

Description

Code

Questions

Attempts

History

Tiled Matrix Multiplication

The coding deadline is 12 days from now (2015-07-10T23:00:00Z) .



Objective

Implement a tiled dense matrix multiplication routine using shared memory.

Prerequisites

Before starting this lab, make sure that:

- You have completed "Matrix Multiplication" MP
- You have completed all week 2 lecture videos

Instruction

Edit the code in the code tab to perform the following:

- allocate device memory
- copy host memory to device
- initialize thread block and kernel grid dimensions
- invoke CUDA kernel
- copy results from device to host
- deallocate device memory
- implement the matrix-matrix multiplication routine using shared memory and tiling

Instructions about where to place each part of the code is demarcated by the `//@@` comment lines.

Suggestions

- The system's autosave feature is not an excuse to not backup your code and answers to your questions regularly.
- If you have not done so already, read the tutorial (/help)
- Do not modify the template code provided – only insert code where the `//@@` demarcation is placed
- Develop your solution incrementally and test each version thoroughly before moving on to the next version

- Do not wait until the last minute to attempt the lab.
- If you get stuck with boundary conditions, grab a pen and paper. It is much easier to figure out the boundary conditions there.
- Implement the serial CPU version first, this will give you an understanding of the loops
- Get the first dataset working first. The datasets are ordered so the first one is the easiest to handle
- Make sure that your algorithm handles non-regular dimensional inputs (not square or multiples of 2). The slides may present the algorithm with nice inputs, since it minimizes the conditions. The datasets reflect different sizes of input that you are expected to handle
- Make sure that you test your program using all the datasets provided (the datasets can be selected using the dropdown next to the submission button)
- Check for errors: for example, when developing CUDA code, one can check for if the function call succeeded and print an error if not via the following macro:

```
#define wbCheck(stmt) do {                                     \
    cudaError_t err = stmt;                                     \
    if (err != cudaSuccess) {                                   \
        wbLog(ERROR, "Failed to run stmt ", #stmt);           \
        wbLog(ERROR, "Got CUDA error ... ", cudaGetErrorString(err)); \
        return -1;                                             \
    }                                                         \
} while(0)
```

An example usage is `wbCheck(cudaMalloc(...))`. A similar macro can be developed while programming OpenCL code.

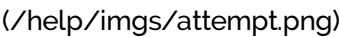
Plagiarism

Plagiarism will not be tolerated. The first offense will result in the two parties getting a 0 for the machine problem. Second offense results in a 0 for the course.

Grading

Grading is performed based on criteria that are specific for each lab.

For each attempt in the attempts tab, you'll see a *submit for grading* button.



GPU ProgrammingMachine Problems ▾Attempts ▾Peer Review ▾Grades ▾Help ▾Profile ▾

Grade

Submit Grade to Coursera

Updated:	a day ago
Total Score	50 out of 100 points
Coding Score:	50 out of 50 points
Coding Score on Coursera	50 out of 50 points
Peer Review Score:	0 out of 50 points

(/help/imgs/grade.png)

Designed for the Heterogeneous Parallel Programming Coursera Course.

Note: you cannot submit code for grading after the corresponding deadlines have passed.

While not required, the library used throughout the course can be downloaded from Github (<https://github.com/abduld/libwb>). The library does not depend on any external library (and should be cross platform), you can use `make` to generate the shared object file (further instructions are found on the Github page). Linking against the library would allow you to get similar behavior to the web interface (minus the imposed limitations). Once linked against the library, you can launch your program as follows:

```
./program -e <expected_output_file> -i <input_file_1>,<input_file_2> -o <output_file> -t <type>
```

6/29/2015 11:19 AM

In case of issues or suggestions with the library, please report them through the issue tracker (<https://github.com/abduld/libwb/issues>) on Github.

Issues/Questions/Suggestions

In case of questions, please post them to the forums. For issues or suggestions, please use the issue tracker (<https://github.com/abduld/wb/issues>).

Deadline

The coding deadline is 12 days from now (2015-07-10T23:00:00Z) .

Note: you cannot submit code and answer questions for grading nor perform a peer review after the corresponding deadlines have passed.

Grading Rubric

100 out of the 100 possible points is reserved for code development.

Dataset

We provide the dataset (`/dataset/3/dataset.zip`) to allow you to develop the machine problem on your own system or to examine the data. This is purely optional however. Consult the tutorial (`/help`) for information about local development.