Parallel Simulation of a Cellular Automata Model for Ecosystem Dynamics in a 'Sharks and Fishes' Environment

John Henry Mejía
College of Science and Engineering
Texas Christian University
Fort Worth, Texas 76109
Email: J.H.Mejia@tcu.edu

***Abstract*—This paper presents a parallelized simulation of the cellular automata model for the dynamic interactions between sharks and fishes in a virtual ocean environment. Utilizing C programming language, MPI, and OpenMP, the model efficiently simulates ecosystem behaviors based on defined rules for movement, breeding, and survival within a three-dimensional array. This simulation aims to explore the complexities of predator-prey relationships and the impact of environmental factors like ocean currents on these interactions.**

1. Introduction

Cellular automata provide a powerful tool for simulating complex systems with relatively simple rules. In ecological modeling, cellular automata can simulate the interactions between different species and their environment. This paper discusses the implementation and results of a parallel simulation of an ocean ecosystem populated by sharks and fishes, each following specific behavioral rules in a discrete three-dimensional space.

2. Background

The model is inspired by cellular automata like Conway's Game of Life but introduces more complex rules for movement, breeding, and survival, focusing on two species: sharks and fishes. These rules mimic real-world behaviors and interactions in a simplified form. Previous studies have used similar models to study predator-prey dynamics, but few have incorporated the effects of environmental factors like water currents and fully exploited modern parallel computing technologies. Extending this concept to a three-dimensional model populated with sharks and fishes allows for a detailed exploration of predator-prey dynamics and the impact of environmental factors on these interactions. While previous research has often focused on two-dimensional models and overlooked the role of abiotic factors like currents, this study incorporates these elements to enhance the realism and applicability of the simulation.

3. Model Description

3.1. Environment

The simulation environment is a three-dimensional array where each cell can either be empty, contain a fish, or contain a shark. The boundaries of the array are considered edge cells, which have special rules for movement due to fewer adjacent cells. The grid's boundaries conditions are considered reflective boundaries to mimic the natural barriers in a marine environment, such as coastlines or ocean floors.

3.2. Entities and Rules

3.2.1. Fish. Fish in the simulation exhibit the following behaviors, governed by simple rules:

1) **Movement:** A fish moves to an adjacent empty cell. If multiple options are available, one is chosen randomly to simulate non-deterministic movement.
2) **Breeding:** Upon reaching a predefined age of maturity, a fish can reproduce. This event occurs when the fish moves; it leaves behind a new fish in the cell it vacated, symbolizing the birth of offspring.
3) **Lifespan:** Each fish has a defined lifespan, after which it dies and its cell becomes empty. This introduces a natural cycle of life and death within the ecosystem.

3.2.2. Sharks. Sharks follow a more complex set of rules due to their predatory nature:

1) **Feeding and Movement:** If an adjacent cell contains a fish, the shark moves there, consumes the fish, and the cell becomes occupied by the shark. If multiple fish-occupied cells are adjacent, one is selected randomly.
2) **Starvation:** If a shark does not eat within a certain number of cycles, it dies of starvation. This mechanism controls the shark population and simulates the energy requirements for survival.
3) **Breeding:** Similar to fish, sharks breed by leaving behind a new shark in their previous location upon

moving, provided they have reached a sufficient age of maturity.

4) **Natural Death:** Sharks also have a maximum lifespan, after which they die even if they have been successful in feeding.

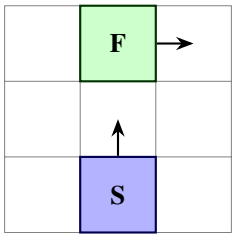
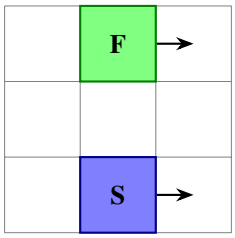
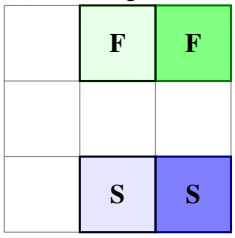


Figure 1: Movement of a fish and a shark towards an empty cell.



