Parallization of the UCLA Parallel PIC Framework

Viktor K. Decyk

UCLA and JPL

Abstract:

Parallelization of the QuickPIC code comes from components provided by the UCLA Parallel PIC (UPIC) Framework. Parallelization of these components will be discussed, particularly the handling of dynamic load balancing.

What are Particle-in-Cell Codes?

PIC codes integrate the trajectories of many particles interacting self-consistently via electromagnetic fields

1. Calculate charge (and current) densities from particles

$$\rho(r) = \Sigma q_i S(r-r_i)$$

2. Solve Maxwell's equation, Spectral or Finite-Difference

$$\nabla \bullet E = 4\pi\rho$$

3. Advance particle's coordinates:

$$m_i(dv_i/dt) = q_i[E(r_i) + v_i \times B(r_i)/c]$$

$$dx_i/dt = v_i$$

A grid is used as a scaffolding to calculate forces

=> the grid reduces calculation to order N

UPIC Framework

Framework: a unified environment containing all components needed for writing code for a specific problem domain

Goal is rapid construction of new codes by reusing tested modules: “Lego” pieces for developing new codes

Designed for student programmers, with many error checks and debugging help

Supports multiple numerical methods, optimizations, different physics models, different types of hardware

Automatically adapts to computer architecture

Above all, hides parallel processing

Layered Approach

Bottom Layer: optimized Fortran 77 routines

- complex but very fast

Middle layer: helper objects work with Fortran 90 arrays

- Fortran 90 arrays powerful, well-understood
- Helper objects describe data, but do not contain data
- Good environment when making many changes (white box)

Higher Layers: high level objects have many properties

- Objects contain great deal of hidden information
- Good environment when making few changes (black box)

Highest Layer: one object represents entire plasma simulation

PLIB: Parallel Particle Simulation Library

A low-level library (60 subroutines) which encapsulate all the communication patterns needed by parallel PIC codes

- designed for high-performance
- communications are separated from physics
- supports both message-passing and shared memory
- supports RISC and vector architecture
- supports linear and quadratic interpolation

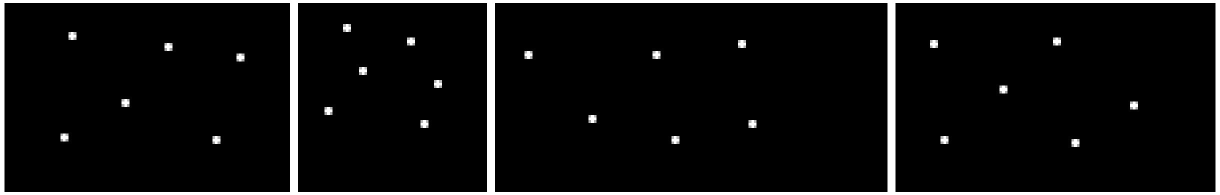
Robust: used in many projects since initial development in 1989.

Fortran77 with Fortran90 wrappers

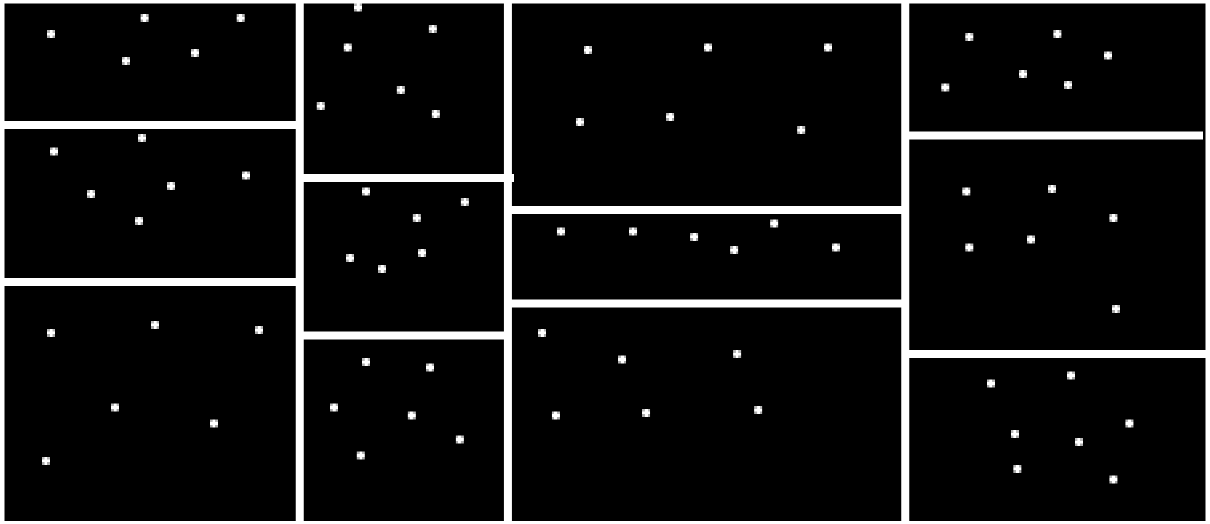
PLIB: Parallel Particle Simulation Library

Initialization Routines	PPINIT, PPEXIT, PPID, DCOMPx family
Particle Manager, guarantees particles in correct domain	PMOVEx family
Field Manager, guard cell accumulation and replication	PxCGUARDx, PxAGUARDx family
Partition manager, moves data between partitions	PFMOVE, REPARTD, FNOFF
Transpose Routines, swapping decompositions	PxTPOSEx family
FFT Routines	PxFFTx, DBLSINx, PHAFDBLx, PZDBLx family
I/O Routines, accumulate and broadcast data from files	PBCAST, PWRITE, PREAD
Miscellaneous Routines	PSUM, PMAX, PSCAN, PTIMERA

