Replicated Concurrency Control and Recovery Design Document

Liu, Jane
jl860@nyu.edu

Christensen, Conrad
cc5608@nyu.edu

December 7^{th}, 2018

1 Project Goal

Implement a distributed database which correctly achieves replicated concurrency control with replication, including: deadlock detection, failure recovery, multi-version concurrency control, maintenance of the `ACID` principles.

2 Requirements

- System with at least Python 3.5 (untested for other versions)

3 Installation and Execution

Note: when running with reprounzip there may be a few warnings on the ld path, or related environement variables. These are a byproduct of the environment, and given the correct output is still produced the warning should be ignored.

- Unzip the turned in file (already done if you are reading this)
- Set up reprounzip directory with the follwing command
  `$ reprounzip directory setup proj_reprozip.rpz pdir`
- Run with reprozip using the following command
  `$ reprounzip directory run pdir < path/to/test_file.txt`
- Or to pass the name of the file directly (note the path to test file must be absolute, or relative to home, or reprounzip may get confused)
  `$ reprounzip directory run pdir --cmdline python3 main.py /abs/path/to/file`
- To lower the amount of output produced (to the minimum level suggested by syllabus)

```
$ reprounzip directory run pdir --cmdline python3 main.py --min-output < test_file.txt
```

- Likewise to disable "skip write to recently recovered site" optimization add `--no-rec-site-opt` argument
- Or to simply run with python
  `$ python3 main.py test_file.txt`

4 System components

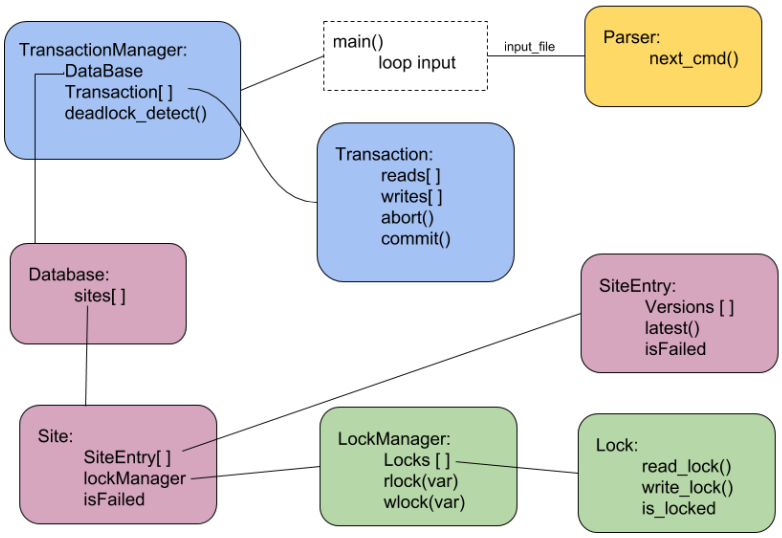

Figure 1: Main components of our distributed database

Refer to figure 1 for an overall design of the components of our project. Communication between components is marked with connecting lines (and the biggest methods and variables are shown in each box). For further description please read the following sections on each component.

Unit tests

Author: Jane Liu

Unit tests were a most important part of the implementation. These were used to ensure no regression happened when fixing bugs or implementing features.

Main logic and Parser

Author: Conrad Christensen

- Open file handle for reading (possibly stdin)
- Parse command line arguments (log levels, output control)
- Parse each line of file and create CommandType/argument pairs
- Call methods on TransactionManager based on CommandType

Transaction Manager

Authors: Conrad Christensen and Jane Liu

- Tracks the current running transactions
- Contains Database object encapsulating sites
- Performs deadlock detection
- Method for each valid command input

Database

Author: Conrad Christensen

- Small wrapper for sites list

Site and SiteEntry

Authors: Conrad Christensen and Jane Liu

- Holds all 20 varaibles as SiteEntry objects
- SiteEntry objects track versions and if they are ready to read
- Sites can be told to fail, and then reject all accesses until they recover
- Each site has a LockManager to control access to variables

Transaction

Authors: Conrad Christensen and Jane Liu

- Stores collection of reads/writes
- Has an abort method to mark transaction for failure
- Commit method which determines if transaction aborts or not
- Has writes variable so all writes can be flushed after transaction commit

LockManager and Lock

Authors: Conrad Christensen and Jane Liu

- LockManager tracks 20 lock variables per site
- Implements `readLock()`, `writeLock()`, `upgrade()` for each variable
- Contains logic to return "waits-for" relations, used for dead lock detection