15440 P4 Report

April 29, 2018

General Description

This project simulates a failure-tolerant system that uses two-phase commit to coordinate between Server and User Nodes in publishing collages.

I. Two Phase Commit

Each commit procedure is handled separately by a single thread.

Phase One: Server sends commit query to User Nodes
User Nodes reply with commit agreements (Yes/ No)

Phase Two: Server sends commit decision (Yes / No / Abort) to User Nodes
User Nodes reply with ACKs

II. Failure Tolerant

The server takes the main responsibility for starting and recovering commit, and conducts the User Nodes in committing and recovering. The Server's log file keeps track of both the information and the progress of all the finished and on-going commits.
The User Node, is only responsible for itself, and only replies to the queries or acts to the commands from the Server. The User Node's log file only keeps track of its own files' status.

Log File Design

For each Server or User Node, log files are maintained in the directory "log" under the working directory.

The **Server** keeps a .txt log file named the same as the image for each commit procedure. The log file for a fully finished commit procedure has the following structure:

```
File Name:<name of image>
Sources:<UserNode_1>:<file_1> <UserNode_2>:<file_2> <UserNode_3>:<file_3> ...
Phase One
Phase Two:<commit_decision>
DONE
```

- After a commit starts, first log commit information including the commit file name and the sources.
- When Phase I starts but before the Server starts to distribute commit queries, log "Phase One", so that to cover the situation when the Server crashes while sending the commit queries.

- When Phase II starts, before the Server starts to distribute commit decisions, but after the image is saved (when commit approved), log “Phase Two” and the commit decision for the same reason as above and for the Server to recover the commit decision if it crashes during Phase II.
- After the Server receives all ACK messages, log “DONE”, indicating the commit has fully finished.

The **User** Node keeps a single log file named “log.txt” which keeps track of its source file status. A source file has three status: agreed to a commit and waiting for Server’s commit decision, formally used by a commit, agreed to a commit but the commit aborted. The log file for a User Node is a list of file names associated with the file status and the image name of the related commit:

<source_file_1>:<commit_file_name>:<status>
<source_file_2>:<commit_file_name>:<status>
<source_file_3>:<commit_file_name>:<status>
...

Each row is logged every time the User Node makes change to a source file status.

Failure Situations and Failure Recovery

There are mainly two kinds of failures:

1. The Server or the User Node itself crashes
2. Reply message from the other side times out after 6 seconds

Case 1 failure situation: This only happens on the Server side. A commit agreement or an ACK message from the User Node times out and is recognized as lost after 6 seconds.
Case 1 failure recovery: If a commit agreement times out during Phase I, Server starts Phase II immediately and distributes commit ABORT decision. If an ACK message times out during Phase II, Server continuously redistribute the commit decision to all User Nodes until it collects all the ACKs.

Case 2 failure situation: This happens on both sides. A Server or a User Node can crash at any time.
Case 2 failure recovery for Server:
The Server reads through each commit log file.
If log file reached “Done” or has not reached “Phase One”, then ignore it.
If log file reached “Phase Two”, redo phase two with the logged commit decision.
If log file reached “Phase One”, start phase two with commit ABORT decision, delete the saved image if it exists.
Case 2 failure recovery for User Node:
The User Node keeps a map from files waiting for commit decision to the file name of its commit.
The User Node reads through the log file. For each <source_file>:<commit_file> pair, add one for a waiting status and subtract one otherwise. Finally, add the pair to the map if the counter is positive.