Algorithmic Development for PIC codes on Advanced Architectures: OpenMP

Viktor K. Decyk and Tajendra V. Singh

UCLA

Particle-in-Cell Codes

Simplest plasma model is electrostatic:

1. Calculate charge density on a mesh from particles:

$$\rho(\boldsymbol{x}) = \sum_i q_i S(\boldsymbol{x} - \boldsymbol{x}_i)$$

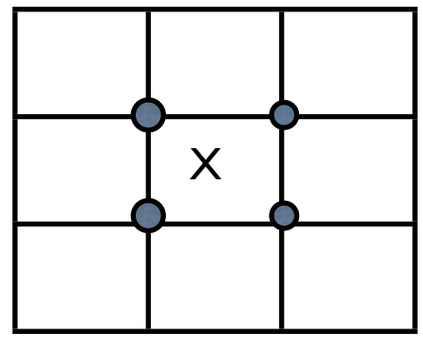
2. Solve Poisson's equation:

$$\nabla \cdot \boldsymbol{E} = 4\pi\rho$$

3. Advance particle's co-ordinates using Newton's Law:

$$m_i \frac{d\boldsymbol{v}_i}{dt} = q_i \int \boldsymbol{E}(\boldsymbol{x}) S(\boldsymbol{x}_i - \boldsymbol{x})\, d\boldsymbol{x} \qquad \frac{d\boldsymbol{x}_i}{dt} = \boldsymbol{v}_i$$

Inverse interpolation (scatter operation) is used in step 1 to distribute a particle's charge onto nearby locations on a grid.



Interpolation (gather operation) is used in step 3 to approximate the electric field from grids near a particle's location.

When running in parallel, data collisions might occur

Designing New Particle-in-Cell (PIC) Algorithms for Shared Memory

Most important bottleneck is memory access
- PIC codes have low computational intensity (few flops/memory access)
- Memory access is irregular (gather/scatter)

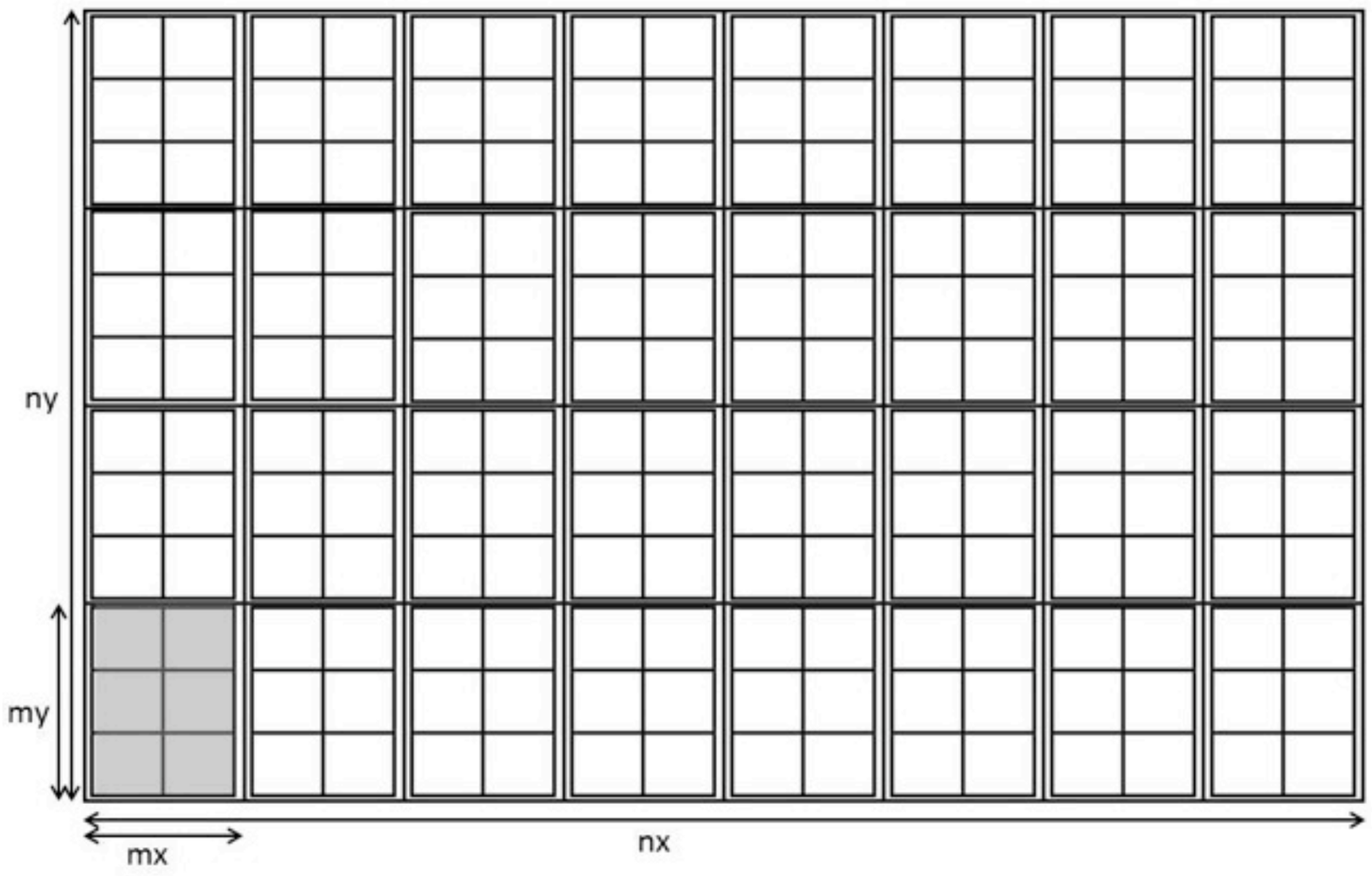
Memory access can be optimized with a streaming algorithm (global data read only once)

