15-618 Project Proposal: Parallel Simulation of Heat Diffusion

Group: Akshay Joshi (akshayjo) & Amogha Hegde (amoghah)

URL: https://akshayjo1811.github.io/15-618-project/

Summary:

We will develop a 2D heat diffusion simulation using finite difference methods, implement parallel versions using both shared address space and message passing programming models, and conduct a detailed performance analysis comparing their scalability, efficiency, and execution characteristics on multi-core CPU clusters available in the GHC lab.

Background:

Heat diffusion is a fundamental physical phenomenon described by the heat equation, a partial differential equation that models how temperature distributes over time in a medium:

$$\partial T/\partial t = \alpha \nabla^2 T$$

Where T is temperature, t is time, α is the thermal diffusivity constant, and ∇^2 is the Laplacian operator. In two dimensions, this equation becomes:

$$\partial T/\partial t = \alpha(\partial^2 T/\partial x^2 + \partial^2 T/\partial y^2)$$

To solve this equation numerically, the finite difference method (FDM) is commonly used. The domain is discretized into a grid, and the partial derivatives are approximated using difference equations. The explicit scheme for the 2D heat equation is:

$$T(i,j)^{(n+1)} = T(i,j)^n + \alpha\Delta t/\Delta x^2[T(i+1,j)^n + T(i-1,j)^n + T(i,j+1)^n + T(i,j-1)^n - 4T(i,j)^n]$$

Where i,j are spatial indices, n is the time step, Δt is the time step size, and Δx is the spatial step size.

This computation is parallelizable because the temperature update at each grid point depends only on its previous value and its neighbors' previous values. The computation can be distributed across multiple processors by dividing the grid into subdomains, with each processor responsible for updating its assigned subdomain. The sequential pseudocode may look something like this:

```
1. Initialize temperature grid with boundary conditions
2. For each time step:
   a. For each grid point (i,j) in a group:
      - Calculate new temperature using the finite difference formula
      - Update grid with new temperatures
```

The Challenge:

While each grid point's update depends only on its neighbors, this creates communication requirements at subdomain boundaries. The main challenge here is the diffusion changes for each particle over time which means the particles in the grid do not have fixed locations, which has to be accounted for making the problem dynamic. These ghost regions must be exchanged between processors. Hence, communication/synchronization overhead will be introduced. Nonetheless, the algorithm exhibits good spatial locality since it accesses neighboring grid points, but the efficiency depends on how the domain is partitioned. For large problem sizes, the ratio of computation to communication is favorable. However, as we increase the number of processors, the subdomain size decreases, potentially increasing the relative communication overhead. Ensuring each processor has an equal amount of work is crucial for performance. Heat diffusion with uniform properties is naturally balanced, but non-uniform domain decomposition or irregular boundary conditions can create imbalances.

This has to be repeated in iteration with a scenario something like this. Let's suppose we have 'n' processors and one processor is assigned to update 1000 cells totally, and in each iteration each processor works on let's say 10 cells after which it has to get the updated matrix which has all the other 10 cells done by other processors to ensure better accuracy. This stage we will have to keep the matrix shared to have communication between processors or each one has to have the entire local copy and data has to be synced with them. We will try out both these approaches and compare the performance and accuracy and note the tradeoffs.

Resources:

The simulator will be built from scratch; nevertheless, if we happen to find a starter code in C/C++ that seems to model heat diffusion, we would utilize it and model it accordingly. Additionally, we are referring to online papers to understand heat distribution behavior.

Goals and Deliverables

Plan to Achieve:

- ➔ Implement a sequential 2D heat diffusion simulation using the finite difference method with various boundary conditions.
- ➔ Develop two parallel implementations - Shared memory parallelization using OpenMP and Distributed memory parallelization using MPI.
- ➔ Conduct a comprehensive performance analysis based on communication overhead, scaling efficiency, synchronization overhead, workload balance.

Hope to Achieve:

- ➔ Implement a hybrid OpenMP+MPI approach and compare its performance with the pure OpenMP and pure MPI.

Platform Choice:

X86-64 multi-core cpu architecture with support for OpenMP & MPI libraries; source code in C/C++

Rough Schedule:

Week 1: Setup, approach design finalise and sequential implementation

Week 2: OpenMP implementation

Week 3: MPI implementation

Week 4: Hybrid implementation of MPI and OpenMP and analysing tradeoffs between them

Week 5: Benchmark testing and final report