Only small number of different communication patterns needed



1D Domain decomposition



2D Domain decomposition

Each partition has equal number of particles

Particle Manager

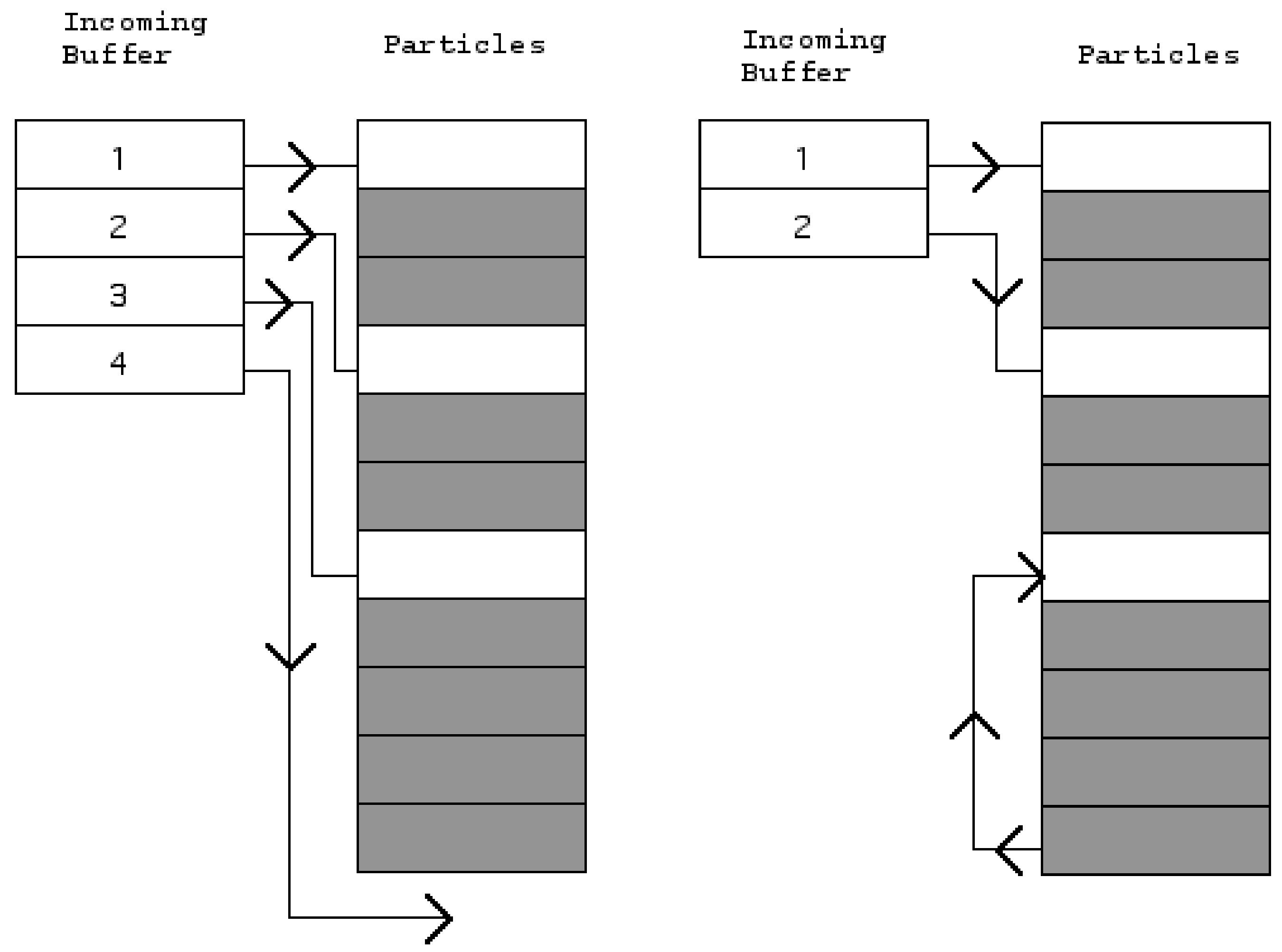
Nodes organized by physical location in space. After update, particles distributed to appropriate node

All holes in particle array are filled

- Particles can move across multiple nodes
- Particle data will be processed in pieces, if memory low
- Particle data will not be overwritten if array is too small

Particle boundaries can be changed drastically

If overflow is detected, partition can be changed without failure



PLIB: Particle Manager Message-Passing

Field Manager

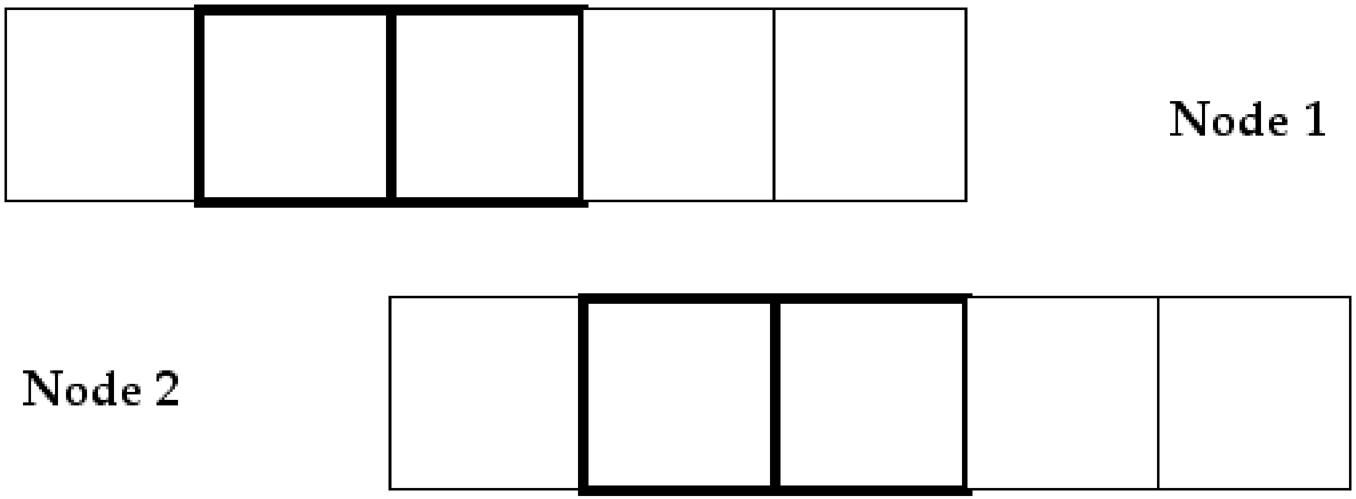
Field quantities (charge density, electric fields) have guard cells to avoid excessive communication

