CSE201 Advanced Programming (Section-A)
Lab Assignment 07

IIIT-Delhi. 17th Nov 2018. Due by 11:59 pm on 19th Nov 2018

This is a take-home lab assignment. No extensions whatsoever will be provided. Any submission after the deadline will not be evaluated. If you see any ambiguity or inconsistency in a question, please seek a clarification from the teaching staff. Please read the entire text below very carefully before starting your implementation.

Plagiarism: All submitted lab assignments are expected to be the result of your individual effort. You should never misrepresent someone else's work as your own. In case any plagiarism case is detected, it will be dealt as per IIITD plagiarism policy and without any relaxations: https://www.iiitd.ac.in/sites/default/files/docs/education/AcademicDishonesty.pdf

Please note that you are **not** allowed to discuss the design/solution of the lab assignment (e.g., backpack discussions, etc.). Anyone who is found doing this will be treated as plagiarism case.

Parallel Programming in Java

A new banking service has just started in India and the name of this bank is **Serious Bank of India** (SBI). They have an online portal where the customers can login and transfer their money from one SBI account to another SBI account. Lately customers have started complaining to SBI that they have to wait for several minutes for their online transfers to complete. SBI's IT team has investigated this complaint and found that the bottleneck is due to slow processing of transactions. At any given time there are millions of transactions waiting to happen but their software performs all the transactions sequentially. They have approached you to help them in parallelizing their software.

Their requirements are as following:

1) There are **10000** accounts in their bank. You should create all the accounts as soon as the banking software is launched. SBI don't want to alter their existing implementation of Account class. Assume that none of the methods inside Account class will be using any multithreading features.

2) A transaction is performed only between two different account holders. A transaction is done by first debiting the amount from the sender's account and then crediting that amount to the destination's account. Once the amount is credited to the destination account, only then the transaction is said to be completed.

3) At any given time there are exactly **10 million** pending transactions in SBI. You can simulate this by first randomly creating all pending transactions before processing those transactions.

4) Once all the transactions are created, the computation required to complete these transactions should be done in parallel. You **must use the best possible parallelization strategy** in your implementation. Your implementation must be able to process pending transactions faster by increasing the thread count.

5) After all the transactions are completed, you must verify if the above step complete with correctness with any number of threads you use (assume maximum number of threads are four). Report this result with appropriate messages ("Processing PASSED" or "Processing FAILED"). Execution time should only be considered for total time taken to complete step 4 above.

Deliverables:

1) Your implementation of SBI software by using an object oriented approach.
2) Brief explanation of why you chose that particular parallelization strategy.
3) Draw these three graphs for your banking software by using threads = 1, 2, 3, and 4: a) execution time graph, b) speedup graph, and c) parallel efficiency graph. You should upload the PDF of these graphs and upload along with your solution on backpack.