Parallel Design of Particle-in-Cell Method

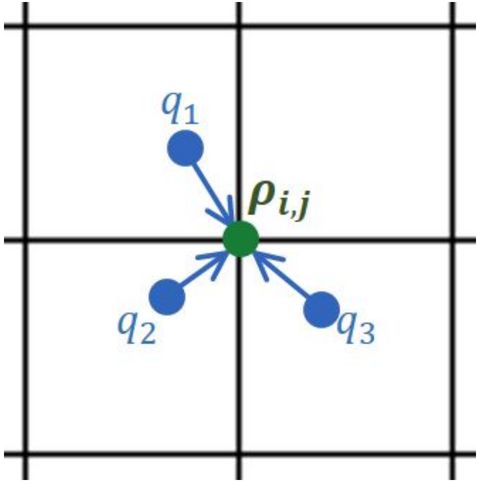
CS 205

Kevin Howarth, Aditi Memani, Hari Raval, Taro Spirig

Sequential Baseline

PIC Algorithm

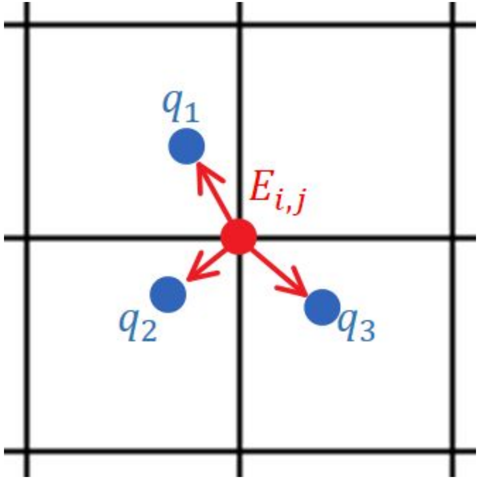
1. Interpolate from Particles to Mesh



2. Solve Discrete Poisson Equation on Mesh

$$\nabla^2 \phi = -\rho, \qquad \nabla \phi = E$$

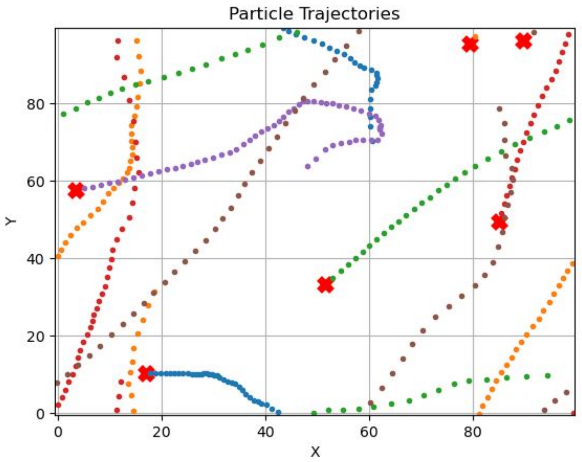
3. Interpolate from Mesh back to Particles

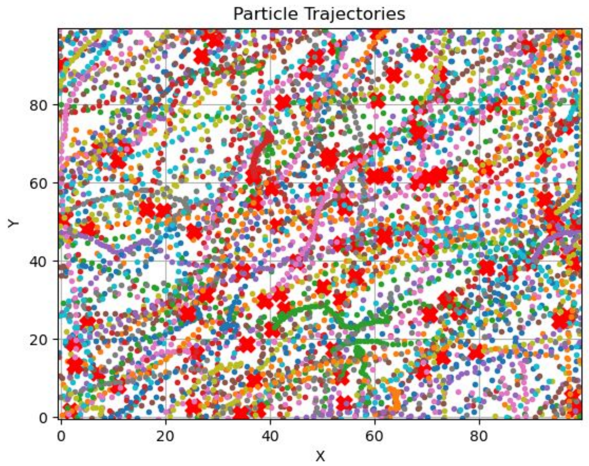


4. Time-step Particle Locations

$$\frac{d\overline{v}}{dt} = qE(\overline{x}), \qquad \frac{d\overline{x}}{dt} = \overline{v}$$

