Some thoughts for particle-in-cell on leadership class computers

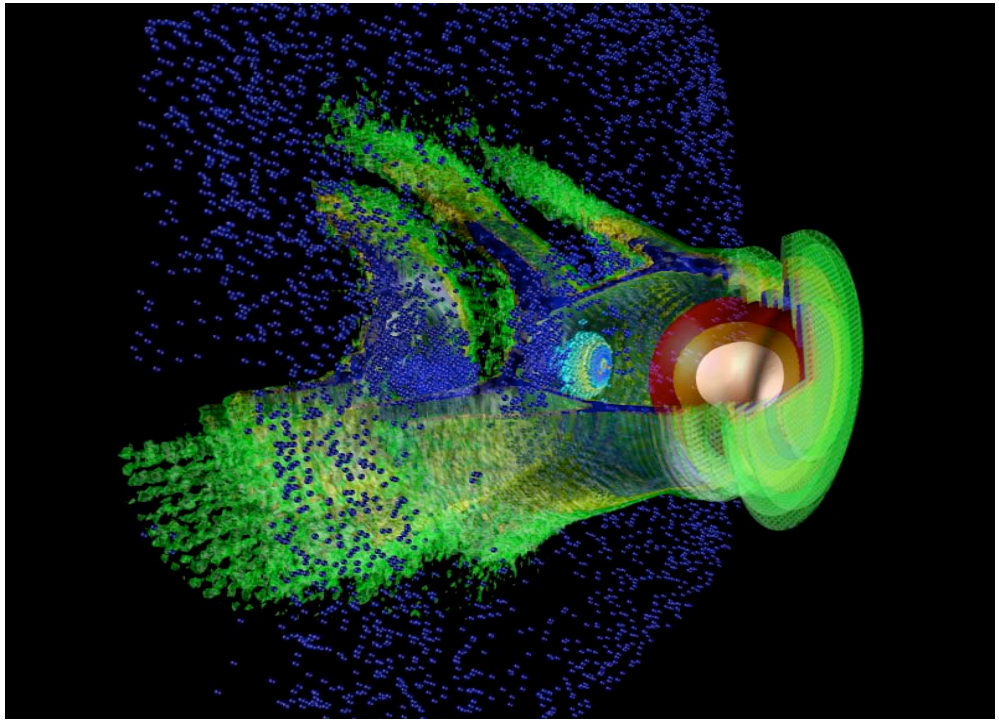
W. B. Mori

University of California
Los Angeles (UCLA)
Departments of Physics and
Astronomy and of Electrical Engineering

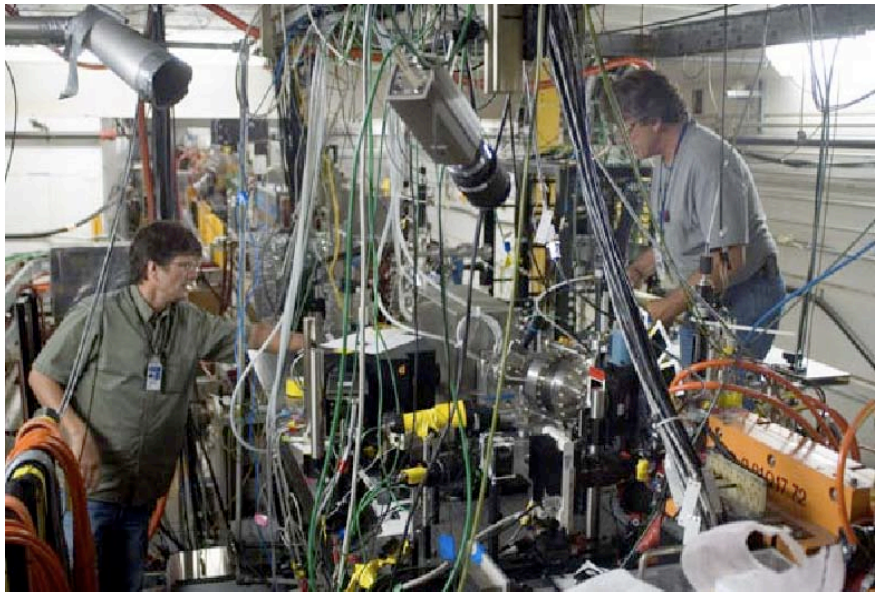
Institute of Digital Research and Education