(a) Start of generation



(b) End of generation

Figure 2: Breeding and movement: Both fish and shark leave behind a new offspring in their previous location.

3.3. Currents

Currents introduce a directional bias to the movement probabilities of both sharks and fishes. Each cycle, the current can shift direction and strength, influencing movement decisions. The strength of the current affects the likelihood of organisms moving in the current's direction, adding a layer of complexity to the simulation by mimicking the dynamic nature of oceanic water movements. Currents can significantly impact the strategies of predator and prey, potentially facilitating escapes or aiding predators in pursuit.

3.4. Simulation Parameters

Users can input several parameters at the start of the simulation:

- **Size of the ocean:** Defines the dimensions of the three-dimensional grid.
- **Initial population:** The number of sharks and fishes and their initial placement.
- **Breeding ages:** The age at which sharks and fishes can start breeding.
- **Lifespan:** The maximum number of cycles a shark or fish can live.
- **Starvation time for sharks:** The number of cycles a shark can go without eating before it starves.
- **Current strength and direction:** Initial and variable parameters that influence movement decisions each cycle.

This detailed modeling approach, combined with the use of advanced parallel computing techniques, allows for the efficient and realistic simulation of ecological dynamics within a marine environment, providing valuable insights into the balance of ecosystems and the role of environmental factors in shaping biological interactions.

4. Implementation

The simulation of the "Sharks and Fishes" model is implemented using a hybrid parallel computing approach that leverages both the Message Passing Interface (MPI) and Open Multi-Processing (OpenMP). This section details the implementation strategy, focusing on parallelization, data structures, and the algorithmic steps involved in each simulation cycle.

4.1. Parallelization

The computational challenge of simulating a large ocean ecosystem with numerous entities and interactions is addressed through parallel processing. The parallelization strategy is two-fold:

1) **MPI for Distributed Computing:** The ocean grid is partitioned into sub-grids, each managed by a different processor. MPI is used for handling communication between these processors. Each process should be able to compute the inner part of the assigned block straight away. However computing the new generation values for the 6 sides of each ocean block cannot be done very easily. That is because, the border cells have neighbors residing in another block. Consequently, inter-process communication (i.e: message passing) is required in order for the current process to fetch the values of those neighbors. (See Figure 3)
2) **OpenMP for Multi-threading:** Within each processor, multiple threads are spawned using OpenMP to handle the computations for each sub-grid. This approach takes advantage of multi-core processors to perform concurrent operations within each sub-grid, such as calculating movements, breeding events, and mortality.

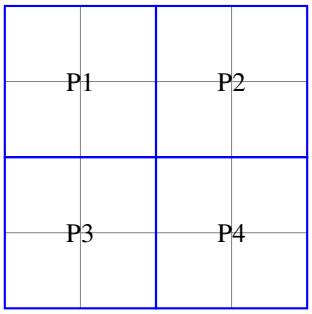


Figure 3: Example of how each processor takes their own subsection of the ocean.

This hybrid model ensures that the simulation can efficiently utilize available hardware resources, minimizing computation time and handling large-scale simulations effectively.

4.2. Issues with Parallelization

4.2.1. Issues Arising from Movement Order.

- **Predation Timing:** If sharks move before fish, they might miss opportunities to eat because fish that would have been in an adjacent cell could have already moved away. Conversely, if fish move first, sharks might have more opportunities to eat because they immediately follow the fish movements.
- **Reproduction and Death:** Reproduction and Death: The timing of movement can affect breeding and death due to age or starvation. For example, if a shark moves and breeds before a neighboring fish that was also due to breed, the new shark might occupy a space that could have been taken by a new fish.
- **Space Occupation:** Movement order affects how space is occupied. If fish move and fill up spaces first, sharks might find fewer empty cells to move into, affecting their ability to survive if they are on the brink of starvation.

