P. Hadjidoukas, C. Papadimitriou
ETH Zentrum, HG E5/G3
CH-8092 Zürich

Set 9 - GPUS II

Issued: May 8, 2017

Question 1: Diffusion on GPUs - Part II

In this exercise we will improve the 2D diffusion code on GPUs.

a) Improve the 2D diffusion kernel of the previous exercise by employing the shared memory of the GPU.

b) Write a kernel for `GetMoment()` to further reduce the memory transfer needed between GPU and CPU by calculating (at least) partial sums on the GPU.
Hint: An important part of the operation is a reduction.
While you can spend a full talk on how to optimize reductions[1], a simple GPU reduction will do here. It is called only every 100th update and will not dominate the execution time of our code. A simple reduction scheme within a block using the shared memory can be achieved by selecting only specific thread indices:

```
__shared__ float data[8];

...
if(threadIdx.x < 4)
    data[threadIdx.x] += data[threadIdx.x + 4];
__syncthreads();
if(threadIdx.x < 2)
    data[threadIdx.x] += data[threadIdx.x + 2];
__syncthreads();
if(threadIdx.x < 1)
    data[threadIdx.x] += data[threadIdx.x + 1];
```

c) Think about good choices for blocksPerGrid and threadsPerBlock of your kernels and explain your choice.

[1] M. Harris, Optimizing Parallel Reduction in CUDA, 2007,
`http://developer.download.nvidia.com/assets/cuda/files/reduction.pdf`