CS 239 programming exercise 1

Department of Computer Science
College of Engineering
University of the Philippines Diliman

1 Objectives

Review and demonstration of the following:

- Some key concepts in authoring massively parallel programs running in a heterogeneous computer system.
- Basics of CUDA and scalable computing.
- Basics of algorithmic parallel thinking.

The objectives are obtained through the use and application of well-known concepts in parallel computing to the (1) GPU and (2) your choice(s) of parameters, e.g. size or dimension of your input(s), blocks in the grid, block of threads. The *design, execution, trade-offs,* and *analyses* of your experiments based on (1) and (2) are the **main highlights** of the exercise and report.

2 Exercises

Main references I suggest is [Kirk and Hwu, 2012] and [NVIDIA, 2025], but you can consult other sources as long as you cite precisely which part(s) is your own work, and which part(s) are not (including where you take those part(s) that are not yours). Perform the citations in the short report to be submitted.

For this exercise, main references are Chapters 3 and 4 of [Kirk and Hwu, 2012] or their corresponding section(s) in [NVIDIA, 2025].

1. Begin your code by querying the properties of the device(s), i.e. GPU(s), that are installed in the system you are using. You can use the `cudaDeviceProp` type, then loop and print all device properties. See e.g. Chapter 4 of [Kirk and Hwu, 2012] or the corresponding section in [NVIDIA, 2025].

2. A matrix addition takes two input matrices `B` and `C` and produces one output matrix `A`. Each element of the output matrix `A` is the sum of the corresponding elements of the input matrices `B` and `C`, that is, `A[i][j] = B[i][j] + C[i][j]`. For simplicity, we will only handle square matrices of which the elements are single-precision floating point numbers. Values for each matrix element of `B` and `C` is a (pseudo) randomly generated number in the host (i.e. CPU) in the range [0,100]. Write a matrix addition kernel and the associated host function that can be called with four parameters: pointer to the output matrix, pointer to the first input matrix, pointer to the second input matrix, and the number of elements in each dimension. Use the following instructions:

- Write the host function by allocating memory for the input and output matrices, transferring input data to device, launch the kernel, transferring the output data to host, and freeing the device memory for the input and output data. Leave the execution configuration parameters open for this step. Make sure you make use of the values obtained from using `cudaDeviceProp` to make your execution parameters more reasonable.
- Write a kernel function with name `kernel_1t1e` that has each thread producing one output matrix element. Fill in the execution configuration parameters for the design.
- Write a kernel function with name `kernel_1t1r` that has each thread producing one output matrix row. Fill in the execution configuration parameters for the design.
- Write a kernel function with name `kernel_1t1c` that has each thread producing one output matrix column. Fill in the execution configuration parameters for the design.
- Analyze the pros and cons of each of the preceding kernels you designed. Compare and contrast each of your kernel with your other kernels, e.g. what we learned in class about warps, divergence. Also analyze your execution parameters based on the properties of your device given by `cudaDeviceProp`, e.g. optimality of your block and grid dimensions. You can also create tables showing and comparing the averaged run time (over, say, 10 runs) of each of your kernels for different matrix dimensions. See report section below for further details.

3 Report

Precisely cite in your report which parts of your work are yours, and which parts are not, whether the part(s) are in code form or design form. Remember that despite the fact that most or the entire source codes for the exercise are well-known e.g. from CUDA and related resources, the *point of the exercise* is not to insult your programming skills or understanding of concepts; the point is to gain *insights* from the experiments, and NOT just report beautiful tables, numbers, charts and the like. Aside from almost certainly citing [Kirk and Hwu, 2012] and [NVIDIA, 2025] in your report, you can cite other sources e.g. articles, books, videos. Please typeset your report, including snippets of your code, using a recommended maximum of 4 pages and using `article` template of LaTeX e.g. `https://www.overleaf.com/latex/templates/a-simple-article-template/gdsdkccmjnxg`.

The recommended way to cite code in your report is using the `typewriter font` to make reading and identification easier. Email me the PDF file of your exercise report. The filename of your report for exercise number `X` as a PDF file should be

`lastname-exerX.pdf`

with **email subject** as `lastname-exerX`. In your exercise report please include at least your complete name, the class and section (e.g. from UVLe or CRS). Those who prefer offline, i.e. download and setup required but not constant Internet connection afterwards, you can download LaTeX for Linux, Mac, or Windows. Those who prefer online, i.e. no download and setup required but constant Internet connection is required, I suggest `https://www.overleaf.com`.

References

[Kirk and Hwu, 2012] Kirk, D. and Hwu, W. (2012). *Programming Massively Parallel Processors: A Hands-on Approach (2nd ed)*. Morgan Kaufmann.

[NVIDIA, 2025] NVIDIA (February 2025). CUDA C programming guide. `http://docs.nvidia.com/cuda/cuda-c-programming-guide`.