After deposit, density deposited in guard cells passed to “owner”

After field solve, electric for magnetic fields replicated to guard cells before advancing particles

Linear and quadratic interpolation supported

For this thin regions, guard cell information does not have be be nearest neighbor



Guard Cells, Quadratic Interpolation

Outlined Cells are unique, others are copies

Partition Manager

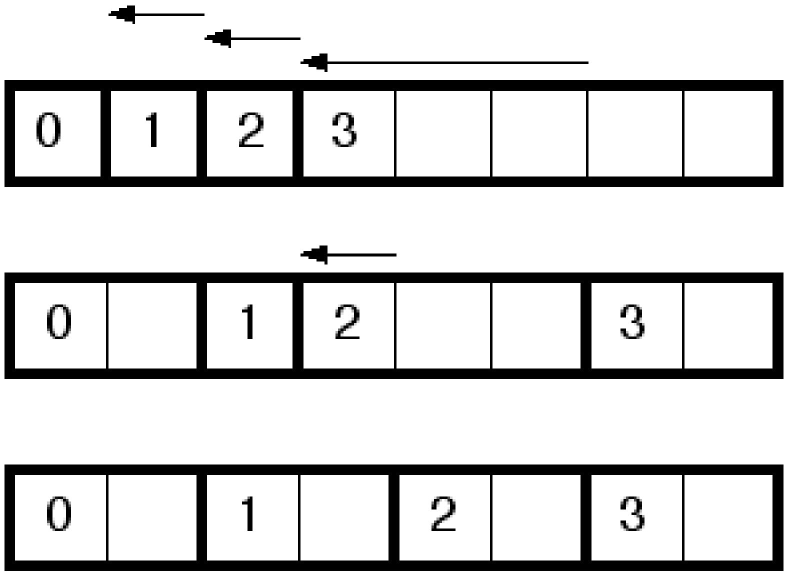
Moves data between non-uniform and partitions

Fields which do not belong, passed to appropriate neighbor, then passed again if necessary. Remembers how many passes are required.

Finds new partitions based on number of particles per cell, accurate to nearest cell. Can also find new partitions from known distribution.

Partition manager is called every time step, since fields needed by particles are in non-uniform partitions, but FFT based solvers require uniform partitions.

Moving Data from Non-uniform to Uniform Partition



First Pass

Second Pass

0,1,2,3 refer to processor number

Arrow indicates data movement

Transpose Routines

Many algorithms can be parallelized by operating on one coordinate which is local (not distributed), then transposing and operating on the other coordinate, which is now local

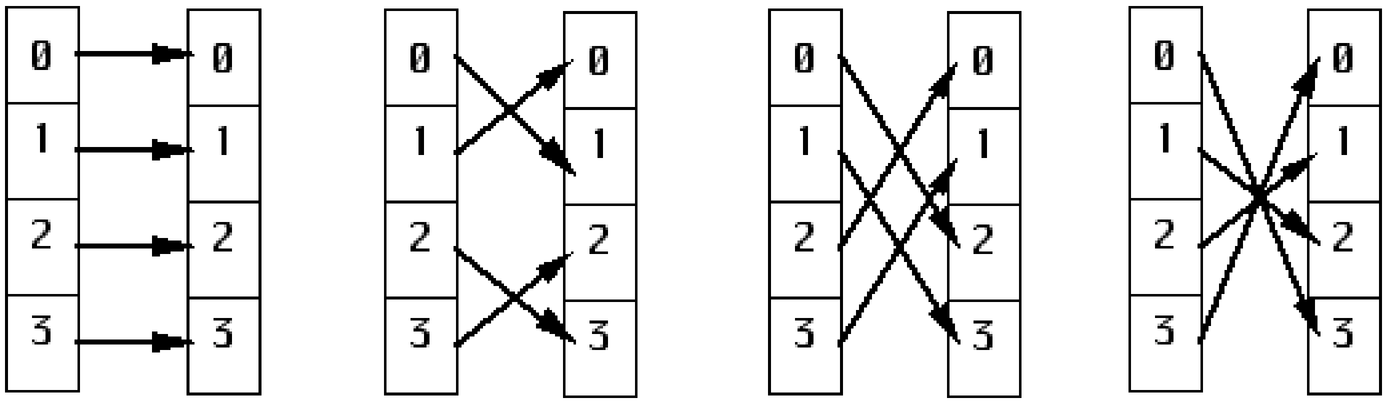
FFT can be parallelized this way
Also, mixed spectral and finite-difference methods

