ECE 428 MP3 Report
Yuhang Ren (yuhangr2)

Repository URL: https://gitlab.engr.illinois.edu/yuhangr2/ece428_mp3_yuhangr2
Commit SHA: d71a3ae1044de4ab81bde5eb0548a761df770608

Cluster number: 39

Instruction:
1. Clone server.go and client.go to multiple clusters. Import config file (in the same format as provided in MP3 webpage) into the folder.
2. For servers, run "go run server.go branch_name config_file". For clients, run "go run client.go client_id config_file". Make sure all servers are set up before clients get in.

Design document
Concurrency control approach:
Concurrency is controlled using an optimistic approach, such that concurrency is checked at commit time. Timestamped ordering is used to achieve serial equivalence. Timestamps, RTS, and TW are stored in each server. Four locks are used: network mutex, account mutex, message mutex and commit mutex, all of which are part of the server struct. network mutex is acquired when messages are sent across different clusters (connection.Write()). Account mutex is acquired when account information stored in each sever is being read or written. Message mutex is a more general lock which is acquired when other data stored in servers are read or written. Commit mutex is acted as a message blocker that let transactions with lower priority wait or wake them up. The wait and wake-up instruction is broadcast by sync.Cond formed by commit mutex.

Abort and Rollback
Transactions are aborted when consistency is violated before a commit. The coordinator aborts the changes it has made in RTS and TW, send abort messages to other servers, and signal other transactions to wake up if blocked by its commit condition. For servers receiving abort messages, they delete changes made in TW if any.
When other transactions try to access partial results from aborted transactions, they are blocked by the commit mutex. The commit mutex is released if commit is success or the blocking transaction is aborted. And before a commit is done, account data are not revised.

Deadlock
Deadlock is avoided by timestamped ordering. As mentioned in lecture, timestamped ordering prevents deadlocks.