2. Bank accounts

Problem

At a bank, we have to keep track of the balance of some accounts. Also, each account has an associated log (the list of records of operations performed on that account). Each operation record shall have a unique serial number, that is incremented for each operation performed in the bank.

We have concurrently run transfer operations, to be executed on multiple threads. Each operation transfers a given amount of money from one account to some other account, and also appends the information about the transfer to the logs of both accounts.

From time to time, as well as at the end of the program, a consistency check shall be executed. It shall verify that the amount of money in each account corresponds with the operations records associated to that account, and also that all operations on each account appear also in the logs of the source or destination of the transfer.

Solution

AccountsCount = 1,000

Method 1: Bank level lock

For the first solution I chose to use a single lock for the whole bank. Each transfer operation locks the bank for the whole duration of the transfer. The consistency checks that run regularly on the main thread also lock the bank while running.

Method 2: Account level lock

For this solution I chose to use separate locks for each account. Each transfer operation first locks the sender's account, adds the funds and logs the transfer, then does the same for the receiver. The consistency checks also lock each pair of accounts independently while running.

Hardware

CPU: Intel® Core™ i7-8750H CPU @ 2.20GHz, 2201 Mhz, 6 Core(s), 12 Logical Processor(s)
RAM: 16 GB

Tests

Average execution time

Method 1 (left) vs Method 2 (right)

1,000,000 operations

- `2` threads: `564ms` vs `434ms`
- `5` threads: `633ms` vs `429ms`
- `10` threads: `624ms` vs `424ms`
- `20` threads: `674ms` vs `440ms`

2,000,000 operations

- `2` threads: `1255ms` vs `783ms`
- `5` threads: `1416ms` vs `792ms`
- `10` threads: `1711ms` vs `815ms`
- `20` threads: `1460ms` vs `794ms`