Limitations

- Cannot use more processors than grids
- Assumes domains are regular

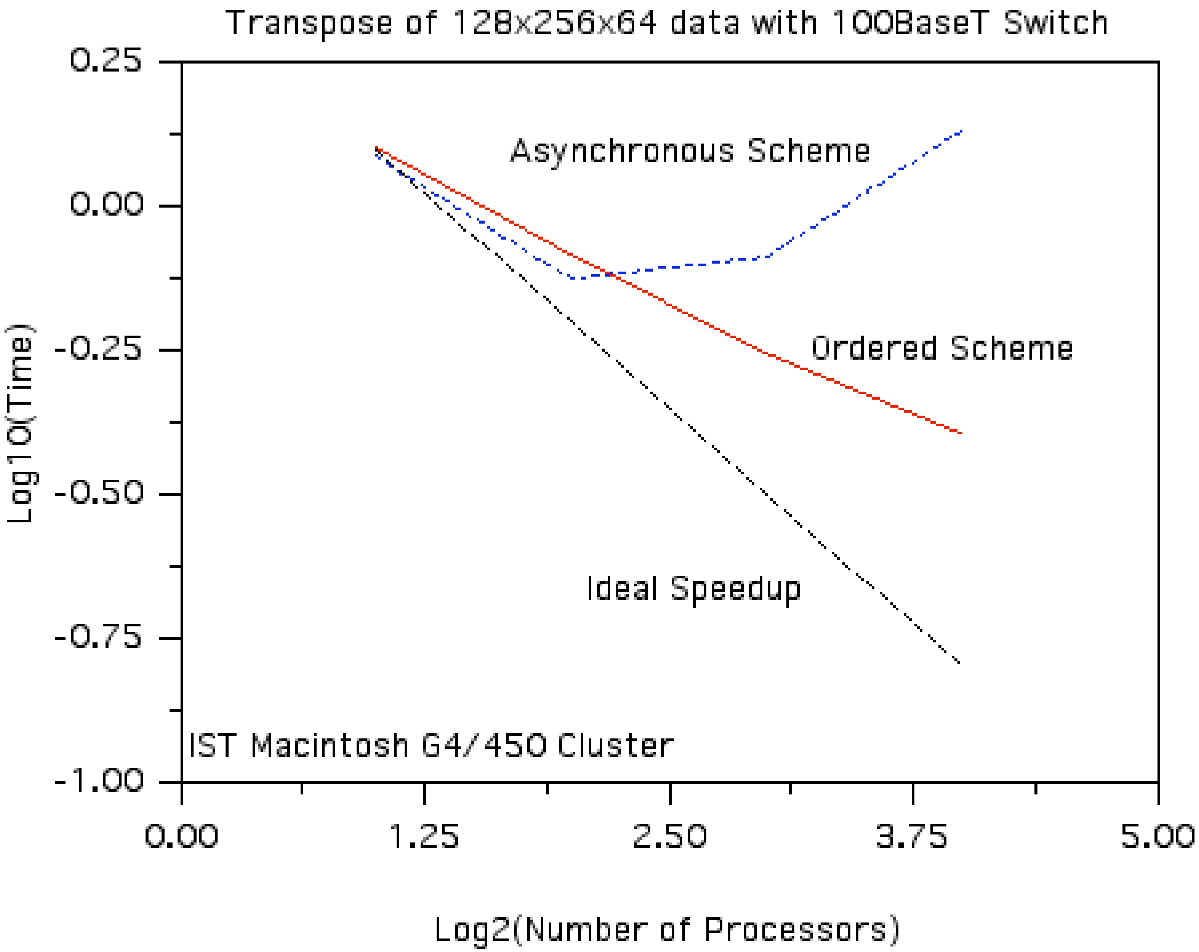
Transpose is very communication intensive, all-to-all

Collision avoidance important for some hardware



Controlled Data Transpose
Full Duplex, No collisions

At any given pass, each process sends and receives to unique processor. Pause at each pass before continuing.


Transpose of 128x256x64 data with 100BaseT Switch
0.25
0.00
-0.25
-0.50
-0.75
-1.00
Log10(Time)
Asynchronous Scheme
Ordered Scheme
Ideal Speedup
IST Macintosh G4/450 Cluster
0.00
1.25
2.50
3.75
5.00
Log2(Number of Processors)

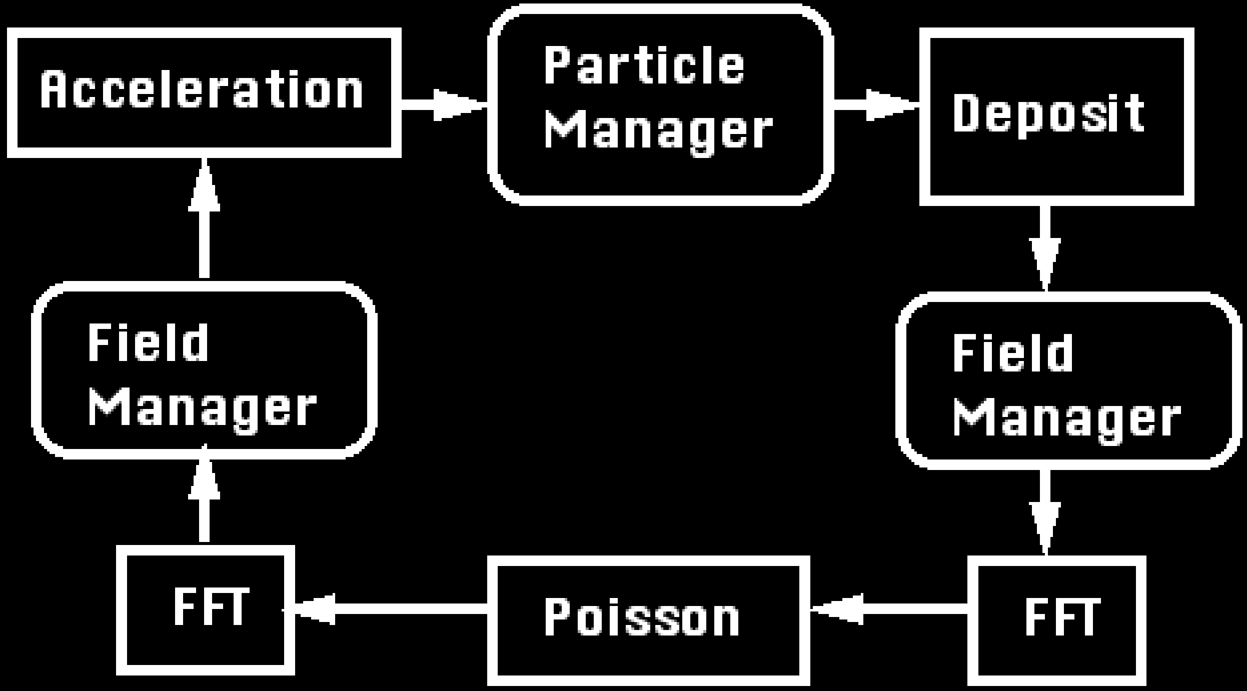
PLIB Philosophy

2D code supports 1D domain decomposition
3D code supports 1D and 2D decomposition

PLIB generally does only data movement, no physics
- Communications patterns can be reused

Physics modules only need to know current data layout
- Physicists can use PLIB without knowing about MPI

Multiple FFTs can be done with single transpose



Main program loop

Rounded boxes: data layout routines contain MPI calls

Square boxes: physics routines do not contain MPI

Encapsulation of Parallelism

PLIB is embedded in various classes in the Middle layer of UPIC

- Provides safe and simpler interface to Fortran77 codes
- Encapsulates parallel data layouts
- Provides polymorphism
- Provides error checking

Some helper classes that use parallelism:

- ufield2d, uniformly partitioned fields (e.g., transpose)
- nfield2d, non-uniform partitioned fields (e.g., partition manager, guard cells)
- fft2d, for Fast Fourier Transforms
- part2d, for particles (e.g., particle manager)