Preliminary Results





Time ⟶ 1.73 s
Number of particles (N_p) ⟶ 6
Number of grid points (N_g) ⟶ 64

Time ⟶ 4.03 s
Number of particles (N_p) ⟶ 100
Number of grid points (N_g) ⟶ 100

All results obtained on Intel Broadwell node on FASRC

Profiling

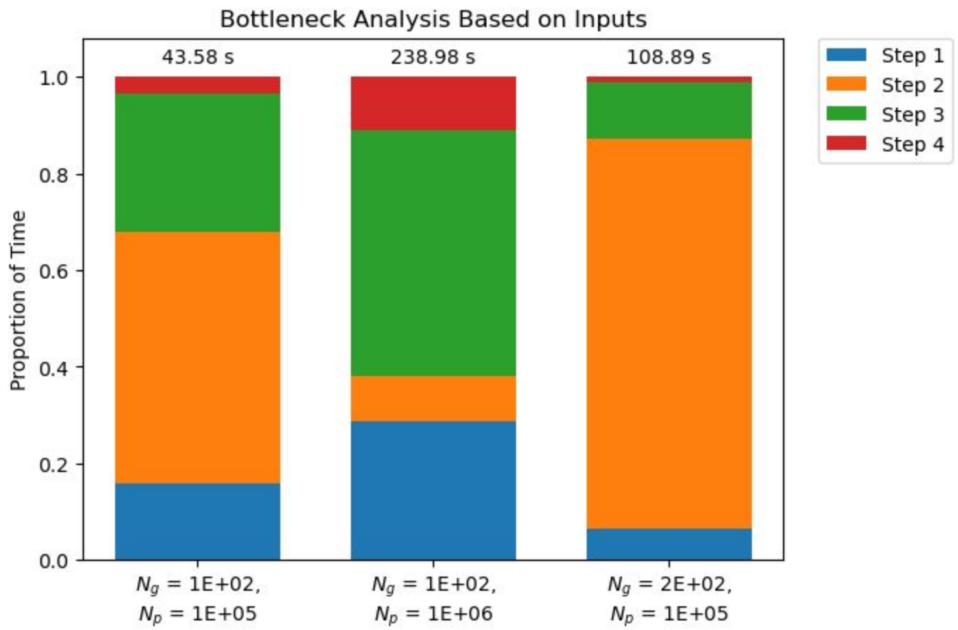
Bottleneck Analysis

Step 1: P2M

Step 2: Gauss-Seidel

Step 3: M2P

Step 4: Timestep



**runtime averaged over 100 algorithm steps*

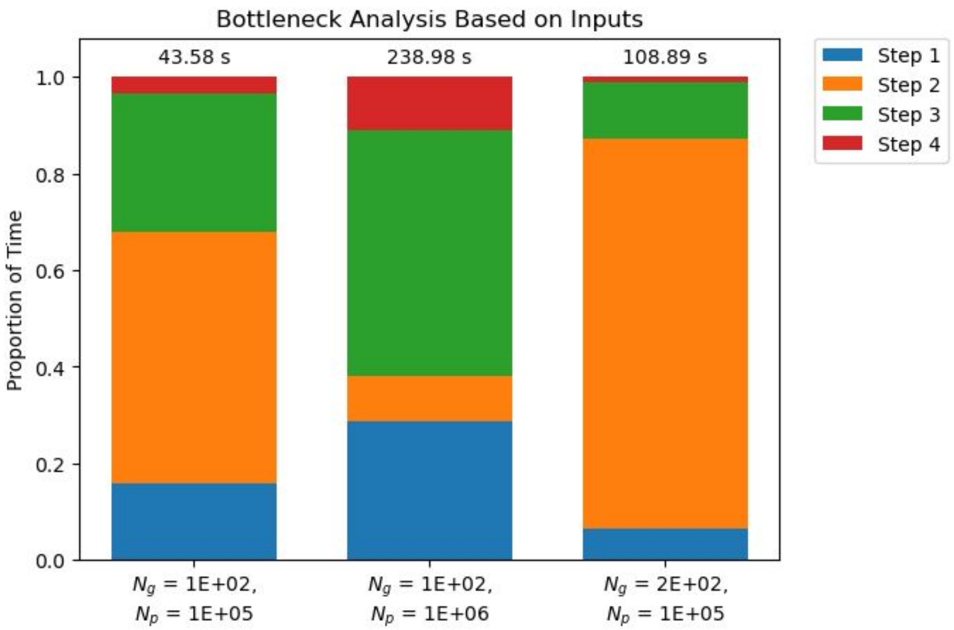
Bottleneck Analysis

Step 1: P2M

Step 2: Gauss-Seidel

Step 3: M2P

Step 4: Timestep



**runtime averaged over 100 algorithm steps*

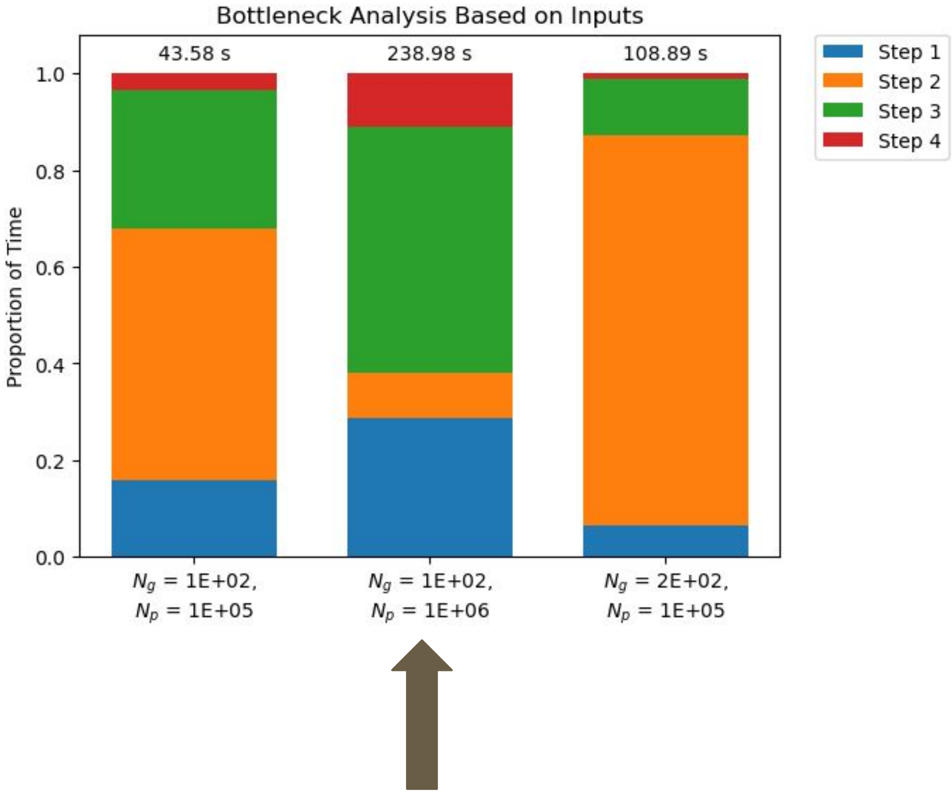
Bottleneck Analysis

Step 1: P2M } **_Bottleneck_**

