Improved Efficiency of Forest Fires Models Using MPI Parallelisation

Introduction

This report investigates a parallelised forest fire simulation, based on cellular automata principles and implemented using MPI (Message Passing Interface). The primary aims of the project are to simulate fire spread through a discretised forest grid, assess the impact of parallelisation on computational performance and evaluate model convergence, with respect to the number of runs M and tree density probability p.

The simulation models a forest as a two-dimensional grid, where each cell may be empty, contain a tree, or represent a burning tree. Fire spreads through the grid following cellular automata rules based on the Von Neumann neighbourhood. The model accommodates random grid generation via probabilistic tree placement, as well as user-defined or file-based input. Simulation outcomes include fire duration, whether the fire reaches the bottom row, and execution time across varying grid sizes and parallelisation levels.

Methodology

Implementation and Tools

The simulation was implemented in C++ and parallelised using MPI. The key algorithmic components include grid initialization with tree density, fire propagation using Von Neumann neighbourhood logic and domain decomposition for distributing work across MPI processes. Each process handles a horizontal section of the grid, exchanging boundary rows with neighbours to preserve model accuracy.

Model Architecture

The model operates on a square grid of size N×N, where each cell is assigned one of three states: 0 (empty), 1 (tree), or 2 (burning tree). Fire is initialised in the top row, with any trees present set alight. At each time step, trees adjacent via Von Neumann neighbours to burning trees catch fire, burning trees are removed and empty cells remain unchanged. The simulation halts when no trees are on fire.

Initialisation can occur through two modes: probabilistic tree generation from user-defined grid parameters, or a predefined grid from an input file. During execution, the model records the total and average number of steps until termination, the total time and the average elapsed time per run, the fraction of the fires that reaches the bottom of the grid and the world size.

MPI Parallelisation Strategy

The MPI implementation uses a hybrid approach, combining MPI_Scatter for initial data distribution and point-to-point communication for neighbour data exchange. The grid is partitioned among processes using vertical strip decomposition, with extra rows allocated to the first few ranks to account for non-uniform division. Each process independently initialises its local section of the forest grid or receives its assigned segment from the root process if the grid is centrally generated or read from a file.

Vertical strip decomposition was chosen over horizontal, due to the increased simultaneity of MPI tasks and over block decomposition due to the squared relationship of boundaries and decreased simultaneity of MPI tasks.

During each simulation step, every process performs a local update of its assigned grid cells based on Von Neumann neighbour rules. To correctly simulate interactions at the boundaries of process domains, each process exchanges its top and bottom rows with adjacent ranks using MPI_Sendrec. This ensures that border updates reflect burning trees in neighbouring partitions.

Termination of the simulation is globally coordinated using MPI_Allreduce to monitor the total number of burning trees. The same collective communication pattern is applied to determine whether the fire has reached the bottom row. Execution times are recorded locally and averaged over multiple runs to assess convergence and performance scaling. The root process collects and outputs summary statistics such as average step count, execution time and the frequency of fire reaching the bottom boundary.

Convergence Analysis

To assess the statistical convergence of the model, we examined how the mean number of steps before the fire burns out changes, with increasing number of repeat runs M. For each value of M, the percentage change in the mean value from the previous M was computed. This was repeated across a range of initial tree probabilities p, with grid size N = 100 held fixed.

