Group Project proposal - Parallelizing Cellular Automaton (Conway's Game of Life)

Hritvik JV, Kaegan Koski

October 27, 2025

Abstract

This project will focus on a Conway's Game of Life (GoL) implementation. This implementation will use various parallelization techniques on a high-performance computing cluster (HPC). The goal is to analyze the performance characteristics of domain decomposition using different levels of partitioning the grid. This is bounded by the speed of transfer between partitions. This analysis will be done via **strong and weak scaling studies**. GoL serves as an ideal example for many stencil-based simulations, requiring rigorous analysis of processes communicating "ghost cell" data. Ghost cells referring to cell data on the edges of our partitions. We will measure quantitative metrics like speedup and parallel efficiency. This work will not only benchmark parallel performance but also establish a foundation for our understanding of similar projects with similar constraints.

1 Introduction and Area Description

1.1 Context: High-Performance Computing and Cellular Automata

HPC uses parallelism to speed up solving problems ripe with opportunities for parallelization. We will the University of Oregon provided HPC cluster to investigate the performance gains and bottlenecks involved with parallelizing a classic cellular automaton Conway's GoL.

GoL, specifically, involves a two-dimensional grid where the state of each cell in the next generation depends on the state of its eight neighbors. The two states being "alive" or "dead". This localized dependency structure makes GoL an excellent base example for **stencil computations**, which are fundamental to problems ranging from fluid dynamics to image processing.

1.2 Motivation and Problem Statement

The parallelization of GoL, typically achieved through **domain decomposition**, has two main factors; computation and communication. Each sub partition of our two-dimensional grid will need to communicate with the perimeter cells of the adjacent grids. These perimeter cells are called "ghost cells".

The execution time (T) for a single simulation step can be modeled as:

$$T = T_{\text{comp}} + T_{\text{comm}}$$

Where T_{comp} is the time spent on local cell rule evaluation(is cee (computation), and T_{comm} is the time spent exchanging ghost cell data (communication).

- **Computational Complexity:** T_{comp} scales with the area of the sub-grid
- **Communication Complexity:** T_{comm} scales with the perimeter of the sub-grid

As the number of processors (P) increases, the ratio of communication surface area to computational volume increases, eventually dominating the total time. You can imagine the case where the subgrid size is 1 and the communication size is 2-4 for every perimeter cell. The motivation for this project is to quantitatively demonstrate this crossover point in terms of the computation time on the HPC cluster, providing practical insights into the scaling behavior.

2 Proposed Methodology: Parallel GoL Implementation

The simulation will be implemented primarily in C or C++ using the **Message Passing Interface (MPI)** library for inter-process communication.

2.1 Domain Decomposition and Data Structure

The global $N \times N$ grid will be partitioned into P rectangular sub-domains, where P is the number of MPI processes. A 2D array of processes will be used to map the computational grid efficiently.

- **Grid Management:** Two arrays will be maintained by each process: a `current_state` array (the inner, computational domain plus the ghost cells) and a `next_state` array (the result array). This double-buffering technique ensures that all cell updates are based on the same generation's rules.
- **Communication Strategy:** We will employ **Periodic Boundary Conditions (BCs)**, where the grid wraps around on all four edges (toroidal geometry). This is the standard BC for GoL and simplifies the scaling analysis as all processes have identical communication requirements.

2.2 Ghost Cell Information Exchange

The efficiency of the parallel GoL rests on optimizing the exchange of boundary data. At the start of each generation update, every processor must exchange the rows and columns that form its perimeter with its eight neighboring processors.

- **Required Exchanges:** Each process needs to communicate with eight neighbors (North, South, East, West, NE, NW, SE, SW).
- **Synchronization:** Utilize information exchange libraries that do not block computation of cells to try and reduce network latency
- **Performance Measurement:** A timer will be used to record the time for each component: total time, T_{comm}, and T_{comp}.

3 Directions of Investigation

The parallel investigation will be structured into two primary scaling studies, followed by an analysis of the communication patterns.

3.1 Investigation 1: Strong Scaling Analysis

Analyze the performance while keeping the **global problem size ($N \times N$) fixed** while increasing the number of processors (P).

