CE 368/468 – Lab 1
Programming Massively Parallel Processors with CUDA

Matrix Multiplication in CUDA

In this first assignment, you will work in teams to write a matrix multiplication code in CUDA. Team sizes, formation and team rules are all outlined in the syllabus. The rules governing tardiness and academic integrity are detailed in the syllabus as well. We intend to follow all these rules.

Install the CUDA SDK at your Wilkinson Lab account

1. Login to a machine at Wilkinson Lab (Tech M338). If you wish to work remotely (e.g., via ssh, PuTTY or FastX), the Wilkinson Lab hostnames are below. FastX is a particularly useful tool that gives you a graphical interface to the machines. Information on how to use FastX is at http://it.eecs.northwestern.edu/info/. Note that you must login to Northwestern's VPN in order to login Wilkinson lab machines.

alfred.ece.northwestern.edu	joker.ece.northwestern.edu
bane.ece.northwestern.edu	killercroc.ece.northwestern.edu
batgirl.ece.northwestern.edu	madhatter.ece.northwestern.edu
batman.ece.northwestern.edu	nightwing.ece.northwestern.edu
clayface.ece.northwestern.edu	poisonivy.ece.northwestern.edu
cobblepott.ece.northwestern.edu	ras.ece.northwestern.edu
freeze.ece.northwestern.edu	riddler.ece.northwestern.edu
gordon.ece.northwestern.edu	robin.ece.northwestern.edu
gotham.ece.northwestern.edu	scarecrow.ece.northwestern.edu
harley.ece.northwestern.edu	selina.ece.northwestern.edu
huntress.ece.northwestern.edu	twoface.ece.northwestern.edu
hush.ece.northwestern.edu	

2. Find which shell you are using. Issue the command:
   ```
   ps -p $$
   ```
   The right-most word of the last line tells you which shell you are using (typically one of `csh`, or `tcsh`, or `bash`, or `sh`)
3. Setup the CUDA environment. If you use `csh` or `tcsh`:
   ```
   source ~hardav/cuda-env.csh
   ```
   or if you use the `bash` or `sh` shell:
   ```
   . ~hardav/cuda-env.sh
   ```
4. Install SDK in your home directory
   ```
   cp -r ~hardav/cuda-11.0 ~/
   cd ~/cuda-11.0
   make
   ```

5. Test by finding out the characteristics of your GPU device
   ```
   cd ~/cuda-11.0/1_Utilities/deviceQuery
   ./deviceQuery
   ```
   The command above should return a list of the capabilities of the machine's GPU. The last line should be "`Result = PASS`".

Install the labs scaffold at your Wilkinson Lab account

6. Download the labs-scaffold tarball from Canvas into some directory in your Wilkinson Lab account. Let's call that directory CE468 in the command lines below.

7. cd to that directory and untar the labs-scaffold tarball:
   ```
   cd CE468
   tar xvf CE468-CUDA-Labs-Scaffold.tgz
   ```

Install lab1 at your Wilkinson Lab account

8. Go to the main labs directory:
   ```
   cd CE468-CUDA-Labs
   ```
9. Download the lab1 tarball from Canvas into the current directory (`CE468/CE468-CUDA-Labs`) in your Wilkinson Lab account.

10. Install the lab1 tarball and compile the lab sources:
    ```
    cd CE468-CUDA-Labs
    tar xvf CE468-CUDA-Lab1.tgz
    make
    ```

Complete the assignment

11. **Remember**: every time you login to the Wilkinson lab to program in CUDA you need to setup the CUDA environment. Always perform step #3 above.

12. **Assignment Task:**
    Edit the following two functions
    `MatrixMulOnDevice(...)` in the source file `matrixmul.cu`
    `MatrixMulKernel(...)` in the source file `matrixmul_kernel.cu`
    to implement matrix multiplication on the GPU device. **Do not change any other piece of the source code**. The size of the matrix is defined such that one thread block will be sufficient to compute the entire solution matrix.

    The source code you will be working on is at :
    `CE468-CUDA-Labs/labs/src/lab1`

    Compiling your code produces the executable:
    `CE468-CUDA-Labs/labs/bin/linux/release/lab1`

13. There are several modes of operation for the application.

    a. No arguments: the application will create two randomly initialized matrices to multiply. After the device multiplication is invoked, it will compute the correct solution matrix using the CPU, and compare that solution with the device-computed solution. If it matches (within a certain tolerance), it will print out "`Test PASSED`" to the screen before exiting.

   b. One argument: the application will use the random initialization to create the input matrices, and write the device-computed output to the file specified by the argument.

   c. Two arguments: the application will initialize the two input matrices with the values found in the files provided as arguments. No output is written to file.

   d. Three arguments: the application will read its inputs from the files provided by the first two arguments, and write its output to the file provided in the third.

   Note that if you wish to use the output of one run of the application as an input, you must delete the first line in the output file, which displays the accuracy of the values within the file. The value is not relevant for this application.

14. Hand in your solution:

    a. Create a tarball of your solution, which includes the contents of the `lab1` source folder provided, with all the changes and additions you made to the source files. Please make sure you do a `make clobber` before submitting; don't include object code or executables, as these are typically large files:

    ```
    cd CE468-CUDA-Labs/labs/src/lab1
    make clobber
    tar cvfz CE468-lab1-YourNames.tgz *
    ```

       where you should replace `YourNames` with the **concatenated full names of the team members** (this simplifies logistics on our end).

    b. Submit your solution tarball via Canvas as a group. To submit as a group, first select an empty Lab Group on Canvas at People → Groups and add all team members (2 or 3) in the group. Then submit your solution on behalf of the entire group. A submission from any group member counts as a submission for the entire group, and all group members will receive the same grade and comments for it. If you submit multiple times, only the latest submission will be graded.

Good luck!