Rise of the Elements: The Parallel Chronicles

CMPE 300 - Analysis of Algorithms
Department of Computer Engineering, Bogazici University
MPI Programming Project

Beyza İrem Urhan

Deadline: 20 December 2024, 23:59

1 Introduction

In a distant universe, a world governed by the elements—Earth, Fire, Water, and Air—is in peril. The elemental lords have been at war for centuries, vying for control over the sacred crystal, a source of unimaginable energy that maintains balance across their realm. As tensions escalate, each faction has rallied its forces for an ultimate confrontation to determine the fate of the crystal and the world.

To simulate this epic battle, we harness the power of parallel computing using the Message Passing Interface (MPI), a widely used standard for message passing in distributed computing environments. Just as the elemental forces interact and clash across the vast battlefield, MPI allows us to coordinate computations across multiple processors, effectively modeling the complex interactions of the factions.

2 The Battlefield

The battlefield is the arena where the elemental factions engage in combat. Understanding its structure and mechanics is essential for implementing the simulation accurately.

2.1 Grid Structure

- The battle takes place on an $N \times N$ grid, where N represents the size of the square battlefield.
- Each cell on the grid can hold a single unit or remain neutral, denoted by `(.)`. Neutral cells serve as potential expansion zones for factions, making them critical for strategic play.
- Units belong to one of four factions:
  - Earth (`E`)
  - Fire (`F`)

  - Water (`W`)
  - Air (`A`)

- The grid has fixed boundaries; units cannot move or attack beyond the edges. This means that units at the edges or corners have fewer neighboring cells, affecting their attack patterns and movement options.

2.2 Coordinate System

- Grid cells are identified by their coordinates (x, y), where x and y range from 0 to $N - 1$.
- The origin $(0, 0)$ is at the top-left corner of the grid.
- The x-coordinate increases to the right, and the y-coordinate increases downward.
- For example, the cell to the right of the origin is $(1, 0)$, and the cell below the origin is $(0, 1)$.

3 Units and Factions

Each faction has units with specific attributes and abilities that are consistent for all units within the faction.

3.1 Earth Units (`E`)

- **Health:** 18 HP
- **Attack Power:** 2 HP damage
- **Healing Rate:** +3 HP when not attacking.
- **Attack Pattern:** Targets direct neighbors (up, down, left, right).
- **Special Ability:** *Fortification*
  - Always reduces incoming damage by 50% (rounded down).
  - This ability is passive and continuously active.
- *Fortification reflects Earth's resilience and stability, allowing it to sustain prolonged engagements and hold key positions on the battlefield.*

Attack Pattern Visualization:

$$\begin{array}{ccc} & \mathbf{X} & \\ \mathbf{X} & \mathbf{E} & \mathbf{X} \\ & \mathbf{X} & \end{array}$$

3.2 Fire Units (`F`)

- **Health:** 12 HP
- **Attack Power:** 4 HP damage (can increase to a maximum of 6 HP using *Inferno*).
- **Healing Rate:** +1 HP when not attacking.
- **Attack Pattern:** Targets all 8 neighboring cells (up, down, left, right, and diagonals).
- **Special Ability:** *Inferno*
  - Each time a Fire unit destroys an enemy unit during the Action Phase, its attack power increases by +1 HP.
  - This increase can occur **once per round per unit**, up to a maximum attack power of 6 HP.
  - The increased attack power persists for the rest of the wave.
  - At the beginning of a new wave, the attack power resets to the base value of 4 HP.
- *Inferno captures Fire's aggressive and destructive nature, rewarding successful attacks with increased power. This makes Fire a high-risk, high-reward faction.*

Attack Pattern Visualization:

$$\begin{array}{ccc} \mathbf{X} & \mathbf{X} & \mathbf{X} \\ \mathbf{X} & \mathbf{F} & \mathbf{X} \\ \mathbf{X} & \mathbf{X} & \mathbf{X} \end{array}$$

3.3 Water Units (`W`)

- **Health:** 14 HP
- **Attack Power:** 3 HP damage
- **Healing Rate:** +2 HP when not attacking.
- **Attack Pattern:** Targets diagonally adjacent cells.
- **Special Ability:** *Flood*
  - At the end of each **wave**, after all rounds have been completed, each Water unit converts one adjacent neutral cell (`.`) into a Water unit.
  - **Selection of the cell:**
    1. Choose the neutral cell with the lowest y-coordinate.
    2. If multiple cells share the same y, choose the one with the lowest x-coordinate.
  - The new Water units have full health and the same attributes.
  - This ability is automatic and does not depend on whether the unit attacked or healed during the wave.

