Tutorial № 1

JI Zhuoran, The University of Hong Kong 28/07/2018

This tutorial provides a gentle introduction to CUDA programming with several "hello world" code. The main objectives in this tutorial are:

- knowing how to compile an project consists of host codes to be executed on the CPU, and kernel codes to be executed on the GPU
- writing a simple code to copy data between the device memory and the host memory
- learning how to do error-checking for kernel code

Problem 1

In this problem, you will learn the basic structure of a kernel function and how to compile and execute a kernel program:

1. Read through the **prac1a.cu** source file and the **Makefile** file to understand how a kernel code is compiled.
2. Compile the textbfprac1a.cu by the following command:

   make prac1a

   and then execute the resulting executable file.

Problem 2

In this problem, you will learning how to copy the data between devices and host.

1. Read section 3.1.1 and section 3.1.2 in CUDA C Programming Guide, understand the examples.
2. Read through the **prac1b.cu**.
3. Replace the comments with your code to copy the input into the devices before launching the kernel and copy the result back after the kernel code finished.
4. compile and check the result

Notice that:

- We allocate three arrays on the device using **cudaMalloc()**:two arrays, **dev_a** and **dev_b**, to hold inputs, and one array, **dev_c**, to hold the result

- Because we are environmentally conscientious coders,we clean up after ourselves with **cudaFree()**
- Using **cudaMemcpy()**,we copy the input data to the device with thep arameter **cudaMemcpyHostToDevice** and copy the result data back to the host with **cudaMemcpyDeviceToHost**

Problem 3

Always check the result codes returned by CUDA API functions to ensure you are getting cudaSuccess. If you are not, and you don't know why, include the information about the error in your question.

1. Read section 4.3 in CUDA C Programming Guide, understand the examples.
2. Read the **cudaError** section in the documentation.
3. Write a function in **prac1c.cu** to determine whether **cudaSuccess** is returned, otherwise, print the error information for debugging.
4. compile and check the result.

Problem 4

Sometimes, you may want to profile the execution time of a kernel function, in this problem, we are going to learn how to measure the execution time.

1. Read section 4.5 in CUDA C Programming Guide, understand the examples.
2. Complete the code in **prac1d.cu** to print the execution time of the kernel function

