Fermi National Accelerator Laboratory
Office of Science / U.S. Department of Energy
Managed by Fermi Research Alliance, LLC

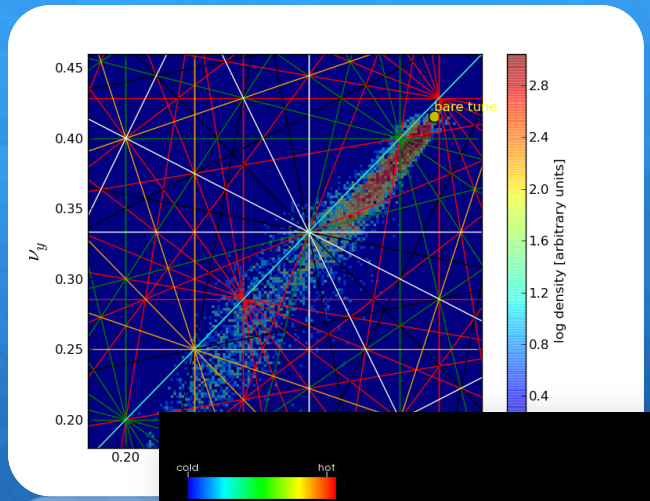
Synergia-CUDA

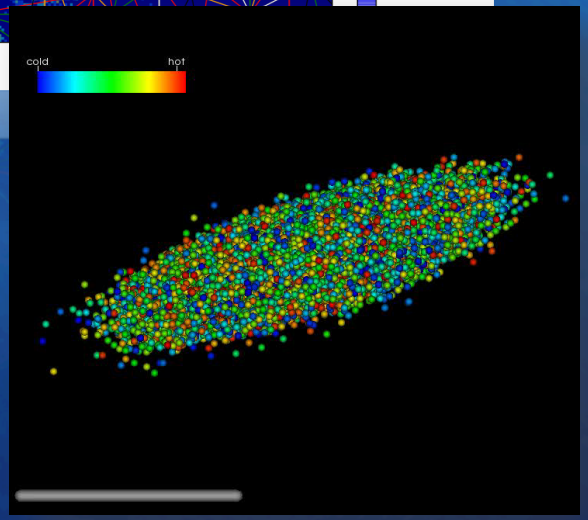
GPU Accelerated Accelerator Modeling Package

Qiming Lu & Jim Amundson
Fermilab

Synergia

Accelerator Modeling and Simulation Package





- Developed by Accelerator Simulation group at Fermilab
- Beam dynamics simulation
  - Single-particle dynamics
  - Multi-particle dynamics where the particle-particle interactions are important :

    Space charge / beam-beam interaction / electron cloud / wakefields

+ Parallel, 3d space charge Particle-in-cell (PIC) code for very large scale accelerator simulations

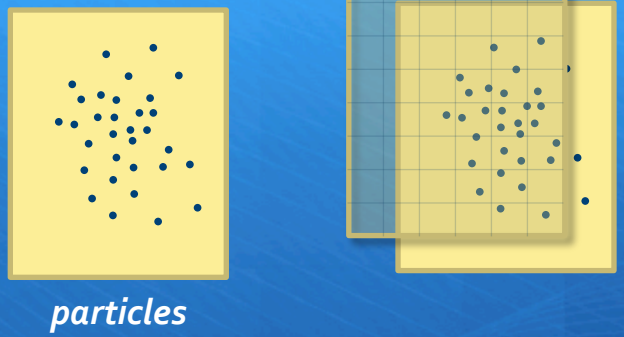
+ ~ 100 million to 1 billion particles

Synergia
Accelerator Modeling and Simulation Package

Outlines

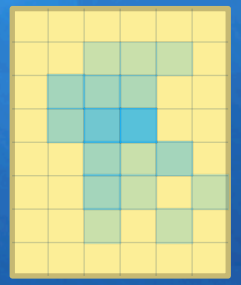
- Particle-in-Cell (PIC) algorithm
- Parallel decomposition schemes for PIC
- High level (MPI) parallelization and optimizations for PIC algorithm
- Fine level (thread) parallelization using CUDA
- Results and conclusions

Procedures of Particle-in-cell Method



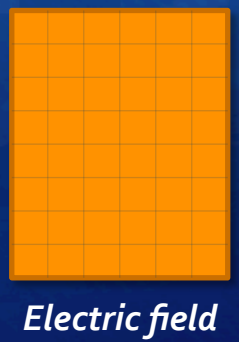
Interpolation of the charges to the filed mesh

1. Deposit Charge



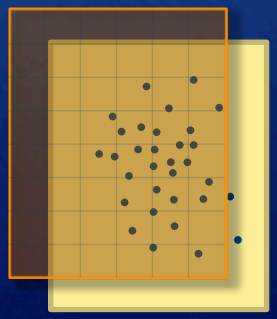
Computation of the fields on grid points

2. Field Solver



- Poisson solver using convolution method
- From the scalar field then solve the electric field

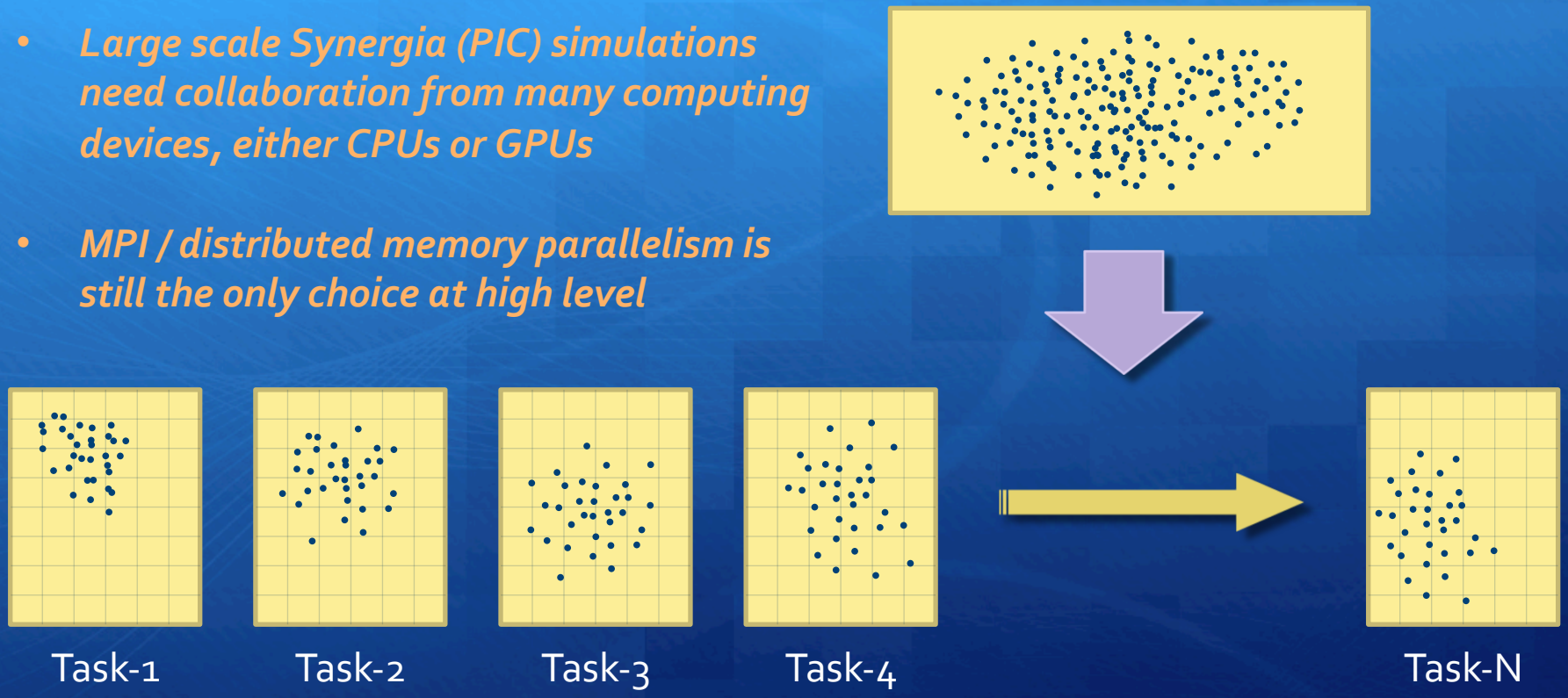
3. Apply Kicks



- Interpolation of the fields from grid points to particle positions
- Integration of the equation of motion

