INSTRUCTIONS

To compile the code,

make

To run the code,

./charmrun +p<number of processes> ./life2d <input file name> <# of generations> <X_limit> <Y_limit> <num_chares_X> <num_chares_Y>

e.g. ./charmrun +p1 ./life2d final.500x500.data 500 500 500 8 8

The output file name is <input file name>.csv, e.g., final.500x500.data.csv

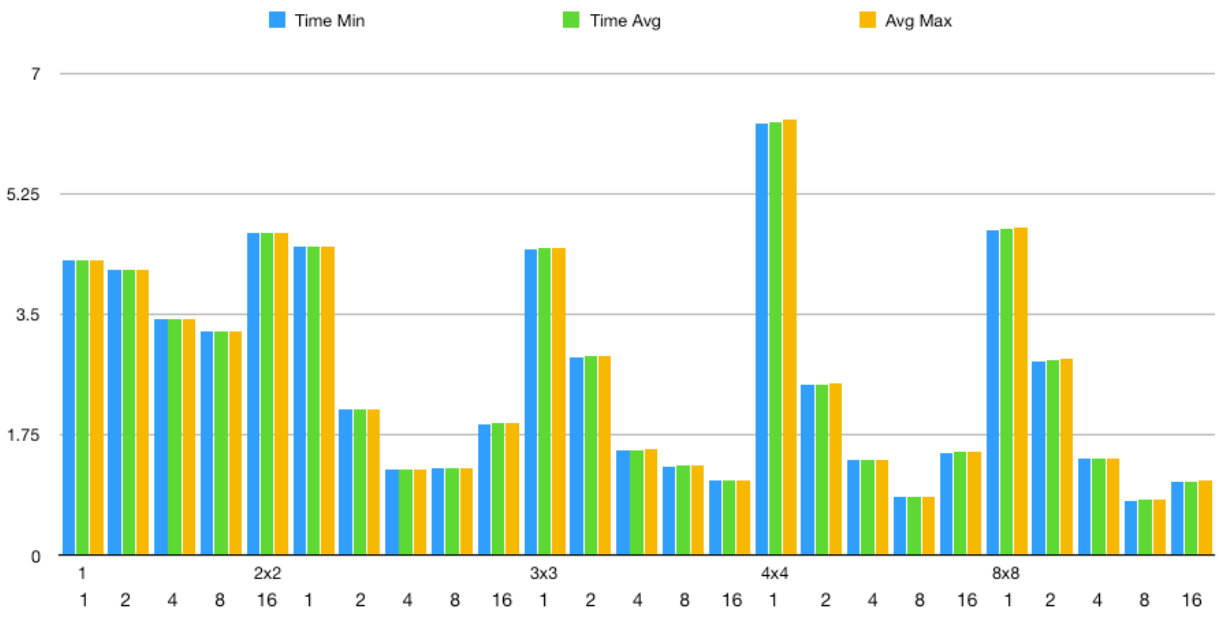
EXPERIMENTS:

Input: final.500*500.data, 500 steps, 500*500 grid

Metrics: Min, Avg and Max times (in seconds) are calculated by reduction operations (contribute) over the individual wall time measurements over generations on different chares.

Chares	Processes	Time Min	Time Avg	Avg Max
1	1	4.287459	4.287459	4.287459
	2	4.146701	4.146701	4.146701
	4	3.428358	3.428358	3.428358
	8	3.251046	3.251046	3.251046
	16	4.671548	4.671548	4.671548
2x2	1	4.478975	4.483668	4.490534
	2	2.123467	2.125924	2.130335
	4	1.240391	1.242033	1.244886
	8	1.263650	1.265565	1.269397
	16	1.910930	1.914057	1.918417
3x3	1	4.451333	4.456451	4.466930
	2	2.884559	2.888040	2.895772
	4	1.532395	1.534535	1.539327
	8	1.296423	1.298444	1.301683
	16	1.080508	1.082954	1.086776

4x4	1	6.272731	6.278323	6.324003
	2	2.480381	2.484423	2.493211
	4	1.377681	1.380777	1.385746
	8	0.845913	0.847650	0.851393
	16	1.492994	1.501624	1.514572
8x8	1	4.727116	4.731527	4.751800
	2	2.817819	2.832761	2.862166
	4	1.401470	1.407150	1.415290
	8	0.798605	0.802966	0.807530
	16	1.070604	1.075660	1.083010



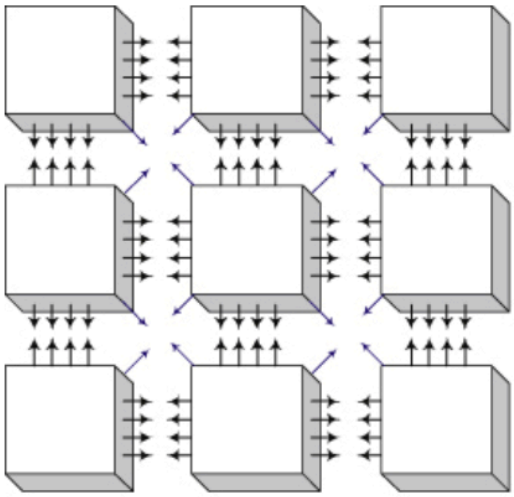
From the table, we can find the best performance is with 8x8 chares and 8 processes. In general 8 processes is an ideal setting from this experiment. Less processes takes more time due to the parallel potential is not fully used. More processes also slow down the speed a little because of the overhead and data locality/transmission.

Good performance goes with more chares. This is because the original matrix is big enough (500x500) which has a good potential to be split into chares. Each of them has still big computation burden so that the overhead and communication cost between chares is not a big concern. Moreover, with more chares, the program is more fine-grained, burden on each processes are more balanced so that we can see a better result with more chares but with same number of processes.

One chare doesn't make sense to parallel because a single chare cannot be spawned to multiple processes. So we can see the time remains relative constant no matter how many processes are available.

2D DECOMPOSITION:

Data distribution: Initially the universal grid is read by chare (0,0), the top-left chare and split into blocks along the rows and columns. The top-left keeps its own block and send others to other chares according to the block row and column id. After the generation ends, data is sent back to chare (0,0) to reconstruct a universal then output.



Global grids are divided into sqrt(nRows*nCols/mpiSize) sub blocks(grids). Each middle block has left, right, bottom, top neighbors to transfer an edge, and have four corner neighbors, bottom_left, bottom_right, top_left, top_right, to transfer only a single corner token. While each edge block and corner blocks are more special. These bound blocks doesn't transfer several edges or corners depending on their location. They follows the arrows to transfer data shown above. In each iteration, chares send edge and corner ghosts first and then receive them. Directions are arguments to help the receiver block recognize the source of the ghost data. Game of Life computation begins once the ghosts are fully updated.