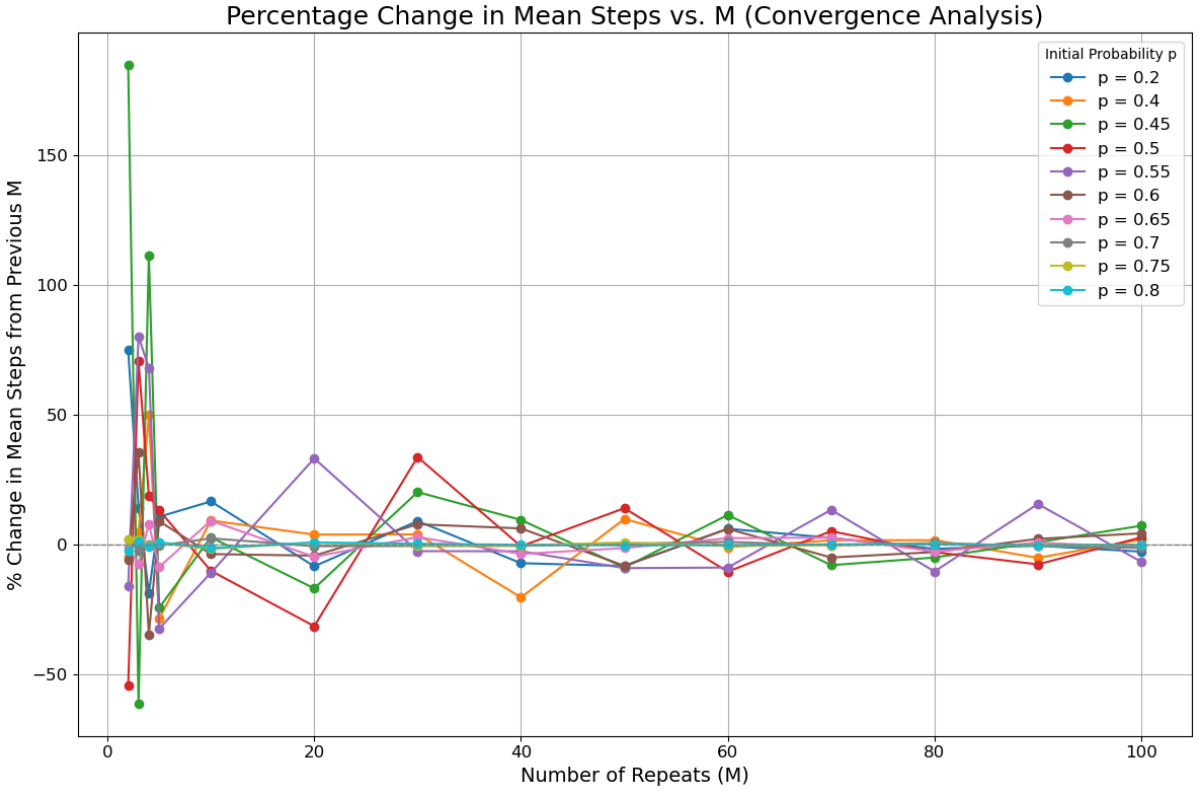

Figure 1 - Percentage change of steps versus Number of repeats M, percentage change in reference to previous M entry.

The results are shown in the plot above. For small M, the variation in mean step count is highly volatile, with percentage changes exceeding 100% for some p-values. However, beyond approximately M = 30, fluctuations stabilise and the percentage change falls consistently within a 20% difference, indicating that the model is beginning to converge.

Some key observations:

- For low probabilities such as p = 0.2, convergence is rapid due to the low density of trees, hence the shorter simulation runtimes.

- Near the critical threshold (p ≈ 0.6), the model shows greater variance and takes slightly longer to converge due to the complex percolation dynamics.
- For higher values of p, convergence is once again faster, likely due to saturation of the grid with trees and a more deterministic burn-through behaviour.

Overall, M = 50 appears to be enough repeats for most p-values, with the percentage change in mean steps remaining small (approx. 10% with the vast majority under 5%) and stable beyond this point. This validates the use of M = 50 for performance benchmarking and further analysis.

3.2 Performance Evaluation

To evaluate the parallel performance of the forest fire model, we measured two key metrics:

$$Speedup = \frac{T_{\{1\}}}{T_{\{p\}}}$$

$$Efficiency(n) = \frac{Speedup(n)}{n}$$

Simulations were run for three grid sizes: N = 50, 100, 500, each using p = 0.6 and M = 50. Performance was tested on up to 64 MPI tasks.