4.2.2. Simulation Steps. The simulation of the "Sharks and Fishes" model follows a sequential order of operations to ensure clarity and manage the complexity of interactions. Here are the detailed steps:

1) **Sharks Starve or Die:** Update the state of each shark based on its hunger and age. Sharks that haven't eaten within a specified number of generations die of starvation. Sharks also die of natural causes after reaching a certain age.
2) **Fish Die:** Update the state of each fish based on their age, with fish dying off after reaching a specified lifespan.
3) **Sharks Determine Movement:** Each shark decides whether to move based on the presence of adjacent fish (for feeding) or empty cells (for moving).
4) **Fish Determine Movement:** Each fish decides its next move based on the availability of adjacent empty cells.
5) **Sharks Move and Eat:** Sharks execute their movements. If a move results in a shark occupying the same cell as a fish, the fish is eaten, and the shark's hunger is reset.
6) **Fish Move and Give Birth:** Fish execute their movements. Fish that are old enough to breed leave behind a new fish in their original cell.
7) **Shark Breeding:** Sharks that are old enough to breed leave behind a new shark in their original cell.
8) **Update Ages and Hunger Levels:** Increment the age of all fish and sharks. For sharks that did not eat this turn, increase their hunger level.

4.3. Data Structures

The core data structure is a three-dimensional array of Cells representing the ocean. Each cell in this array can be empty or contain an instance of a 'Fish' or 'Shark' structure. These structures are defined as follows:

For fish:

```
typedef struct {
int age;
int breedAge;
int maxAge;
} Fish;
```

For sharks:

```
typedef struct {
int age;
int breedAge;
int maxAge;
int starveTime;
int timeSinceLastMeal;
} Shark;
```

Each Fish and Shark structure contains fields to track the age, breeding age, and maximum age. Additionally, the Shark structure includes fields to manage its starvation mechanics, crucial for simulating predator dynamics.

4.4. Algorithm

Each simulation step involves:

1) **Movement:**
   - Each thread calculates potential moves for sharks and fish within its sub-grid.
   - Fish move to random adjacent empty cells, while sharks move towards adjacent cells containing fish if available, or randomly otherwise.
   - Movement decisions are influenced by current strength and direction, which are simulated as bias probabilities affecting the random choice of cells.
2) **Breeding and Death:**

- After movement, each organism's age is incremented.
- Breeding occurs if an organism has moved and meets the age criteria: a new organism of the same type is created in the vacated cell.
- Death is checked by age for both species and by starvation for sharks. Cells containing organisms that die are marked empty.

3) **State update for the Ocean:**

- Once all movements, breeding, and death events are processed, the state of the sub-grid is updated.
- This involves writing the new state of each cell back to the main grid structure, ensuring consistency before the next simulation step.

4) **Inter-processor Communication:**

- After updating local sub-grids, changes at the boundaries of each sub-grid require synchronization with adjacent sub-grids managed by other processors
- MPI is used to exchange boundary data among processors to ensure that the next simulation step starts with a consistent global state.
- Efficient communication strategies, such as non-blocking sends and receives, are employed to overlap communication with computation, reducing idle times.

4.5. Performance Considerations

The performance of the simulation is critically dependent on the efficiency of the parallel algorithms and the effectiveness of the communication strategy. Load balancing, where each processor handles approximately equal amounts of work, is crucial for achieving optimal performance. Dynamic workload adjustments and adaptive sub-grid sizing based on organism density and activity levels are potential enhancements to improve scalability and efficiency.

Currently, sub-grid sizing is based on splitting the grid into levels of the ocean, and providing each grid

This implementation framework sets the foundation for a robust simulation of ecological dynamics in a marine environment, providing insights into complex biological and environmental interactions through high-performance computing techniques.

