GPU Programming with Directives Exercise
Approximating Pi with OpenACC

1 Introduction

This exercise involves redoing the `pi` example with OpenACC instead of OpenMP.

2 Compiling and Running

The code is contained in `openacc_pi.tar`. The tar file can be fetched from GitHub by cloning the course repository with the following command:

```
git clone ...
```

Alternatively, the file can be found on ARCHER2 and copied into your `/work/` directory with the command:

```
cp ...
```

Unpack the file with the command: `tar -xvf pi.tar`. There is only a Fortran version of this exercise because offloading using OpenACC directives on ARCHER2 is only supported by the Cray Fortran compiler. Therefore, the following modules must be loaded prior to compiling the code:

```
module load PrgEnv-cray
module load rocm
module load craype-accel-amd-gfx90a
module load craype-x86-milan
```

Makefiles are provided to assist with compilation. In order to compile the code, simply run `make`, and to remove the generated object files and executable, run `make clean`.

Slurm scripts are also provided and should be submitted with the command `sbatch jobscript.slurm`. Before the first submission, the script must be edited to include the correct budget code, which is the project code for the course, e.g. `ta123`.

3 Offloading

Begin by compiling and running the code. Initially, it will only run on the CPU since the directives used to offload the code to the GPU have not been added yet.

Then, experiment with the following:

- offload the loop to the GPU with the `parallel` construct. Ensure that data is properly mapped to and from the GPU with clauses such as `copyin`, `copyout` and `copy`. Furthermore, remember that the code inside the `parallel` construct will be executed by every gang.
- parallelise the `do` loop at the varying levels using the following clauses: `loop worker`, `loop gang`, and `loop gang worker`. When applicable, experiment with the number of gangs using the `num_gang` clause.
- compare the performance of the `pi` exercise with OpenACC to the `pi` exercise with OpenMP. However, please note that the `pi` exercise with OpenMP was compiled with the AMD compiler and the `pi` exercise with OpenACC was compiled with the Cray compiler. Therefore, for a more fair comparison, the `pi` exercise with OpenMP should be recompiled with the Cray compiler and then rerun.