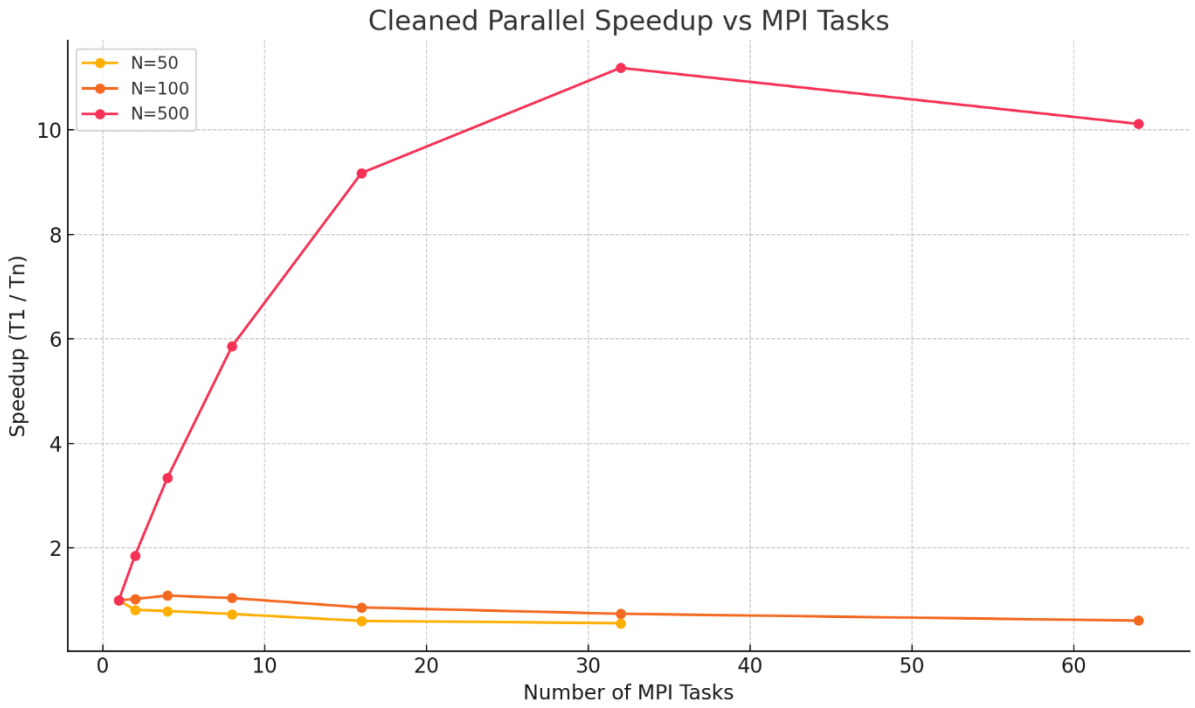

Figure 2 - Parallel Speedup vs. MPI Tasks for N=50, N=100 and N=500.