UPIC Framework

Codes built using UPIC Framework:

QuickPIC: Quasi-static code for plasma accelerators

- approximation that everything is moving at the speed of light
- coupled 2D and 3D parallel codes

QPIC: a 2D quantum PIC code

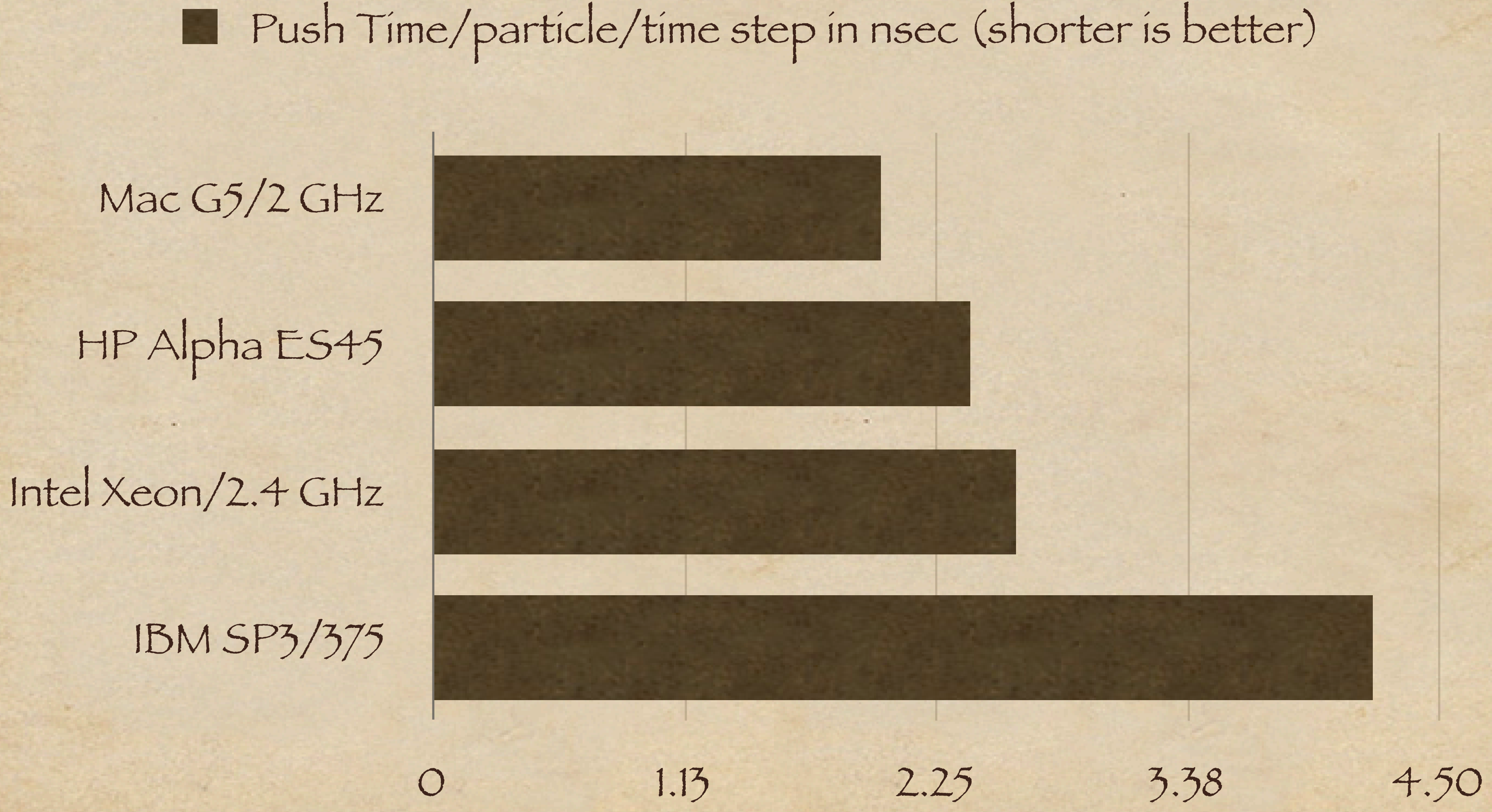
- uses semi-classical approximation based on integrating many Feynman paths

