Assignment: MPI

In this assignment, use MPI running on up to 4 nodes (64 cores total) of BlueCrystal Phase 3 to make your Lattice Boltzmann code go as fast as possible

MPI

- You can re-use your optimised serial code from OpenMP, or start from the original in the GitHub repository
- Your submission will be made via the website and should include:
  1. A **four** page report in PDF form, including:
     a. Your name and user id or candidate number;
     b. A description of your MPI design, including an analysis of data layouts for an optimal distributed memory MPI version (arrangement for halo exchange etc);
     c. Description of your efforts to optimise parallel performance;
     d. A comparison of parallel performance vs. optimised serial code and show MPI scaling from 1 to 64 cores (4 nodes);
     e. Include your very best performance for all four test cases, averaged over some number of runs;
  2. The working code you used to generate the results in your report.
- Results must be within acceptable tolerances.

Rules for performance results

- Your timings must be for the total time around the main loop, ignoring overhead for printfs etc, i.e.:

```
/* start timing here */
for (ii=0; ii < params.maxIters; ii++) {
    timestep(params, cells, tmp_cells, obstacles);
    av_vels[ii] = av_velocity(params, cells, obstacles);
}
/* stop timing here */
```

- Results must be written out at the end (but don’t time this part!)
- Results must pass the results checking script

Submission requirements

- Your **report** which must be in a file called "**report.pdf**"
- Your **source code files**, e.g. "**d2q9-bgk.c**" etc
- Your **makefile**, called "**Makefile**"
- Your **job submission script**, called "**mpi_submit**"
- If you need to modify the default environment (e.g. you've used a different compiler, or you wish to set some environment variables for MPI), then you should create an **env.sh** file containing any **module load** or **export** commands that need to be run
- Your output filenames must remain unchanged from the example, i.e they must be **final_state.dat** and **av_vels.dat** (don't submit these)
- We must be able to reproduce the best runtimes in your report by compiling and running the code that you submit
- Don't zip these files up, instead submit them as separate files in SAFE

Testing your code

- We run all your submitted codes using an **auto testing script**
- To make this work you **must** to stick to the requirements for file names for the output
- Make sure you test your code against all four problems:
    1. **input_128x128.params**
    2. **input_128x256.params**
    3. **input_256x256.params**
    4. **input_1024x1024.params** ***NEW FOR MPI (see GitHub)***
- We will test your code on **4 nodes**
- Once you've got it working, if you are interested in going further, modify your code so that it can work on any number of nodes. Note that for some node counts the problem size will no longer exactly divide across the nodes, and your code will need to correctly handle this situation
- Use the test script to make sure your code produces correct results for each problem
- Example serial code timings (on one core of phase 3, compiled with -O3):
    - **(105s) input_128x128.params**
    - **(213s) input_128x256.params**
    - **(855s) input_256x256.params**
    - **(3477s) input_1024x1024.params**

Plagiarism checking

- We will check **all** submitted code for plagiarism using the MOSS online tool
  - MOSS is clever enough to ignore the example code you're all given
  - MOSS will spot if any of you have worked together or shared code, so **don't!**
- We'll also check all submitted reports using the TurnItIn tool, which will find if any of you have shared text
- So don't copy code or text from each other! You **will** get caught, and then ***both*** the copier and original provider will get a **0** for the whole assignment.

Getting good marks

- You'll get marks for:
  - A well written, comprehensive report
  - An MPI code that is fast and scales well

- Have fun writing your first distributed memory parallel programs!