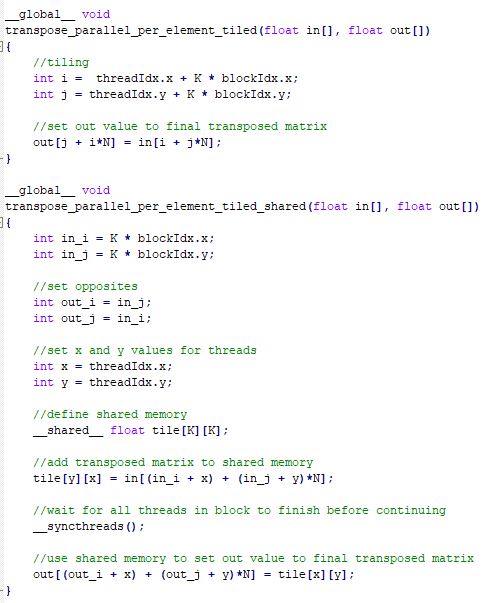
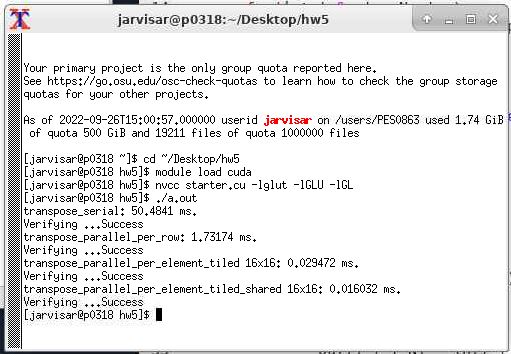
Adam Jarvis Assignment 5

For this assignment, I was tasked with creating two different methods for transposing a matrix in CUDA/C.

Below is a screenshot of my final code:

After running the code on OSC, the console outputs the amount of time it takes for execution.



Based on the above output, we can see that transpose\_serial takes the longest amount of time while transpose\_parallel\_per\_element\_tiled\_shared takes the shortest amount of time. Transpose\_serial is allocated a single thread for the entire operation. Transpose\_parallel\_per\_row is allocated one thread per row of the output matrix. Transpose\_parallel\_per\_element\_tiled is allocated one thread per element in KxK threadblocks. transpose\_parallel\_per\_element\_tiled\_shared is also allocated one thread per element in KxK threadblocks but uses shared memory.

Using the above descriptions, it’s obvious to see that allocating more threads generally leads to a shorter execution time. Similarly, based on the above console output, we can also see that using shared memory instead of global memory also leads to shorter execution time.

All in all, utilizing multiple threads and a shared memory model when transposing a large matrix in CUDA/C can lead to drastically shorter execution times.