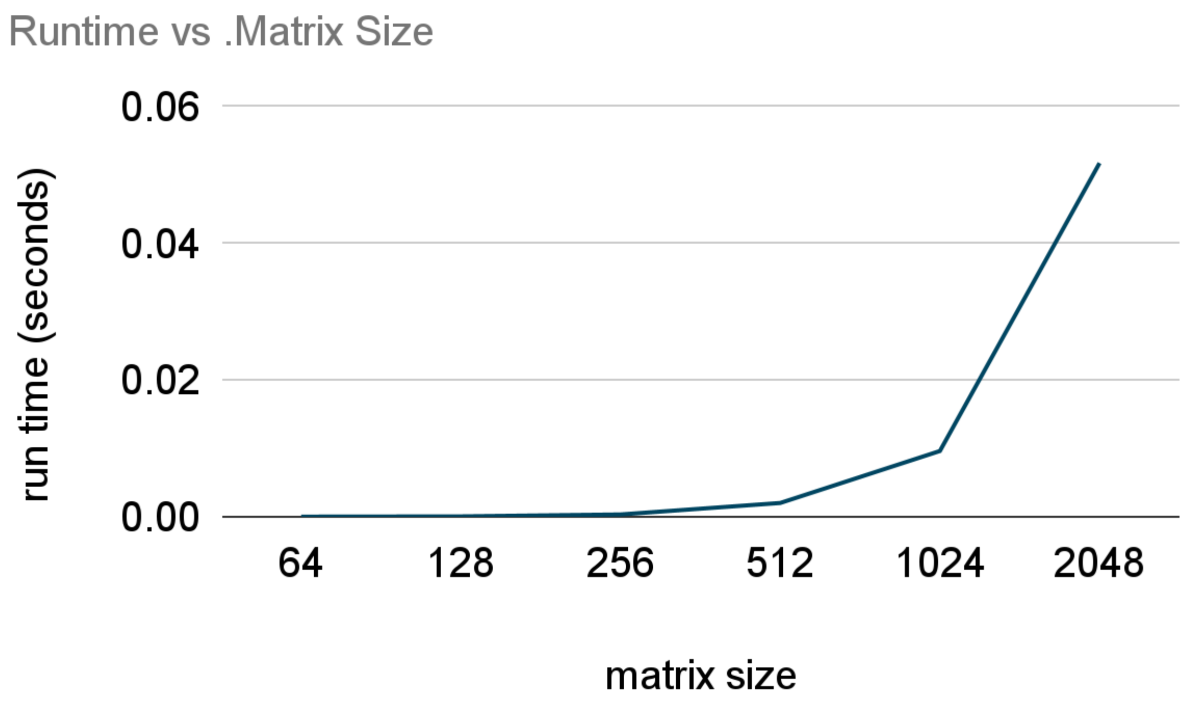
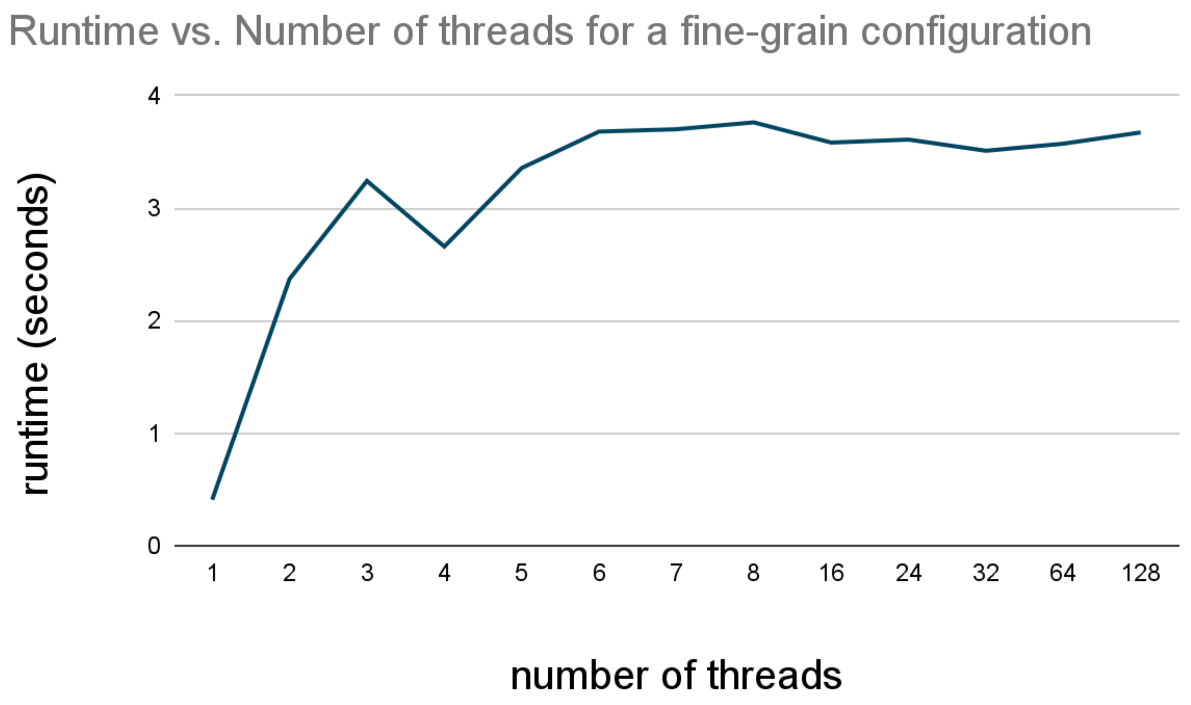
231P HW5 Report
Student 1 Name: Xinyun Shen ID: 3080 5314
Student 2 Name: Dongjue Zu   ID: 3256 9266

