Wildfire-sim Project Proposal - Joshua Brown (jdb2252)

Summary

Parallelize a cellular automaton-based wildfire spread simulation on a 2D grid. The simulation models fire propagation across terrain based on fuel levels, wind direction, terrain type, and neighboring cell states.

Background & Motivation

Wildfire modeling is crucial for disaster preparedness and resource allocation. While professional models use complex physics, cellular automaton approaches provide computationally tractable alternatives that still capture realistic fire spread patterns used in rapid assessment scenarios.

This project is inspired by the increasing frequency and severity of wildfires globally. The grid-based nature with local cell interactions makes it naturally suited for parallel processing, while stochastic elements and multiple influencing factors provide interesting algorithmic challenges.

Serial Algorithm

The simulation uses a 2D grid where each cell represents a land area (30m × 30m) with:

- **State**: Unburned, Burning, or Burned
- **Fuel level**: 0.0 to 1.0 (vegetation density)
- **Terrain type**: Forest, grassland, water, urban
- **Elevation**: For slope effects

Each time step:

1. For each burning cell, examine adjacent neighbors
2. Calculate ignition probability for unburned neighbors:

```
P_ignite = base_prob × fuel_factor × wind_factor × slope_factor × random_factor
```

3. Apply state transitions (Unburned → Burning → Burned)
4. Track statistics (burned area, spread rate)

Fire spreads based on fuel availability, wind (faster downwind), and slope (faster uphill). The algorithm uses **Data.Vector** for grid storage and iterates until fire extinguishes or reaches max time steps.

Parallelization Plan & Anticipated Difficulties

Some ideas:

1. **Spatial domain decomposition**: Partition grid into regions, each thread updates its region in parallel, synchronize between time steps
2. **Red-black checkerboard**: Process alternating cell patterns (red then black) to eliminate dependencies within each phase
3. **Par monad**: Explicit task creation for finer control over granularity

Functional-specific ideas:

- Use Strategies library (**parMap, parListChunk**) for coarse-grained parallelism
- Implement with REPA arrays for automatic stencil parallelization
- Have threads process N local steps before synchronizing with shared grid

Anticipated difficulties:

- **Boundary synchronization**: Cells at region edges need neighbor data from other regions
- **Load imbalance**: Active fire regions have more work than empty regions
- **Random number generation**: Need reproducible parallel RNG for testing
- **Stochastic validation**: Non-deterministic results require statistical testing
- **Convergence detection**: Coordinating when fire is completely extinguished

Input Data

Programmatically generated. Wildfire simulations are expensive even for 500×500 grids, so I/O is not a bottleneck. I'll also use Test scenarios (all generated in code with deterministic seeds for reproducibility)