Step 2: Gauss-Seidel

Step 3: M2P } **_Bottleneck_**

Step 4: Timestep



**runtime averaged over 100 algorithm steps*

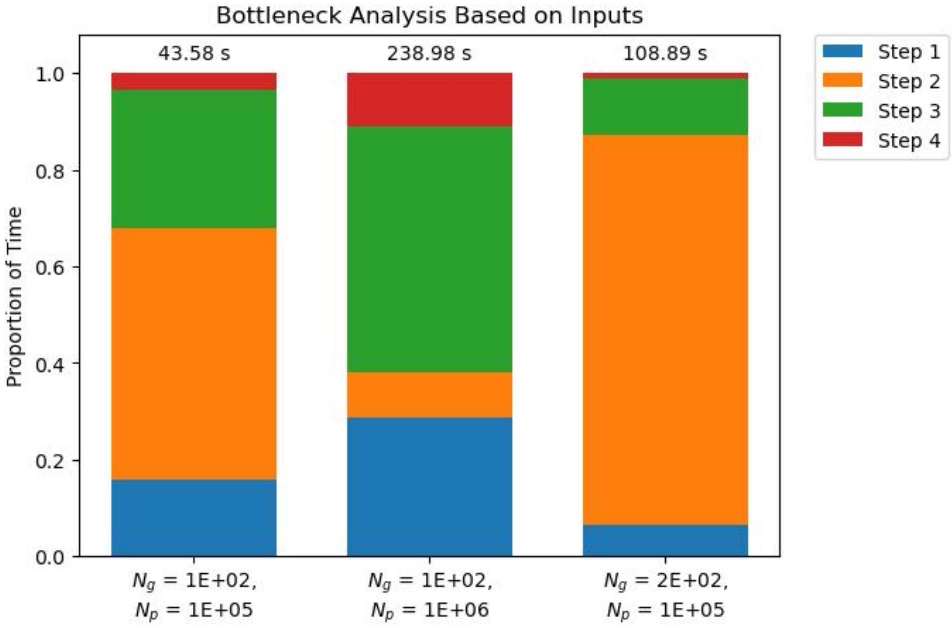
Bottleneck Analysis

Step 1: P2M

Step 2: Gauss-Seidel } **_Bottleneck_**

Step 3: M2P

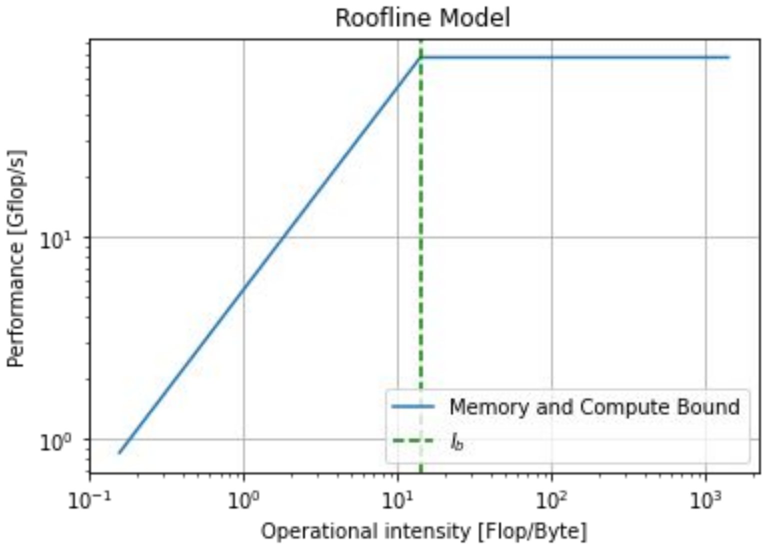
Step 4: Timestep



**runtime averaged over 100 algorithm steps*

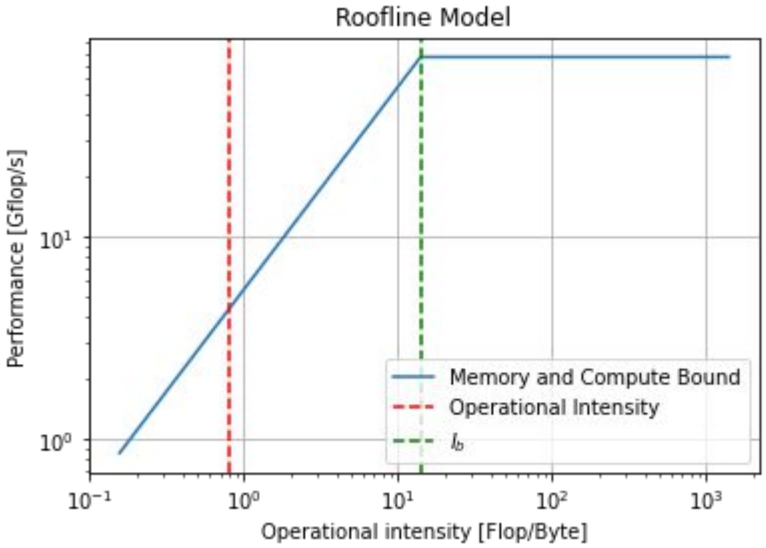
Roofline Model

- **Broadwell Architecture:**
  - Single Precision: π = 1075.2 Gflop/s
  - Peak memory bandwidth: β = 76.8 GB/s
  - Ridge Point: I_b = 14.0
- **Memory Bound:**
  - $I < I_b$
- **Compute Bound:**
  - $I > I_b$



Roofline Model

- **Algorithm Complexity for Gauss-Seidel Red Black: $O(n^2)$**
  - **Black**: n(n/2)
  - **Red**: n(n/2)
- **Number of Operations: 19 flops**
  - Addition/Subtraction: 16
  - Multiplication: 3
- **Memory Performance: 6 memory operations**
  - Read: 5
  - Write: 1
- **Operational Intensity: I = 19/24**
  - Our problem is memory bound: $I < I_b$



Parallel Model

Parallel Model

- **OpenMP and Shared Memory Model**
  - Limitations to this model as the density of the grid and the number of particles increase

Data Structures

Particles

```
Particle() {

        std:: vector < float > xx;
        std:: vector < float > xy;
        std:: vector < float > vx;
        std:: vector < float > vy;
        std:: vector < float > Ex;
        std:: vector < float > Ey;
}
```