BEPS: an interactive 2D PIC code for teaching plasma physics

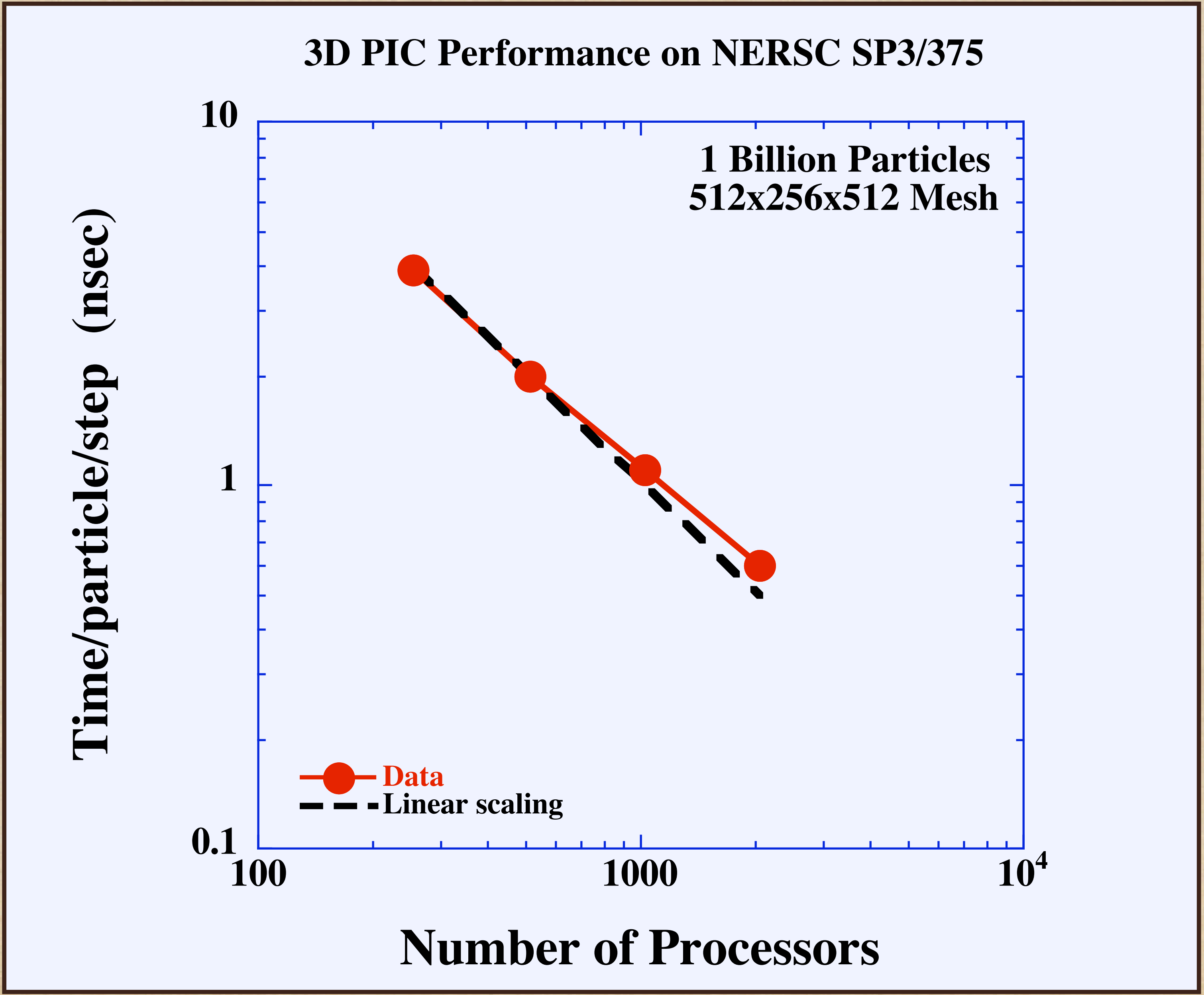
Hipass: 2D for ionospheric modification experiments