1. A characterization of the single-threaded transposer execution time as the size of the matrix increases. Add plots and figures as necessary to explain your results.



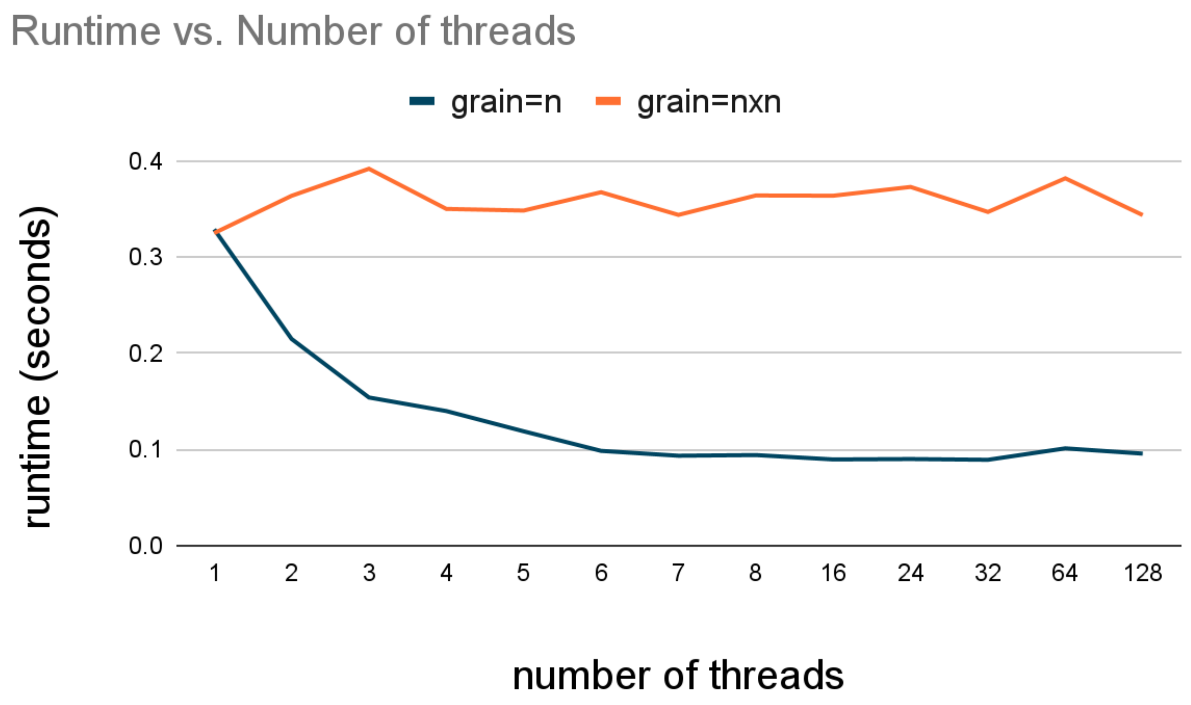
From the above image, we can see that the larger the matrix size is, the slower the program execution will be in the single-threaded transposer scenario.

