University of Basel

Lecturer:
Prof. Dr. Florina M. Ciorba

Computer Lab. U1.001
Spiegelgasse 1,
CH-4051 Basel

Assistants:
Aurélien Cavelan, Ph.D.
Danilo Guerrera, M.Sc.
Ahmed Eleliemy, M.Sc.
Ali Mohammed, M.Sc.

http://informatik.unibas.ch/fs2018/lecture-high-performance-computing/

High Performance Computing (17164-02) **Spring Semester 2018**

Assignment 4: Introduction to GPU programming *(20 Points)*

Starting Date: April 19, 2018
Deadline: May 2, 2018 - 23:59:59

Objectives:
1. Understand the memory transfers between the host (CPU) and the device (GPU).
2. Understand how to launch 2D kernels.
3. Exploit the massive parallelism offered by GPUs.

All experiments should be performed on Scicore K80 GPU node.
Ensure that your optimizations do not affect the correctness of the results.
The delivered solution should be in one tar file.

Using the GPU node on Sciscore:
1. To login:
`ssh username@login.scicore.unibas.ch`
2. To compile: use the provided scripts
- Task 1 code (pure CUDA code), use *compile_t1.sh* file.
- Task 2 code (hybrid CUDA and MPI code), use *compile_t2.sh* file.
3. To run: use the provided scripts
- Task 1, use `sbatch job_t1.sbatch`
- Task 2, use `sbatch job_t2.sbatch`

1 Launching 2D kernels and exploiting the massive parallelism *(10 Points)*

Given the file *T1.cu* which contains a basic skeleton for developing a matrix multiplication program on GPU, show the right way of launching the matrix multiplication kernel in 2D fashion.
Attention: The target device (GPU) on which the code will be executed has a limit of 1024 threads per block.

a. Implement the five TODO parts in the given skeleton within the *main* function. *(5 Points)*

b. Implement the TODO part in the *matrix_mult_kernel* function. *(3 Points)*

c. Run and obtain the total program execution time for Matrix size $N \times N = 32 \times 32, 64 \times 64, 128 \times 128, 512 \times 512, and\ 1024 \times 1024$. *(2 Points)*

2 Experiencing parallelism using MPI + OpenMP + CUDA *(10 Points)*

Given the file *T2.c* which contains a basic skeleton for developing a matrix multiplication and matrix addition program, you are required to distribute the matrix addition among the MPI ranks and offload the matrix multiplication calculation to one GPU device that is managed by one MPI rank. All calculations (matrix addition and multiplication) should be performed in parallel.

a. Implement the missing parts in *T2.c* to distribute, calculate, and collect the results of the matrix addition among the MPI ranks. *(3 Points)*

b. Use OpenMP directives to parallelize the matrix addition within each MPI rank in *T2.c* . *(1 Point)*

c. Offload the matrix multiplication to the GPU and collect the results in *T2.c*. *(1 Point)*

d. Implement all the TODO parts in the CUDA source file *T2.cu* to invoke the matrix multiplication kernel asynchronously. *(3 Points)*

e. Run and obtain the total program execution time using 2 MPI ranks, 2 threads per rank and 1 GPU for Matrix size $N \times N = 1024 \times 1024$. *(2 Points)*