Scalable Cloud-in-Cell Interpolation Using OpenMP

1 Problem Statement

You are given a set of scattered points in a 2D domain with known values:

$$S = \{(x_i, y_i, f_i) \mid i = 1, 2, \ldots, N\}$$

The objective is to interpolate these values onto a structured $M \times M$ mesh using the Cloud-in-Cell (CIC) method. For simplicity, each point has $f_i = 1$.

2 Structured Mesh Description

The target structured mesh consists of regularly spaced grid points:

$$G = \{(X_i, Y_j) \mid X_i = i\Delta x,\ Y_j = j\Delta y,\ i, j = 0, 1, \ldots, M-1\}$$

where:

$$\Delta x = \frac{X_{\max}}{M}, \quad \Delta y = \frac{Y_{\max}}{M}$$

The domain is normalized such that $0 \le x_i \le X_{\max}$ and $0 \le y_i \le Y_{\max}$.

3 Interpolation Technique

Using bilinear interpolation, each scattered point contributes to four surrounding grid points in the mesh. For a point (x, y) in a cell with corners:

- (X_i, Y_j)
- (X_{i+1}, Y_j)
- (X_i, Y_{j+1})
- (X_{i+1}, Y_{j+1})

Weights are computed as:

$$\begin{aligned}
dx = \frac{x - X_i}{\Delta x}, &\quad dy = \frac{y - Y_j}{\Delta y} \\
w_{i,j} &= (1 - dx)(1 - dy) \\
w_{i+1,j} &= dx(1 - dy) \\
w_{i,j+1} &= (1 - dx)dy \\
w_{i+1,j+1} &= dx \cdot dy
\end{aligned}$$

Each of these weights is used to proportionally distribute the value of f_i to the corresponding grid points.

4 Parallelization Objectives

The primary goal of this project is to optimize the interpolation process for high-performance computing platforms by:

- Exploiting thread-level parallelism using OpenMP
- Avoiding race conditions through thread-local data buffers
- Improving spatial locality to enhance cache performance
- Achieving strong scalability across increasing thread counts

5 Implementation Steps

1. Read input binary file containing grid and particle information.
2. For each iteration, load a new set of scattered points.
3. Use bilinear interpolation to deposit values onto the grid using thread-local arrays.
4. Reduce per-thread contributions to obtain the final interpolated mesh.
5. Write the output mesh to a file.

6 Applications

- Scientific simulations: e.g., fluid dynamics, astrophysical simulations
- Computer graphics and surface reconstruction from point clouds
- Meteorological and oceanographic visualization from sensor data
- Medical imaging: reconstruction from CT or MRI scans
- Data preprocessing in machine learning for sparse sensor networks