2. A characterization of the multi-threaded transposer execution time for a fine-grain configuration. Keep the number of exchanges per thread constant, grain=1. Keep the size of the matrix constant, n=7,000. Variate the number of threads in {1; 2; 3; 4; 5; 6; 8; 16; 24; 32; 64; 128} Add plots and figures as necessary to explain your results



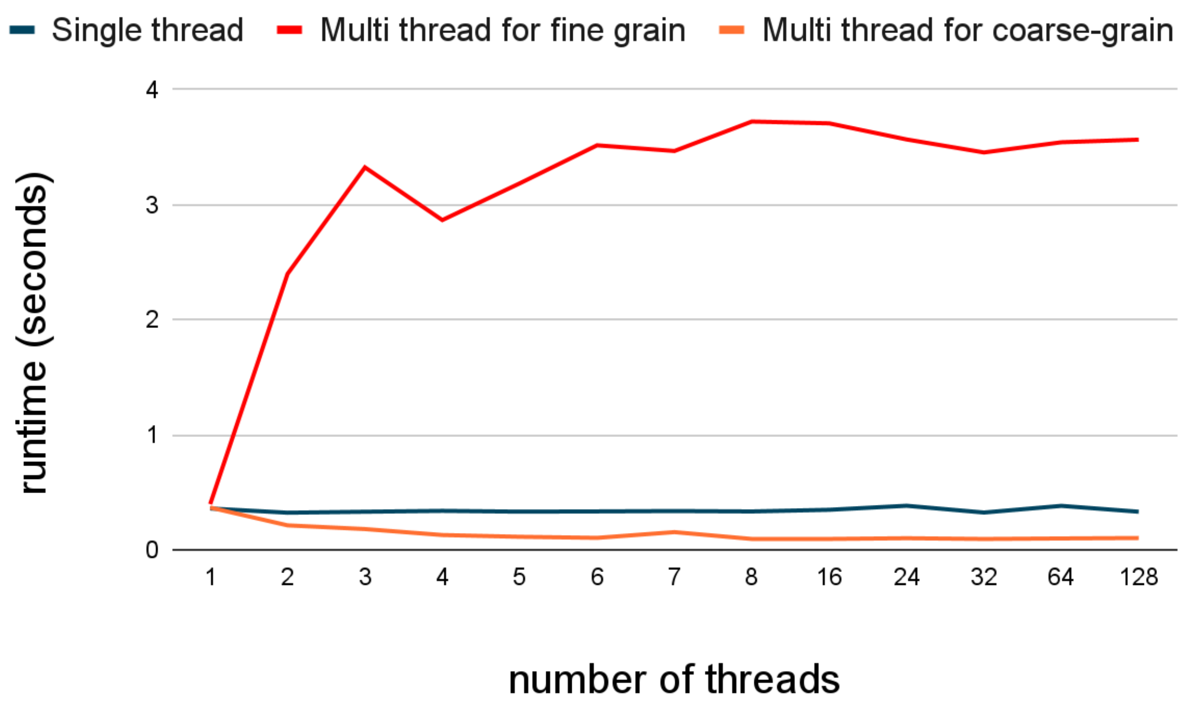
Overall, we can find that as the number of threads increases, the execution time increases. Because the overhead of using multi-thread increases as the number of threads increases, which takes up most of the execution time. Only when the number of threads is 4, we can find a slight decrease in the runtime. This is because our computer is a 4-core CPU.