P_1	P_2	P_3	P_4	P_5	...	P_n

$\in \mathbb{R}^{N_p}$

```
std:: vector < Particle > Particles;
```

Data Structures

Particles

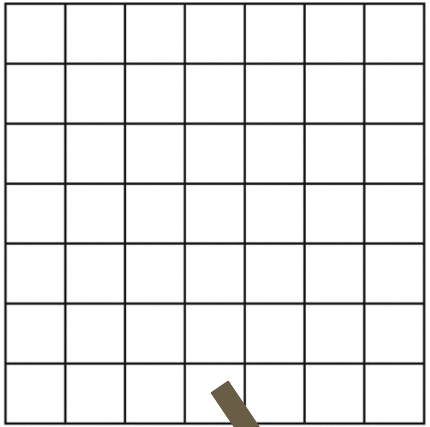
```
Particle() {

        std:: vector < float > xx;
        std:: vector < float > xy;
        std:: vector < float > vx;
        std:: vector < float > vy;
        std:: vector < float > Ex;
        std:: vector < float > Ey;
}
```

P_1	**P_2**	**P_3**	**P_4**	**P_5**	**...**	**P_n**

$\in \mathbb{R}^{N_p}$

std:: vector < Particle > Particles;

Computational Mesh



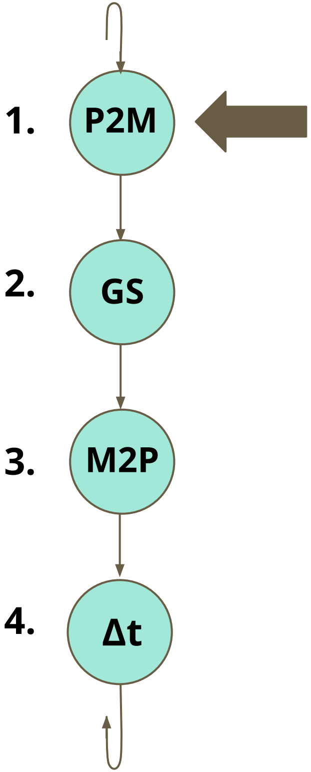
Charge	**Potential**	**Electric Field**

$\in \mathbb{R}^{N_g \times N_g}$

*float** qGrid, pGrid, eGrid;*

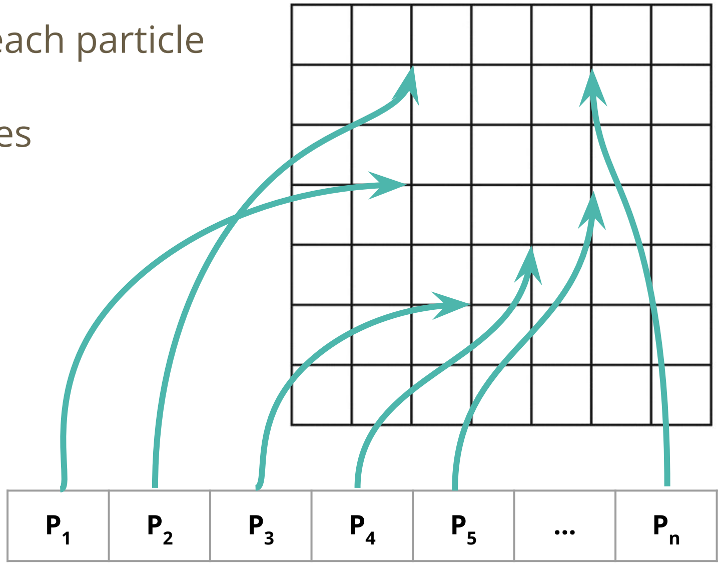
Parallel Design

Algo. Steps



Parallel Design of P2M

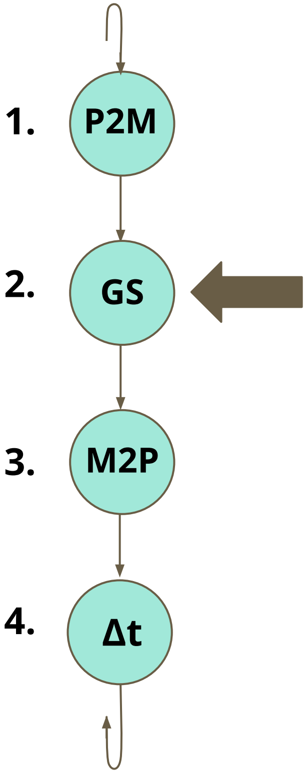
- Interpolation is independent for each particle
- Parallelize over number of particles



