Lab 1: Matrix-Matrix Multiplication

Objective

Implement a tiled dense matrix multiplication routine (C=A*B) using shared memory in CUDA. A and B matrix elements can be randomly generated on the host. Your code should be able to handle arbitrary sizes for input rectangular matrices.

Instructions

Make a new folder for your project. Copy and modify the code of matrixMul in CUDA sample code to include the following key functions:

1. allocate device memory
2. copy host memory to device
3. initialize thread block and kernel grid dimensions
4. invoke CUDA kernel
5. copy results from device to host
6. deallocate device memory
7. implement the matrix-matrix multiplication routine using shared memory and tiling algorithm
8. handle boundary conditions (thread divergence) when dealing with arbitrary matrix sizes

Your final executable can be run using the following command:

.TiledMatrixMul -i <rowDimA> <colDimA> <colDimB>

About input parameters:

<rowDimA> is the row dimension of the rectangular matrix A.
<colDimA> is the col dimension of the rectangular matrix A.
<colDimB> is the col dimension of the rectangular matrix B.

Questions

(1) How many floating operations are being performed in your dense matrix multiply kernel if the matrix size is N times N? Explain.

(2) How many global memory reads are being performed by your kernel? Explain.

(3) How many global memory writes are being performed by your kernel? Explain.

(4) Describe what further optimizations can be implemented to your kernel to achieve a performance speedup.

(5) Suppose you have matrices with dimensions bigger than the max thread dimensions allowed in CUDA. Sketch an algorithm that would perform matrix multiplication algorithm that would perform the multiplication in this case.

(6) Suppose you have matrices that would not fit in global memory. Sketch an algorithm that would perform matrix multiplication algorithm that would perform the multiplication out of place.

Report Submission

Your Lab report should be submitted along with the source code folder (zipped) via Canvas no later the required project due date. In the report, you have to

1. Answer all the above questions.
2. Include important implementation details for developing your CUDA program.
3. Demonstrate extensive experimental results, such as compute throughputs (GFLOPS), using different combinations of matrix sizes and thread block sizes. Matrix size may not be multiples of the thread block size.
4. Discuss your results in your report using the knowledge learnt from our classes.