Mori@physics.ucla.edu

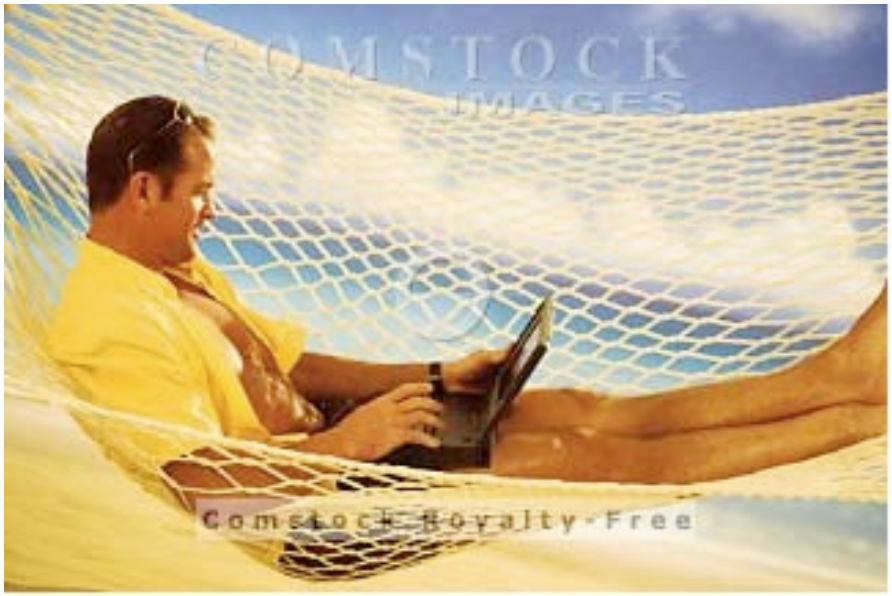
And much of the material from the Joule Metric study is from:
R.A. Fonseca
GoLP/IPFN, Instituto Superior Técnico, Lisboa, Portugal



