ECSE 420 Parallel Computing **Dennis Liu #260581270**
Assignment #1 **Yijie Zhou #260622641**

Question 1.1 and 1.2

Sequential and parallel multiplication static methods can be called with two matrices of the same size.

Question 1.3

The sequentialVsParallel() method is the test method which executes matrix multiplication of two 2000 by 2000 matrices both sequentially and in parallel with 4 threads. The execution progress and time used in milliseconds is printed on the console.

Question 1.4

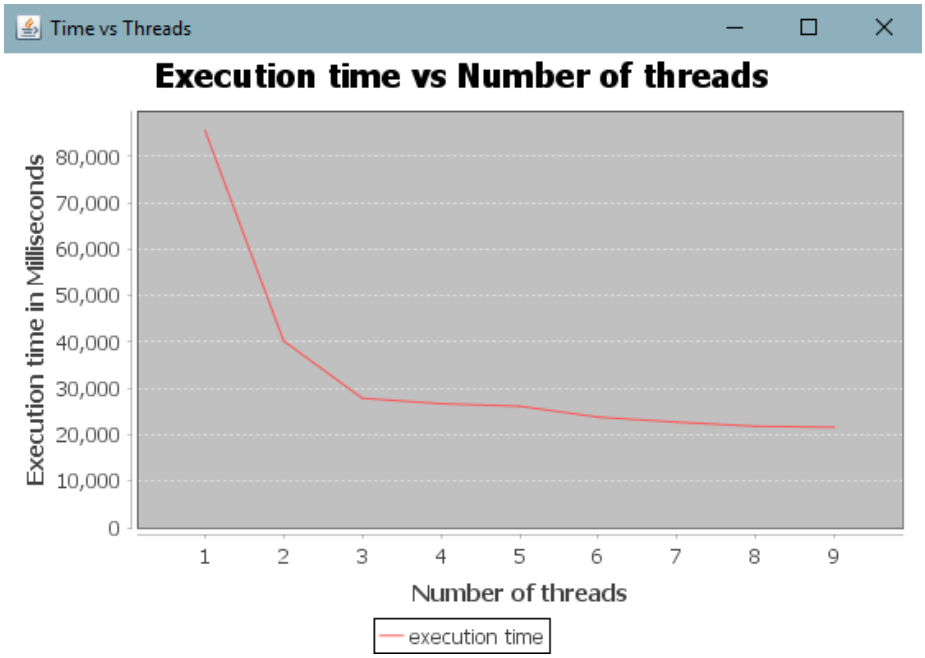
The timeVsThreads() method is the test method which executes matrix multiplication of two 2000 by 2000 matrices in parallel with increasing number of threads from 1 to 9. The execution progress and time used in milliseconds is printed on the console. A 2D line graph is also generated with JFreeCharts library after the processes are finished.

Question 1.5

The timeVsSize() method is the test method which executes matrix multiplication of matrices sequentially and in parallel with increasing size. The matrices have sizes from 200 by 200 to 2000 by 2000, with increments of 200 elements. The execution progress and time used in milliseconds is printed on the console. A 2D line graph is also generated with JFreeCharts library after the processes are finished.

Question 1.6

Comment on Question 1.4



The sequential process of the matrix multiplication completed around 77000ms, whereas the processing times with 2 to 9 threads varied from 40000ms to 21000ms. This speedup rate matches the definitions of Amdahl's Law:

$$S_{speedUp} = \frac{1}{(1-p) + \frac{p}{n}} \tag{1}$$

where p is the fraction of the execution in parallel, and n is the number of threads. The speedup calculations during our testing for when using 2, 4, and 8 threads are:

$$\frac{77761}{40206} = \frac{1}{(1-p) + \frac{p}{2}} \tag{2}$$

$$p = 0.966 \tag{3}$$

$$\frac{77761}{26692} = \frac{1}{(1-p) + \frac{p}{4}} \tag{4}$$

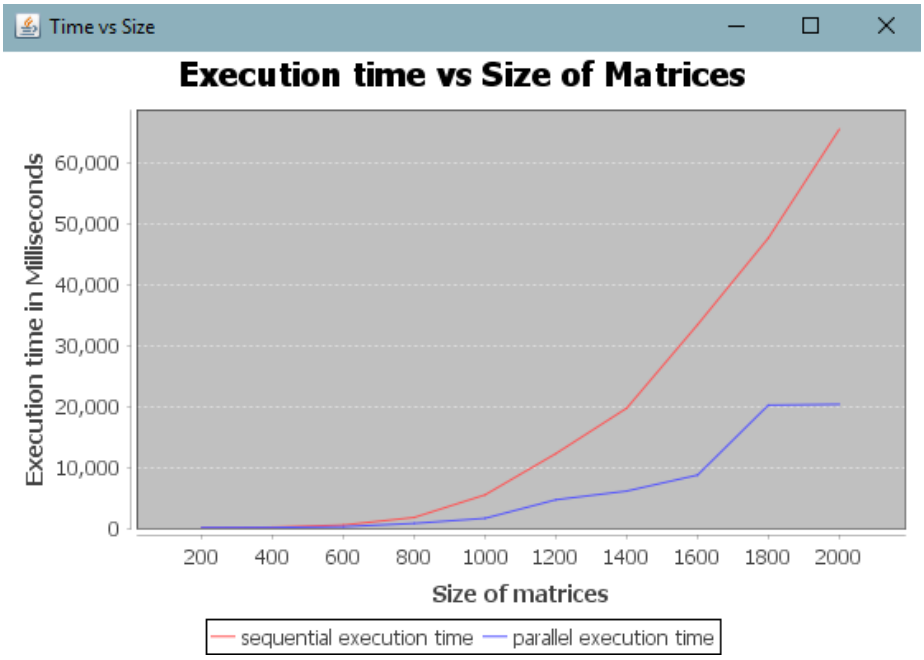
$$p = 0.876 \tag{5}$$

$$\frac{77761}{21836} = \frac{1}{(1-p) + \frac{p}{8}} \tag{6}$$

$$p = 0.822 \tag{7}$$

The decreasing speedup when each new thread is added, and the decreasing parallel fraction of the process, follow the law of diminishing returns of Amdahl's Law. The speedup ratio will diminish as the total throughput heads towards the limit of $\frac{1}{1-p}$.

Comment on Question 1.5



The execution time of the sequential process grows exponentially as the size of matrices increases. This is due to the fact that there are $O(n^3)$ computations to be done. The execution time of the parallel process also grows exponentially, however, its speed is much faster. An ideal divide-and-conquer algorithm running on a machine with infinite threads has a critical path length of $O(log_2 n)$ steps; thus the algorithm implemented without divide-and-conquer will grow with rates between $O(n^3)$ and $O(log_2 n)$.

Question 2.1

My demonstration of deadlock, each process is holding a different resource for exclusive access, and each one is requesting access to the resource held by the other. This has a consequence of circular waiting, none of those processes will release the resource, the program halted because of the deadlock.

Question 2.2

Using try lock to acquire the lock, if dead lock occurs, release current resource and try to acquire the lock again after a while. This eliminates the circular wait, possible design solution implemented in runner_deadlockFree and Question2_deadlockFree class.

Question 3.1

Demonstration in Question3 and Philosopher class. Using synchronized block, each fork is a shared object, concurrency is solved by monitoring method, but deadlock do exist.

Question 3.2

To eliminate circular wait, last philosopher reaches right fork first instead of left fork. Synchronized block is still be used. Demonstration: Question3_deadlockFree and Philosopher_deadlockFree

Question 3.3 and 3.4

Using a fair lock instead of a synchronized block, thus every thread calling lock() is now queued, this increase fairness to avoid starvation. The program is written for n philosophers as well. Demonstration: Question3_fairlock and Philosopher_fairlock

Question 4.1

$$S_{speedUp} = \frac{1}{(1-p) + \frac{p}{n}} \tag{8}$$

$$S_{speedUp} = \frac{1}{40\% + \frac{60\%}{N}} = 2.5 \tag{9}$$

Question 4.2

$$S_n = \frac{1}{0.3 + \frac{0.7}{n}} \tag{10}$$

$$S'_n > 2 * S_n \tag{11}$$

$$\frac{1}{\frac{0.3}{k} + \frac{0.7}{n}} > 2 * \frac{1}{0.3 + \frac{0.7}{n}} \tag{12}$$

$$k > \frac{0.6}{0.3 - \frac{0.7}{n}} \tag{13}$$

Question 4.3

$$\frac{1}{\frac{1-p}{3} + \frac{p}{n}} = \frac{1}{1 - p + \frac{p}{n}} * \frac{1}{2} \tag{14}$$

$$1 - p = 1 - \frac{5}{5 - \frac{3}{n}} \tag{15}$$