CSE 5441 LAB3 Report

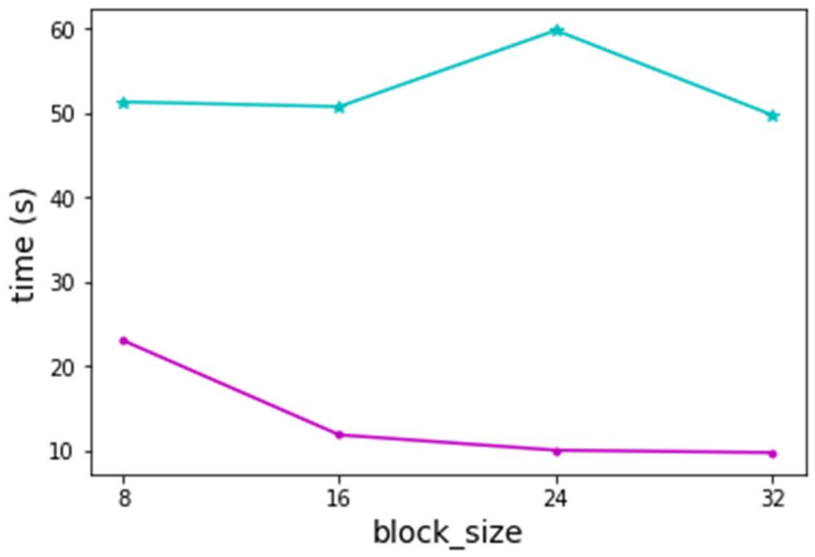
Neng Shi

A brief explanation describing how my changes ensure mutual exclusion

The strategy is submatrix multiplication, each of which can be done in a block and use the shared memory.

I declared an array “temp” in shared memory whose size is 2 * block_size * block_size, which is used to store the submatrix. There is one loop in the “mmul” function, which is shown as follows, and each time we process one submatrix from A and one submatrix from B. First, the submatrices from the global matrix “A” and “B” are copied to the shared memory. We synchronize all the threads to ensure all the data is available. Second, we calculate the dot product by operating in the shared memory. After the loop, the calculated dot product is copied to the output matrix “C”.

A chart which illustrates the scalability of my shared memory-based solution



One block size which gave me the best perfomance & the reason

When block_size = 32, it gives the best performance because it is the max block_size that can be allocated and let the shared memory be well utilized.

Any unusual or unexpected results

No.