Yuta Kato
6.814
Lab 4 Writeup
3 November 2014

I collaborated with Arjun Narayanan.

Design Decisions:

Locking Granularity:

I implemented page-level locking. I chose to do locking at the level of pages, because I expected to make the fewest changes to the existing code and therefore I expected to correctly achieve locking. In previous code, I was already calling BufferPool.getPage to load pages, so implementing page-level locking naturally made sense.

Deadlock Detection Policy:

I wanted to implement a simple, but reasonable deadlock detection policy, so I decided to implement a timeout. In areas of my code that block to acquire locks, specifically in BufferPool.getPage, I keep track of how long a transaction waits for a lock. After 10 seconds, I elect to throw a TransactionAbortedException. I decided to wait for 10 seconds because I did not want to wait too long to break deadlocks, and I also did not expect the acquiring and releasing of locks as a process to take very long. I blocks for locks within while loops that have no sleep time, so while the cpu is strained, I expect unblocking to be fast in the cases that there are no deadlocks.

Deadlock Resolution:

I decide to abort the transaction that is waiting for a lock rather than aborting others and attempting to complete the “current” transaction. I thought that this would provide the cleanest (code-wise) way to resolve deadlocks.

I worked on this lab for 11 hours.