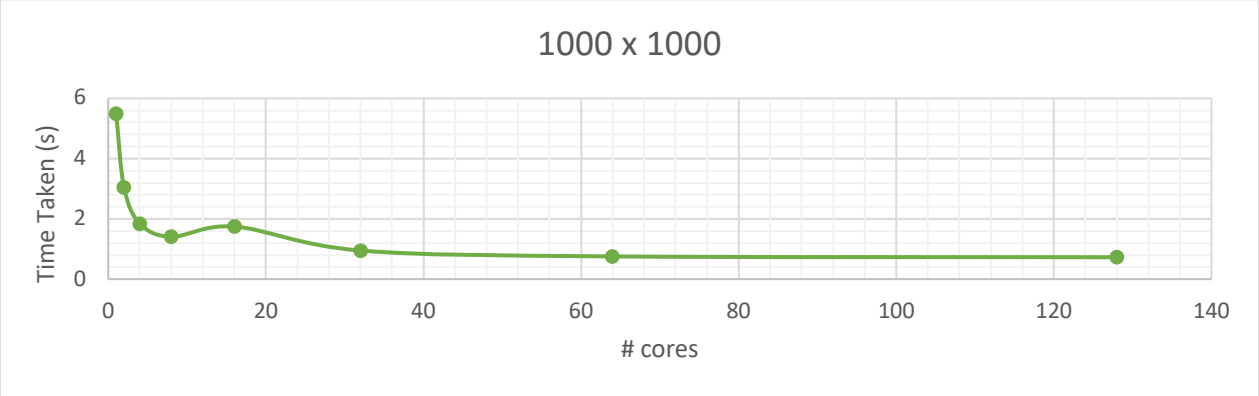
ACSE 6 – Report

Description

The GitHub repository contains the main (game.cpp) and post processing(post_pro.ipynb). In the beginning of game.cpp, the user can specify: the Boolean variable for periodic, integer variables for global domains size (imax & jmax) and an integer variable for the number of times games will be played (no_steps). When executed, files with iteration number and indices will be written out (example: it_3_row_1_col_0.txt).

Post_pro.ipynb is responsible for merging files for each iteration into a singular file according to indices such as iteration_3.txt. It also converts these singular files into images (.jpg format) for easier analysis and creates a GIF video for the whole game.

