GPU Programming 2019/20

Tutorial 11

In this tutorial we will extend the Cuda program that we wrote in the tutorial 7 in order to illustrate how one can execute two several kernels on the device using streams.

0.) Download the skeleton code and generate the build system using `cmake` (Under Linux and MacOS you can use a package manager (apt, brew, port,...) to install `cmake`. Then type `cmake ..` on the command line from the `./build` directory to generate the make file. On Windows `cmake ..` generates a Visual Studio solution).

1.) Measure the performance of the kernel function `addAddC()` for $n = k \cdot 1024$, where $k = 1, 8, 16, 17, 32, 33, 64$. Discuss the results taking into account the number of streaming multiprocessors in the device.

2.) Implement a new kernel function `addSubC()` analogous to `addAddC()`. Launch both kernels in sequence for several values of n and confirm that a kernel launch is a asynchronous command.

3.) Use *CUDA streams* in order to run both kernels concurrently. By considering the number of streaming multiprocessors in the device, try to predict for which values of n the concurrency of the kernels is no longer effective.

Please finish the implementation until next week (week of 20/01/2020).