My experimental colleagues view of the world



Life of an experimentalist



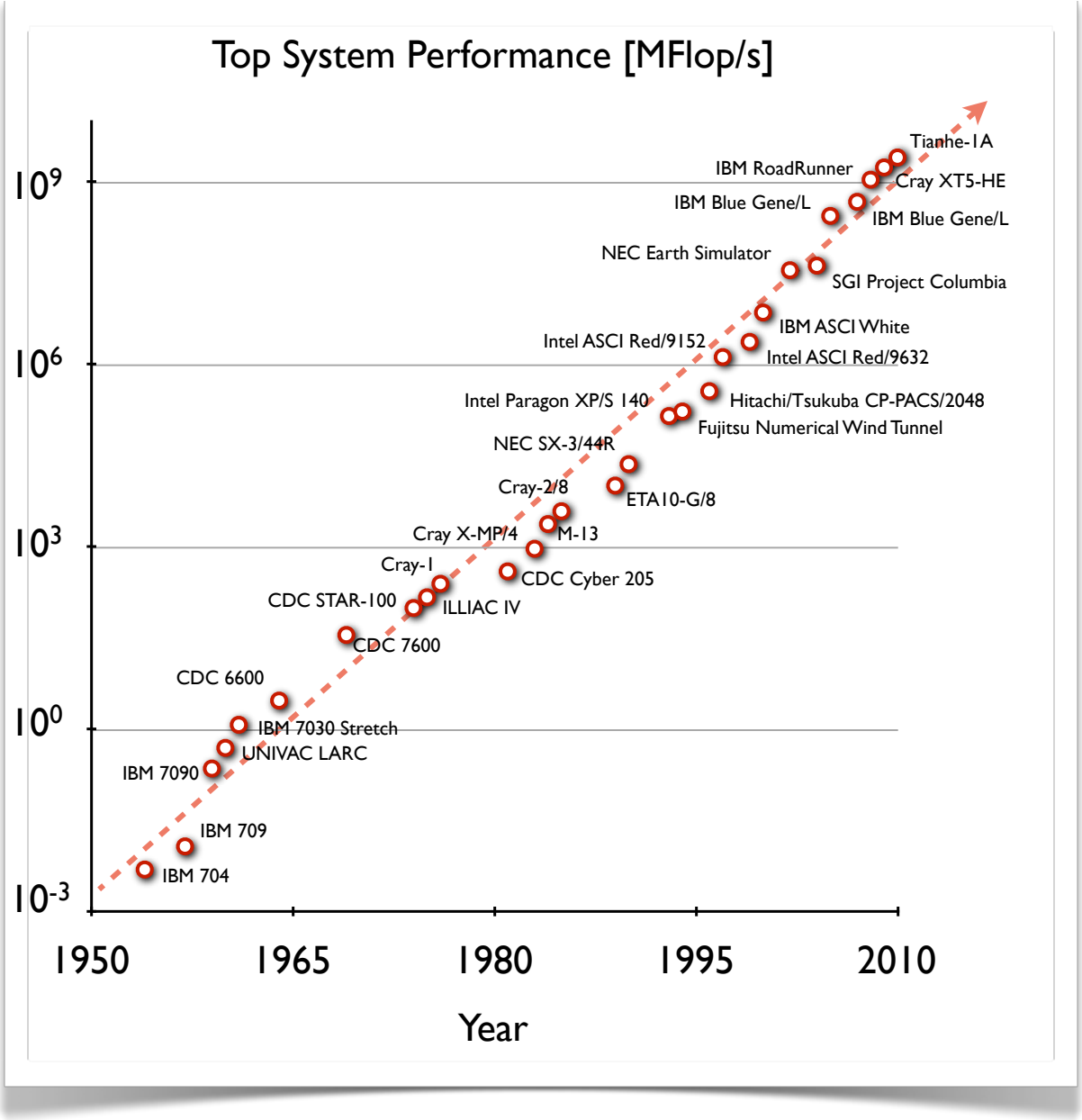
Life of a simulationist

But we face a tsunami of advances...

...in theory, in computational methods, in new hardware, and in new algorithms

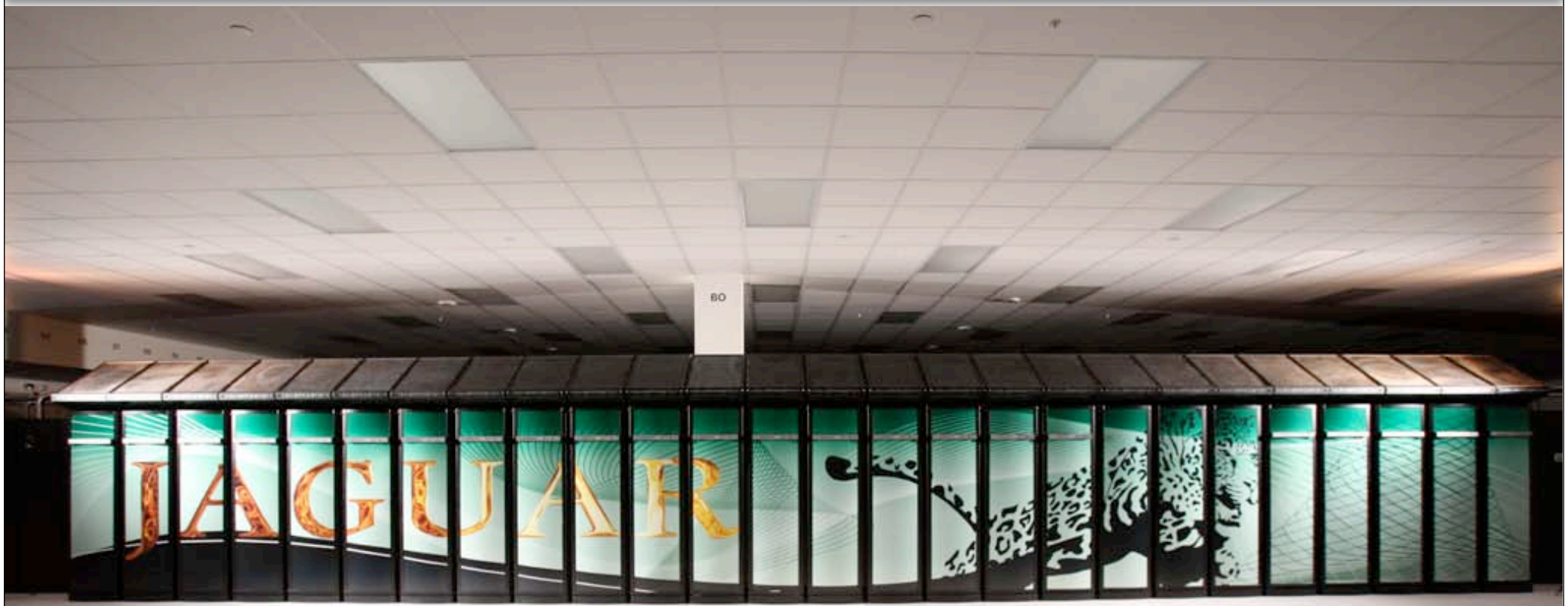
Contrary to the perception of my experimental colleagues, we rarely get to "relax" . This workshop offers us a rare opportunity for us to think in a "quiet room".

