the following:

- **ReadMe file**
- **p2p.fsx**

In order to execute in terminal or cmd use the following command
dotnet fsi p2p.fsx <no-of-nodes> <no-of-requests>

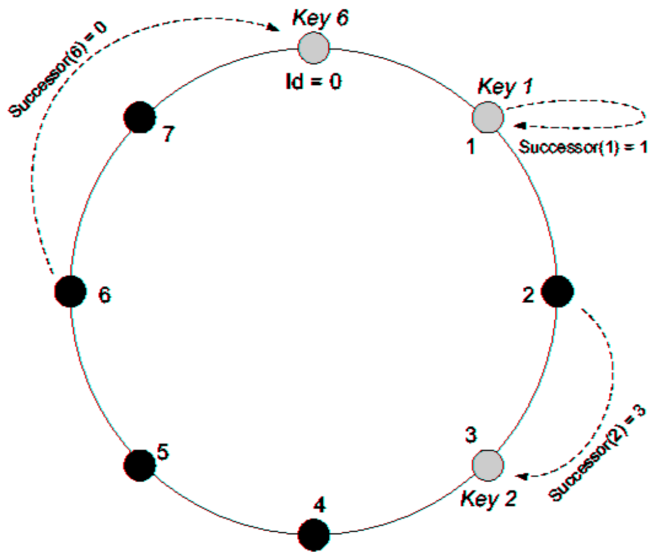
Example:
dotnet fsi p2p.fsx 1000 18

What is Working?

Chord Protocol:
The Chord protocol uses Consistent hashing to assign an identification to each node's IP address and file key. It is a ring of digits ranging from 0 to 2.^(m-1).

Assigning keys to a node is done as follows:

1. When a node is assigned the key K, it is the first node whose identifier equals K in the identifier space.
2. A K's successor node is the first node discovered clockwise from K in the chord ring.

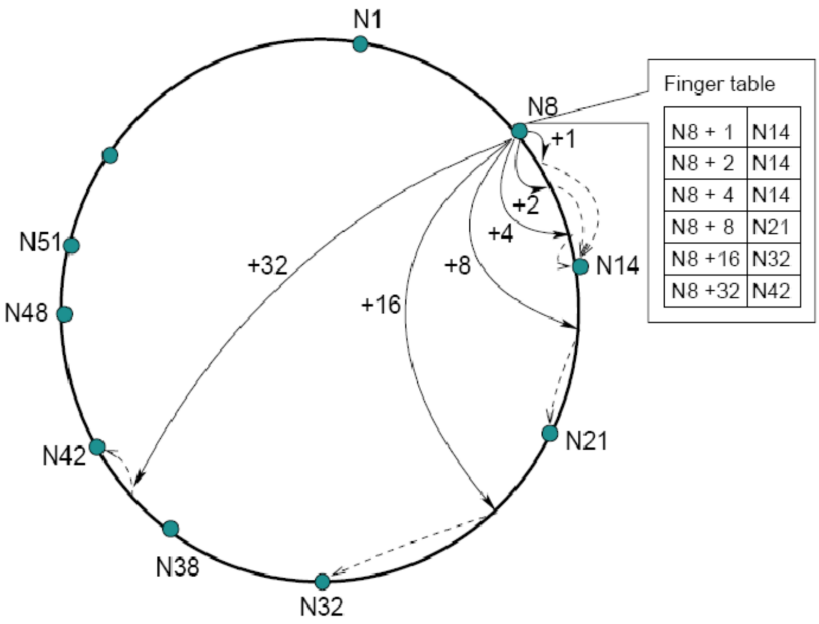


3. For example Successor of 2 would be 3.
4. When a peer joins the ring, some keys previously allocated to the predecessor are now assigned to the new peer.
5. When a peer leaves, the keys assigned to the departing peer are transferred to the departing peer's successor.

Finger Table:
In chord, a finger table is a routing table that is used to speed up the lookup process. Each node maintains a finger table with m entries**.**

The lookup process is done as follows:

1. If the key falls between n and successor(n), the successor(n) is returned.
   For example, if we **lookup(41)** finger tables show that the successor node is 42(between 40 and 42).



2. Alternatively, another lookup is performed in the node that is closest to n.
   For example, if we **lookup(39)** and it is less than 40, we assign node 32 to manage.
   Therefore, the route follows 8 -> 32 -> 38 -> 42.

This **lookup algorithm** progresses the query at least halfway, owing to the finger table's nodes being separated by powers of two. Finally, within **O(log n)** hops, a resource or file can be located.

All of the APIs defined in the paper, such as join, find successor, and stabilize, as well as optimizations such as finger table, are functional. To simplify complexity, the join and stabilize APIs have been combined into a single join. The join operation has to be performed sequentially to minimize the need for periodic stabilization.

What is the largest network you managed to deal with?

We are able to run the code with 10000 nodes.