Report 5: Homework Chordy

Nagasudeep Vemula

October 9, 2019

1 Introduction

In this assignment we implement a distributed hash table by using the chord algorithm.Fundamentally this involves constructing a ring network with nodes having a list of storage that can store the key/value pairs.We implement this in erlang with the core of the chord protocol but in a simplified manner.This assignment will show how nodes are defined and structured in the ring and also introduce a storage module to demonstrate how keys are shared between the nodes.

2 Main problems and solutions

The first thing to decide is the flow of initialization.In the starting, when we create a new node we need to set its successor as itself and its predecessor as nil.The stabilize function then runs automatically and sends succesor a request message to ask for its predecessor.If we are not returned as the result then we need to notify the successor of the same.So on running stabilize the first node will be its successor and on adding any new nodes,it will take the first node as its successor and then stabilize happens and the process repeats.

For solving the problem of storage when a new node joins we use the split functionality to split the storage and then we check each of the items,if they are found to be in the range of new predecessor to the node itself,these key value pairs then belong to the node itself and the other pairs are sent to the new predecessor.

Another problem is to search for the key.We lookup for the key in the nodes of the ring.For example if we search the first node A and the key doesnt belong in the range from the first node's predecessor's key to its own key then it will pass the searching to A's successor and repeat the process.

3 Implementations

I implemented 2 modules primarily the first one handling the construction of the ring and stabilizing it and the next module dealing with introduction of a storage system.

3.1 First Implementation

The first implementation namely node1 is for handling construction of the ring while new nodes are being added to the network.A key module is also implemented with the method between/3 i.e between(key,from,to) which tells us if a key is in the range (From,To],notice that its not inclusive of the predecessor.Another method is introduced for generating random keys in the range 1-10000000. The between functionality is as follows:-

```
    between(Key, From, To)->
  case From==To of
    true->
      true;
    false->
      if From>To ->
        ((From<Key)and(Key=<1000000000))or((0<Key)and(Key=<To));
        To>From ->
          (From<Key)and(Key<To)

      end
  end.
```

The next important aspect to handle is the continuous stabilizing of the ring that is being constructed.This is essentially to ensure the correct order is maintained in the DHT despite new additions to the network.This is implemented using the stabilize/3 method.In the initial 3 cases we just notify the node and return back the original successor.In the final case we check if the key is between current node and successor or not.In case its true,we update our successor and send request for node to take us as his predecessor.Else,the original successor is returned and notify occurs.

```
    stabilize(Pred, Id, Successor) ->
  {Skey, Spid} = Successor,
  case Pred of
     nil ->
      Spid ! {notify, {Id, self()}},
      Successor;
      {Id, _} ->
      Successor;
      {Skey, _} ->
      Spid ! {notify, {Id, self()}},
      Successor;

      {Xkey, Xpid} ->
      case key:between(Xkey, Id, Skey) of
        true ->
          Xpid ! {request, self()},
          {Xkey,Xpid};

        false ->
          Spid ! {notify, {Id,self()}},
          Successor
      end
  end.
```

The next part is to implement the notify method as given in the assignment description,following the remaining guidelines,node1 is completely implemented. Testing is done for the same,using a small ring made with 3 nodes:-

```
     N1=test:start(node1).
<0.84.0>
3> N2=test:start(node1,N1).
<0.86.0>
4> N3=test:start(node1,N1).
<0.88.0>
5> N1 ! probe.
443584618 probe
723040206 6> 945816365 6>
 Time = 16>
N1 ! state.
Id:443584618
 Predecessor:{945816365,<0.88.0>}
 Successor:{723040206,<0.86.0>}
state
```

3.2 Second Implementation

The second implementation,node2 involves the addition of store which will simulate key storage for the system and this module store involves the procedures merge and split the stores when adding nodes to the ring.Key,Value pairs are stored in lists. The implementations of merge and split are as follows:

```
  merge(Entries,Store)->
lists:merge(Entries,Store).

split(Key,Store)->
lists:partition(fun(A)->{B,_}=A , B>Key end, Store).
```

We use the merge and partition functions from the lists library.Partition lets us split the list into 2 parts,one which satisfies the given function and the other that doesnt.This condition will split the list until a certain key.The next step is the implementation of the add and lookup procedures taking the guidelines in the documentation.

In order to modify the notify procedure the key store has to be split upon notification of a new predecessor.Here the purpose of the split function is to check which keys in the store are less or equal to the splitting key.The keys meeting this condition are the ones that will be used in handover and the rest will be kept.The modification to be made is thus to return the store that the node will need to keep and send the other list to the predecessor.

```
  handover(Store, Nkey, Npid) ->
{Keep, Leave} = store:split(Nkey, Store),
Npid ! {handover, Leave},
Keep.
```

4 Conclusions

This assignment gave me an introduction to the chord protocol which is a real world solution to a distributed system problem.Using this protocol,searching becomes much easier as we dont have to parse the entire list of keys and we can just check the keys of nodes to know if its in that node.Instead of an O(n) operation it gets reduced to O(logn) which is described in the paper provided. The problems faced by such systems include load balancing,decentralization,scalability,availability and other issues that arise in designing of peer to peer networks.