Running Pencil-Code on Dardel GPUs

This document lists instructions to run pencil-code on the Dardel supercomputer (PDC Sweden). This set of instructions was prepared with enormous help from Matthias Rheinhardt (MR) and Touko Puro (TP).

1. We will assume that Pencil-Code was obtained using git, since the GPU version with svn is not supported as of now. Get the latest version of pencil-code and all submodules from git using:

```
git pull
git submodule update --init --remote
```

2. On Dardel, we load the following dependencies:

```
ml rocm
ml bison
ml flex
```

3. We also need cmake (version >3.21). However, at the time of writing this document, such a version of cmake was not available on Dardel via the module load function which was also compatible with the above dependencies. My workaround was to install cmake using conda. Similarly, I was using python installed using conda.
4. We prepare a run directory as usual and add the following line to Makefile.local:

```
GPU = gpu_astaroth
```

5. In cparam.local we specify the number of processors (ncpus) as the number of GPUs we want to run on. A suggestion from MR is to use approx 256^3 cells per GPU.
5. We then setup src in our run directory and compile using appropriate flags:

```
pc_setupsrc
pc_build -f compilers/Cray_MPI LDFLAGS+='-Wl,--no-relax' FFLAGS+=-O1
```

7. A sample job script is given below for which ncpus = 64.

```
#!/bin/sh

#SBATCH --account=naiss2025-3-69
```

```
#SBATCH --partition=gpu
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=8

#SBATCH --gpus-per-node=8      # allocates one gpu per MPI rank
#SBATCH --cpus-per-task=8      # eight cpus per task for multithreading
##SBATCH --exclusive           # avoids other jobs to run on the node
(s)
##SBATCH --mem=0               # requests all available node memory

#SBATCH --time=23:50:00
source src/.moduleinfo
export LD_LIBRARY_PATH=${CRAY_LD_LIBRARY_PATH}:$LD_LIBRARY_PATH
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
export OMP_MAX_ACTIVE_LEVELS=2
export OMP_PROC_BIND=close,spread
export OMP_WAIT_POLICY=PASSIVE
./start.csh
#pc_start -f hosts/lumi/lumi.csc.fi-Cray.conf
export MPICH_GPU_SUPPORT_ENABLED=1 # MPICH only
./run.csh
#pc_run -f hosts/lumi/lumi.csc.fi-Cray.conf
```

Tips:

1. Keep the cadence of the diagnostic output low (or, equivalently, keep it1 in run.in high) since the diagnostic output is calculated on CPUs. The speed of many of my runs in the beginning was limited due to this being less optimal.