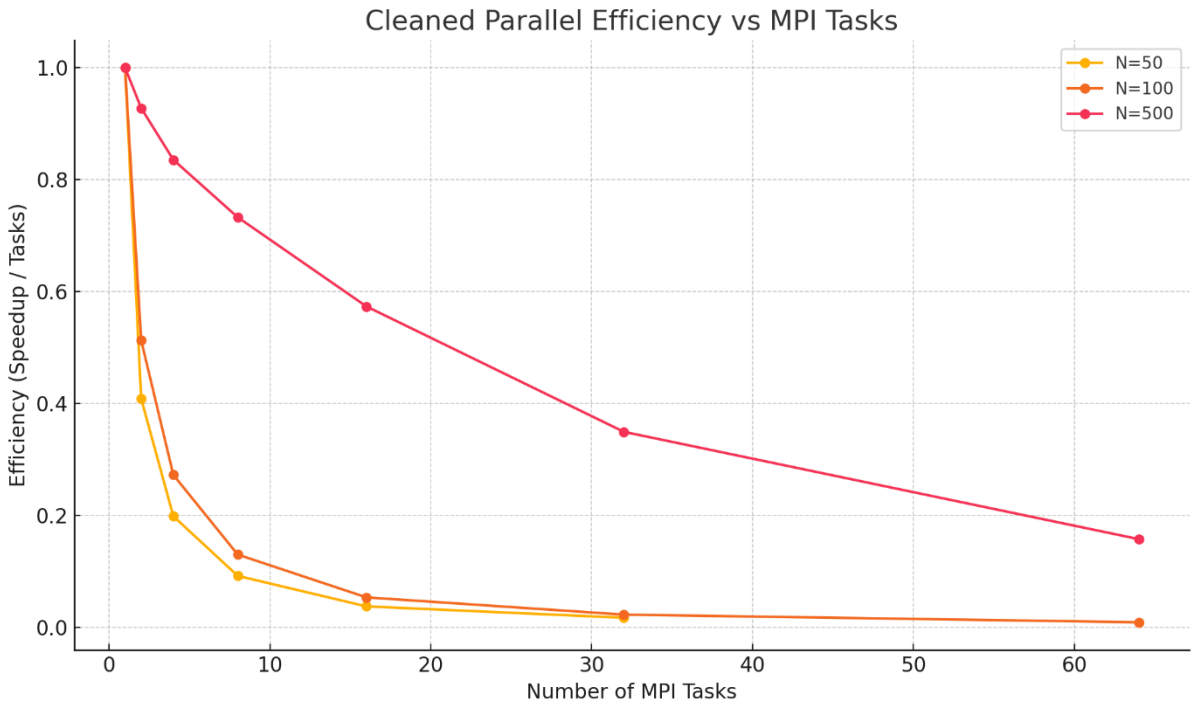

Figure 3 - Parallel Efficiency vs. MPI Tasks for N=50, N=100, and N=500.

Observations:

- For N = 500, the model scaled efficiently up to 32 MPI tasks, reaching a speedup of ~11. Efficiency stayed above 0.35, highlighting good use of parallel resources.
- For N = 100, performance plateaued earlier, with little gain beyond 8 tasks and efficiency rapidly declining after that point.
- For N = 50, parallel performance was poor across the board, with minimal speedup and efficiency dropping near zero after 4 tasks.

These results indicate that parallelisation is most beneficial for large grids, where the computational load per task is high enough to outweigh communication overhead.

Discussion

The convergence results demonstrate that averaging over 50 independent runs (M = 50) is generally sufficient to obtain reliable estimates of model behaviour. This is consistent across most values of p, although slightly more repeats might be warranted near the critical percolation threshold $p_c \approx 0.6$, where variance is naturally higher.

In terms of performance, the forest fire model displays strong scaling only for sufficiently large problem sizes. The parallel efficiency is directly tied to the grid size:

Larger grids (e.g. N = 500) allow for significant parallel speedup as the computation-to-communication ratio is high.

Small grids suffer from excessive communication overhead relative to the computational work, resulting in diminished or negative returns from parallelisation.

The use of a straightforward MPI communication strategy — row-wise domain decomposition with blocking MPI_Sendrecv — prioritised correctness and ease of implementation but limited scalability

due to synchronisation delays. This further reinforces the importance of matching the parallel strategy to both problem scale and available resources.

Conclusion

This project examined both statistical convergence and parallel performance of a cellular automaton forest fire model implemented in C++ with MPI.

- Convergence: For most probabilities p, the percentage change in the mean number of steps stabilised well below 10% by M = 50, validating this choice as a reasonable compromise between accuracy and runtime.
- Performance: The model benefits substantially from MPI-based parallelisation for large grids (up to 32 tasks for N = 500), but smaller grids fail to scale due to communication overhead.
- Best practices: Effective parallel performance is only achievable when the workload per task is sufficient to amortise communication costs, and careful selection of task count is critical to resource efficiency.

These insights are essential for optimising simulations of stochastic cellular automata where repeat runs, and large-scale data processing are required.

Future Work

Several extensions could be pursued to improve and generalise the current implementation:

- Advanced MPI Schemes: Introduce non-blocking communication (MPI_Isend, MPI_Irecv) to overlap communication and computation and reduce idle time.
- 2D Domain Decomposition: Extend the current row-wise slicing to block-wise (2D) partitioning to reduce communication volume and balance workloads better.
- Hybrid MPI + OpenMP Parallelism: Implement hybrid parallelisation to take advantage of multi-core nodes within HPC clusters.
- Profiling and Optimisation: Apply tools like Valgrind or perf to identify bottlenecks and optimise critical communication paths.
- Larger Problem Sizes: Extend the analysis to N > 1000 and 128+ tasks to explore limits of scalability and assess the feasibility of large-scale simulations.
- Alternative Initialisation Strategies: Investigate the impact of different fire ignition patterns and heterogeneous tree distributions on both convergence and parallel performance.
- These directions offer opportunities to enhance the model's robustness, scalability, and realism, making it a more powerful tool for exploring stochastic processes in computational science.