ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

High Performance Computing for Science and Engineering II

Spring semester 2015

M. Troyer
ETH Zürich, HIT G 31.8
CH-8093 Zürich

Set 8 - GPUs

Issued: May 4, 2015
Hand in: May 11, 2015

Question 1: GPU - Hello World

We will do a little vector addition to get started on the GPU.

a) Get access to a computer with NVIDIA CUDA. If your laptop/machine has a NVIDIA GPU you may install the CUDA toolkit on your own machine[1].

b) Complete the `Makefile` in `src/vector_addition` to compile the given code with the NVIDIA CUDA compiler.

For now the program will perform a vector addition on the CPU and do nothing on the GPU. It will then compare the results of the CPU and the GPU. As you do nothing on the GPU, running the program will cause many error messages.

c) Complete the empty CUDA kernel in `vector_addition.cu` and add the kernel call in the `main()` function to perform a vector addition on the GPU.

Run the program on a computer with an NVIDIA GPU. The error messages should be gone.

Note: Beware of out-of-bounds accesses.

Question 2: Diffusion on GPUs

In this exercise we will port the well known 2D diffusion using stencils on GPUs.

a) Implement a kernel for the propagation using only global memory. You may copy the serial CPU code `diffusion2d_serial.cpp` we provided in `src/diffusion_gpu` to `diffusion2d_cuda.cu` as a skeleton. The function `PropagateDensity()` should call the CUDA kernel instead of doing the compuation. Adapt the other functions of the class to minimize the data transfers between the CPU and the GPU.

Hint: `threadIdx` and `BlockIdx` offer more than one dimension.

b) Check your results by comparing to the serial code and report the achieved speed compared to the given OpenMP implementation `diffusion2d_openmp.cpp`.

c) Improve your kernel by employing the shared memory of the GPU.

Solution codes for a CUDA implementation of the 2D diffusion problem with and without shared memory can be found in `solution/diffusion2d_cuda.cu` and `solution/diffusion2d_cuda_shared.cu`, respectively. A speed comparison on Brutus (24 core AMD Opteron 6174, NVIDIA Tesla M2090 GF100), reveals a speedup of up to 3x (Tab. 1).

[1] `https://developer.nvidia.com/cuda-downloads`

Version	time	speed-up
OpenMP	263.0s	1.0x
CUDA	130.6s	2.0x
CUDA with shared mem	89.2s	2.9x

Table 1: Execution time of the 2D diffusion solution for system size $2^{11} \times 2^{11}$.

Summary

Summarize your answers, results and plots into a short PDF document. Furthermore, elucidate the main structure of the code and report possible code details that are relevant in terms of accuracy or performance. Send the PDF document and source code to your assigned teaching assistant.