Parallelization: Conway's Game of Life

By Aaron Weeden, Shodor Education Foundation, Inc.

I. Abstract

This module teaches: 1) Conway's Game of Life as an example of a cellular automaton, 2) how cellular automata are used in solutions to scientific problems, 3) how to implement parallel code for Conway's Game of Life (including versions that use shared memory via OpenMP, distributed memory via the Message Passing Interface (MPI), and hybrid via a combination of OpenMP and MPI), 4) how to measure the performance and scaling of a parallel application in multicore and manycore environments, and 5) how cellular automata fall into the Structured Grid "dwarf" (a class of algorithms that have similar communication and computation patterns).

Upon completion of this module, students should be able to: 1) Understand the importance of cellular automata in modeling scientific problems, 2) Design a parallel algorithm and implement it using OpenMP and/or MPI, 3) Measure the scalability of a parallel code over multiple or many cores, and 4) Explain the communication and computation patterns of the Structured Grid dwarf.

It is assumed that students will have prerequisite experience with C or Fortran 90, *nix systems, and modular arithmetic.

II. Pre-assessment Rubric

This rubric is to gauge students' initial knowledge and experience with the materials presented in this module. Students are asked to rate their knowledge and experience on the following scale and in the following subject areas:

Scale

1 – no knowledge, no experience
2 – very little knowledge, very little experience
3 – some knowledge, some experience
4 – a good amount of knowledge, a good amount of experience
5 – high level of knowledge, high level of experience

Subject areas

Cellular Automata
Parallel Algorithm Design
Parallel Hardware
MPI Programming
OpenMP Programming
Using a Cluster
Scaling Parallel Code

III. Conway's Game of Life and cellular automata: Motivation and Introduction

The cellular automaton is an important tool in science that can be used to model a variety of natural phenomena. Cellular automata can be used to simulate brain tumor growth by generating a 3-dimensional map of the brain and advancing cellular growth over time [1]. In ecology, cellular automata can be used to model the interactions of species competing for environmental resources [2]. These are just two examples of the many applications of cellular automata in many fields of science, including biology [3][4], ecology [5], cognitive science [6], hydrodynamics [7], dermatology [8], chemistry [9][10], environmental science [11], agriculture [12], operational research [13], and many others.

Cellular automata are so named because they perform functions automatically on a grid of individual units called cells. One of the most significant and important examples of the cellular automaton is John Conway's Game of Life, which first appeared in [15]. Conway wanted to design his automaton such that emergent behavior would occur, in which patterns that are created initially grow and evolve into other, usually unexpected, patterns. He also wanted to ensure that individual patterns within the automaton could dissipate, stabilize, or oscillate. Conway's automaton is capable of producing patterns that can move across the grid (gliders or spaceships), oscillate in place (flip-flops), stand motionless on the grid (still lifes), and generate other patterns (guns).

Conway established four simple rules that describe the behavior of cells in the grid. At each time step, every cell in the grid has one of two particular states: ALIVE or DEAD. The rules of the automaton govern what the state of a cell will be in the next time step.

Like all cellular automata, the rules in Conway's Game of Life pertain to cells and their "neighbors", or the cells to which a cell is somehow related (usually spatially). The collection of a cell's neighbors is known as the cell's "neighborhood". Two examples of neighborhoods are shown in Figure 1. The code in this module uses the latter of these two, the "Moore neighborhood", in which the 8 cells immediately adjacent to a cell constitute the cell's neighbors.

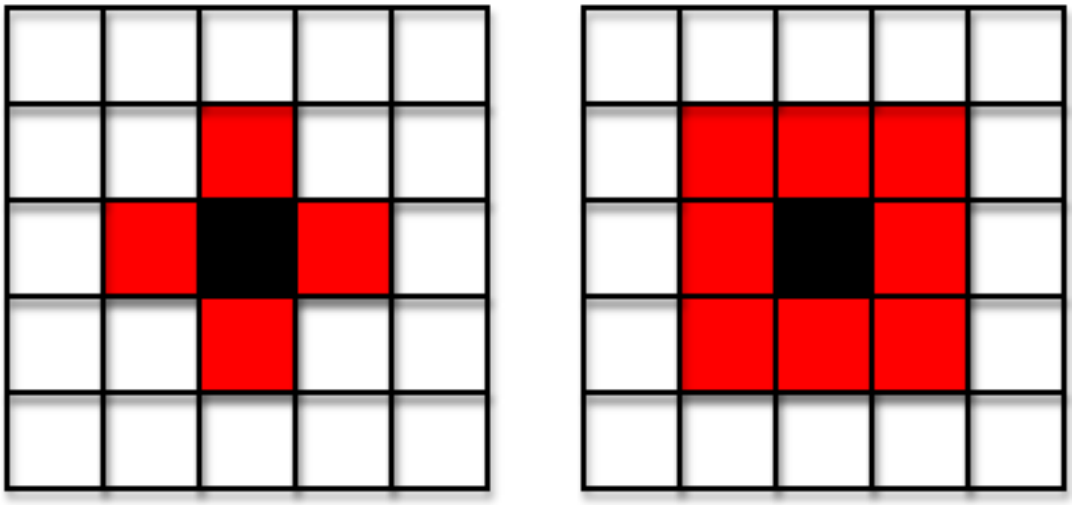

Figure 1: Two types of neighborhoods of a black cell, consisting of red neighbor cells: von Neumann neighborhood (left) and Moore neighborhood (right).