- *Flood represents Water's gradual and inevitable expansion, allowing it to grow by assimilating nearby neutral territory at the end of each wave.*

Attack Pattern Visualization:

```
X . X
. W .
X . X
```

3.4 Air Units (`A`)

- **Health:** 10 HP
- **Attack Power:** 2 HP damage
- **Healing Rate:** +2 HP when not attacking.
- **Attack Pattern:** Targets all neighboring cells (up, down, left, right, and diagonals), and can ”skip over” neutral cells to attack the next enemy unit in that direction.
- **Special Ability:** *Windrush*
  - Before the Action Phase, Air units may move to an adjacent empty cell (up, down, left, right, or diagonal).
  - **Movement Criteria:**
    1. Select the position that allows the unit to attack the most enemy units.
    2. If moving does not provide any advantage (i.e., does not increase the number of enemy units in attack range), the unit remains in its current position.
    3. If multiple positions qualify, choose the one with the lowest y-coordinate.
    4. If still tied, choose the one with the lowest x-coordinate.
  - This movement does not prevent the unit from attacking in the same round.
- *Windrush and the strategic repositioning reflect Air's speed and elusiveness, allowing it to dynamically adapt to the battlefield and maximize its effectiveness.*

Attack Pattern Visualization:

Default Attack Pattern:

```
X X X
X A X
X X X
```

Attack Pattern Over Neutral Cells:

```
X . X . X
. . . . .
X . A . X
. . . . .
X . X . X
```

*Note: Air units can attack over any number of consecutive neutral cells (.) in any direction to target the next enemy unit (**X**).*

4 Phases

Each wave consists of R rounds. In each round, the following phases occur:

4.1 Movement Phase (Air Units Only)

- Air units may move to an adjacent empty cell based on their *Windrush* ability.

4.2 Action Phase

- All units decide whether to **attack** or **skip** based on their health and strategic considerations.
- Units that choose to attack calculate and queue their damage outputs.
- Units that choose not to attack will be eligible to heal in the Healing Phase.
- **Skipping Attacks:**
  - **Health Threshold:** Units will skip attacking if their health falls below 50% of their maximum health to prioritize healing.
  - **No Valid Targets:**Units skip attacking if no enemy units are within their attack range.

4.3 Resolution Phase

- **Damage Application:**
  - Apply all queued damage to the units simultaneously.
  - Earth's *Fortification* reduces incoming damage by 50% (rounded down).
  - Units with $HP \leq 0$ after damage application are removed from the grid.

4.4 Healing Phase

- Units that did not attack during the Action Phase recover health according to their healing rate.

5 Execution Instructions

For this project, you are required to implement the simulation in Python using the MPI framework. Below are the compilation and execution instructions, along with the input and output file formats.

5.1 Compilation and Running

- Install **mpi4py** if it is not already installed:

  ```
  pip install mpi4py
  ```

- To run the program with P processes:

  ```
  mpiexec -n [P] python main.py input.txt output.txt
  ```

  Here, `[P]` is the total number of processes, including one for the manager and $P-1$ for the workers.

- Ensure you provide the required input and output files as command-line arguments.

5.2 Input and Output Files

5.2.1 Input Format

The input file `input.txt` specifies the simulation parameters and unit placements:

- The first line contains four integers:

  ```
  N W T R
  ```

  where:

  - N: Grid size ($N \times N$).
  - W: Number of waves.
  - T: Number of units per faction per wave.
  - R: Number of rounds per wave.

- Each subsequent block provides the unit placements for one wave, adhering to the following format:

  ```
  Wave 1:
  E: x1 y1, x2 y2, ..., xT yT
  F: x1 y1, x2 y2, ..., xT yT
  W: x1 y1, x2 y2, ..., xT yT
  A: x1 y1, x2 y2, ..., xT yT

  Wave 2:
  ...
  ```

- For each faction, list the coordinates of their units for that wave. Ensure that:

  - No two units occupy the same cell.
  - Coordinates are within the grid bounds $[0, N-1]$.

5.2.2 Output Format

The output file `output.txt` contains the final grid state after all waves are completed. Each line represents a row in the grid, with cells separated by spaces:

```
E . F W .
. W . . F
A . E . W
```

Here:

- `E, F, W, A`: Cells occupied by Earth, Fire, Water, or Air units.
- `.`: Neutral (empty) cells.

Ensure that the output grid reflects the state after all movements, attacks, healing, and special abilities have been processed across all waves.

6 Manager and Worker Responsibilities

The simulation uses a manager-worker model, with one manager process (`rank = 0`) and $P - 1$ worker processes.

6.1 Manager Responsibilities

1. Parse the input file `input.txt`.
2. Distribute the grid and unit data evenly among worker processes.
3. Collect results from the workers and generate the final grid state.
4. Write the final grid state to `output.txt`.

6.2 Worker Responsibilities

1. Receive grid partitions and unit data from the manager.
2. Simulate movements, attacks, and special abilities for the assigned grid.
3. Communicate with neighboring workers for boundary data.
4. Send results back to the manager at the end of each wave.

7 Partitioning Strategies (20 Bonus Points for Checkered Partitioning)

Efficient grid partitioning is crucial for balancing workloads and minimizing communication between processes. You have the option to choose between two partitioning strategies:

7.1 Striped Partitioning (Standard)

- The grid is divided horizontally, and each processor is assigned a set of contiguous rows.
- This minimizes communication to top and bottom neighbors but may lead to uneven workloads if unit distribution is uneven.

