Project Four Design Report

1. Protocol between Server and UserNodes

There are four types of transmit message in this project:

1. Server sends queries to UserNodes, denoted by opcode 1
2. UserNodes send back votes to Server, denoted by opcode 2
3. Server send vote decision to UserNodes, denoted by opcode 3
4. UserNodes send back ACK to Server, denoted by opcode 4

In UserNodes, there is a function “deliverMessage” to receive message from server, and invoke a corresponding method to respond to the message according to the opcode.

In Sever, a class “ServerMessageHandler” override method “deliverMessage” to receive message from UserNodes, and put the content to corresponding List according to the opcode.

2. Timeout thresholds

For Phase One: After server sends queries to nodes, if the server doesn’t get a vote message from a certain node within 3 seconds, the server will assume that the message was lost and this turn of vote is aborted. Then, the server will send the decision “false” to all nodes and abort this operation.

For Phase Two: After the server receive all agreement of nodes in Phase One, the server will send the decision “true” to all nodes. For the server side, all nodes are expected to send ACK to the server, if the server doesn’t receive a certain node ACK message, the server will continue resending decision to this node until it gets ACK from it. This is based on an assumption that a node will recover all status even if it restarts.

3. How you handle lost message

1. If a query sent to UserNodes or a vote sent to Server is lost, Phase One aborts, and the server will send decision = “false” to all nodes.

2. If a vote decision sent to UserNodes or ACK sent to Sever is lost, the Server will continue resending the decision to this node until server gets ACK.

4. How you recover from node failures

At UserNode side, UserNodes hold a Map as a lock for each local file corresponding to each commit file. Every time the content of the lock is changed (file is deleted, add a new lock), UserNode will flush this change to log. When a node fails and starts to recover, it will reimplement the operation in log in order and finally reach the same status as before the crash.

At Server side, Server will flush the commit status to log at following timestamp:

1. before Phase One(“Filename”, “Source”, “img”)
2. Between Phase One and Phase Two(“PhaseOneDone”, “decision”)
3. After Phase Two (“commit”)

If the server crash at Phase One, after recover, the server will send abort to all nodes.

If the server crash at Phase Two, after recover, the server will load the data stored in log (“Filename”, “Source”, “img”), and continue Phase Two based on the decision in Phase One.

04/28/2020

Zihao ZHOU