An issue arises for the cells on the edges of the grid, since they will not have a full neighborhood. Cells on the sides only have 5 neighbors, and those in the corners only have 3 neighbors. There are numerous ways to solve this problem. In this module, we resolve the issue by modeling the grid not as a rectangle, but as a torus that wraps around on the top, bottom, and sides. In this arrangement, the cells in the top row have the cells in the bottom row as their neighbors to the north, the cells in the bottom row have those in the top row as their neighbors to the south, the cells in the left column have the cells in the right column as their neighbors to the west, and the cells in the right column have the cells in the left column as their neighbors to the east. A toroidal grid can be simplified by including "ghost" rows and columns, which are copies of the rows and columns on the opposite sides of the grid from the edge rows and columns. A ghost corner is determined by copying the corner cell that is opposite the ghost corner on the diagonal. The method of using ghost rows and columns is depicted in Figure 2.

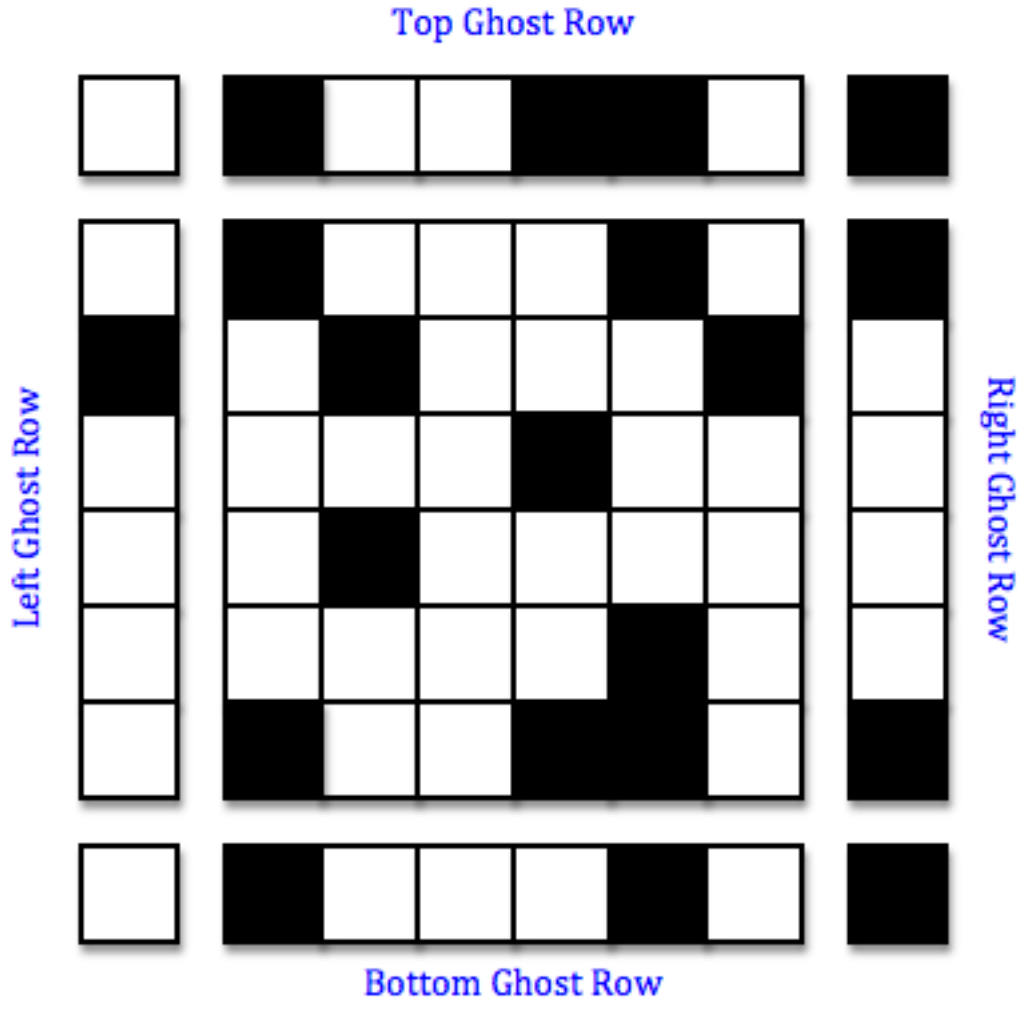

Figure 2: A toroidal grid simplified with ghost rows and columns

The rules of Conway's Game of Life are as follows:

- If a cell has fewer than 2 ALIVE neighbors, it will be DEAD in the next time step.
- If an ALIVE cell has 2 or 3 ALIVE neighbors, it will be ALIVE in the next time step.
- If a cell has more than 3 ALIVE neighbors, it will be DEAD in the next time step.
- If a DEAD cell has 3 ALIVE neighbors, it will be ALIVE in the next time step.

The automaton begins by initializing the states of the cells randomly and ends after a certain number of time steps have elapsed.

For a small board, a cellular automaton can be simulated with pencil and paper. For even a reasonably small board, however, working by hand becomes cumbersome and the use of a computer is needed to run the simulation in an acceptable amount of time. A coded implementation of Conway's Game of Life running on a single computer can simulate a fairly large grid in a very small amount of time.

To simulate an even bigger grid, one needs to employ more processing power than is available on a single processor. The concept of parallel processing can be used to leverage the computational power of computing architectures with multiple or many processors working together.

Conway's Game of Life is an interesting problem to parallelize because of the boundary conditions involving ghost rows and columns and because neighbor cells are tightly coupled; that is, neighbors relate to each other directly and require calculations to be performed on groups of them. This leads to interesting parallel communication patterns and requires a modest amount of computation as well.

Quick Review Questions:

1. What determines the state of a cell in a given time step?
2. What is the purpose of having "ghost" rows and columns?