3. A characterization of the multi-threaded transposer execution time for a coarse-grain configuration. Experiment with different values for the parameter grain, particularly grain=n and grain=(n×n)/threads. Keep the size of the matrix constant, n=7,000. Variate the number of threads in {1; 2; 3; 4; 5; 6; 8; 16; 24; 32; 64; 128} Add plots and figures as necessary to explain your results



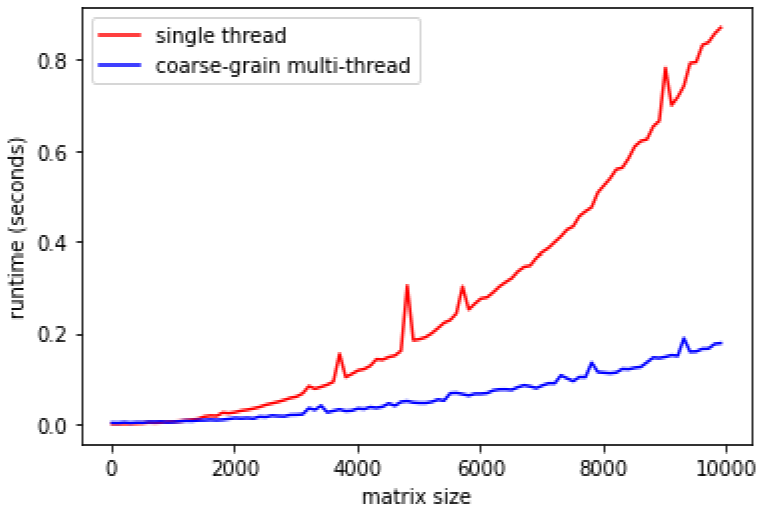
As the number of threads increases, when grain equals n, which is the size of the matrix, the runtime will gradually decrease. Since we use a 4-core computer, when there are 4-6 threads, the running time is the least.
However, as the number of threads increases, if the grain equals n * n, the runtime will fluctuate between 0.3s to 0.4s. Because when the grain size is larger than the total number of elements, all work will be assigned to one single thread, which means more threads will not lead to less running time.

4. Fixing n=7,000, grain=1 for fine-grain, and grain=n for coarse-grain; graph the speedup obtained by using the fine and coarse grain multi-thread version with respect to the single-threaded, as the number of threads used in the multi-threaded versions variates in {1; 2; 3; 4; 6; 8; 16; 32; 64; 128}. Explain your results.



For the coarse grain multi-thread version, we can find that as the number of threads increases, the runtime first decreases and then becomes stable and the runtime is faster than the single-threaded version.
However, for the fine grain multi-thread version, we can find that as the number of threads increases, the execution time increases, except for 4 threads.
Moreover, the runtime for the fine-grain multi-thread version is much slower than the coarse-grain multi-thread version. It is because, for the fine-grain multi-thread version, the overhead brought by multi-thread takes up much more time than the transposition operation itself since only one element is exchanged in each round of calculation.

5. Consider matrices of size n from 10 to 10,000. Graph the speedup obtained by the fastest coarse-grain transposer with respect to the single thread version. Explain your results.



From the above image, we can find that as the matrix size increases, both the run time of the coarse-grain and single thread transposer increases. However, overall, the coarse-grain transposer is much faster than that of the single thread, because we fully utilized all 4 cores in the CPU to do the work parallelly.