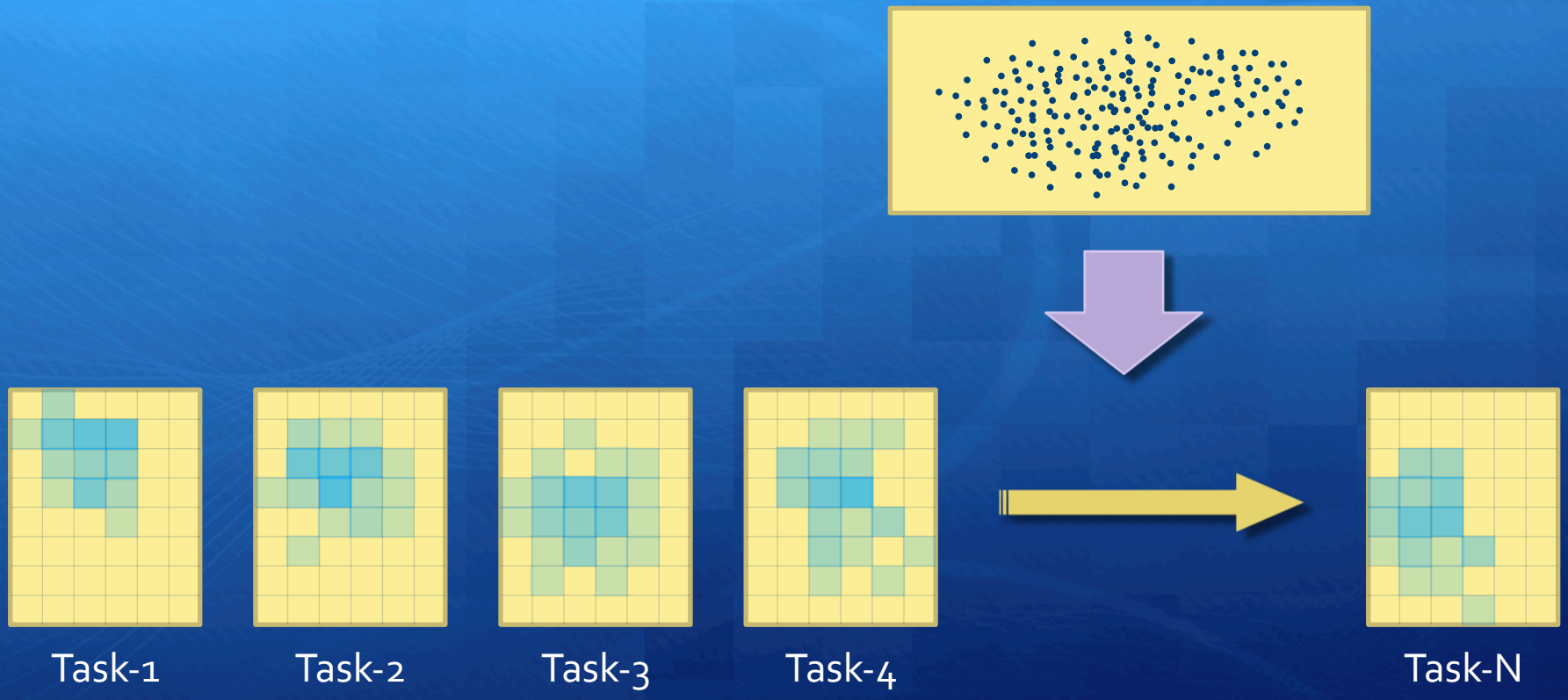
Parallel PIC at High Level o. parallel decomposition

- *Large scale Synergia (PIC) simulations need collaboration from many computing devices, either CPUs or GPUs*
- *MPI / distributed memory parallelism is still the only choice at high level*



- *Randomly distribute particles into MPI tasks*
- *Each task has a duplicated spatial domain*

