Final Project Report: Game of Life

Dante Buhl

June 13, 2024

PCAM for the Game of Life

P The problem caan be partitioned into a smallest task size of updating a single cell in the grid.

C The communication needed for this problem involves getting the neighbors of a cell (or sub-grid) from other processors.

A Agglomeration involves assigning sub-grids of the original matrix to update rather than having a processor do one cell at a time. This reduces the communication needed by a lot.

M Depending on it being a 1D or 2D decomposition, tasks are assigned going downwards first and then left to right. So if its a 1D column decomposition then taks will be assigned left to right, if its a 2D decomposition, tasks will be assigned from the top down until the bottom of the matrix is reached, and then left to right, repeating the veritcal process. The algorithm will assign rows and column sizes evenly unless the partitioning size doesn't evenly divide the grid size (in which case the end tasks are assigned an extra row/column).

Instructions for Running the Code

a) There are 3 versions of this code; a serial version, a parallel 1D column decomposition (parallel_1d), and a parallel 2D decomposition (parallel_2d). All of the versions read in initial states in the same way. The executable is ran with an input file called IC or IC.dat (this is seen in the gol.slurm files in each subdirectory). In order to compile the code just type make in each directory (in the serial version, the makefile also runs the code). This will produce an executable called driver.exe. Note that the makefile should include the compiler flag -cpp and -DHB. This essentially runs the code on a version of MPI which is supported by Hummingbird. Then to run the code (on hummingbird) simply type sbatch gol.slurm. This will work as long as the IC file is formatted properly.

In order to format the IC file properly follow the guidelines here. The first line of the IC file should be 2 integers, each in 6 digit cells. For example, if you would like a 20 x 20 grid the first line should be, "____20____20" (latex does spacing weirdly, the underscores should be spaces). Then the following 20 lines after that should be the matrix transposed. That is the first column of the matrix should be the second line of the IC file. Then the second column on line 3 and so on. Note that an active cell is labeled as 1 and an unactive cell is labeled as 0. Also note that there should be no spaces between the cells. If this is done correctly, then the code will read the IC file perfectly and perform the rest of the procedure. If you would like to aadjust the number of timesteps, go into the code and change the variable "nt" to whatever you would like.

On output, the code produces 2 files, "gol.dat" and "params.dat". "Gol.dat" contains the matrix values at every update printed in the same format as the input except with spaces between the cells, and "params.dat" contain the gridsize and number of timesteps (plus the IC).

In order to visualize the output, open the matlab file gol.m and run this code within the same directory. It will read in the two output files and create a movie. In order to adjust the speed of this movie, change the command "movie(M,1,*)" to whatever you would like, where the number in place of the star will indicate the frames per second of the movie.
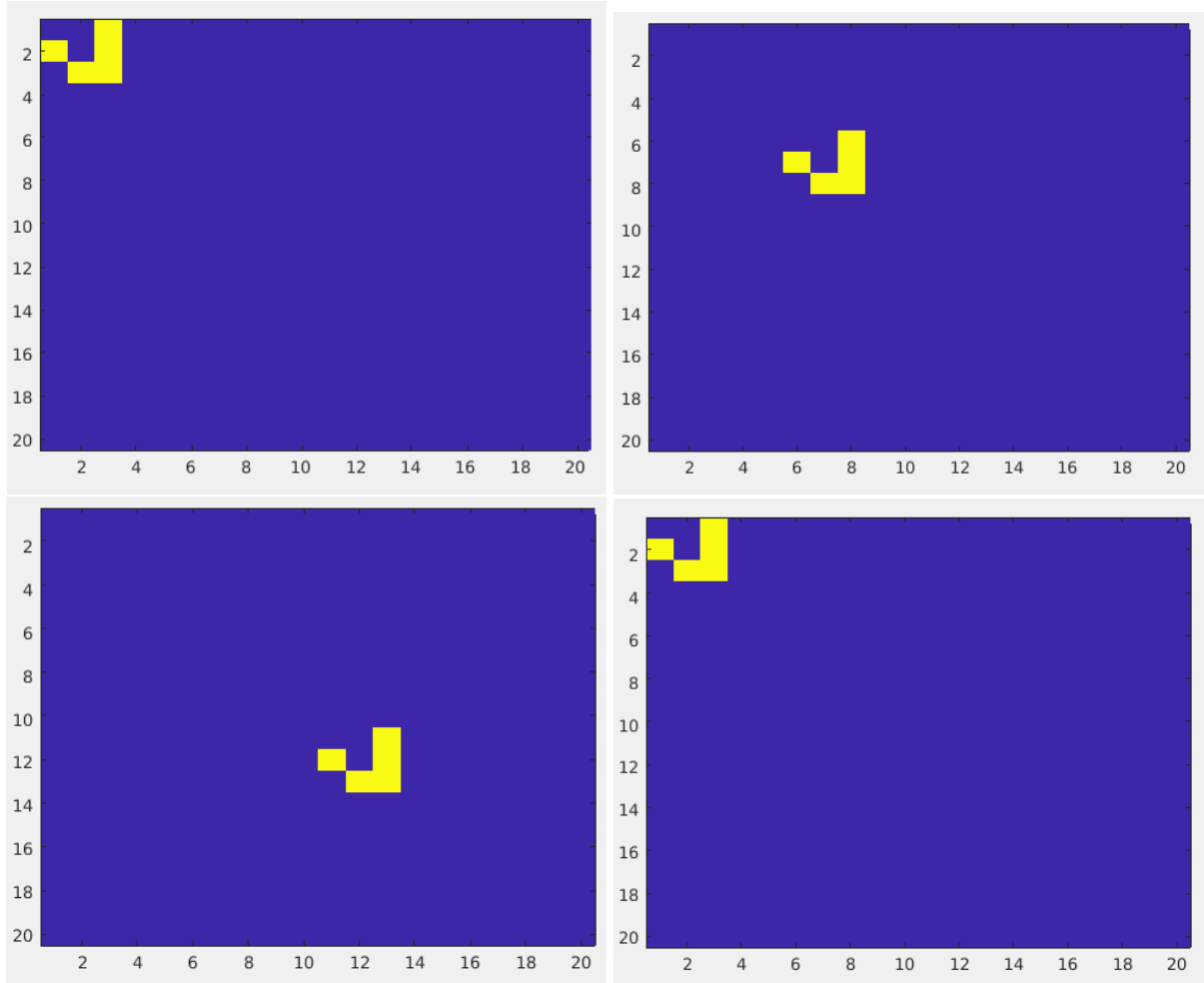
Figure 1: Output from the glider IC at t = 0 (top left), t = 20 (top right), t = 40 (bottom left), and t = 80 (bottom right).

That should be everything you need to know. Note that the 2D decomposition only works on an even number of processors, a perfect square number of processors, and a multiple of three number of processors. Information about how the decomposition works can be found in the subroutine "mpi_decomp_2d" within the "pgameoflife.f90" module file.

Glider Test on a 20x20 grid

a) Figure 1 shows the glider test at all 4 timesteps requested. As we can see at t = 80, the state has returned to to the initial state in the top left corner.

Performance Model

a)