MC^2: a 3D PIC code for cosmology

3D Electrostatic particle benchmark
127 million particles, 256 processors

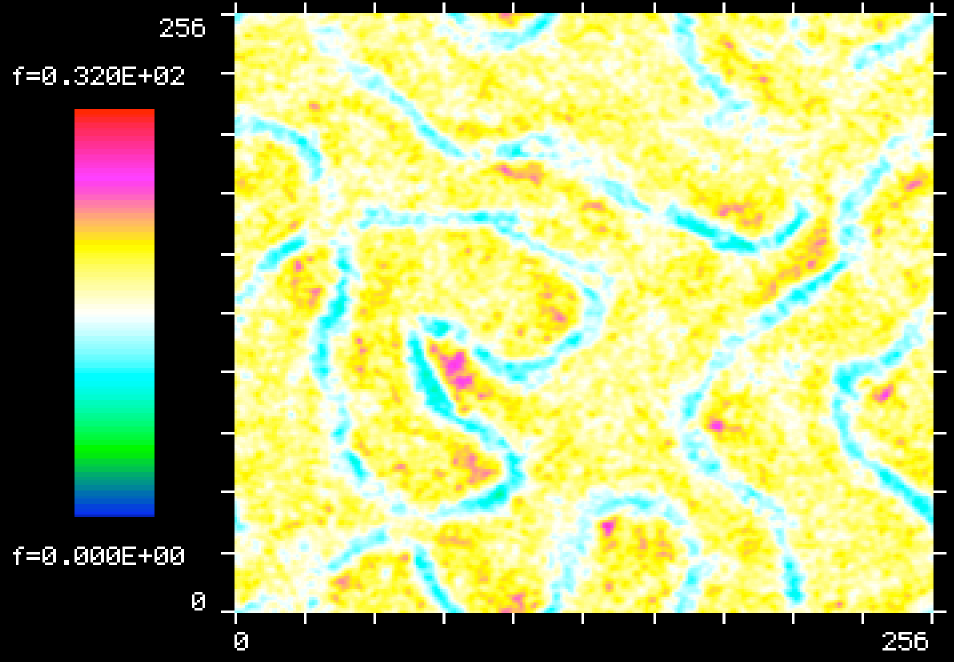


Scaling of Large PIC Electrostatic Benchmark

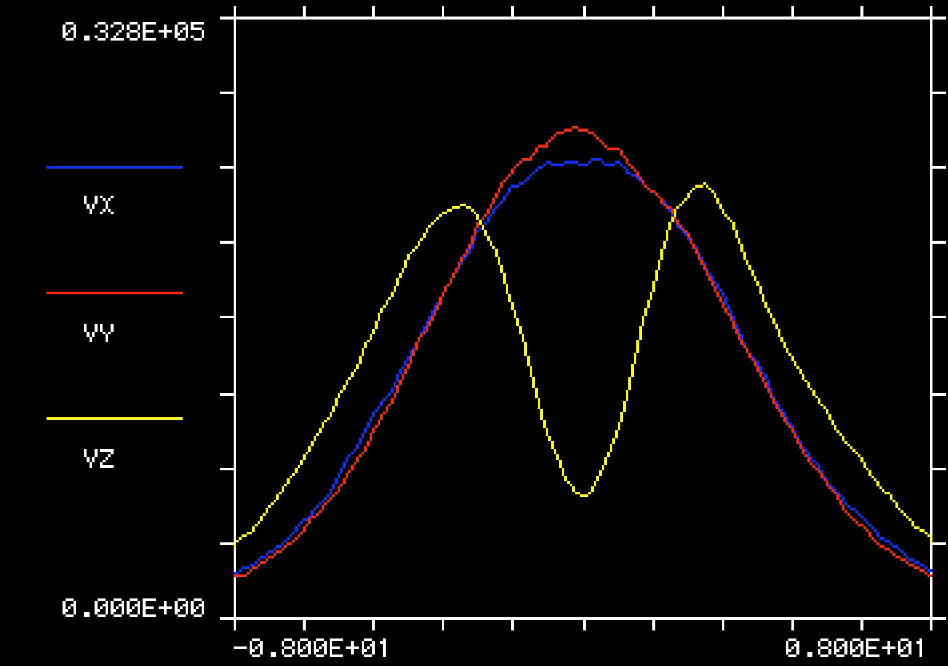


Status of UPIC Framework

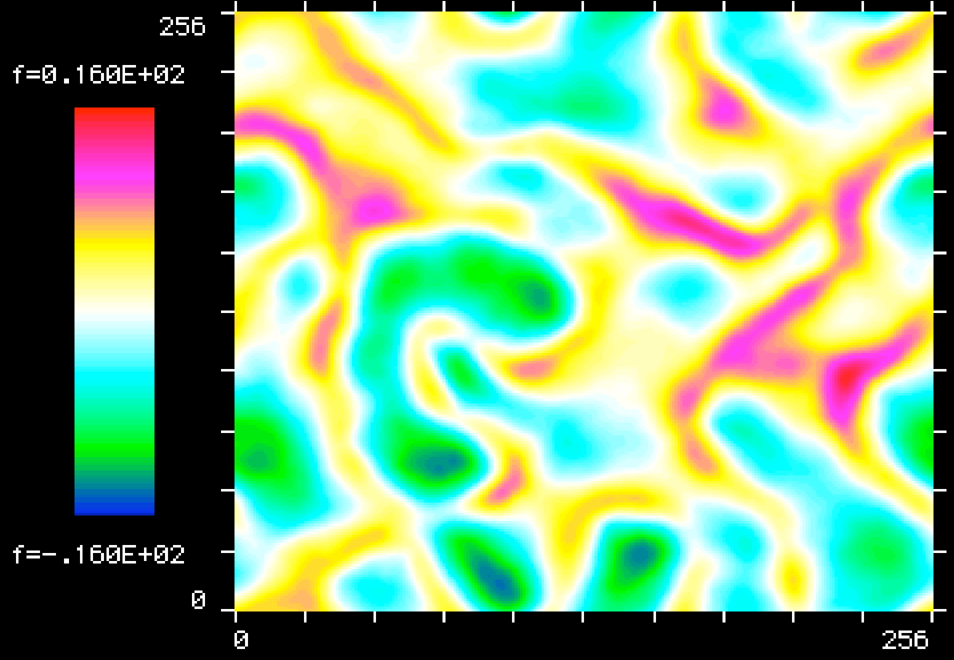
<table>
<tr><th>Currently available</th><th>Planned for future</th></tr>
<tr><td colspan="2">Force Types</td></tr>
<tr><td>Electrostatic, Electromagnetic, Darwin
Relativity Option</td><td>Gyrokinetic, Collisions</td></tr>
<tr><td colspan="2">Field Boundary Conditions</td></tr>
<tr><td>Periodic, Conducting,
Mixed Periodic/Conducting</td><td>Open (Vacuum), Moving Window</td></tr>
<tr><td colspan="2">Particle Boundary Conditions</td></tr>
<tr><td>Periodic, Reflecting,
Mixed Periodic/Reflecting</td><td>Absorbing/Emitting, Ionization</td></tr>
<tr><td colspan="2">Initialization</td></tr>
<tr><td>Arbitrary function n(x)n(y)n(z),
2 species</td><td>Arbitrary function n(x,y,z),
N species</td></tr>
</table>



DENSITY
T =   2000



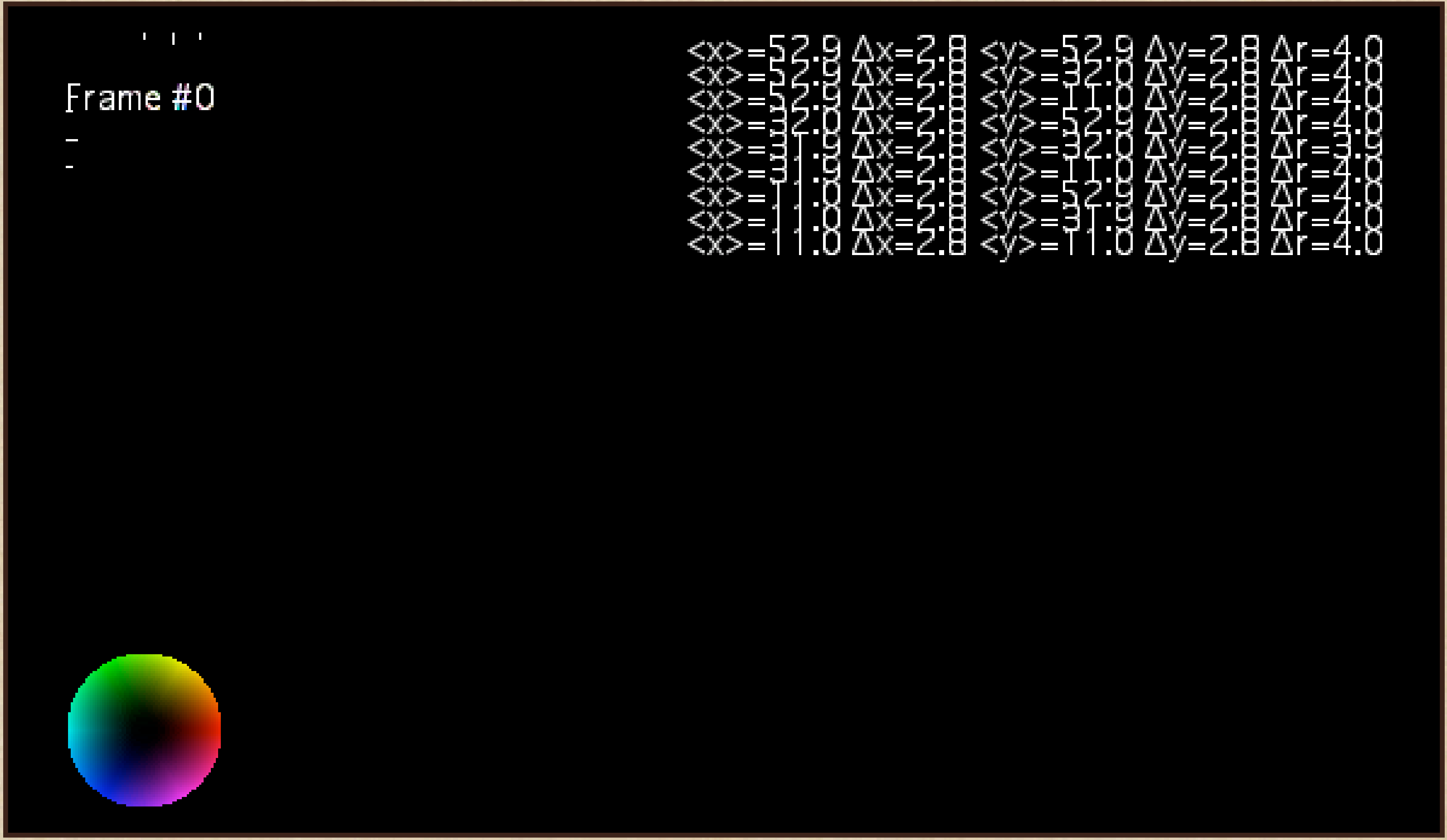
ELECTRON VELOCITY DISTRIBUTIONS VERSUS V, T =   2000
VTX =   3.79310 VTY =   3.72331 VTZ =   4.58008