Parallel PIC at High Level 1. charge deposition

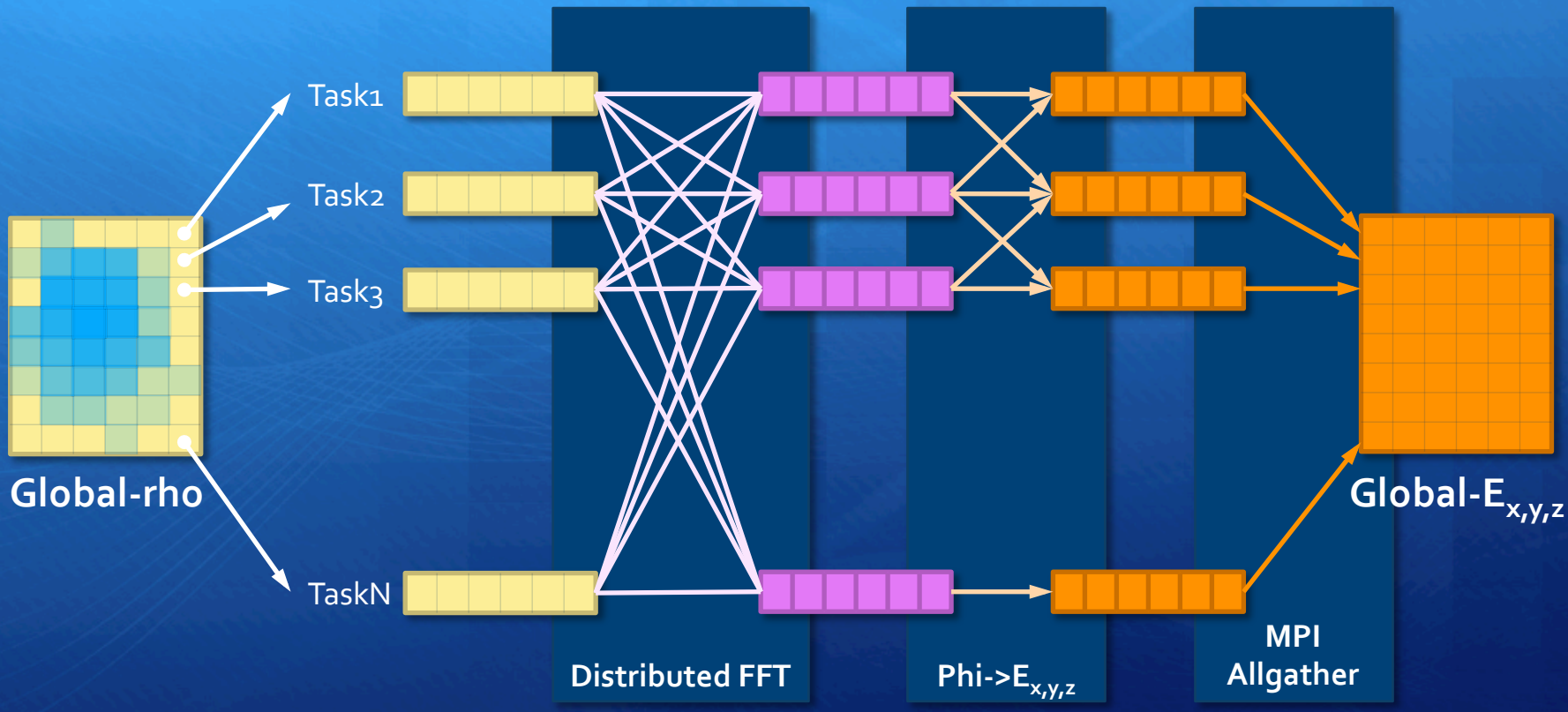


- ***Distributed charge deposition***

Parallel PIC at High Level 1. charge deposition

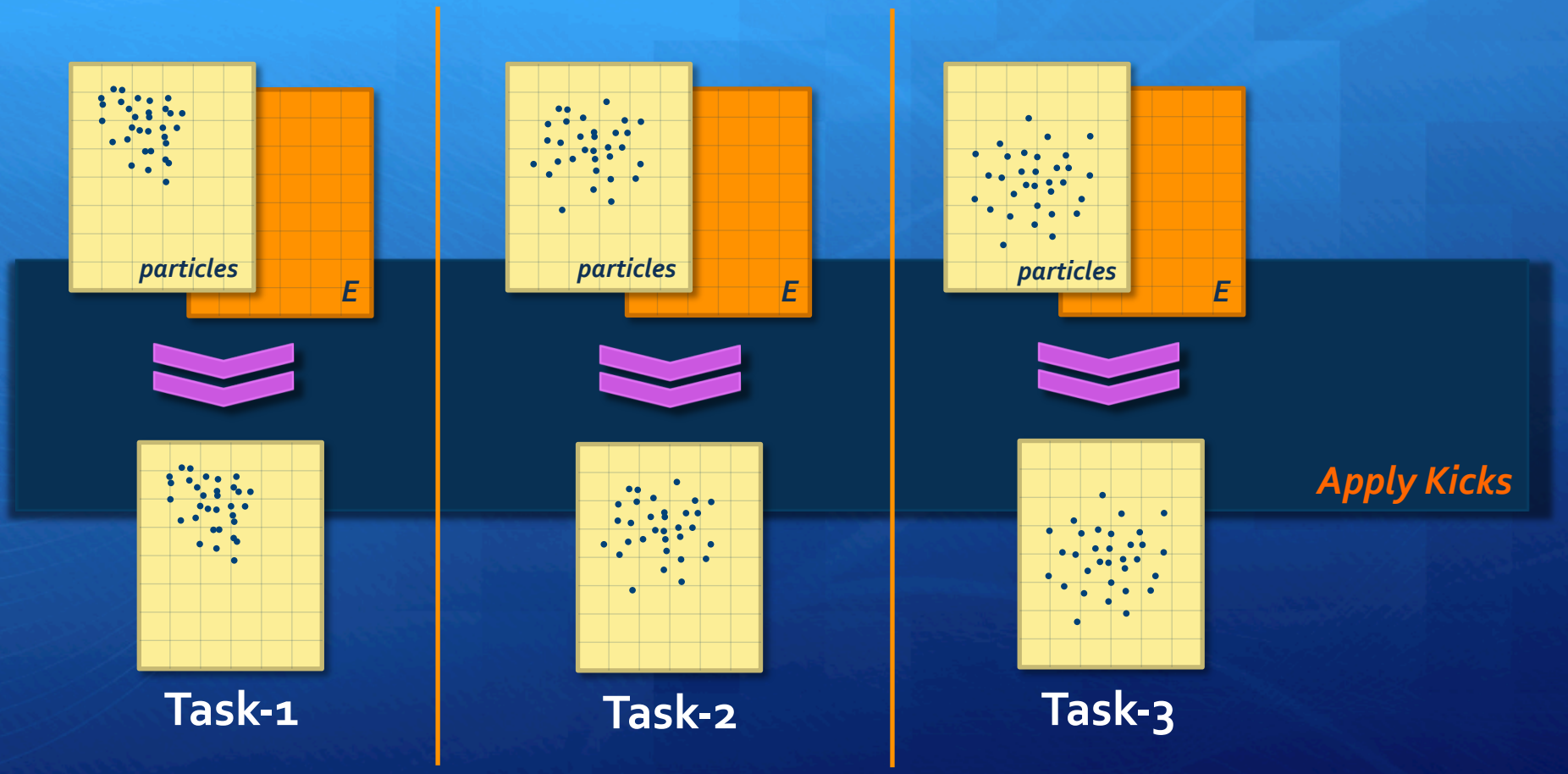
- *Collecting local charge densities via MPI_Allreduce()*

Parallel PIC at High Level 2. Field Solver

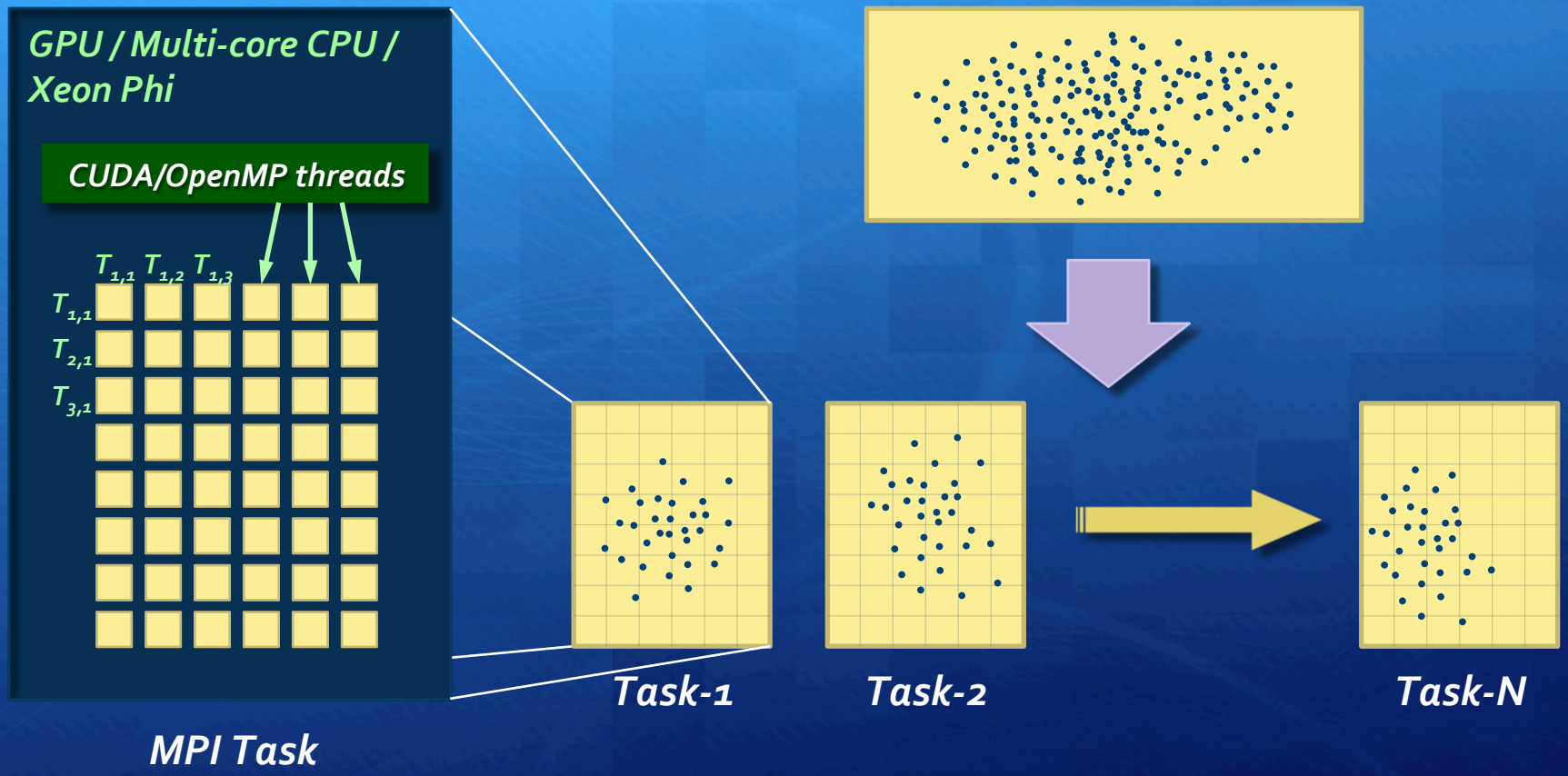


- *Spatial decomposition in field solver*
- *Global-rho -> phi -> local-En -> global-En*

Parallel PIC at High Level 3. apply kicks



Hybrid Parallel Synergia: MPI + CUDA



- *Hybrid = high level MPI + share memory parallelism at finer granularity*
- *Interchangeable computing kernels – possible for heterogeneous computing*

