Parallel Computing/Programming Assignment #3: Massively Parallel Game of Life

Christopher D. Carothers
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street
Troy, New York U.S.A. 12180-3590
Email: chrisc@cs.rpi.edu

February 26, 2016

DUE DATE: Noon, Friday, March 11th, 2016

1 Assignment Description

For this assignment, your are use the parallel Conway's Game of Life program you wrote in assignment 2 and port it to the Blue Gene/Q and conduct a strong scaling performance study.

1.1 Review: Basic Rules

The Game of Life is an example of a Cellular Automata where universe is a two-dimensional orthogonal grid of square cells (with WRAP AROUND FOR THIS ASSIGNMENT), each of which is in one of two possible states, *ALIVE* or *DEAD*. Every cell interacts with its eight neighbors, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur at each and every cell:

- Any live cell with fewer than two live neighbors dies, as if caused by under-population.
- Any live cell with two or three live neighbors lives on to the next generation.
- Any live cell with more than three live neighbors dies, as if by over-population.
- Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

The initial pattern constitutes the seed of the system using a Random Number Generator (RNG) at time 0. Use the RNG provided in the assignment 2 code template (yes, a new template!). The first generation is created by applying the above rules to every cell in the seedbirths and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a "tick" The rules continue to be applied repeatedly to create further generations.

Note, a "tick" starts with $Cell(0,0)$ and ends with $Cell(N-1,N-1)$ in the serial case. When we get to parallel things will get much more interesting! (see below).

1.2 Adding Additional Randomness

Same as assignment 2.

1.3 Parallelization Approach

First, this time to ensure that there is more true "randomness" among the cells, each row will have it's own RNG seed set. According to the template, every MPI rank will initialize 16,384 RNG streams. These streams are about 2^{70} calls apart.

SO, EACH ROW WILL HAVE IT'S OWN RNG STREAM!

All other parts are the same as in assignment 2.

1.4 Experiments

You will conduct a strong scaling experiment using "AMOS", the CCI Blue Gene/Q system. Here, you will have 64 MPI rank per Blue Gene/Q node.

- Run a 8192x8192 cell universe using 64 MPI ranks (1 BG/Q nodes) for 100 ticks with a threshold of 0%.
- Run a 8192x8192 cell universe using 128 MPI ranks (2 BG/Q nodes) for 100 ticks with a threshold of 0%.
- Run a 8192x8192 cell universe using 256 MPI ranks (4 BG/Q nodes) for 100 ticks with a threshold of 0%.
- Run a 8192x8192 cell universe using 512 MPI ranks (8 BG/Q nodes) for 100 ticks with a threshold of 0%.
- Run a 8192x8192 cell universe using 1024 MPI ranks (16 BG/Q nodes) for 100 ticks with a threshold of 0%.
- Run a 8192x8192 cell universe using 2048 MPI ranks (32 BG/Q nodes) for 100 ticks with a threshold of 0%.
- Run a 8192x8192 cell universe using 4096 MPI ranks (64 BG/Q nodes) for 100 ticks with a threshold of 0%.
- Run a 8192x8192 cell universe using 8192 MPI ranks (128 BG/Q nodes) for 100 ticks with a threshold of 0%.

Re-perform the above set of experiments for thresholds of 25%, 50% and 75%.

Next, plot execution time (Y-axis) as a function of the number of MPI ranks (X-axis) grouping each threshold as a graph line. Do you observe any performance differences as the threshold changes? Also, in your report, discuss the max speedup relative to the execution time for the 64 MPI rank case you obtain as well as at what point is your parallel efficiency the greatest (i.e., speedup / MPI ranks for each case and which case yields the highest value).

Next, for a threshold of 50% run the following series of experiments where you have only 32 MPI ranks per node.

- Run a 8192x8192 cell universe using 32 MPI ranks (1 BG/Q nodes) for 100 ticks with a threshold of 50%.
- Run a 8192x8192 cell universe using 64 MPI ranks (2 BG/Q nodes) for 100 ticks with a threshold of 50%.
- Run a 8192x8192 cell universe using 128 MPI ranks (4 BG/Q nodes) for 100 ticks with a threshold of 50%.
- Run a 8192x8192 cell universe using 256 MPI ranks (8 BG/Q nodes) for 100 ticks with a threshold of 50%.
- Run a 8192x8192 cell universe using 512 MPI ranks (16 BG/Q nodes) for 100 ticks with a threshold of 50%.
- Run a 8192x8192 cell universe using 1024 MPI ranks (32 BG/Q nodes) for 100 ticks with a threshold of 50%.
- Run a 8192x8192 cell universe using 2048 MPI ranks (64 BG/Q nodes) for 100 ticks with a threshold of 50%.
- Run a 8192x8192 cell universe using 4096 MPI ranks (128 BG/Q nodes) for 100 ticks with a threshold of 50%.

Plot the execution time (Y-axis) as a function of the number of BG/Q nodes used (NOT MPI ranks as you did before) for the 50% threshold data. Plot a graph line for the 32 MPI ranks per BG/Q node data and a separate graph line for the 64 MPI ranks per BG/Q node data obtained in the previous experiment. Do you observe any performance change when using only 32 MPI ranks per Blue Gene/Q node? Describe these changes in performance and provide your reasoning as to why the performance might have changed when going from 64 MPI ranks per BG/Q to 32 MPI ranks per BG/Q node.

2 HAND-IN INSTRUCTIONS

Keep/put your assignment C code and write-up with graphs in PDF format on the class server, `kratos.cs.rpi.edu` under the `assignment3` sub-directory on your Linux account.

We will use an automated script to collect the assignments. So it is extremely important you place your solution in the right directory name, `assignment3`.

PLEASE PUT YOUR ASSIGNMENT UNDER THE `assignment3` SUB-DIRECTORY OR ELSE IT WILL NOT BE COLLECTED CORRECTLY AND COULD RESULT IN SUBSTANTIAL POINT LOSS OR NOT GRADED AT ALL.

ADDITIONALLY, KEEP A BACKUP COPY OF YOUR CODE, DATA and REPORT/GRAPHS ON YOUR OWN SYSTEM IN CASE KRATOS.CS.RPI.EDU FAILS DURING THE ASSIGNMENT PERIOD.