The evolution of state-of-the-art HPC systems cannot be ignored



Memory also has increased!

Petascale computing has arrived and exascale is around the corner



- Jaguar (jaguarpf)
  - 18688 compute nodes
  - dedicated service/login nodes
  - SeaStar2+ network
- XT5 Compute node
  - Dual hex-core AMD Opteron 2435 (Istanbul) at 2.6 GHz
  - 16GB DDR2-800 memory
- Complete system

  224256 processing cores

  300 TB of memory

  Peak performance **2.3 PFlop/s**

Outline

- **Introduction/Motivation**
- **The particle-in-cell approach**
- **Code developments for HEDP modeling in HPC systems**
- **Conclusions and perspectives**

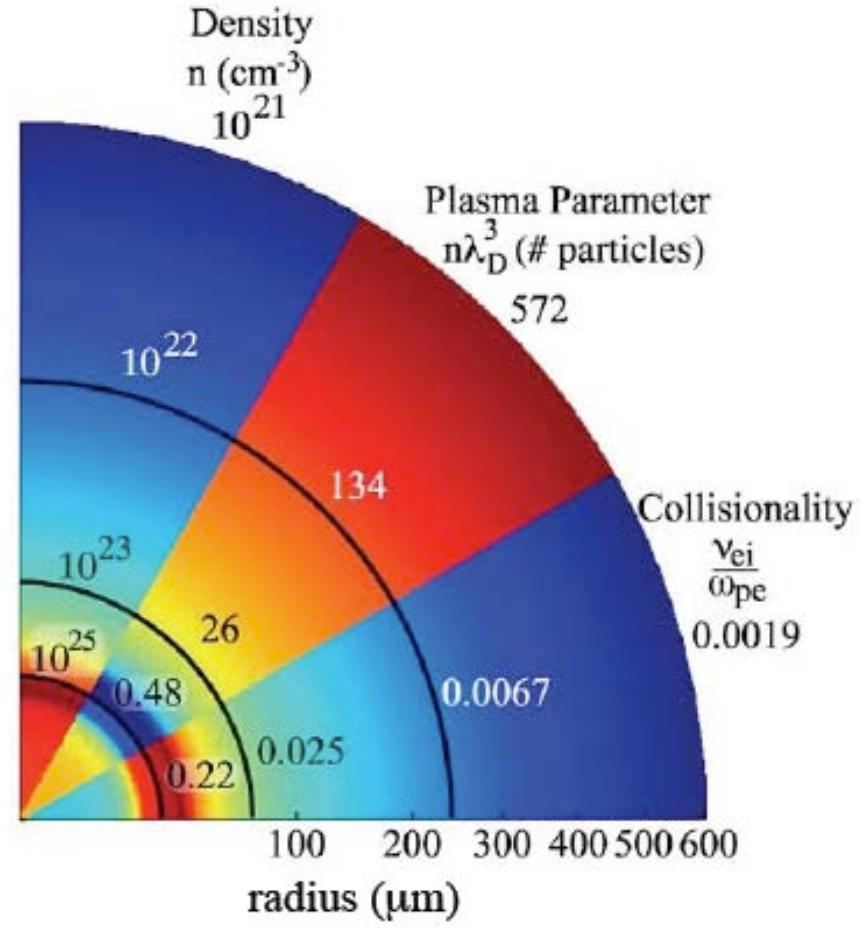
Outline

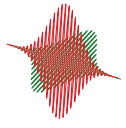
- **Introduction/Motivation**
- The particle-in-cell approach
- Code developments for HEDP modeling in HPC systems
- Conclusions and perspectives

Advanced ICF modeling is demanding due to different scales involved

eg. fast Ignition or shock ignition

Typical compressed target





Laser duration = 10 ps - 10 ns

Computational requirements for PIC

Physical size

Box size: 1 mm
Cell size: 5 Å
Duration: 10 ps
Time step: 1 as (10^{-18} s)

Numerical size

cells/dim: 2×10^6
particles/cell: 100 (1D); 10 (2D); 1 (3D)
time steps: 10^6

Particle push time: 1 ms

Computational time

1D - 2×10^3 CPU days
2D - 5×10^8 CPU days ~ 10^6 CPU years
3D - 2×10^{11} CPU days ~ 7×10^8 CPU years

What is high energy density plasma physics?