Parallel PIC + CUDA

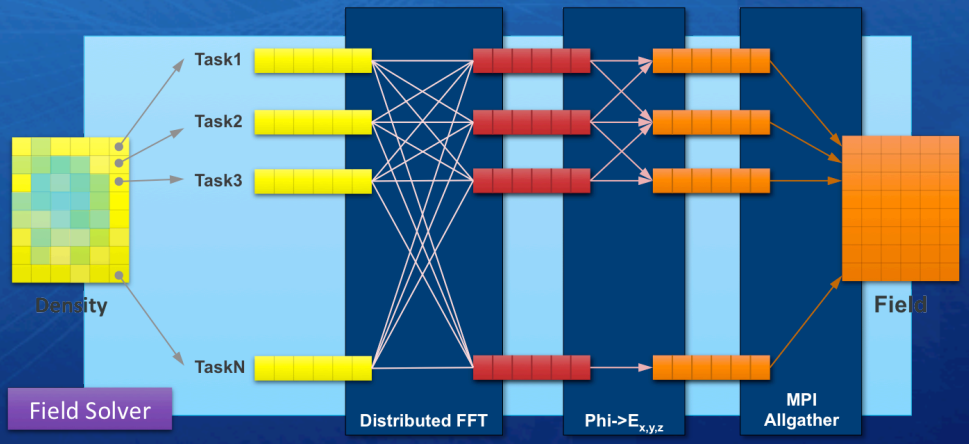
Communication Avoidance (Redundant Field solver)

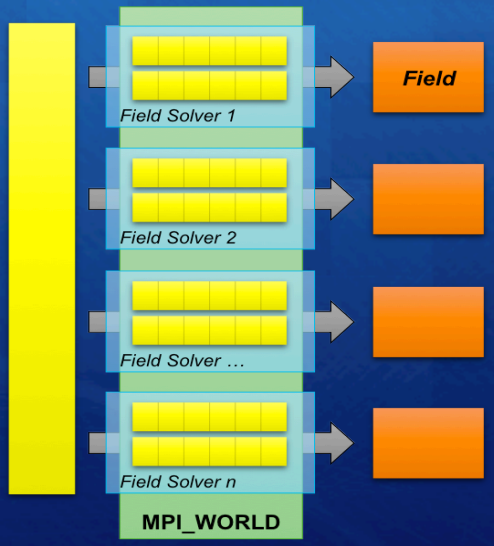
GPU-GPU communication is expensive

- Cross node GPU-GPU communication is even more expensive

Field solver demands high volume of data exchanges

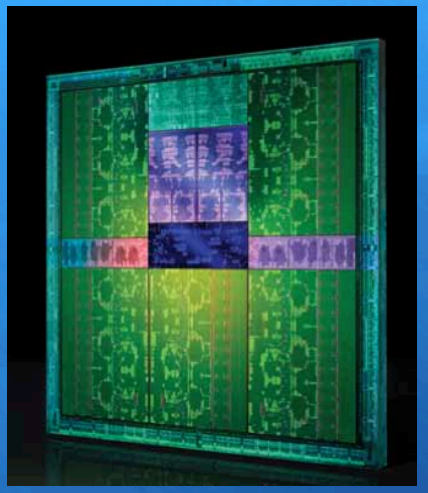
- FFT, field calculations and distributions, etc.





Have redundant field solver on every computing node to reduce the complexity of communication, and also keep data exchanges within the node boundary

Thread Parallelization of PIC



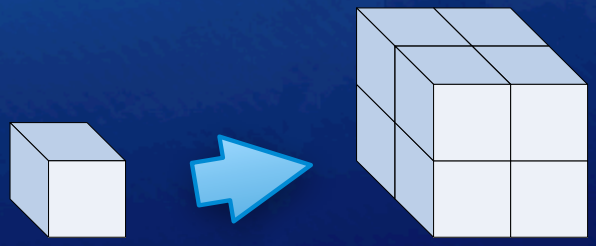
1. ***Particle statistics and diagnostics:***

   - Calculate the *mean* and *standard deviation* of coordinates and momentums for particles in the bunch

Multi-threaded parallel reduction

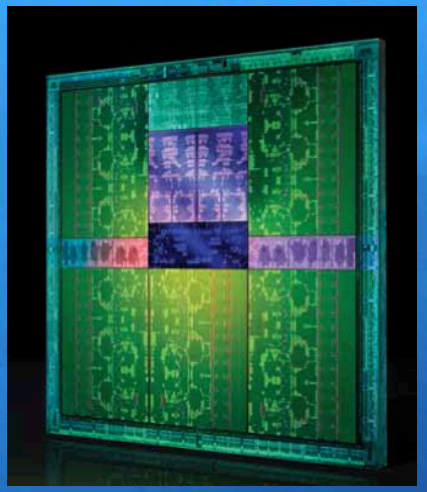
2. ***Charge deposition:***

   - One macro particle can contribute up to 8 grid cells



Collaborative updating algorithm: An interesting topic, will talk more about it later

Thread Parallelization of PIC



3. **Field solver -- Green function:**

$$G(x,y,z) = (x^2 + y^2 + z^2)^{-1/2}$$

Independent and easy parallelization with CUDA / threads

4. **Field solver -- convolution:**

$$\hat{\rho}(u,v,w) = \iiint_{\infty} \rho(x,y,z) e^{-2\pi i(ux+vy+wz)} dxdydz$$

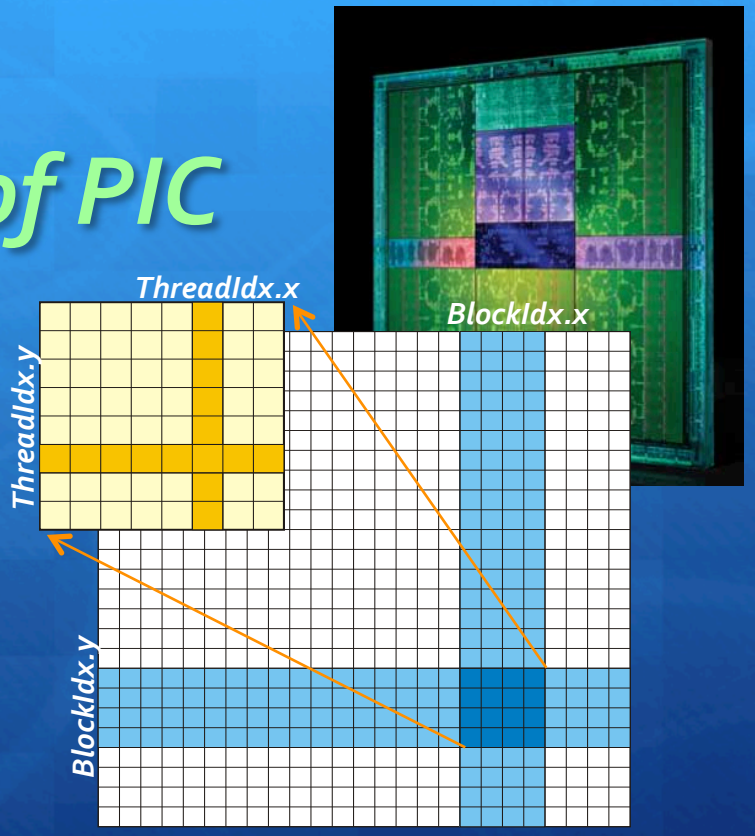
$$\hat{G}(u,v,w) = \iiint_{\infty} G(x,y,z) e^{-2\pi i(ux+vy+wz)} dxdydz$$

$$\hat{\varphi}(u,v,w) = \hat{\rho}(u,v,w) \times \hat{G}(u,v,w)$$

$$\varphi(x,y,z) = \iiint_{\infty} \hat{\varphi}(u,v,w) e^{2\pi i(ux+vy+wz)} dudvdw$$

Convolution method: FFT

Thread Parallelization of PIC



5. Field solver -- electric field:

$$E_x(x,y,z) = -\partial\varphi(x,y,z)/\partial x$$
$$E_y(x,y,z) = -\partial\varphi(x,y,z)/\partial y$$
$$E_z(x,y,z) = -\partial\varphi(x,y,z)/\partial z$$

- *Be careful with memory access pattern for better cache efficiency*
- *CUDA can use shared memory to halve the global memory access*

