ACSE6 Parallel Programming

Parallel Panicking
https://github.com/acse-2020/group-project-parallel-panicking

The Game of Life

Authors:
Iñigo Basterretxea Jacob
Nina Kahr
Orkan Sezer

Module coordinators:
Dr. A. Paluszny Rodríguez
Dr. S. Dargaville

February 17, 2021

Imperial College London

1 Code Structure & Optimisation

The code is structured in such a way that allows the user to interact with the main script "main.cpp" to either play the game making some adjustments (e.g. grid size, iterations, whether they want output), or produce time-performance analyses of different aspects of the code (e.g. grid size, number of cores, etc.). The "learn.h" "play.h" scripts interact with the "Grid" class.

With regards to optimising the game, each iteration of the game is performed in parallel by dividing the grid into a number of threads, so that each thread acts on a row of the matrix. As for parallelising the output from the game, it is not feasible to perform write to a single file from multiple threads, so the optimisation of output in the form of .dat and .bmp files involves storing previous iterations of the grid and then performing the output operations in parallel every number of iterations equal to the number of cores used. This of course constrains the performance of our code, as very long strings have to be stored in a vector at each iteration. Our "Grid" class contains a string property called "data", which is populated with new cell states as the parallel game iterations occur, if the user decides to produce some file output. When writing the file the whole string is used, which is much less time-consuming than iterating through a vector.

2 Serial vs. parallel execution time as a function of increasing grid size

As the grid size is increased, the ratio between the performance of the parallel and serial is stabilised at around 3 (see Figure 2). This ratio is well below the theoretical limit of 8 for 8 cores, perhaps due to running the game only for 1 iteration for each grid size, or likely because the time-measurements include the grid generation, which should be similar for both cases.

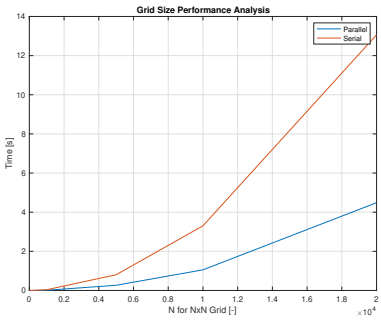

Figure 1: Execution time as a function of increasing grid size $N \times N$.

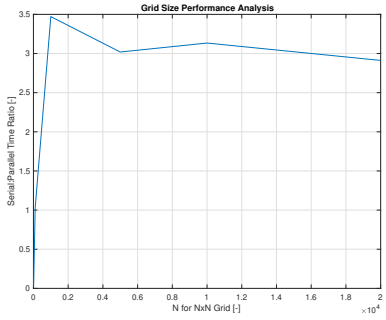

Figure 2: Ratio of serial to parallel execution time as a function of increasing grid size $N \times N$.

3 Performance as generations progress

Execution time increases in a linear manner with generations as show in Figure 3, whilst the ratio of execution time for the parallel and serial cases seems to be around 5.5 (see Figure 4) for 8 cores. This is less than the theoretical limit of 8.

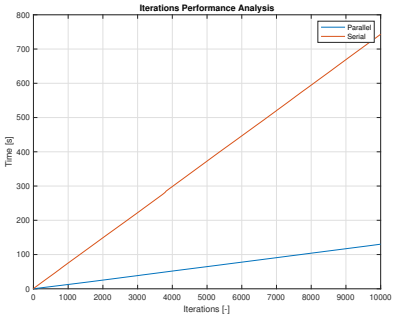

Figure 3: Execution time as a function of increasing iterations. 8 cores & 2000 rows/columns.

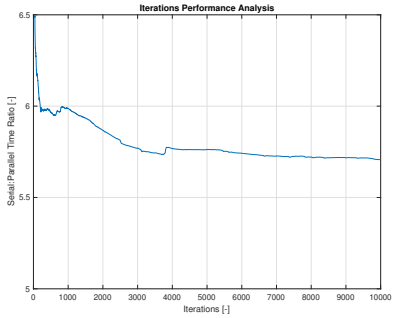

Figure 4: Ratio of serial to parallel execution time as a function of increasing iterations. 8 cores & 2000 rows/-columns.

4 Change of execution time with number of cores

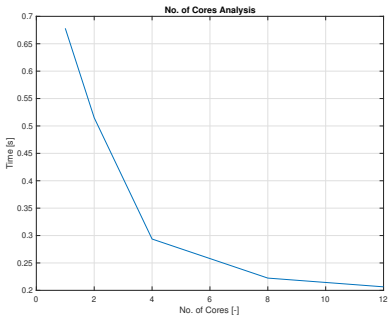

Figure 5: Execution time as a function of cores. 5000 rows/columns.

5 Performance of file writing

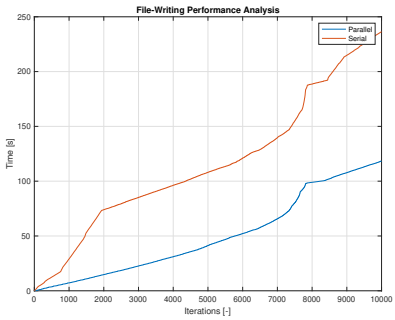

Figure 6: Comparison of .dat file-writing execution time with increasing iterations. 8 cores & 2000 rows/columns.

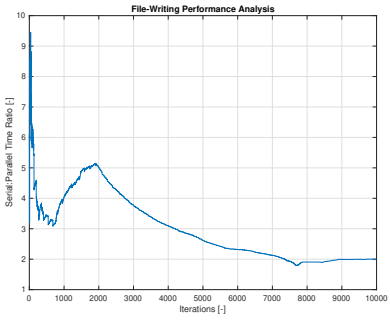

Figure 7: Ratio of serial to parallel file-writing execution time with increasing iterations. 8 cores & 2000 rows/-columns.

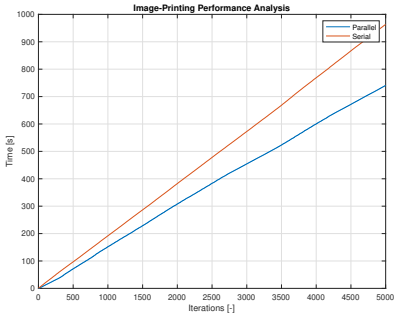

Figure 8: Comparison of execution time as a function of increasing iterations when printing images. 8 cores & 2000 rows/columns.

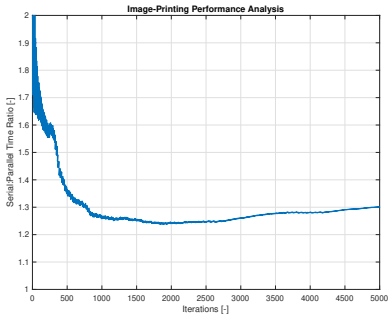

Figure 9: Ratio of serial-to-parallel image-printing execution time with increasing iterations. 8 cores & 2000 rows/columns.

A Division of Work & Specifications

- Nina: function to print .bmp files, parallelised game, doctests, gif, modified file-writing to write whole strings
- Orkan: sparse grid class, tests, documentation, parallelised game
- Iñigo: parallelised image-printing and file-writing, parallelised game, plots, designed user interface in main.cpp, learn.h, play.h
- Compiler(s) used: g++, OpenMP version: 5.0, operating system(s): macOS Catalina, number of logical cores of computer(s):12 (Iñigo), 4 (Nina and Orkan)