Questions regarding test Performance output:

Serguey Khovansky

1. Why does the NO_LOCKING test succeed for one thread and fail for 2-20 threads?

For the case of multiple threads, it is possible that at a point in time different threads operate on the same account. This simultaneously exerted multiple influences might corrupt data in the account, for example one thread might partially overwrite the input of another thread resulting in a number that makes no sense. For the case of a single thread at each point of time only one method can change a given account, so in this case issues similar to what we observe in the multiple threads do not occur.

2. How do you explain the change in performance for LOCK_BANK as threads increase from 1 to 20.

In this case, we lock the whole method transferLockingBank() of the bank object. As a result, regardless of the account ids each thread must wait its turn until the currently running thread (operating over its own account ids) finishes this method. It means that at each time moment only one thread performs the method transferLockingBank(). In addition, the program needs time to switch from one thread to another. This outcome implies that the total execution time for 20 threads might exceed the total execution time for 1 thread, e.g. for the case of 1 thread the total execution time can be 2.3 sec but for 20 threads it can be as much as 4.4 sec (or the transactions per msec might decline from 2172 for 1 thread to 1120 for 20 threads). Overall, it is apparent that the method transferLockingBank() coupled with multiple threads tends to slow down the system rather than to accelerate it.

3. How does the performance of LOCK_BANK compare to that of LOCK_ACCOUNTS? How do you explain this?

In the case of LOCK_ACCOUNTS, we lock the methods that implement operations with accounts having specific ids. As a result, threads that change different account ids can function simultaneously. It means that at each time moment several threads might perform the method transferLockingAccounts(). Consequently, under this design the total execution time tends to decline with the number of threads, such that the number of transactions/msec increases with the increase in the number of threads. In other words, the method transferLockingAccounts(), that corresponds to LOCK_ACCOUNTS, coupled with multiple threads tends to speed up the system. In contrast, the LOCK_BANK tends to slow down the system, see my discussion about it in the previous question.