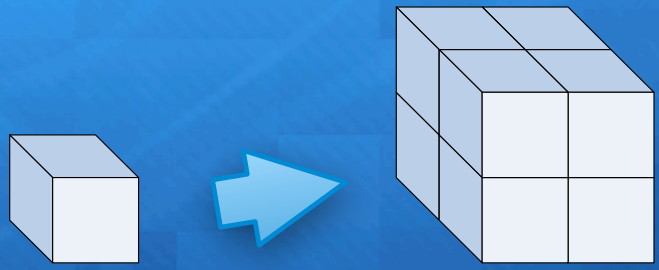
6. Apply kick:

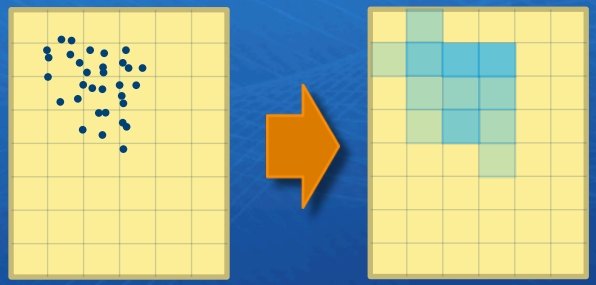
- Advance the *position* and *momentum* for each particle in the bunch

- *Keep particles sorted to mesh grids for better data locality and grouped access of Ex, Ey, and Ez*

Charge Deposition in Shared Memory

One macro particle contributes up to 8 grid cells in a 3 dimensional regular grid





Collaborative updating in shared memory needs proper synchronization or critical region protection

Due to enormous number of particles, collision-free deposition is preferred

CUDA

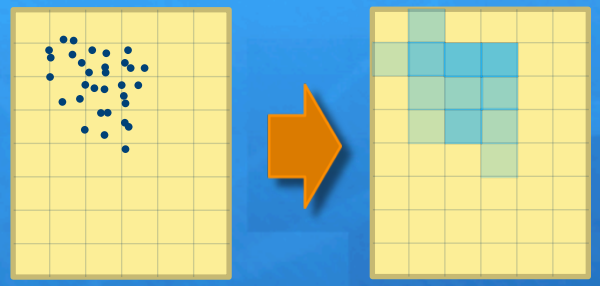
- *No mutex, no lock, no global sync*
- *Atomic add – yes, but not for double precision types*

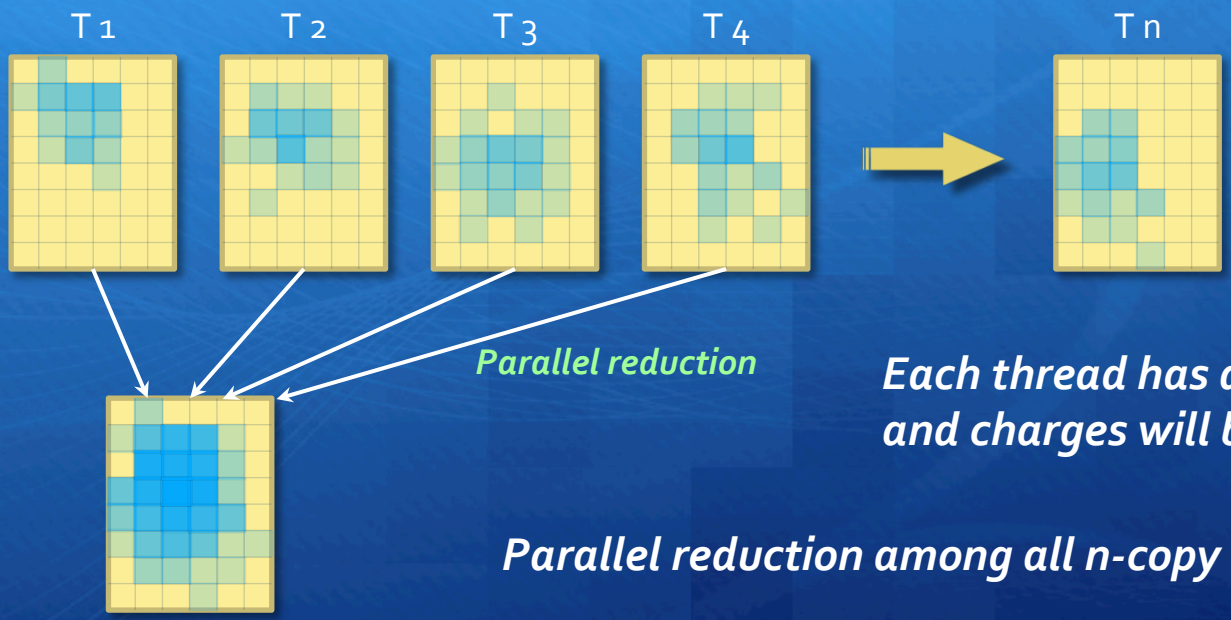
OpenMP

- *#pragma omp critical*
- *#pragma omp atomic*
- *Both very slow*

Charge Deposition in Shared Memory

Distributed Deposition





Each thread has a duplicated spatial grid, and charges will be deposit to that grid only

Parallel reduction among all n-copy of spatial grids

Limitations

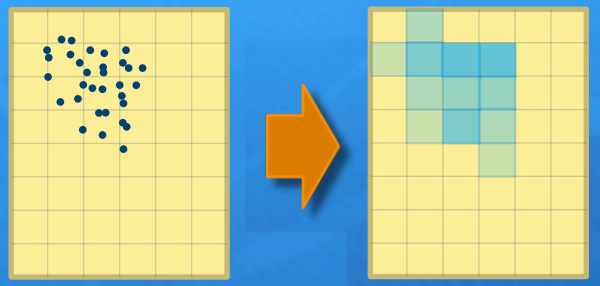
High memory usage
Limited thread concurrency
Reduction overheads increases with number of threads

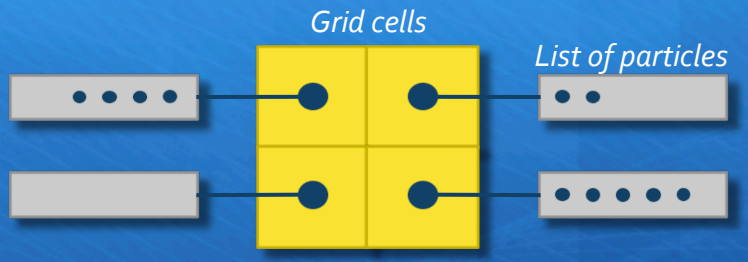
Advantages

Collision-free
Minimum overhead at low thread counts

Charge Deposition in Shared Memory

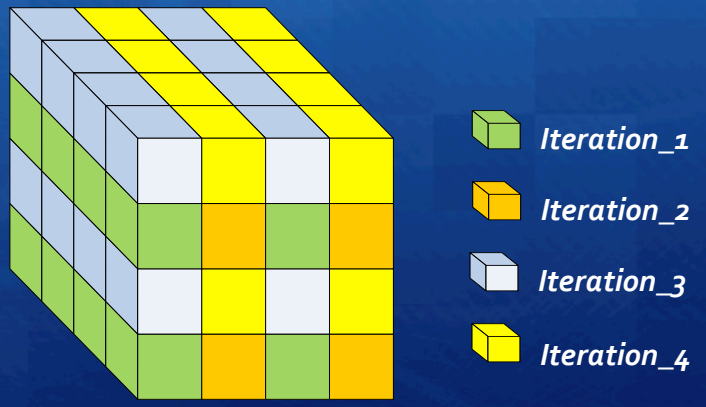
Interleaved Deposition





1. Sort particles into their corresponding cells

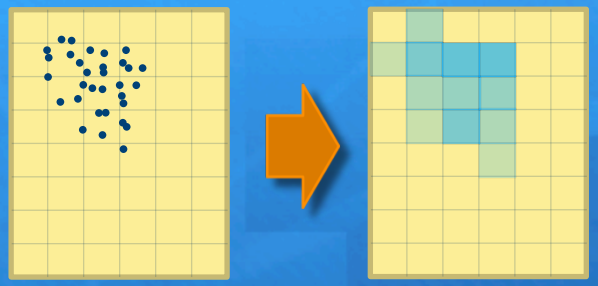
2. Deposit based on color-coded cells in an interleaved pattern (red-black)