POTENTIAL
T =   2000

2D BEPS Code:

Relativistic Electromagnetic Two Stream Instability

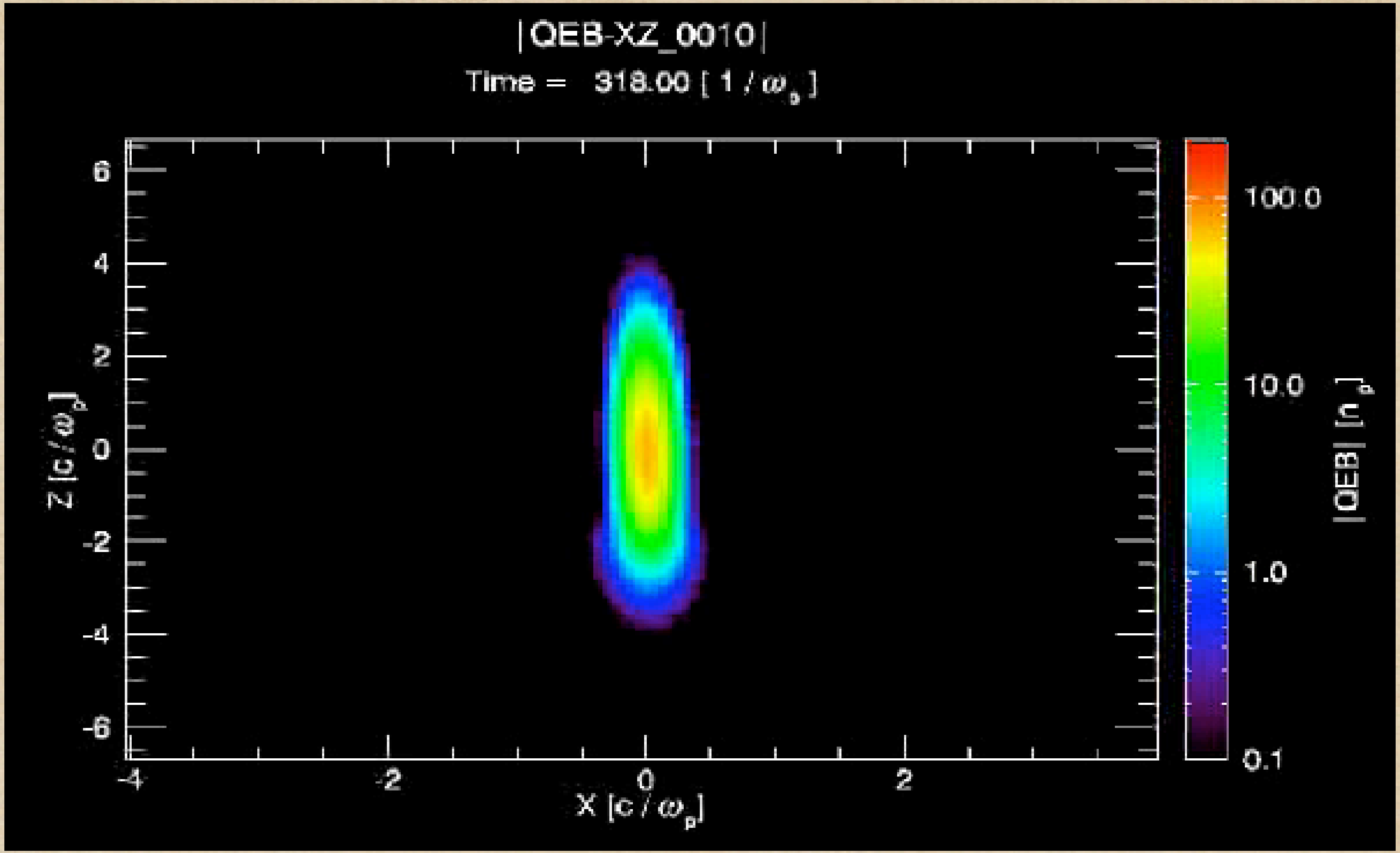


```
Frame #0
```

```
<x>=52.9 Δx=2.8 <y>=52.9 Δy=2.8 Δr=4.0
<x>=52.9 Δx=2.8 <y>=32.0 Δy=2.8 Δr=4.0
<x>=52.9 Δx=2.8 <y>=11.0 Δy=2.8 Δr=4.0
<x>=32.0 Δx=2.8 <y>=52.9 Δy=2.8 Δr=4.0
<x>=31.9 Δx=2.8 <y>=32.0 Δy=2.8 Δr=3.9
<x>=31.9 Δx=2.8 <y>=11.0 Δy=2.8 Δr=4.0
<x>=11.0 Δx=2.8 <y>=52.9 Δy=2.8 Δr=4.0
<x>=11.0 Δx=2.8 <y>=31.9 Δy=2.8 Δr=4.0
<x>=11.0 Δx=2.8 <y>=11.0 Δy=2.8 Δr=4.0
```

QPIC: 2-D Quantum Collisions

J. Tonge, D. Dauger, & V. Decyk



QuickPIC: Hosing instability in plasma wakefield accelerator

Chengkun Huang and E162 Collaboration