PIC codes can implement a streaming algorithm by keeping particles ordered by small tiles
- Minimizes global memory access since field elements need to be read only once.
- Cache is not needed, gather/scatter can be avoided.
- Deposit and particles update can have optimal stride 1 access.

Designing New Particle-in-Cell (PIC) Algorithms:
Particles ordered by tiles, varying from 2 x 2 to 16 x 16 grid points

We created a new data structure for particles, partitioned among threads blocks:

```
dimension ppart(idimp,npmax,num_tiles)
```



Designing New Particle-in-Cell (PIC) Algorithms: **Push/Deposit Procedures**:

Within a tile, all particles read or write the same block of fields.
- Before pushing particles, copy fields to fast memory
- After depositing charge to fast memory, write to global memory
- Different tiles can be done in parallel.

Each tile contains data for the grids in the tile, plus guard cells: an extra column of grids on the right, and an extra row of grids on the bottom for linear interpolation.

For push, parallelization is easy, each particle is independent of others, no data hazards
- Similar to MPI code, but with tiny partitions

For deposit, parallelization is relatively easy if each tile is controlled by one thread
- Care has to be taken in adding fast tile memory to global memory for the guard cells
- Atomic updates (which treat an update as an uninterruptible operation) are one approach

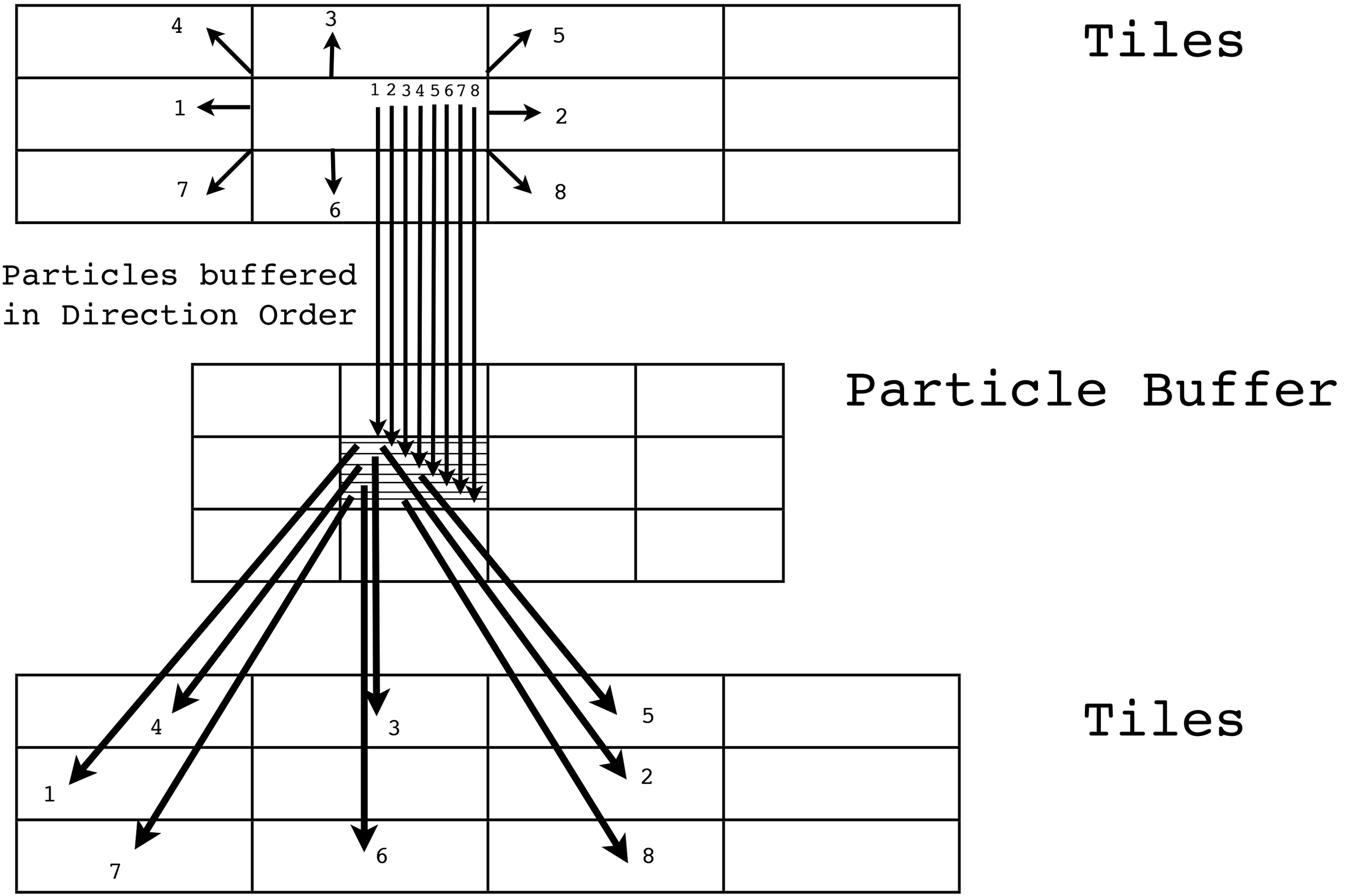
Designing New Particle-in-Cell (PIC) Algorithms: **Maintaining Particle Order**

Three steps:
1. Particle Push creates a list of particles which are leaving a tile
2. Using list, each thread places outgoing particles into a buffer it controls
3. Using lists, each tile copies incoming particles from buffers into particle array

- Less than a full sort, low overhead if particles already in correct tile
- Can be done in parallel