Read charge from particles, write to grid

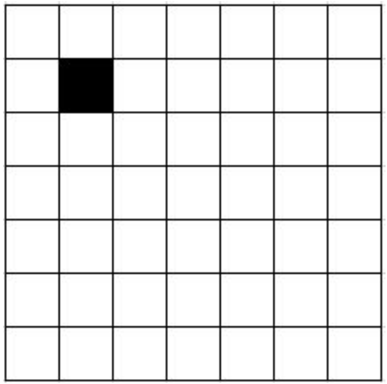
Parallel Design

Algo. Steps



Parallel Design of Gauss-Seidel

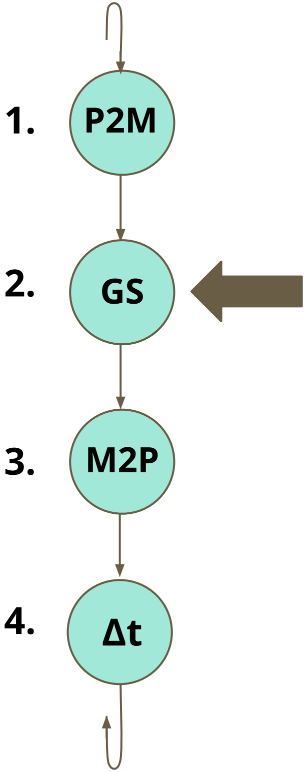
- Gauss-Seidel corresponds to stencil: read from neighbors, write to self
- Parallelization over number of grid points
- Red-Black ordering:
  1. Parallelize over black step
  2. Synchronize
  3. Parallelize over red step
  4. Synchronize



To get new value at black point...

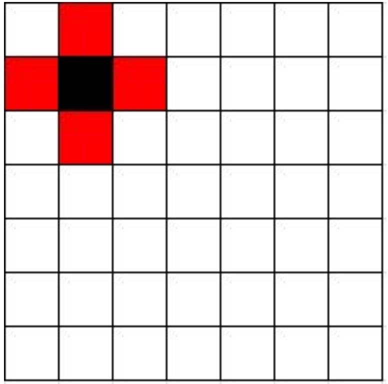
Parallel Design

Algo. Steps



Parallel Design of Gauss-Seidel

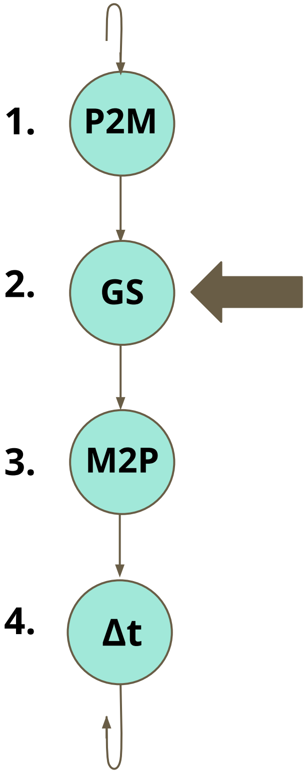
- Gauss-Seidel corresponds to stencil: read from neighbors, write to self
- Parallelization over number of grid points
- Red-Black ordering:
  1. Parallelize over black step
  2. Synchronize
  3. Parallelize over red step
  4. Synchronize



...read from red neighbors and write to self.

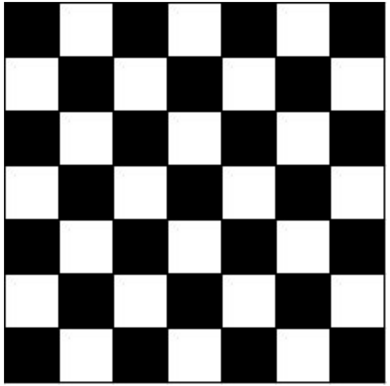
Parallel Design

Algo. Steps



Parallel Design of Gauss-Seidel

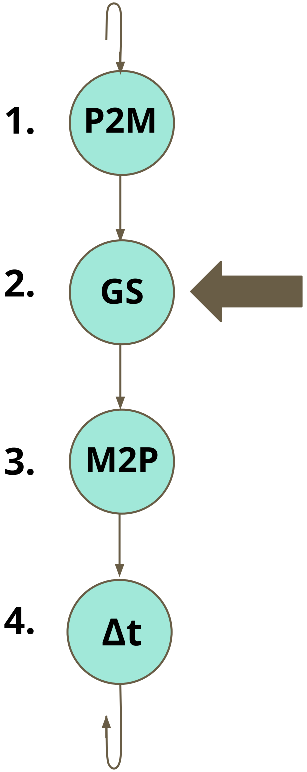
- Gauss-Seidel corresponds to stencil: read from neighbors, write to self
- Parallelization over number of grid points
- Red-Black ordering:
  1. Parallelize over black step
  2. Synchronize
  3. Parallelize over red step
  4. Synchronize



All black points only read from neighbors and write to self - no race condition!

