ASPP coursework 1

Please see Learn for submission deadlines.

Remember what you submit must be your own work. Anything not your own work which you have include should be correctly referenced and cited. You must not share this assessment's source code nor your solutions.

Academic integrity is an underlying principle of research and academic practice. All submitted work is expected to be your own. AI tools (e.g., ELM) should not be used for this assessment.

Please see further information and guidance from the School of Informatics: https://informatics.ed.ac.uk/taught-students/all-students/your-studies/academic-misconduct

Summary

Your goals:

1. Understand the core algorithm (`step()` in `wave_cpu.cpp`) and estimate the maximum performance acheiveable on a single A100 GPU available on the EIDF.
2. Using both OpenMP (in `wave_omp.cpp`) and CUDA (in `wave_cuda.cu`) adapt the CPU-only code and make it run correctly and efficiently on a single GPU.
3. Describe this in a brief report.

As part of marking, your code will be compiled and run automatically. It is essential you pay careful attention to the instructions for submission.

Problem

Here we are solving the wave equation in 3D, using the simplest method: an explicit, second-order in both space and time finite difference code. The model has a variable speed of sound and a simple boundary damping to reduce reflections in the x and y directions.

When run, the code will produce three (one each for CPU, OpenMP, CUDA) output files in HDF5 format which can be read by VTK, or tools based on it such as ParaView, and visualised. It will also produce timing and performance information to standard output.

Set up

Clone this repository on the EIDF (please see Learn for the path), remembering to use your PVC directory.

Configure with CMake. Note: for marking, we will use a build mode of `Release`, which is the default, but you may wish to include basic debug info for testing while doing your development.

```
cmake -S src -B build-dev -DCMAKE_BUILD_TYPE=RelWithDebInfo
```

Compile

```
cmake --build build-dev
```

The unmodified code will run on the ASPP VM (as it doesn't use the GPUs!) but once you start work, you will need to use the compute nodes. We have provided very basic job descriptions for testing and profiling - you should try parameters other than the defaults as we will use others to mark.

The application first runs a CPU reference simulation then the CUDA and OpenMP versions, comparing the results of the latter two to the former with some tolerance. It then reports on timing information.

Usage:

```
awave [-shape int,int,int] [-dx float] [-dt float] \
      [-nsteps int] [-out_period int] [output_base]
```

- `-shape` is the size of the simulation domain in grid points
- `-dx` grid spacing in metres
- `-dt` time step in seconds
- `-nsteps` number of time steps
- `-out_period` number of time steps between writing data
- `output_base` the base (i.e. no extension) for the output files

For your purposes changing the the shape should be sufficient (we may change the time and space resolution during marking, but this must be done with some care to ensure numerical stability, understanding of which is not relevant to this course).

Requirements

You need to prepare a brief report and adapt the code (only the files `wave_cuda.cu` and `wave_omp.cpp`) to use a single GPU with the best performance you can. We recommend using the `NVIDIA-A100-SXM4-40GB` GPU type (not the MIG GPU slices) once you are looking at performance as this will be used for testing.

Report

This should be in PDF format, with a maximum of three pages and a font size of at least 10pts. Please include these three sections below. Introduction and conclusions are not required. Remember to properly cite any sources. [Suggested length allocations in brackets.]

1. Estimate the best performance (for example in site updates per second, where a "site update" is performing one time step's integration for a single grid point) achievable on one A100 GPU (`NVIDIA-A100-SXM4-40GB`) on the EIDF. This should be based on your analysis of the code (the function `step` in `wave_cpu.cpp` for example) and the theoretical performance limits of the device as published by NVIDIA. [Half a page or less]
2. Explain the choices you made and how they give good performance, with reference to the hardware, programming models, and performance evidence gathered along the way to your solution. You should cover a range of problem sizes from 32–1000 along any dimension. Ensure you comment on how your observed performance compares to your estimate from part 1. [1-2 pages]
3. Comment on the differences between your two implementations, in terms of performance and ease of implementation, as well as any other points you consider signficant. Make a recommendation, with reasons, as to which programming model you would use if you could pick only one. [0.5-1 pages]

Code

Adapt the code to use the GPU efficiently with both OpenMP and CUDA, based on the material in the course.

a) Use OpenMP target offload in the file `wave_omp.cpp`

b) Use CUDA in the file `wave_cuda.cu`

Both versions will be assessed for correctness, performance, and clarity. It is important, therefore, that you test and profile this on the EIDF full A100s (`NVIDIA-A100-SXM4-40GB`) which we will use for marking.

You will be submitting *only* the files mentioned in (a) & (b) above, so please localise your changes to those files (you can always check with `git status`) as we will keep the rest of the application as is supplied in the repo. The supplied code has further advice in the source files.

Clarity: your modifications will be marked for good software engineering practice. Recall that you are explaining to an experienced programmer how the CPU and GPU will coordinate to solve a problem.

Correctness: the driver program compares both GPU results to those from the (unmodified) serial CPU version. We will compile and run this against a range of problem shapes, thus you should also in your testing. Ensure the program correctly manages any resources it uses.

Performance: we will use several problems of different sizes, taking the best performance run in each case. Sizes will be between 32 and 1000 inclusive along each dimension.

Submission

Please create a gzipped tar file which unpacks as follows:

```
examno/wave_cuda.cu
examno/wave_omp.cpp
examno/<sensible report name>.pdf
```

We have provided a script that will create this for you:

```
./make_submission.sh -h
Usage: ./make_submission.sh (-h |-e <exam number> -r <path/to/report>)
Create a submission-ready tarball as aspp-cw1-<exam number>.tar.gz
  -h print this message and exit without doing anything
  -e your exam number, the one that starts with a 'B' and has six numbers after
  -r your report in PDF format please

Both -e and -r options are required to produce a tarball.
```

Marking rubric

Marks for the report will be awarded according to the following scheme:

80% An excellent report, fully addressing all points above, and presenting convincing evidence/results. The report is written and presented to a professional standard.

70-80% A very good report, addressing all points to a good standard, and presenting evidence and results that supports the claims made, irrelevant text is minimal. The report is well written and presented.

60-70% A good report, addressing all the points with only minor omissions or confusion. Some irrelevancies, but not detracting from the report generally. Evidence and results support the points made, but perhaps not to the extent claimed. Writing and presentation generally good.

50-60% An OK report, addressing all of the points required at least partially but perhaps showing some faulty reasoning, minor confusions or longer passages of irrelevant material. Evidence/results are present but not convincing or contains minor inaccuracies. Writing and presentation are generally satisfactory but with some brief lapses.

40-50% A poor report with either significant omissions, failures in reasoning or that contains significant irrelevant material. Evidence or results have significant omissions or do not support the claims made. Writing and presentation have significant flaws.

<40% A weak effort which is incomplete or incomprehensible

Marks for the codes will be awarded according to the following scheme:

80% Correct, very well-designed code which, clearly and succintly, explains how the CPU and GPU coordinate to get performance, very good performance, likely using one or more advanced techniques.

70-80% Correct well-designed code with clear explanation of CPU/GPU use, and good performance.

60-70% Correct code, with generally clear GPU/CPU use and good performance. Likely doing the basics well or attempting the more advanced techniques with minor problems.

50-60% Code with minor errors, some oversights in explaining GPU/CPU use and some design and/or performance issues. May fail to properly consider an important performance issue.

40-50% Code with more serious errors, poor design and/or poor performance. No serious attempt to make clear the how GPU/CPU coordinate.

<40% Badly broken or incomplete code and/or very bad design. Description/comments missing or incomprehensible.