- Essentially message-passing, except buffer contains multiple destinations

In the end, the particle array belonging to a tile has no gaps
- Particles are moved to any existing holes created by departing particles
- If holes still remain, they are filled with particles from the end of the array

Particle Reordering



Evaluating New Particle-in-Cell (PIC) Algorithms with OpenMP: **Electrostatic Case**
2D ES Benchmark with 2048x2048 grid, 150,994,944 particles, 36 particles/cell
Tile size = 16x16. Single precision

```
Hot Plasma results with dt = 0.1
              CPU:Intel i7   OpenMP(12 cores)
Push             20.7 ns.       1.92 ns.
Deposit           9.4 ns.       0.99 ns.
Reorder           1.6 ns.       0.20 ns.
Total Particle   31.7 ns.       3.11 ns.

The time reported is per particle/time step.
The total particle speedup with 12 cores was 10.2 compared to 1 core
Field solver takes an additional 9%.
```

Evaluating New Particle-in-Cell (PIC) Algorithms on OpenMP: **Electromagnetic Case**
2-1/2D EM Benchmark with 2048x2048 grid, 150,994,944 particles, 36 particles/cell
Tile size = 16x16. Single precision

```
Hot Plasma results with dt = 0.04, c/vth = 10, relativistic
              CPU:Intel i7    OpenMP(12 cores)
Push             68.4 ns.        6.00 ns.
Deposit          43.2 ns.        4.29 ns.
Reorder           0.6 ns.        0.10 ns.
Total Particle  112.2 ns.       10.39 ns.

The time reported is per particle/time step.
The total particle speedup with 12 cores was 10.8x compared to 1 core.
Field solver takes an additional 8-10%.
```

Further information available at:

V. K. Decyk and T. V. Singh, "Particle-in-Cell Algorithms for Emerging Computer Architectures," Computer Physics Communications, 185, 708, (2014), available at http://dx.doi.org/10.1016/j.cpc.2013.10.013.

http://www.idre.ucla.edu/hpc/research/

Source codes available at:
https://idre.ucla.edu/hpc/parallel-plasma-pic-codes/