Take-Home Exam CUDA_BMM

Parallel Programming & Architectures

Consideration

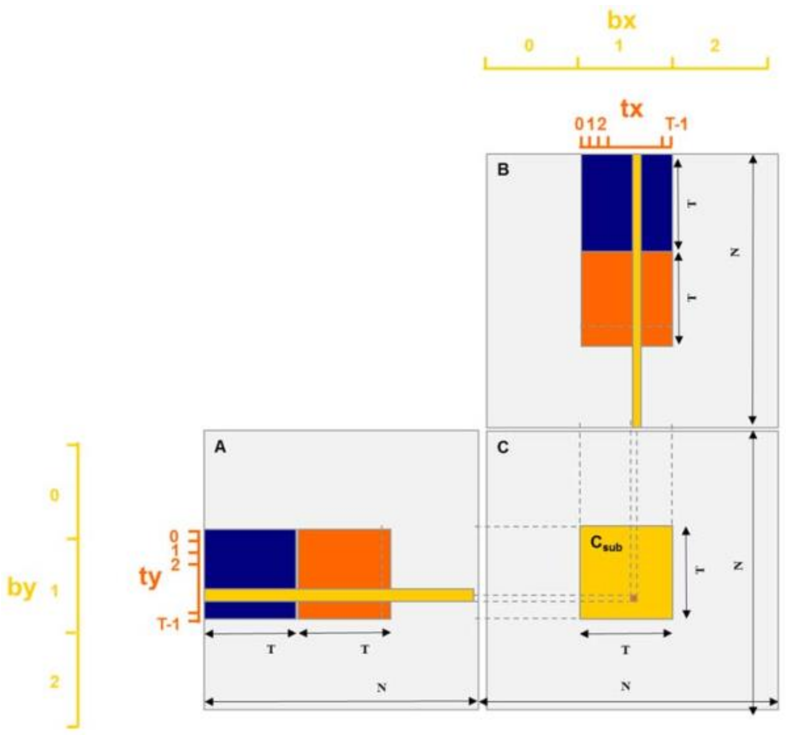
- ✓ Your code is automatically graded using a script, and therefore, if your file/folder names are wrong you will receive a grade of **zero**. Please read and follow the instructions carefully. Common mistakes include
  - Different file or folder names
  - Different formatting of input or output
  - Not paying attention to case sensitiveness of C++ and Linux
- ✓ Go to the folder ~/the/cuda_bmm/ in your home directory on the server and put your codes in this directory and remove any compiled binaries and test cases.
- ✓ Make sure your code compiles and runs without any error **on the server**. Your grade will be **zero** if any compile or runtime error occurs on the server. **Any!**
- ✓ The provided test cases, examples and sample codes (if any) are only to better describe the question. They are **not** meant for debugging or grading. It is your responsibility to think of and generate larger and more complex test cases (if necessary) in order to make sure your software works correctly for all possible scenarios.
- ✓ Start early and don't leave everything to the last minute. Software debugging needs focus and normally takes time.
- ✓ Just leave your final programs on the server. **Don't** email anything!
- ✓ Your grade is divided into several parts. In all cases, if you miss **correctness** (i.e. your code doesn't satisfy desired functionality), you miss other parts (e.g. speed, coding style, etc.) too. This rule is applied separately for each section of a take-home exam. So for example, in cuda_mm, your code might not be correct for M >= x but still you will get your grade for lower M values.
- ✓ Talking to your friends and classmates about this take-home exam and sharing ideas are *OK*. Searching the Internet, books and other sources for any code is also *OK*. However, copying another code is **not OK** and will be automatically detected using a similarity check software. In such a case, grades of the copied parts are **multiplied by -0.5**. Your work must be 100% done only by yourself, and you **should not** share parts of your code with others or use parts of other's codes. Online resources and solutions from previous years are part of the database which is used by the similarity check software.

bmm.cu – Block Matrix Multiply

Grade: *20%* correctness, *80%* speed

A and B are *"float"* matrices of size $N \times N$, where $N = 2^M$. We would like to calculate $C = A \times B$. Parameter M should be a command line argument to the *main()* function. Your program must work correctly for any value $10 <= M <= 13$. Note that larger arrays (M>=14) do not fit into the GPU global memory. Your program must fill A and B with random float values between *-8.0f* and *+8.0f* using *srand()* and *rand()* functions.

You must use the block matrix multiply algorithm explained in class. Each block computes one square sub-matrix C_{sub} of size $T \times T$. Each thread computes one element of C_{sub}. See the following figure (Chapter 5 in David Kirk's book). Parameter T should be defined on top of bmm.cu using *#define* directive. Your program must work correctly for *T=4, 8 and 16*.



tx = threadIdx.x ty = threadIdx.y

bx = blockIdx.x by = blockIdx.y

Check correctness of your calculations by comparing the final values from GPU with results given by a serial matrix multiply function executed in CPU. Use the provided gpuerrors.h, *gputimer.h*, *bmm_main.cu*, *bmm.h* and *bmm.cu* files to start your work. Only modify *bmm.cu*. Check the speed of your calculations as shown in the provided *bmm_main.cu* file.

Compile: *nvcc -O2 bmm_main.cu bmm.cu -o bmm*

Execute: *./bmm M*

Note that N is equal to 2^M.