MPI: Project - Game of Life

You are going to write an MPI parallel version of the Game of Life!

"Game of Life" simulation by John Conway

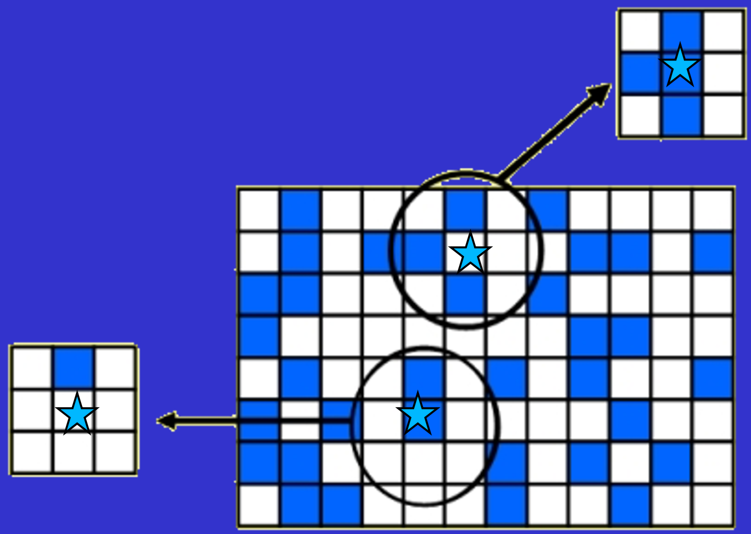
2D array of cells.

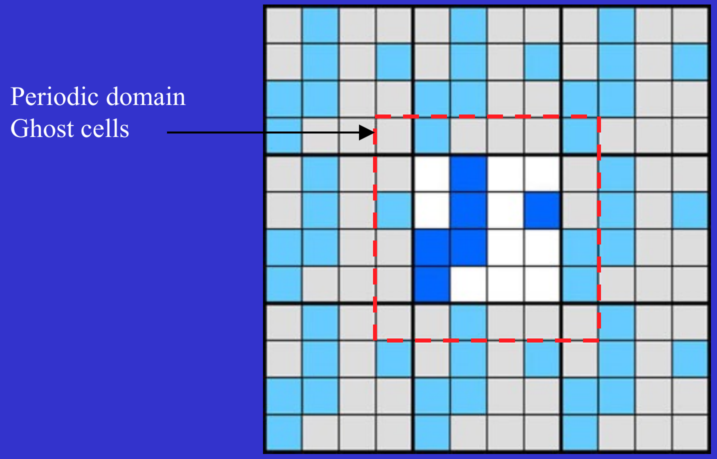
Each cell can have one of two possible states: alive or dead

Initialise with random states and then evolve according to rules:

- If exactly 3 neighbours are alive, cell will be alive (if already alive, remains alive; if was dead, becomes alive)
- If exactly 2 neighbours are alive, no change in cell status.
- All other cases, cell is dead (if was dead, remains dead; if was alive, becomes dead).

Assume periodic boundary conditions (don't forget you need diagonals)





MPI: Project - Game of Life

I suggest you proceed with the following steps:

1. Write an outline of the steps needed to compute the game

2. Write a serial version of the game. To debug, use a small grid (e.g. 4x4) and a single step. You will need this code to check the correctness of your parallel version and to compare speeds. Keep a track (print out) of the total number of alive cells -- this can be your simple metric for whether things are working or not. I suggest you use ghost cells for the boundary conditions.

3. Add the MPI initialisation and finalisation routines to your code and call it a parallel version. Doing this makes a code that runs the same program redundantly on all the processors you specify. Print the rank and the metric to make sure all is working.

Pass level

4. Do a domain decomposition of the problem to parallelise the problem truly. Since you are programming in Fortran, I recommend initially dividing by columns. This is because of the way arrays are contiguous in memory (i.e. column major in Fortran). Use ghost cells for interior boundaries. Debug with a partition of 2.

5. Mega version: Divide the domain the OTHER way. This means you will need to use derived data types, since these are not the natural ways for the language.

100% level

6. Giga version: Decompose the problem in both directions.

7. Tera version: Use a virtual topology. Use MPI_CART_CREATE, and use MPI_CART_SHIFT to create the ghost cells.

8. Peta version: Use parallel I/O to write out the solution grid and visualise it. There will be extra credit for any cute ways of visualizing ☺

MPI: Project - Game of Life

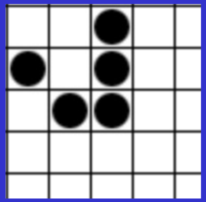
Report: due Wed June 8th at 11:59pm

Write a SHORT report that

1. Describes the PCAM process that you went through to design your code (can be BRIEF: <0.5 page)

2. Documents everything your code needs to run so that I can run it and check it on grape. Include a hardcopy of your code.

3. To prove that your code works in parallel, start with a 20x20 grid and initialise the grid with the configuration shown here (to the right) at the top left of your grid((and nothing elsewhere). Run the code for 4 steps and notice that this returns back to the same pattern but with everything shifted diagonally by 1 square. Now run the code for 80 steps. The pattern should return to where it started. Show the grid at steps 0,20,40,80 when the code is run on 4 processors.



4. Come up with a simple performance model and test it out. Remember, you have the communication time constants (t_s and t_w) from your latency program (earlier HW) and the computation time (t_c) can be calculated from your ones.f90 (earlier HW) too.

MPI: Project - Game of Life

To submit your project:

1. Make a directory wherever you are working called AM250_project_<your_lastname>

2. Copy ALL the necessary files to run your code(s), including a compiled binary executable, the batch jobfile if necessary, etc, to this directory and include a `README.TXT` that describes (very briefly) the commands necessary to compile and run your code, e.g.

   "On grape, do the following to compile and run my code:

   To change the grid size and the number of processors, edit `my_fortran.f90` and change `ngrid` and `nprocs`

   `mpif90 –o my_exec my_fortran.f90`

   `qsub my_batchfile.cmd` or `mpirun –np 4 my_exec`

   The output data will be in `my_datafile`

   To look at the data, type `more my_datafile`"

3. Copy your report in PDF form into your user name directory too. Call it `report_username.pdf`

4. Tar the directory into a single tarfile named by your last name:

   ```
   cd ..   (if you start from your code directory)

   tar cvf <your_lastname>.tar AM250_project_<your_lastname>
   ```

5. Submit the tarfile to the Canvas page, AND SUBMIT THE REPORT AS A SEPARATE FILE TOO.