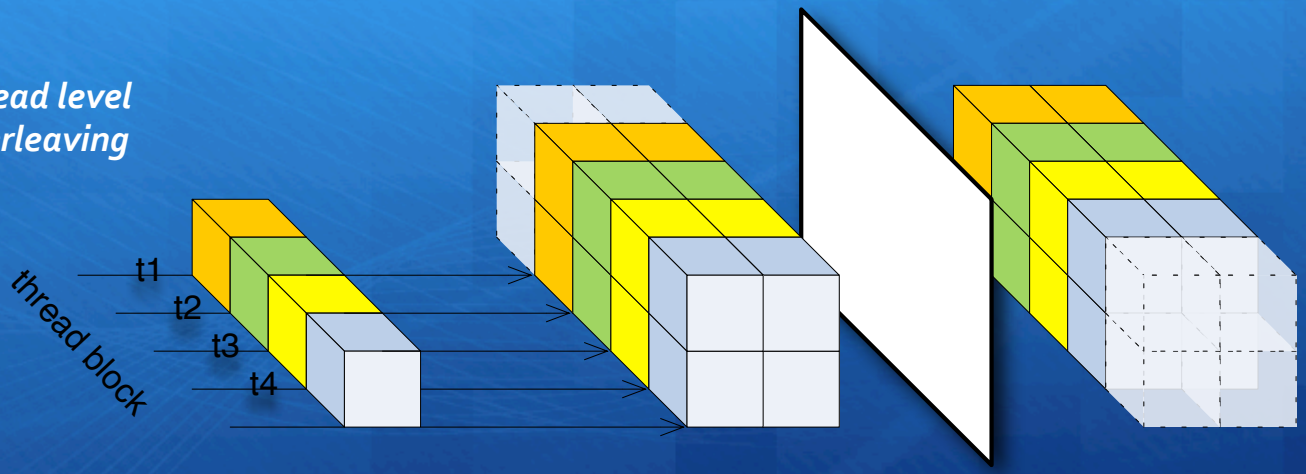


- *Parallel bucket sort (concurrency ~ number of cells)*
- *Indexed list for quick accessing particle data and reduced memory movement*

Charge Deposition in Shared Memory

Interleaved Deposition





Limitations

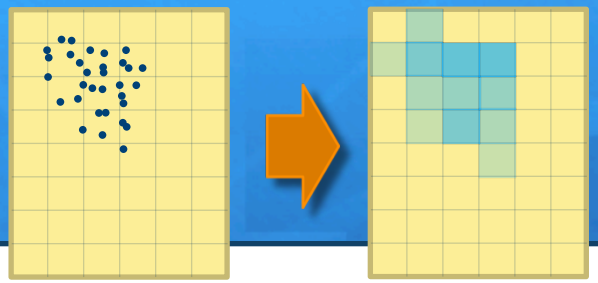
Significant overhead for sorting particles when thread counts are low

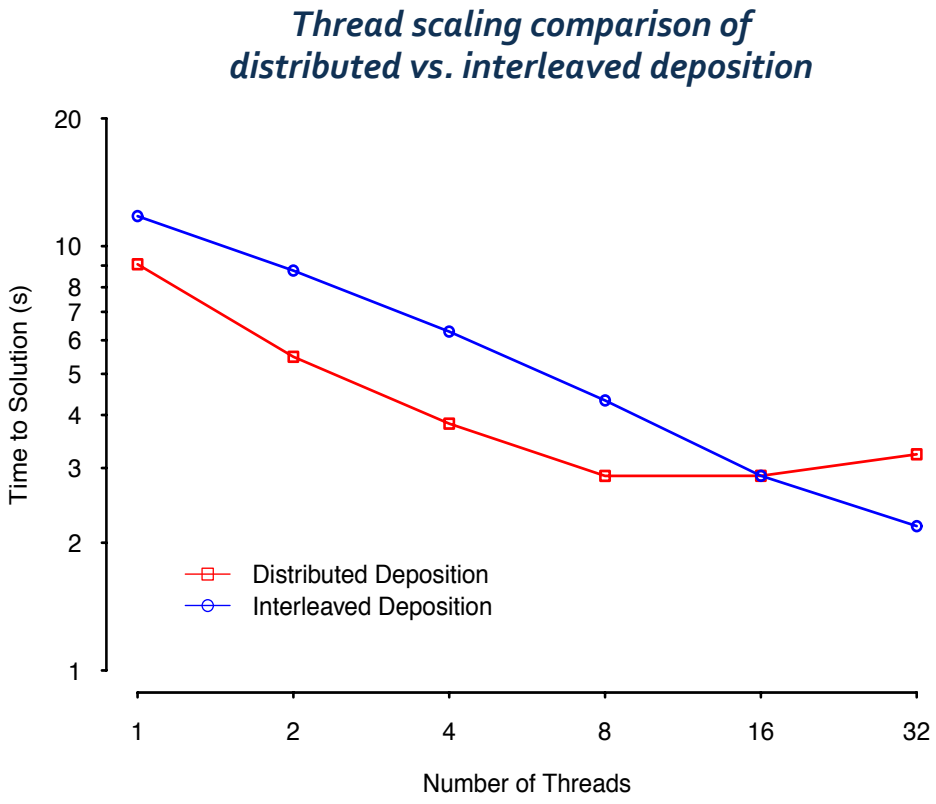
Advantages

Collision-free
High thread concurrency, very good scalability
Scalable sorting overhead
Fixed memory usage
Better data locality at moving particles

Charge Deposition in Shared Memory

Comparison





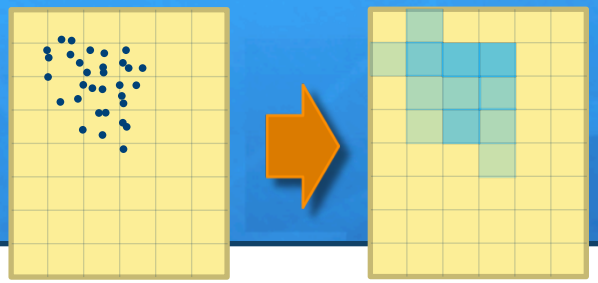
Distributed deposition :

