MPHYG002 Coursework 2: Parallelising Conway's Game of Life

Serial Solution

Build a C++ implementation of Conway's Game of Life. See for example https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life [5 Marks]

Marks Scheme:

- Valid code: 1 mark.
- Readable code: 1 marks.
- Appropriate unit tests: 1 marks.
- Well-structured project layout and build system: 1 mark.
- Use of version control: 1 mark.

Organising remote computation

Define an appropriate output file format, and use a script in a dynamic language (Python, MATLAB, R, Ruby, Perl etc) to create an mpeg video of your results. Use a script to deploy your serial code on a remote cluster with qsub. [5 marks]

Marks scheme:

- Output file works: 1 mark.
- Visualiser works: 1 mark.
- Automated deployment script with fabric or bash: 1 mark.
- Output file and script organisation support reproducible research: 1 mark.
- Valid job submission script: 1 mark.

Shared memory parallelism

Parallelise your code using OpenMP. Create a graph of performance versus core count on a machine with at least 12 cores. Comment on your findings. [5 Marks]

Marks scheme:

- Valid OpenMP parallelisation: 1 mark.
- Preprocessor usage so that code remains valid without OpenMP: 1 mark
- Script to organise job runs and results for performance measurement: 1 mark.
- Clear and meaningful scaling graph: 1 mark.
- Discussion: 1 mark

Distributed memory parallelism

Parallelise your code using MPI, with a 2-dimensional spatial decomposition scheme. Create a performance graph with at least 12 cores. Comment on your findings. [5 marks]

Marks scheme:

- Valid MPI parallelisation: 1 mark.
- Valid 2-dimensional decomposition scheme: 1 mark.
- Unit tests to exercise decomposition sheme: 1 mark.
- Performance graph, with script to organise measurement runs and clear graph: 1 mark
- Discussion: 1 mark

Accelerators

Accelerate your serial solution using at least one of CUDA Thrust, CUDA C or OpenCL. Experiment with different thread counts and performance optimisations. Measure speedup compared to the serial code, either on a cluster or using a graphics card on your own computer. Comment on your findings. (There is one mark available for a second accelerator solution.) [5 marks]

Marks scheme:

- Valid accelerator parallelisation: 1 mark
- Clean, readable, tested code: 1 mark
- Optimisation by exploring different thread counts and other configurable parameters: 1 mark
- Speedup analysis and discussion: 1 mark
- Second accelerator parallelisation: 1 mark.

Submission of the assignment

You should submit your solution via Moodle, including the entire code repository with version control history. Your solution should include a text report with a discussion of your project and a clear explanation of how to build and invoke your code – the marker will deduct marks if the code cannot easily be built and run as indicated. Your report should be broken down into sections that reflect those of the assignment, and should include your graphs, performance measurements, and discussion of any problems or interesting design decisions you encounter along the way. You should separate your solutions for each section with version control revision numbers, tags or branches and describe these in the report.