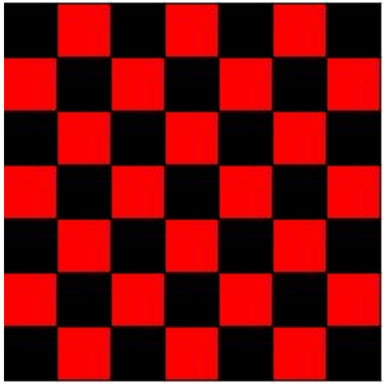
Parallel Design

Algo. Steps



Parallel Design of Gauss-Seidel

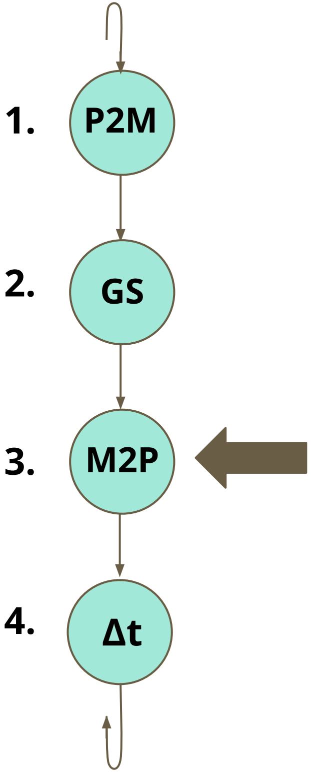
- Gauss-Seidel corresponds to stencil: read from neighbors, write to self
- Parallelization over number of grid points
- Red-Black ordering:
    1. Parallelize over black step
    2. Synchronize
    3. Parallelize over red step
    4. Synchronize



All red points only read from neighbors and write to self - no race condition!

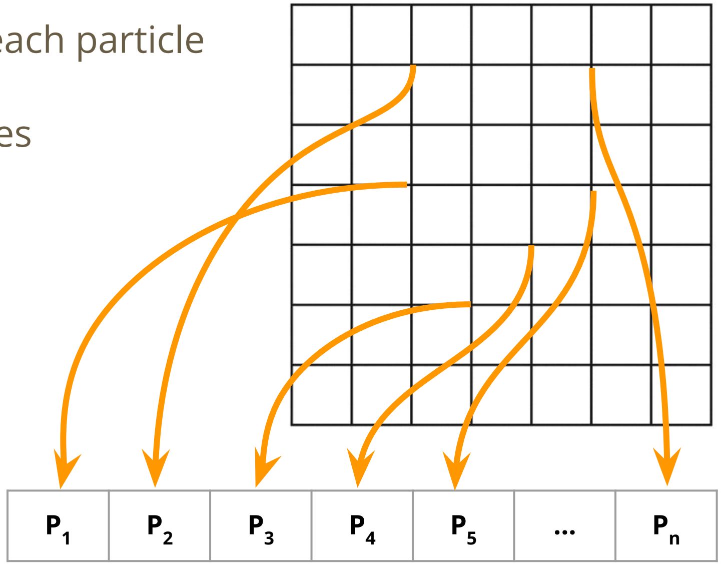
Parallel Design

Algo. Steps



Parallel Design of M2P

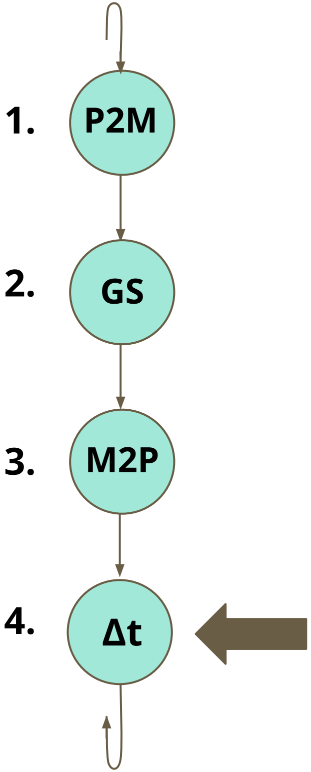
- Interpolation is independent for each particle
- Parallelize over number of particles