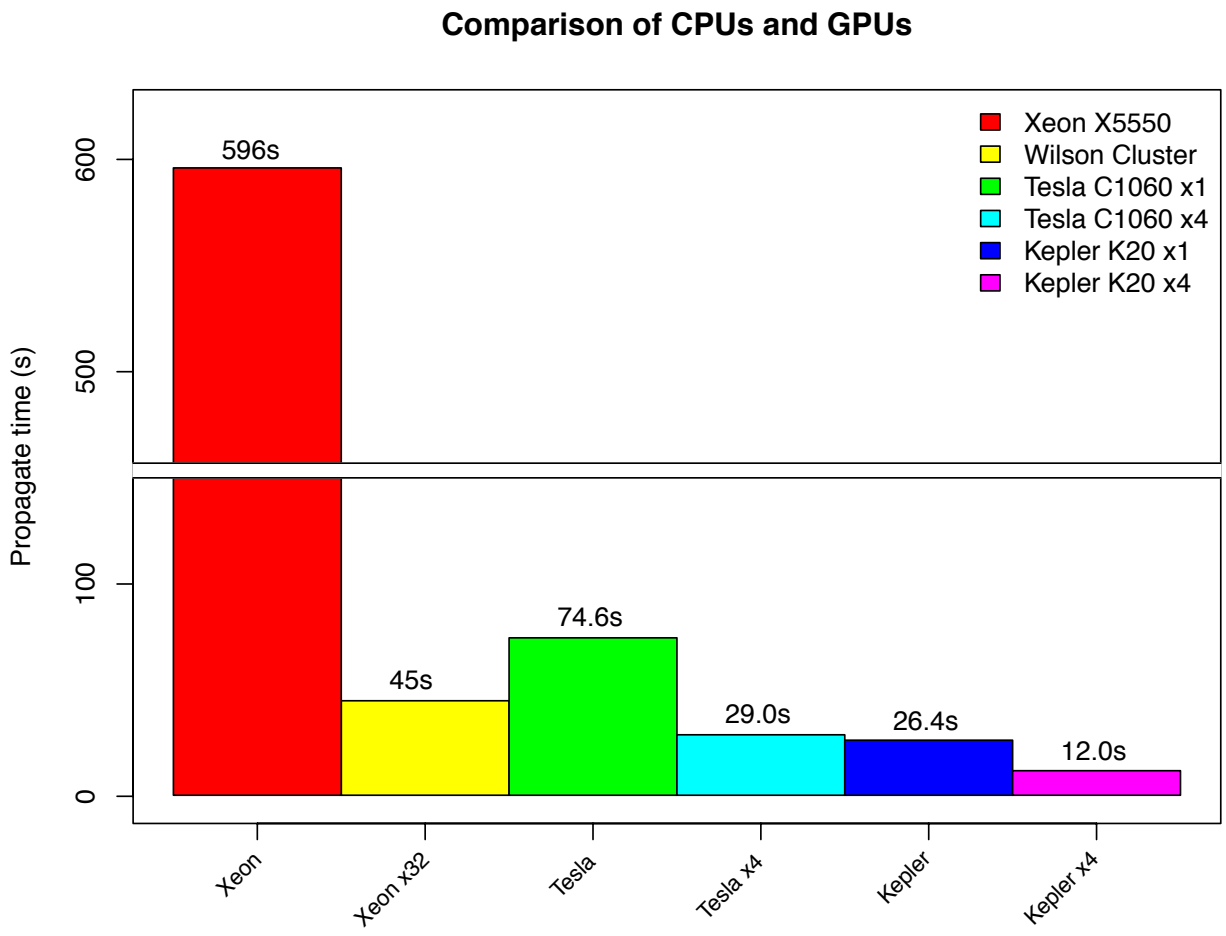
suitable for low thread counts, such as OpenMP for multi-core CPU

Interleaved deposition :

suitable for high thread counts and limited memory capacity, such as GPU and Intel MIC

Synergia on GPU Performance benchmarks





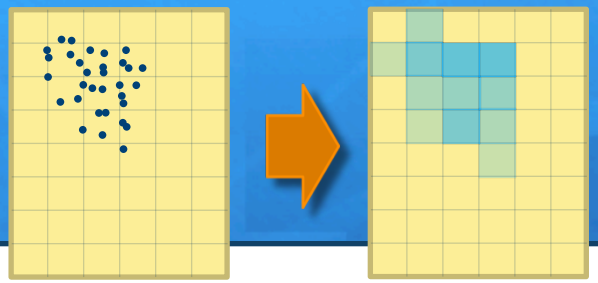
8x speed up on 1x Tesla
20x speed up on 4x Tesla

23x speed up on 1x Kepler
50x speed up on 4x Kepler

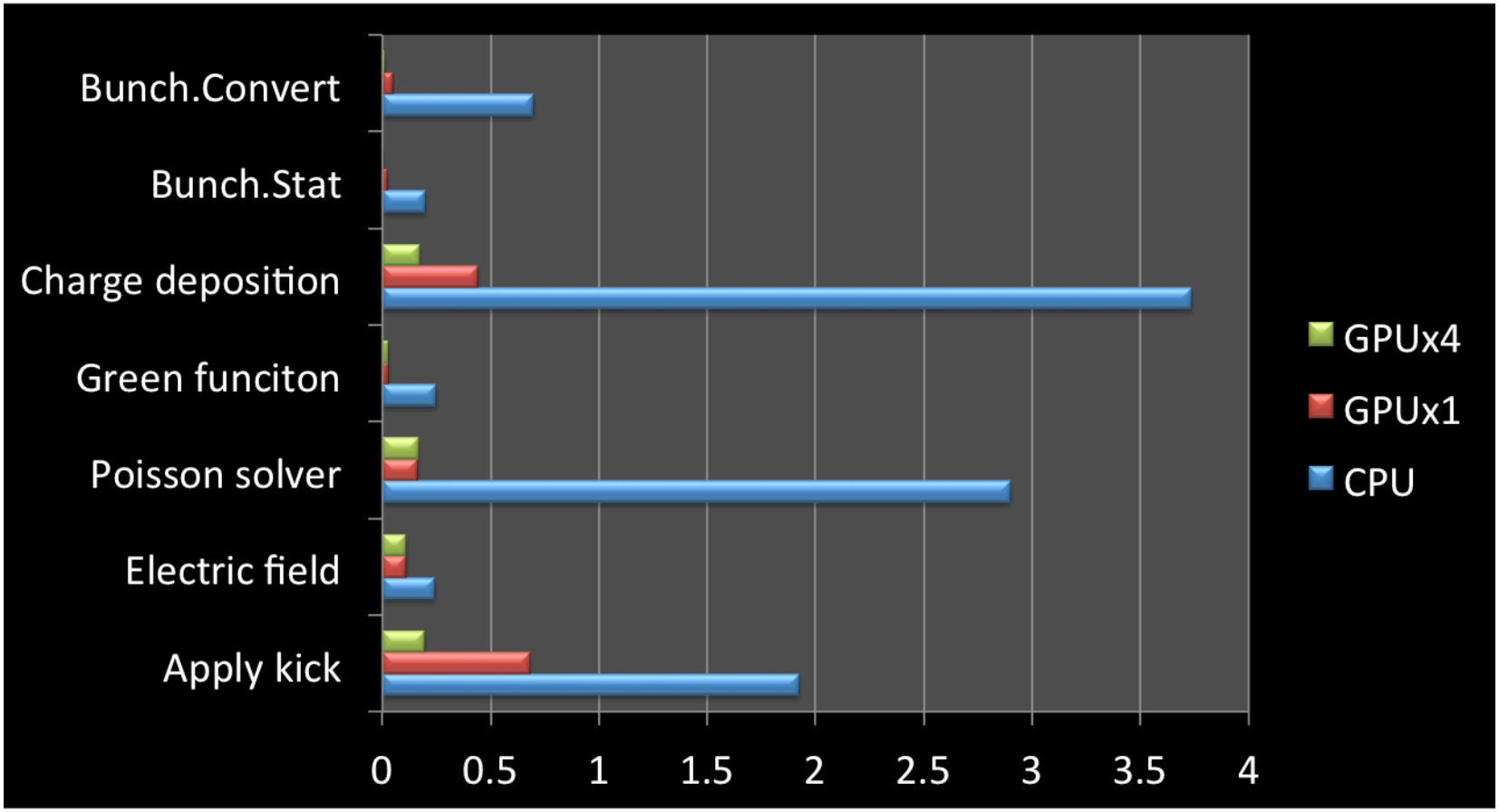
A single Kepler outruns a Xeon cluster with 16 nodes and 128 cores

