Project for the PDC summer school 2023

August 2023

1 Description

In this project, the task is to work in groups of 1-3 students and parallelize the "energy storms" code by using OpenMP, MPI, and CUDA/HIP. In addition, the group must write a short report (maximum five pages in the standard latex template) where they describe the techniques they used and reason around possible future opportunities for parallelization.

The energy storms project was developed by Arturo Gonzalez-Escribano and Eduardo Rodriguez-Gutiez in Group Trasgo, Universidad de Valladolid (Spain) and a handout with a more detailed description of the software is provided as a separate document.

2 Formal requirements

For a passing grade, the group must do the following.

- Develop one version of the code parallelized with OpenMP.
- One version parallelized with MPI.
- One version running on a GPU, using either the HIP or CUDA programming language.
- Verify that the codes are correct and able to reproduce the results from the sequential baseline.
- Compare the different versions with regard to run time and parallelization strategy.
- Reason around possible future optimization strategies.
- Hand in a report by the 6th of September 23:59 AOE, detailing the work and providing a link to a repository with the code and clear instructions on how to compile and run it. The final report and a link to the code should be sent by email addressed to makarp@kth.se and markidis@kth.se.

The groups are encouraged to discuss and talk about different issues regarding the project, but are not allowed to share or help other groups with coding or writing the report. We put a particular emphasis on the coding part and expect everyone in the group to take an active part in writing your software.

3 Hints

- A natural way of going about parallelization for OpenMP and GPUs is to work from the bottom up, first doing inner loops in parallel and then working your way up.
- If using Dardel, please do not run on the login node, make sure to allocate resources with `salloc`/`sbatch` and run the software with `srun`.
- Consider where and how memory is allocated and accessed. This is the major difference between shared-memory programming (OpenMP), distributed memory (MPI), and managing memory between the host and device when programming accelerators (CUDA/HIP).
- If you use HIP rather than CUDA, please change the makefile accordingly. There is a file called energy_storms_hip.hip provided.
- The Makefile needs to be changed if using a system with compiler wrappers (such as Dardel), remember to change e.g. gcc to cc in this case.
- Remember to debug the code often and check for race conditions in the parallel versions. Compile the code in debug mode and run small test commands such as
```
./energy_storms_omp 35 test_files/test_01_a35_p5_w3 \
test_files/test_01_a35_p7_w2 test_files/test_01_a35_p8_w4
```
- Make sure that not all ranks or threads execute the same work, the work needs to be split between the different processes. The code needs to be *parallelized*, not simply the sequential baseline running on several processes at once.