High Performance Computing
Exercise Sheet 8

HS 25
Dr. Douglas Potter

http://www.ics.uzh.ch/
Teaching Assistants:
Cesare Cozza, cesare.cozza@uzh.ch
Mark Eberlein, mark.eberlein@uzh.ch

Issued: 07.11.2025
Due: 14.11.2025

In the current exercise session, we will run first simple `OpenACC` and `CUDA` programs as a preparation for next sessions which will address both approaches more specifically. Instead of the computing cluster `Eiger` we use `Daint` available via daint.cscs.ch. You may want to edit your ssh config file to easily log into `Daint` similar to what we have done for `Eiger`.

Exercise 1 Compute PI - OpenACC

Log into Daint and clone the folder `exercise_session_8` from the course repository. It contains folder `openacc` which contains all the code necessary to start with this exercise.

In order to successfully compile and run the OpenACC codes on daint, few modules need to be loaded. First use `module load cray` to load the cray module, then load the `nvhpc` module. To compile OpenACC code use the following compiler flags:

```
nvc -O3 -acc -gpu=cc90
```

which specifies the correct GPU architecture. Remember to write a `sbatch` script to launch your program on one node using `srun`.

Commit: Modify the serial version of the code in `cpi_openacc.c` to become OpenACC parallel. Run the code and comment on the output. Has each iteration the same run time?

Exercise 2 Compute PI - CUDA

You will find the `cuda` code `cpi_cuda.cu` code in `exercise_session_8/cuda` folder of the course repository. Make sure the `nvhpc` module is loaded. Compile `cpi_cuda.cu` using the Nvidia Cuda Compiler `nvcc`. Each node requested on the GPU partitions contains one `NVIDIA Tesla H100` GPU. You should address the corresponding GPU architecture when compiling CUDA codes, by adjusting the nvcc flag:

```
nvcc -arch=sm_90
```

- Run the code and compare the execution time with your results from exercise 1. Are the times of each iteration the same? Why (not)?

Exercise 3 CUDA - Blocks and Threads per Block

In this exercise, we will investigate how the CUDA code scales with different numbers of block and threads per block. Run the code from previous exercise for the following setups:

$$\texttt{NUM\_BLOCK} = \{60, 120, 180, 240, 300, 360, 420, 600\}$$
$$\texttt{NUM\_THREAD} = \{16, 32, 48, 64, 80, 96, 112, 128, 144, 160\},$$

and observe the elapsed time (you can adapt the `cpi_cuda.cu` code to directly explore the setups, or use a bash script). Comment on the dependence of time the code took to run with respect to the number of block and threads.

- How does the scaling behave?
- Can you explain the results you obtained?

BONUS: visualize the obtained time dependence (e.g. 2D colormesh plot).

Commit: Push your code (and bash script if you have made any) together with the results and comments about the runtime study.