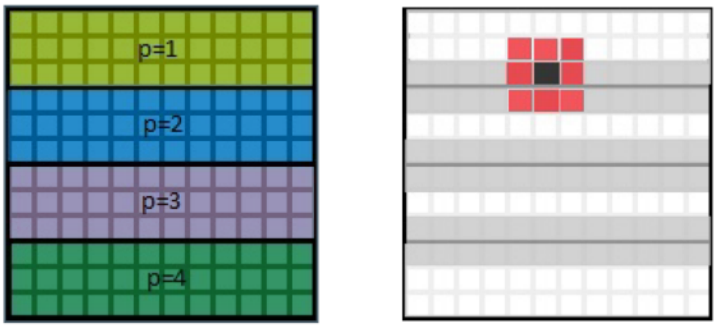

Figure 1: Striped Partitioning Example

7.2 Checkered Partitioning

- The grid is divided into smaller square blocks, forming a checkerboard pattern. Each processor manages one or more of these blocks.
- This strategy ensures a more balanced workload across processors but increases the complexity of communication, as processes have multiple neighbors.

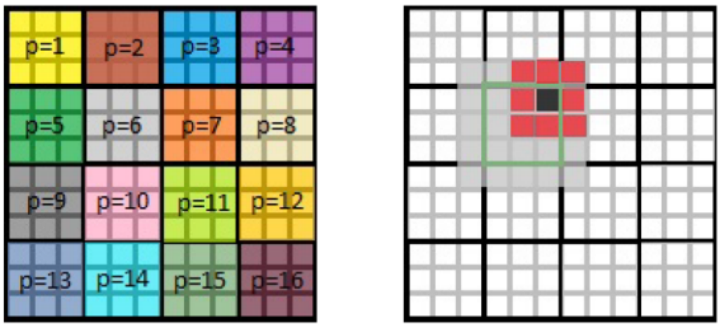

Figure 2: Checkered Partitioning Example

8 Communication Between Workers

- **Boundary Communication:**
  - In striped partitioning, processors exchange top and bottom boundary rows with their neighbors.
  - In checkered partitioning, processors exchange rows, columns, and corners with adjacent neighbors.

- **Avoid Deadlocks:**
  - Use an even-odd communication scheme:
    1. Odd-ranked processors send first, even-ranked processors receive.
    2. Roles reverse in the next phase.

9 MPI Functions and Performance

9.1 MPI Functions

You will need to use the following MPI functions:

- `MPI_Init`, `MPI_Finalize`
- `MPI_Comm_rank`, `MPI_Comm_size`
- `MPI_Send`, `MPI_Recv`

9.2 Performance Considerations

- Striped partitioning performs well with fewer communication overheads.
- Checkered partitioning balances workloads but may slow down due to increased communication complexity.
- Avoid deadlocks by implementing the even-odd communication strategy.

10 Submission Guidelines

- **Project Collaboration:**
  - This project is designed to be completed by a pair of two students.
  - Both students are expected to contribute significantly to all aspects of the project.
- **Submission Deadline:** 20 December 2024, 23:59. **No late submissions will be accepted.**
- **Deliverables:**
  - A report in PDF format covering:
    - Design decisions and assumptions.
    - Explanation of partitioning strategy used (Striped or Checkered).
    - Analysis of communication costs and synchronization.
    - Test results with sample inputs.
  - Source code with proper documentation.
  - A README file summarizing:
    - Compilation and execution instructions.

    * Any known issues or assumptions.
  - **Individual Contribution Document:**
    * Each student must submit a document titled `StudentNumber_WorkDone.pdf`, beginning with the line ”I worked on the following parts in this project:”.
    * The document should clearly explain the parts you have worked on in not less than 10 lines.
    * Do not write general comments such as ”we worked on the project together.” Be explicit about your contributions.

- **Code Quality:**
  - Your code should be self-explanatory by means of either choosing self-explanatory variable/function names or explicit comments.
  - Follow the good programming practices you have learned in your previous courses.
  - Use functions and loops to prevent/eliminate duplicate patterns in your code.

- **Tool Usage:**
  - Your submission must be your own work. Any similarity with another submission will be considered as cheating.
  - The use of Large Language Models (LLMs) and similar tools is prohibited in both project codes and project reports.

- **Submission Procedure:**
  - Each student in the group should upload a single zip file on Moodle, named `StudentNumber.zip`, containing:
    * `StudentNumber.pdf` (Project report)
    * `StudentNumber.py` (Program code)
    * `StudentNumber_WorkDone.pdf` (Individual contribution document)
    * `README.txt` (Summary and instructions)
  - Ensure all files follow the specified naming conventions.
  - Submit your deliverable as a zip file that includes your report and implementation codes in separate folders.

- **Academic Integrity:**
  - Each group must complete the project themselves without collaborating with other groups.
  - All code and reports must be your own work.
  - Plagiarism and unauthorized collaboration are strictly prohibited.