CprE 308
Section 3
Project 2

Sean Gordon
Sgordon4

November 12, 2019

This project focused on the integration of multithreading with many of the other topics we've worked with previously. I haven't had much experience with multithreading in C, and it has been a while since I made such a large project in the language. As such, syntax presented the largest hurdle to the project's completion.

Overall, I learned a fair amount about correct programming techniques with multithreading, and a queue was reinforced as an important data structure for these types of applications. I will certainly remember what I learned when building future projects.

1) Which technique was faster - coarse or fine grained locking?
In practice my coarse grain implementation was only slightly faster than my fine grain, but generally coarse grain would beat out fine-grain by a fair margin.

2) Why was this technique faster?
Rather than looping through each account in a TRANS request, the jobs need only acquire one mutex.

3) Are there any instances where the other technique would be faster?
If each TRANS request was on a different account so that they would never intersect, there would be no waiting with a fine-grain approach.

4)What would happen to the performance if a lock was used for every 10 accounts? Why?
Performance would increase, as that strikes the middle ground between coarse and fine grain for this application. On average there would be less looping over locks, but there is still the chance a thread would still loop 10 times for a TRANS request.

5) What is the optimal locking granularity (fine, coarse, or medium)?
Medium is optimal, as long as it is tailored to the current algorithm.

The variable `COARSE_LOCK` at the top of `bank_server.c` switches between coarse and fine locking.

Performance Measurements - `time ./testscript.pl ./bank_server 10 1000 0`

Non-coarse: ————————————————

real 0m25.015s
user 0m0.005s
sys 0m0.010s

real 0m25.015s
user 0m0.011s
sys 0m0.004s

real 0m25.013s
user 0m0.008s
sys 0m0.005s

realAvg = 25.0343

Coarse: ————————————————

real 0m25.015s
user 0m0.014s
sys 0m0.000s

real 0m25.014s
user 0m0.005s
sys 0m0.009s

real 0m25.012s
user 0m0.012s
sys 0m0.000s

realAvg = 25.031