Exercise #2: CUDA vector add

The goal of this second exercise is to understand how to create a simple CUDA application. In this case, we will implement a vector addition method using the GPU.

To enable CUDA development on todi, run the command “module load cudatoolkit”. This command will activate the nvcc compiler and the environment required to run CUDA applications.

Download or copy to your home directory the content of the “cuda” directory from the exercise course page.

There is a single file (vectorAdd.cu) that contains both the host and device code. To compile it, simply run:

nvcc vectorAdd.cu

An a.out file will be generated. If you try to run it, you will get an initialization error (“unable to set device”). You need to allocate some resources this time, in order to get access to a GPU. Simply run:

salloc –N 1

to book one node for 1 hour. You should now be able to run the a.out example just compiled through:

aprun a.out

Have a look at the source code (e.g., using *gedit*) to understand what this application is doing. By reusing the CscsTimer class previously introduced, measure the time taken to copy memory to/from the device and to run the kernels.

Use a larger set of vectors. How can you optimize the thread grid and thread block sizes used to run these kernels? How far can you push the vector sizes?

Try to optimize memory access by using pinned memory instead of pageable memory. Do you see any speedup?

Try to use asynchronous memory copy. What do you have to take in account when using asynchronous memory copy?

Refactor the source code into a quadratic equation solver. Given a, b and c, it should return x (you can cheat and always return a single solution even when two are available).