- ◇ **Objective:** To determine the maximum speedup achievable for a fixed, large grid size (e.g., $50,000 \times 50,000$ cells) and identify the point where communication overhead negates the benefits of additional parallelism.
- ◇ **Procedure:** Run the simulation for G generations (e.g., $G = 1000$) using $P = 1, 2, 4, 8, 16, 32, \ldots$ processors.
- ◇ **Metric:** Calculate **Speedup (S_P)** and **Parallel Efficiency (E_P):**

$$S_P = \frac{T_1}{T_P} \quad ; \quad E_P = \frac{S_P}{P}$$

  Where T_1 is the wall-clock time on one processor and T_P is the time on P processors.
- ◇ **Expected Result:** We anticipate a non-linear efficiency curve, falling off as P becomes large.

3.2 Investigation 2: Weak Scaling Analysis

This study holds the **workload per processor fixed** while simultaneously increasing both the problem size and the number of processors.

- ◇ **Objective:** To verify that the parallel implementation is scalable, meaning that for a fixed amount of work per processor, the execution time remains approximately constant as the total problem size grows.
- ◇ **Procedure:** Starting with a baseline sub-grid size (e.g., 1000×1000), run the simulation for $P = 1, 4, 9, 16, \ldots$ processors, with the total grid size scaling proportionally ($N \times N = P \times (1000 \times 1000)$).

- **Metric: Weak Efficiency (E_{weak}):**

$$E_{\text{weak}} = \frac{T_{\text{baseline}}}{T_P}$$

  Where T_{baseline} is the time taken by the single-processor run with the baseline sub-grid size.

- **Expected Result:** Since the computational complexity ($O(1)$ per processor) and the communication complexity ($O(1)$ per processor) remain relatively constant, we expect the weak efficiency to remain high (close to 1) for a large number of processors.

3.3 Investigation 3: Communication Overhead Analysis

A critical element of the analysis is the breakdown of time spent on computation versus communication.

- **Focus:** Using the collected T_{comp} and T_{comm} metrics, we will plot the ratio $\frac{T_{\text{comm}}}{T_{\text{comp}}}$ against the number of processors (P).
- **Objective:** Quantify the impact of the communication latency

4 Expected Results and Future Work

4.1 Anticipated Outcomes

The successful completion of this project is expected to yield the following results:

R1 High-Performance Parallel Solver: A fully functional MPI implementation of the Game of Life capable of simulating massive grids (up to $100,000 \times 100,000$)

R2 Quantitative Scaling Data: Data analysis for both strong and weak scaling studies, illustrating the relationship between processor count, grid size, and efficiency.

R3 Communication Breakdown: A quantitative analysis demonstrating the point at which inter-process communication time surpasses computation time.

4.2 Future Directions and Extensions

The methodology developed here provides a general framework for parallelizing many types of cellular automata and stencil-based problems. Time permitting we would like to implement the following as well:

Exploration of Multi-State Cellular Automata (CA)

A compelling extension is to implement a multi-state CA where cells can have $K > 2$ states (e.g., 0=Dead, 1=Young, 2=Adult, 3=Old).

→ **Research Focus:** This extension shifts the focus to an analysis of **computational complexity**. The rule evaluation step becomes more complex, increasing the T_{comp} relative to T_{comm}.

→ **Parallel Impact:** By increasing the computational work per cell (higher T_{comp}), the threshold at which communication dominates performance is pushed further out, potentially leading to better strong scaling for a higher number of processors.

Visualization

Visualizing the simulation by animating it will be helpful in understanding how the rules can be played around with to achieve more complex emergent effects

- **Parallel I/O (MPI-IO):** Periodically save the grid state using **MPI-IO**. This allows all P processes to write their local data simultaneously to non-overlapping sections of a single output file (e.g., a Portable Pixmap, `.ppm`, file)
- **Visualization:** The resulting sequence of files will be post-processed using a local Python script to generate an animation (GIF or MP4) of the simulation's evolution, confirming that patterns evolve correctly across processor boundaries.