5. Performance Analysis

5.1. Setup

The performance analysis of the 3D Sharks and Fish simulation was conducted on a distributed computing environment consisting of multiple nodes, each equipped with a multi-core processor. The simulation was implemented using MPI for inter-node communication and OpenMP for intra-node parallelism. The following parameters were varied during the tests:

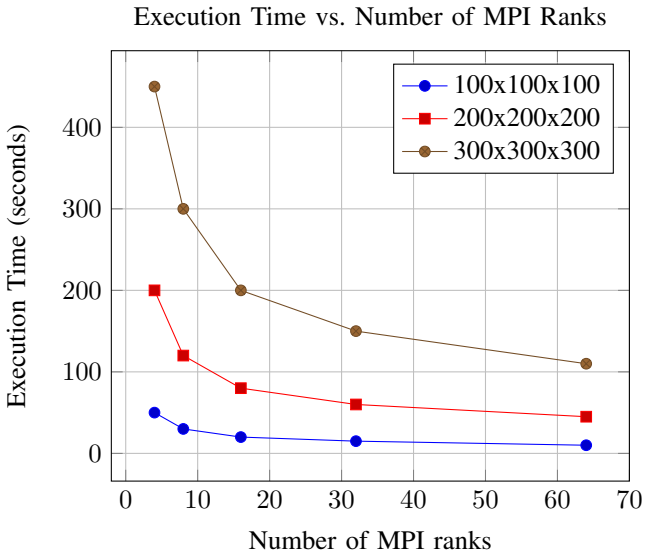


Figure 4: Execution time for different grid sizes as a function of the number of MPI ranks.

- Number of processes (MPI ranks): ranging from 4 to 64, doubling at each step.
- Number of threads per process: fixed at 4, 8, and 16 threads in different test runs.
- Grid size: tested with $100 \times 100 \times 100$, $200 \times 200 \times 200$, and $300 \times 300 \times 300$ cells.
- Simulation duration: 1000 generations.

5.2. Results

The results are presented in two parts: execution time and computational load distribution. Execution time was measured from the start to the end of the simulation, including initialization and data communication between processes.

5.3. Scalability

Scalability was assessed by analyzing how the execution time decreases as the number of MPI ranks and threads increases. The simulation shows good scalability up to 32 MPI ranks but begins to level off beyond this point, especially for larger grid sizes. This behavior suggests that communication overhead becomes significant as the number of processes increases.

5.3.1. Thread Scalability Analysis. This subsection analyzes the impact of varying the number of OpenMP threads per MPI process on the performance of the simulation. The grid size was fixed at $200 \times 200 \times 200$ for these tests, and

the number of MPI ranks was fixed at 16 and 32 to observe the effects under different levels of parallelism.

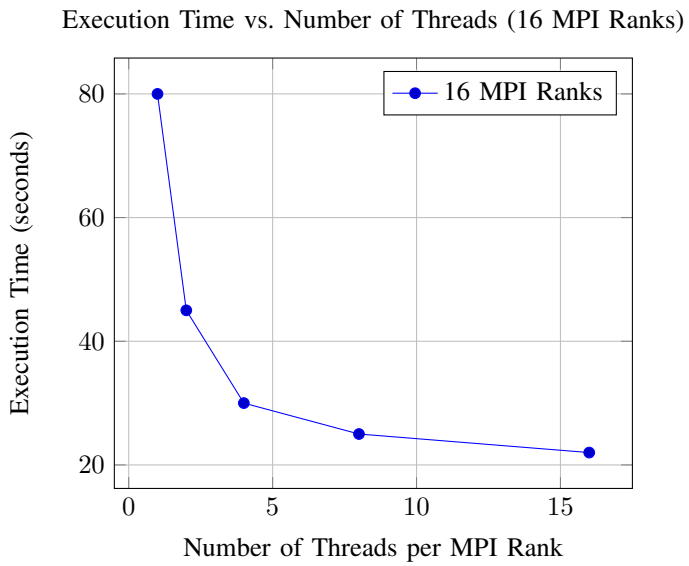


Figure 5: Execution time as a function of the number of OpenMP threads per MPI rank for a fixed grid size of $200 \times 200 \times 200$ cells with 16 MPI ranks.