Read electric field from grid, write to particles

Parallel Design

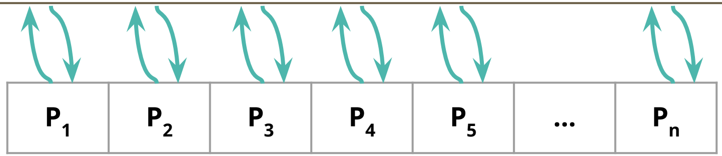
Algo. Steps



Parallel Design of Timestepping

- Timestep particles via Leapfrog method
- Timestepping is independent for each particle
- Parallelize over number of particles

$$v_{t+1} = v_t + \Delta t a_t, \qquad x_{t+1} = x_t + \Delta t v_{t+1}$$

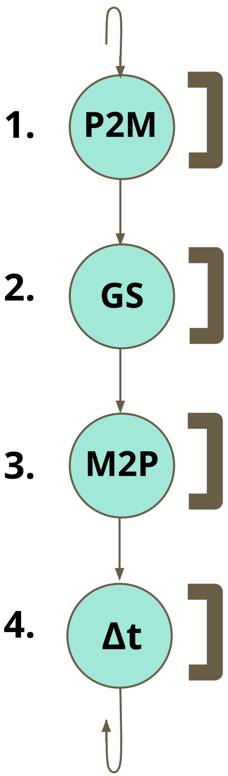


Particle reads from and writes to self

Implementation

Implementation

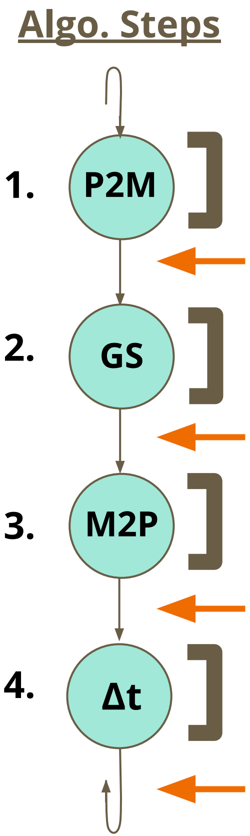
Algo. Steps



Parallel Design Considerations

- Parallelize individual steps of algorithm ( ])

Implementation

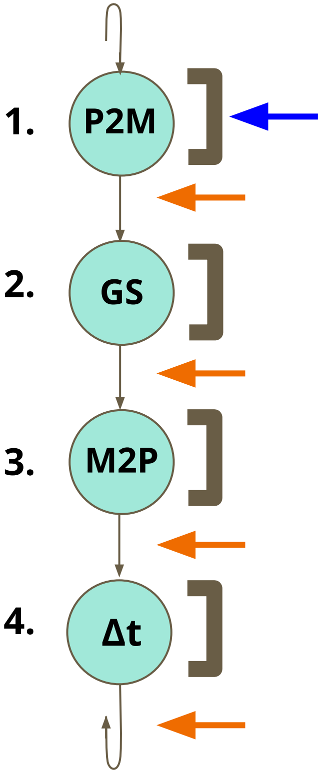


Parallel Design Considerations

- Parallelize individual steps of algorithm ( ])
- Synchronization necessary between steps (←)

Implementation

Algo. Steps

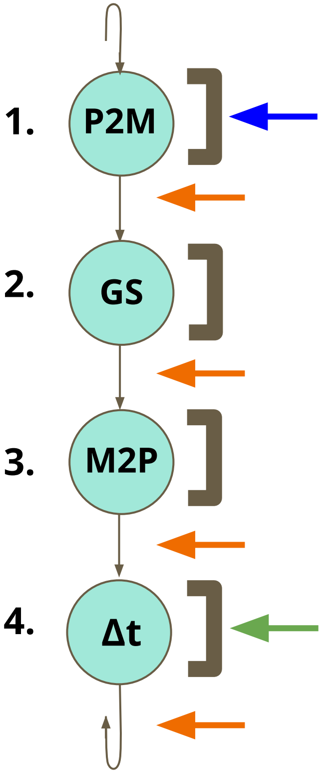


Parallel Design Considerations

- Parallelize individual steps of algorithm ( )
- Synchronization necessary between steps ( )
- Possible race condition in P2M - Multiple particles interpolate to same mesh point ( )
  - Solution: thread-private grids, reduce at end

Implementation

Algo. Steps



Parallel Design Considerations

- Parallelize individual steps of algorithm ( )
- Synchronization necessary between steps ( )
- Possible race condition in P2M - Multiple particles interpolate to same mesh point ( )
  - Solution: thread-private grids, reduce at end
- Possible load imbalance in time-stepping - particles may require multiple periodic wraparounds ( )
  - Solution: dynamic scheduling

Thank You