Project 1

This project aims to test the correctness and performance when using multi-threading.

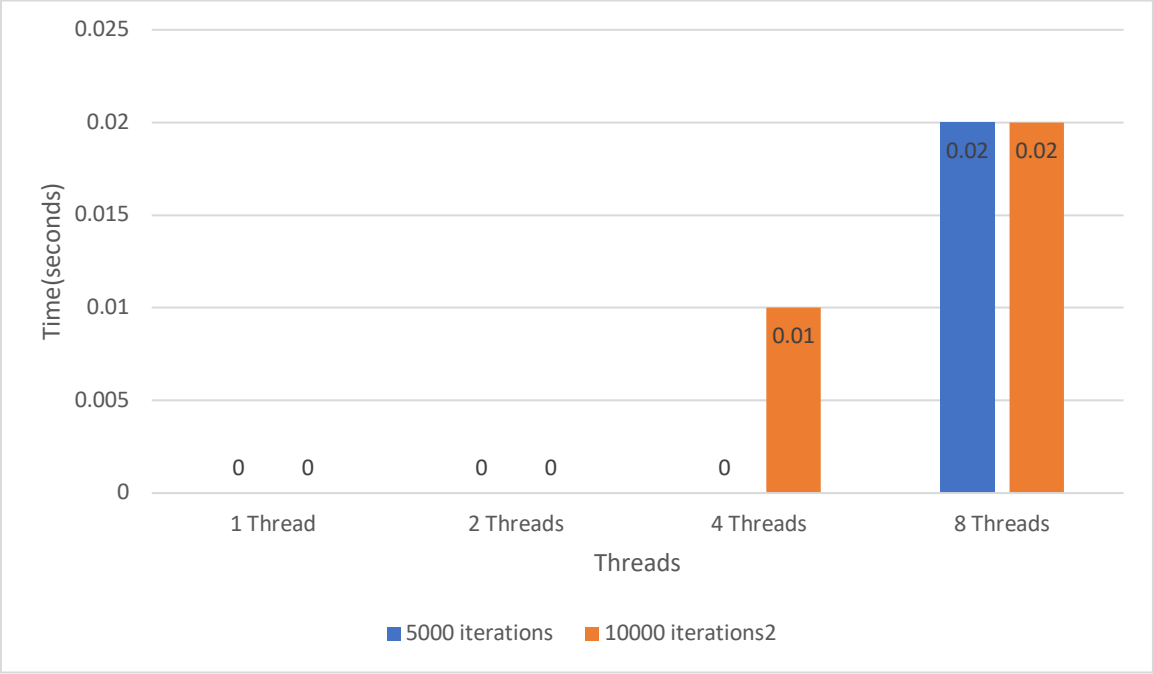
First, define a simplemap_t of types <int, float> that represents a collection of bank accounts, each account has a unique ID of type int, and each account has an amount of fund of type float.

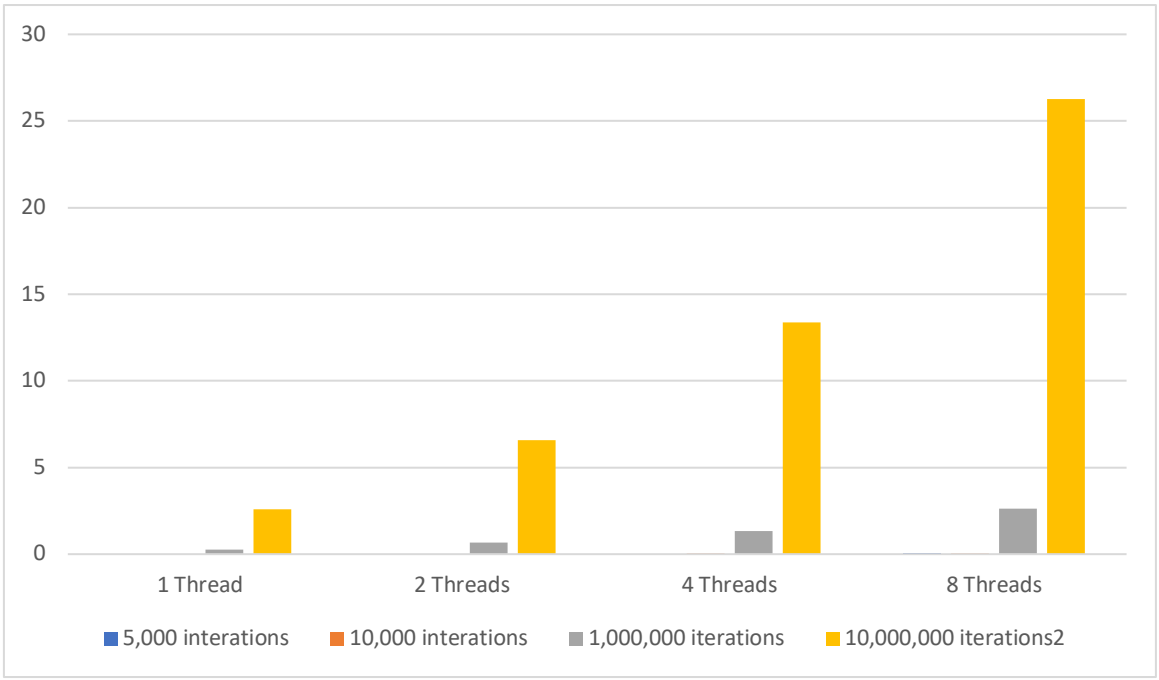
Second, use multi-threading in two functions, “deposit” and “balance”. The function of “deposit” selects two random bank accounts and an amount. This amount is subtracted from the amount of B1and summed to the amount of B2. The function of balance sums of all the accounts amount.

Third, in “do_work” functions, for one loop the function “deposit” should be called with 95% of the probability. The rest 5% the function “balance” should be called.

Moreover, I use lock in “deposit” function, and lock_guard in “balance” function to achieve “read” and “write” operation atomically.

For my experiment result, when the iteration is smaller (graph 1), it does not have a big change when increasing threads. When the iteration is bigger (graph 2), it’s more expensive when increasing threads. I think the reason is that the mathematics we use is trivial, currently the cost of creating threads is more expensive. If adding some operations which could take a lot of CPU, then it will be another story.




30
25
20
15
10
5
0
1 Thread
2 Threads
4 Threads
8 Threads
5,000 interations
10,000 interations
1,000,000 iterations
10,000,000 iterations2