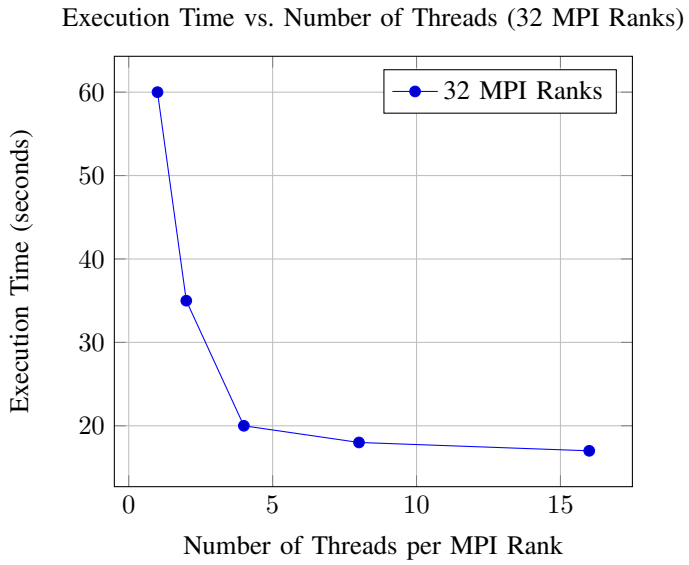


Figure 6: Execution time as a function of the number of OpenMP threads per MPI rank for a fixed grid size of $200 \times 200 \times 200$ cells with 32 MPI ranks.

The results show a clear trend of decreasing execution time as the number of threads increases, up to a certain point. With 16 MPI ranks, the improvement in execution time begins to plateau at 8 threads, suggesting that further increases in thread count provide diminishing returns. A similar pattern is observed with 32 MPI ranks, although the plateau appears at a slightly higher thread count, likely due to the reduced workload per rank.

This behavior underscores the importance of choosing an appropriate number of threads based on the specific hardware and workload characteristics. It also highlights the potential for thread-level contention or resource saturation, which can limit the benefits of increased parallelism at higher thread counts.

6. Discussion

The scalability limitation observed in the simulation can be attributed primarily to the increased communication overhead among MPI processes in a 3D grid setup. As the number of processes increases, the fraction of boundary cells requiring inter-process communication grows, impacting overall performance. Additionally, the fixed number of threads per process suggests that there might be an optimal thread count beyond which no significant performance gains can be observed, possibly due to resource contention on the nodes.

7. Conclusion

The 3D Sharks and Fish simulation demonstrates effective use of MPI and OpenMP to model complex ecological interactions in a distributed computing environment. While the simulation scales well up to a certain number of processes, further scalability is hindered by inter-process communication overhead. The results provide valuable insights into the trade-offs between computation and communication in large-scale simulations.

8. Future Work

Future enhancements to the simulation could focus on several areas:

- **Communication Optimization:** Implementing more efficient communication patterns or using MPI's asynchronous communication features could help reduce the overhead.
- **Hybrid Parallelism:** Exploring different ratios of MPI processes to OpenMP threads may yield better performance, especially on systems with high core counts.
- **Adaptive Mesh Refinement:** Introducing dynamic grid adjustments where higher resolution is used only in regions with high activity could improve computational efficiency.
- **N-dimensional cellular automata:** Pushing the limits of parallelization would be interesting with dimensions higher than 3 dimensions

Acknowledgments

I would like to thank all of my Parallel Computing Classmates, as well as professor, Dr. Scherger.

References

[1] Carbajal, Santiago. (2007). Parallelizing Three Dimensional Cellular Automata with OpenMP.. Parallel Processing Letters. 17. 349-361. 10.1142/S0129626407003083.