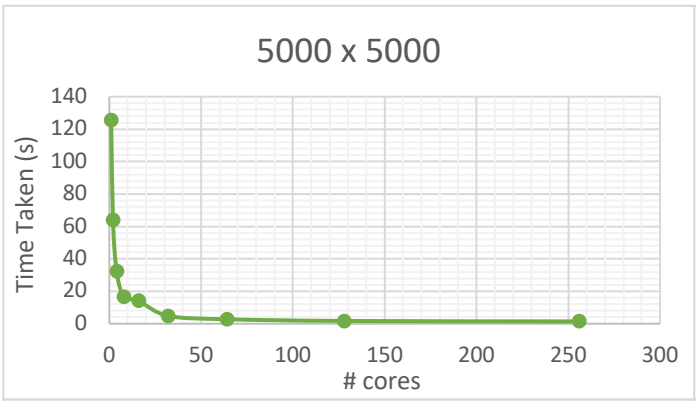
Analysis

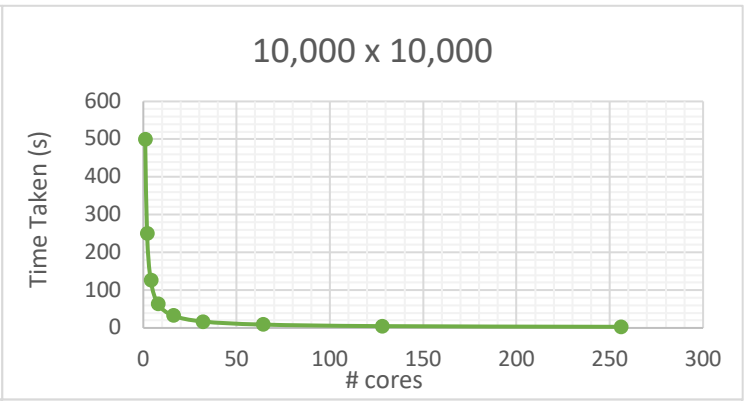


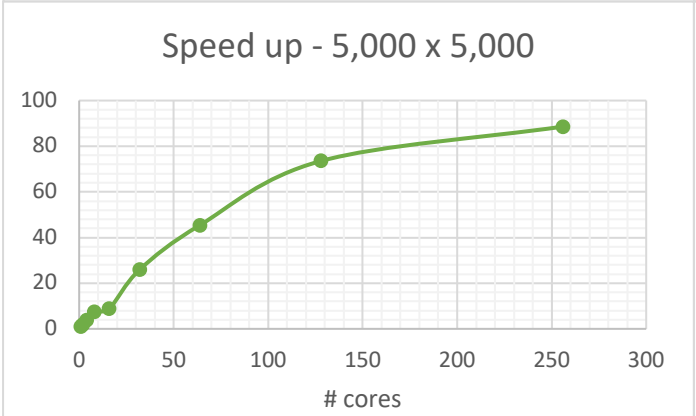
The HPC system was used to analyze the Efficiency/Speedup changes with varying the size of the problem and the number of cores used. my_script.pbs was used to send commands to the system for the run. The number of cores for all the tests usually increase exponential of 2 (1, 2, 4, 16, 32, 64, 128..). All tests have been iterated 100 times. The graphs are available in the HPC Folder to be viewed in more detail.

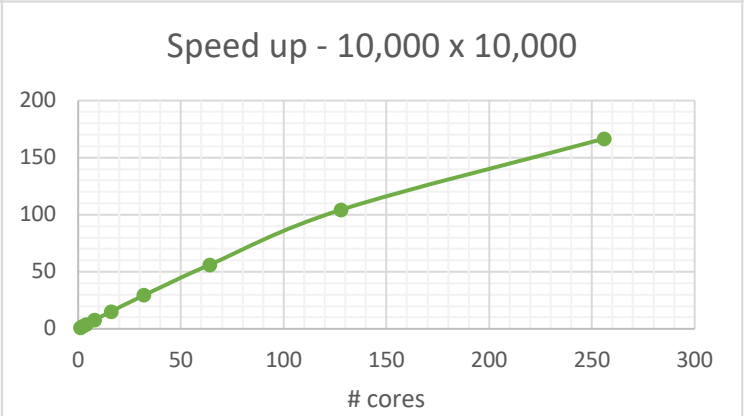
Generally, one would assume that more cores the better however the graph above demonstrates that for a domain of smaller size, after 32 cores the speed-up is not large enough to bother requesting more cores.

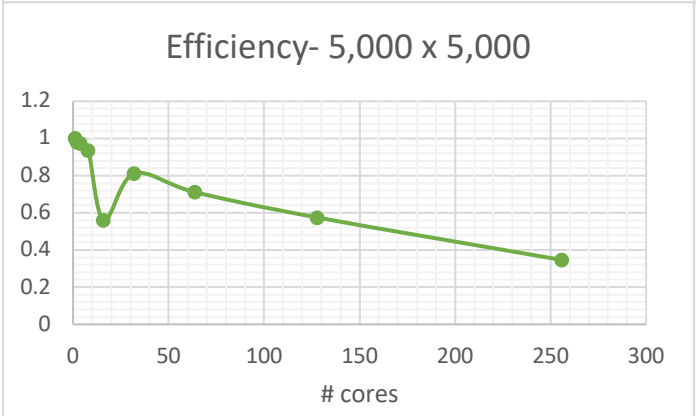
This experiment also pointed a flaw with the HPC system in regard to timing tests. As seen in the graph, 12 cores took more time than 8 cores. There are two main reasons for this. First, the system does not assign the same computer to the user for running thus comparing different runs becomes problematic (some runs can get slower computers). Secondly, the domain for the tests is randomized thus they are not running the exact same live/dead ratio.

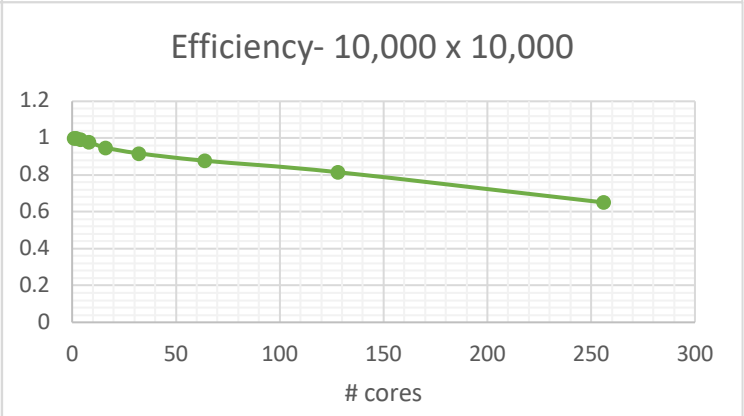












Speed-up has been calculated using the formula: $\frac{time\ taken\ in\ serial}{time\ taken\ on\ multiple\ cores}$

Efficiency has been calculated using the formula: $speed - up * \frac{1}{no\ of\ cores}$

According to the graphs above it can be seen that there is a nearly linear relationship for the speedup however the rate will slow down when it reaches an optimal number of processors. Such as when the cost of communication is more expensive then the computation.

The efficiency graphs indicate that for both sizes increase in core number will yield in a more efficient computation. Note that the dip here is observed for the same reasons discussed in the previous page.