Distributed Systems CS-675

Project 2 - Report

By:

Komatireddy Kirankumar Reddy

Instructions:

This implementation of the project can be setup using the following commands.

javac RegistryCreator.java

java RegistryCreator

rmiRegistry 8989 &

javac Server.java

java Server <Ip address of the system on which the server is working> <Ip address of the node that is already known to exist in the chord network>

The second argument of the system can be left blank if the server that you are setting up is the first in the network.

Assumptions made during the designing of the system.

- At a time there can be only 8 systems online. This is dictated the number of bits that are used to identify each node and key, which has been set to 3 in this implementation as Values.m. Values being the name of the file and m being the variable.
- The Identity of the at least one of the nodes in the network is known to the node that is about to join the network.

System Design:

The design of the system has the clients invoking the methods in the servers through a registry which stores the calendar objects and as well as the Chord objects respective to the calendars. Since the constraint is on the replicas to not exist on the same server, a replica of each of the calendars can exist on all of the servers.

Once a new node tries to join the network it contacts the node that it knows of and asks it to find out its predecessor and successor and then uses the fetched table to calibrate its fingers on the network.

Following that is the method Update_Other_Nodes that tries to the find the nodes that can precede the new node by 2^{i-1} and notifies them to change their finger values. The design enforces the propagation of this node's presence to be reflected (by transferring the replicas assigned to the keys to the new node) on to each of the chord networks established for each of the calendars separately. In this way a node join is handled.

Once a node has joined, **High availability** is offered by fault detection using a heartbeat message broadcasted by the primary backup on to the other replicas, and when a certain replica hasn't heard the heart beat message from the primary backup it times out and sends a message to the primary backup to

check if it is alive. The only way to make this fault detection method work on a UDP based network (Which can incur packet loss) and not make both or either the primary backup or the replica initiate the methods for fault tolerance just because of a loss of a packet is to adjust the time out values, keeping the timeout value for the primary back at least 3 times the time needed to transfer a message transit time.

In case the primary backup fails then the replica pertaining to the chord that first detected the fault will initiate the fault handling method and assume that the former primary backup storing node's successor is the primary backup.

When the respective fingers will be updated over the chord network pointing to the successor of the former primary node the rest of the nodes will come to know that the primary node has failed.

Replication Transparency is achieved by maintaining sequential consistency by implementing distributed commit in the form of 2 phase commit, this enables the user to access the same content from any of the replicas.

Partial Failures

Case 1: If a node fails right after the node that is holding the primary backup fails and before the successor of that node is elected as the new primary node.

Limitations: The address resolution of the new assigned primary backup node to reestablish the heart beat detection would be difficult and time consuming because in a very large CHORD network not every node knows where the desired node is. All it can do is point in the right direction and reduce the search time by logn.

Instructions:

Instructions to run the client server module to schedule a group event on a single server are as follows:

1. Javac Registry_Creator.java
2. Java Registry_Creator
3. rmi registry 8989 &
4. javac rmi_Server.java
5. java rmi_Server
6. javac Ui.java
7. java Ui <Ip address of the server>