Name: Mason Walls
Section: 7
University ID: 910333448

Project 2 Report

Summary:

10pts

This project was challenging. Making sure all of the memory was correct, waiting and broadcasting setup, and all of the mutex work. In the end, I learned how difficult multithreading can be. My program isn't perfect. It works well with certain ratios of threads to accounts, but not all. I believe I did my best and met almost all of the requirements of the project.

Part II:

6.2:

5pts Average run time for each program (use the "real" time):

Using trans and check wait times:
Fine Grained Average Wait: 574.511 seconds
Coarse Grained Average Wait: 10,017.005 seconds

6.3:

3.2.1:

3pts Which technique was faster - coarse or fine grained locking?

Fine grained locking was faster.

3pts Why was this technique faster?

Fine grained was faster because there was less waiting required. Instead of waiting for one of the many accounts, a thread had to wait for the entire bank to re-open.

3pts Are there any instances where the other technique would be faster?

Coarse grained would be faster if there was only one thread. This way, there isnt any waiting for threads. The single thread gets all of the banks attention.

3pts What would happen to the performance if a lock was used for every 10 accounts? Why?

The performance would be slower than fine grained but faster than coarse grained. There are less locks to wait for than coarse grained. This would be more difficult to program though.

If we had endless memory, fine grained locking would be ideal. This gives the least waiting time for each thread, as the likelihood of a specific account being locked is low. If memory isnt endless, medium grained locking would be best. Coarse grained should be a last resort.

3pts Discuss the probable "optimal" locking granularity (fine, coarse, or medium)?