All use double precisions

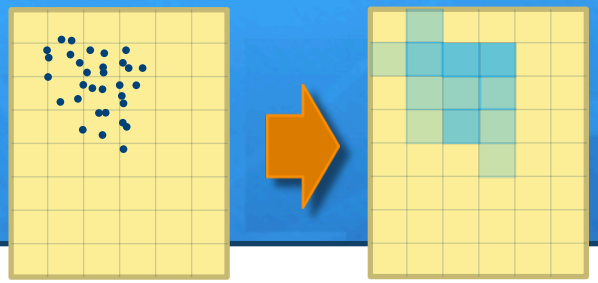
Synergia on GPU Performance Breakdown

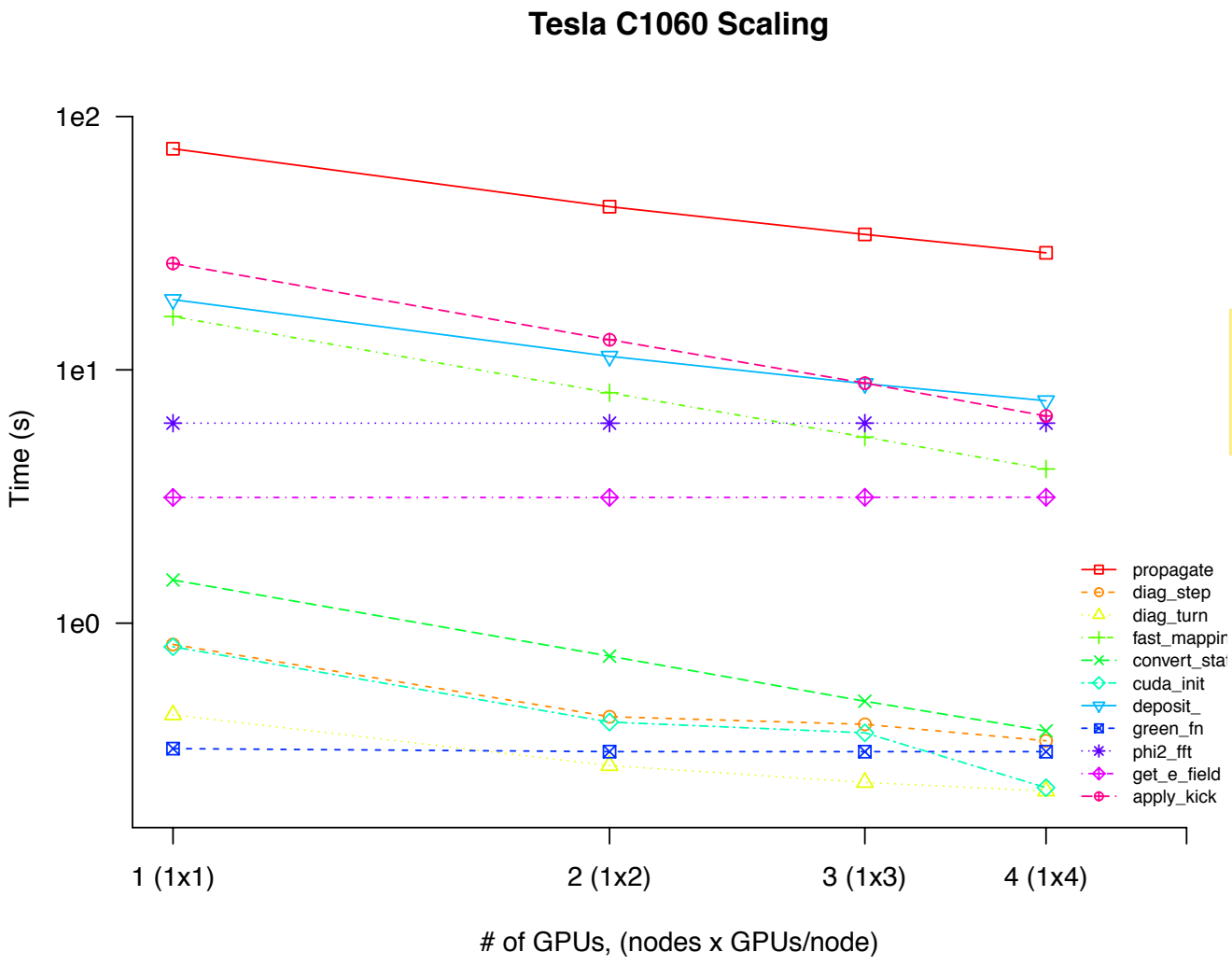


- 1. Intel Xeon X5550, single process @ 2.67GHz;
- 2. NVidia Tesla C1060, 30 streaming multi-processors @ 1.30GHz in a single GPU
- 3. Nvidia Tesla C1060 x 4



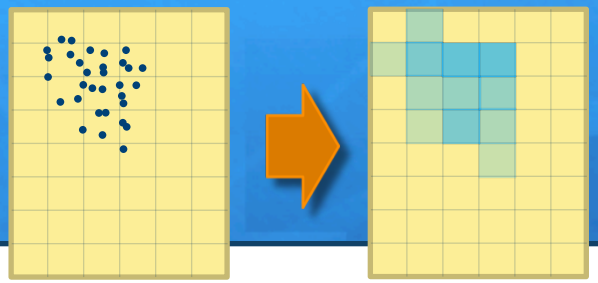
Synergia on GPU
Multi-GPU Scaling on C1060

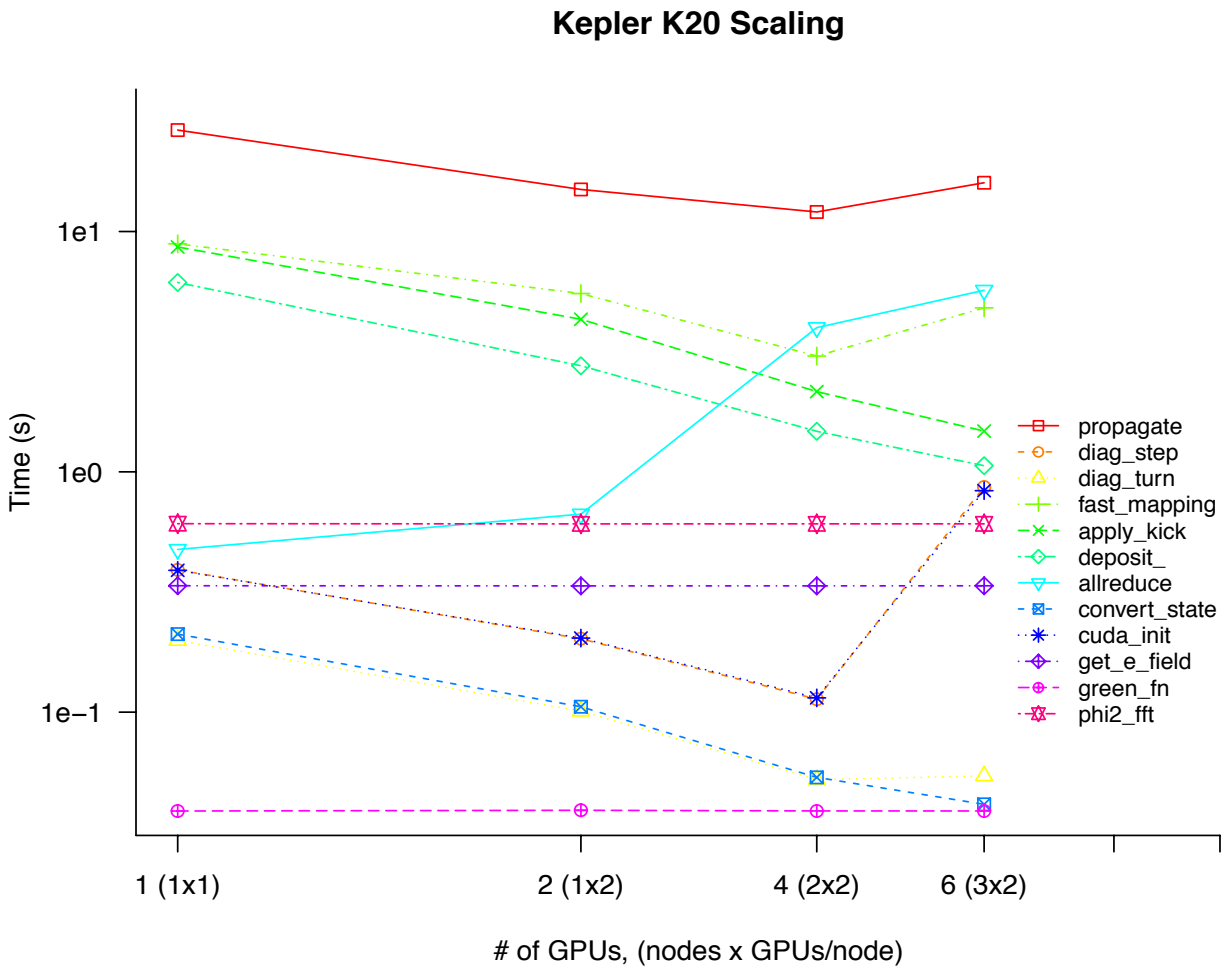




One field solver per Tesla GPU card, therefore the Poisson solver does not scale

Synergia on GPU
Multi-GPU Scaling on K20





2x Kepler K20 per node

Communication spikes when multiple nodes are used