

## Parallel Programming – Lab 2

The purpose of this lab exercise is to familiarize yourself with the basics of CUDA programming. You will modify the programs discussed during the lecture. They all implement “scalar multiplication”, that is the multiplication of each element in a (very large) array of floating point numbers by a floating point constant. The four versions of the program are:

1. `scalarMultHost` : execute scalar multiplication on the host (CPU)
2. `scalarMultGbl` : use a very small kernel, consisting of the maximum number of threads on a single block.
3. `scalarMultBlocks` : use a very large kernel, consisting on the maximum number of threads that can be accommodated by the GPU.
4. `scalarMultBlockShared` : attempt a common optimization strategy by cooperatively loading data from global memory to shared memory before the computation.

You are required to implement the *running average* of each consecutive 100 elements in a matrix of 100,000,000 elements, that is

$$B[i] = (A[i] + A[i+1] + \dots + A[i+99]) / 100.0 \text{ for } i = 0 \text{ to } 99,999,900$$

Note that you will be required to run the kernel several times over the input array and that you will need to pad the array with data to fit it evenly over your last kernel, depending on the parameters that you choose.

You need to create four versions of the program as in the lecture, that is in the end you will need to implement four programs.

### Tasks:

1. In order to verify the correctness of your implementation you are required to output the **first 10,000** and the **last 10,000** elements of your resulting arrays.
2. Compare the results of all programs for consistency of output, either manually (use diff) or using a small program. *Note that small arithmetical errors might be introduced by the GPU.*
3. Log the GFLOP throughput of each implementation in the table below:

Program	GFLOPs
<code>runningAvgHost</code>	
<code>runningAvgGbl</code>	
<code>runningAvgBlock</code>	
<code>runningAvgShared</code>	