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

Hritvik JV

October 27, 2025

Abstract

This proposal outlines a project focused on the high-performance implementation and analysis of Conway's Game of Life (GoL) cellular automaton using the Message Passing Interface (MPI) on a high-performance computing (HPC) cluster. The primary objective is to investigate the performance characteristics of domain decomposition in a bandwidth-bound problem, specifically through detailed **strong and weak scaling studies**. GoL serves as an ideal proxy for many stencil-based scientific simulations, requiring rigorous analysis of inter-process communication overheads associated with "ghost cell" exchange. The expected results include quantitative metrics like speedup and parallel efficiency, alongside a robust visualization pipeline for qualitative analysis. This work will not only benchmark parallel performance but also establish a foundation for exploring more complex, non-uniform cellular automata models in future research.

1 Introduction and Area Description

1.1 Context: High-Performance Computing and Cellular Automata

High-Performance Computing (HPC) relies on parallelism to solve problems intractable for single processors. For this project, we will leverage the architecture of a target HPC cluster to investigate the performance gains and bottlenecks associated with parallelizing a classic cellular automaton (CA): Conway's Game of Life (GoL).

Cellular automata are discrete models often used in computational science to simulate complex system dynamics using simple, localized rules. GoL, specifically, involves a two-dimensional grid where the state of each cell (alive or dead) in the next generation depends solely on the state of its eight immediate neighbors. This localized dependency structure makes GoL an excellent archetype for **stencil computations**, which are fundamental to fields ranging from fluid dynamics (e.g., Finite Difference Methods) to image processing.

1.2 Motivation and Problem Statement

The parallelization of GoL, typically achieved through **domain decomposition**, presents a crucial challenge in balancing computation and communication. The core problem is that the state update for cells located at the boundaries of a processor's assigned sub-grid requires data (the "ghost cells" or "halo") from adjacent processors.

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 (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 ($O(N^2/P)$, where N^2 is the total grid size and P is the number of processors).
- **Communication Complexity:** T_{comm} scales with the perimeter of the sub-grid and the latency of the network ($O(N/\sqrt{P})$).

As the number of processors (P) increases in a strong scaling scenario, the ratio of communication surface area to computational volume increases, eventually dominating the total time. The motivation for this project is to quantitatively demonstrate this crossover point on the HPC cluster, providing practical insights into scaling behavior for a bandwidth-bound, stencil-based problem.

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 Exchange Protocol

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:** We will utilize **non-blocking communication primitives** (e.g., `MPI_Isend` and `MPI_Irecv`) followed by a collective wait (`MPI_Waitall`). This strategy allows the computation for non-boundary cells to overlap with the data transfer of the ghost cells, potentially hiding communication latency.
- **Performance Measurement:** A high-resolution timer (e.g., `MPI_Wtime`) will be used to record the wall-clock time for each component: total time, T_{comm}, and T_{comp}.

2.3 I/O and Visualization Pipeline

- **Parallel I/O (MPI-IO):** The grid state will be saved periodically 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), which is crucial when dealing with extremely large grids.
- **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.

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

This study holds 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 sharply as P becomes large due to the dominating $O(N/\sqrt{P})$ communication cost.

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:** Calculate **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: Analysis of Communication Overhead

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 high-speed interconnect. By isolating communication time, we can determine the true efficacy of the non-blocking communication strategy employed.

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 robust, fully functional MPI implementation of the Game of Life capable of simulating massive grids (up to $100,000 \times 100,000$) on the HPC cluster.

R2 Quantitative Scaling Data: Complete datasets for both strong and weak scaling studies, clearly 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, providing empirical validation of the theoretical scaling models ($O(N^2)$ computation vs. $O(N)$ communication).

R4 Visual Confirmation: Time-lapsed visualizations of GoL evolution that qualitatively confirm correct parallel behavior and successful parallel I/O operation.

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.

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.