Name: Michael Becker

Section: 1

University ID: 371615257

**Programming Project 2 Report**

**Summary (10pts):**

This project uses a multithreaded bank server that manages account transactions through two different synchronization strategies. In this project I learned how to implement and use fine-grained and course-grained locking. The server allows client threads to check account balances and perform transactions. Some of the key parts of this project are the technical components. Using CHECK and TRANS as the main operations. Now to go into more detail about fine-grained locking. It uses a separate mutex for each bank account and allows access to different accounts. It essentially enables parallel processing. Now for coarse grained locking it uses a single mutex, in my code, “bank\_mutex.” The request queue is managed with mutex and a conditional variable and worker threads process the requests coming in. Overall, I had fun implementing these two methods and had some struggles along the way as well, but I gained a deeper understanding of them.

**6.2:**

**(5pts) Average runtime for each program (use the “real” time)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Implementation | Run 1 | Run 2 | Run 3 | Average |
| Fine-grained | 4.7 | 4.5 | 4.9 | 4.7 seconds |
| Coarse-grained | 77.1 | 76.8 | 77.4 | 77.1 seconds |

**6.3:**

1. **(3 pts) Which technique was faster – coarse or fine-grained locking?**

The find grained locking was faster

1. **(3 pts) Why was this technique faster?**

Fine-grained locking is faster because it only locks the accounts involved in a transaction as opposed to all of them (like coarse/ bank\_lock). Worker threads don’t have to wait for certain operations to be completed. If a CHECK operation was ran there is no need to wait for other transactions.

1. **(3 pts) Are there any instances where the other technique would be faster?**

Coarse-grained would be faster if the system had less accounts or if the system involved the same accounts.

1. **(3 pts) What would happen to the performance if a lock was used for every 10 accounts? Why?**

If there was a lock used for every 10 accounts it would be a middle ground… If you look at our implementation it’s a lock for every account and a lock for all accounts. So performance would be somewhere in the middle. More transactions could occur at the same time than coarse but less than fine. It would most likely be considered medium-grained.

1. **(3 pts) What is the optimal locking granularity (fine, coarse or medium)?**

It would depend on the scenario. For work that would contain transactions with many accounts fine-grained would be the best. But medium might be good where overhead is important.