Why are both plasma-based acceleration and the nonlinear optics of plasmas considered high-energy density plasma research?

- High energy density, means high pressure
  - What is a high pressure?
    - MBar? GBar?
  - Need a dimensionless parameter
    - In plasma physics an important parameter is the number of particles in a Debye sphere (which is directly related to the ratio of the kinetic energy of an electron to the potential energy between particles). It measures the discreteness of the plasma.

$$\frac{4\pi}{3} n\lambda_d^3 \equiv N_D = 2.1 \times 10^3 \frac{T_{keV}^2}{P_{MBar}^{1/2}}$$

  - When the pressure exceeds ~1 MBar then the discrete nature of the plasma becomes important:
    - N_D is not “infinite”

- Discreteness makes developing computational methods difficult

How can we describe the plasma physics?

Hierarchy of descriptions (ignoring quantum effects)

- Klimontovich equation ("exact").
- Ensemble average the Klimontovich equation
  - Leads to Vlasov Fokker Planck equation (approximate)
- Take the limit that N_D is very very large
  - Vlasov equation
- Take moments of the Vlasov or Vlasov Fokker Planck equation
  - "Two" fluid (transport) equations
- Ignore electron inertia
  - Single fluid or MHD equations

Klimontovich description ("exact")

Klimontovich equation

$$\frac{D}{Dt}F = 0$$

$$F(\vec{x},\vec{v};t) = \sum_i^N \delta(\vec{x}-\vec{x}_i(t))\delta(\vec{v}-\vec{v}_i(t))$$

$$\left[\frac{D}{Dt} \equiv \partial_t + \vec{v}\cdot\nabla_x + \vec{a}\cdot\nabla_v\right]$$

$$\vec{a} \equiv \frac{d}{dt}\vec{v} = \frac{q}{m}\left(\vec{E} + \frac{\vec{v}}{c}\times\vec{B}\right)$$

Maxwell's equations

$$\frac{\partial}{\partial t}\vec{E} = \nabla\times\vec{B} - \frac{4\pi}{c}\vec{J}$$

$$\frac{\partial}{\partial t}\vec{B} = -\nabla\times\vec{E}$$

$$\vec{J}(\vec{x},t) = \int d\vec{v}\; q\vec{v}\; F(\vec{x},\vec{v},t)$$

Descriptions of a plasma

Not practical to follow every particle

- Ensemble average of Klimontovich equation

- Ensemble averages

$$F = F_s + \delta F \quad F_s \equiv < F >$$

$$\vec{J} = \vec{J}_S + \delta\vec{J}$$

$$\vec{B} = \vec{B}_S + \delta\vec{B}$$

$$\vec{E} = \vec{E}_S + \delta\vec{E}$$

$$\vec{a}_s = \frac{q}{m}\left(\vec{E}_s + \frac{\vec{v}}{c} \times \vec{B}_s\right)$$

$$\left.\frac{D}{Dt}\right|_s = \frac{\partial}{\partial t} + \vec{v}\cdot\nabla_x + \vec{a}_s \cdot \nabla_v$$

Equation for smooth F

$$\left.\frac{D}{Dt}\right|_S F_S = - < \delta\vec{a}\nabla_v \delta f >= \left.\frac{d}{dt}\right|_{col} F_S$$

$$\left.\frac{D}{Dt}\right|_s \delta F = -\delta\vec{a}\ \nabla_v F_s + \delta\vec{a}\ \nabla_v \delta F - < \delta\vec{a}\nabla_v \delta F >$$

Descriptions of a plasma

Where does the Vlasov equation fit into this

- Smooth F descriptions
- Vlasov Fokker Planck

$$\left.\frac{D}{Dt}\right|_S F_S = - < \delta\vec{a}\nabla_v \delta f >= \left.\frac{d}{dt}\right|_{col} F_S$$

$$\left.\frac{D}{Dt}\right|_s \delta F = -\delta\vec{a}\cdot\nabla_v F_s$$

Sometimes called the quasi-linear approximation

Vlasov Equation

$$\left.\frac{D}{Dt}\right|_s F_s = 0$$

ONLY TRUE IF $N_D >>> 1$

Descriptions of a plasma

Where does the Vlasov equation fit into this

- Smooth F descriptions
- Vlasov Fokker Planck

$$\left.\frac{D}{Dt}\right|_S F_S = - < \delta\vec{a}\nabla_v \delta f >= \left.\frac{d}{dt}\right|_{col} F_S$$

$$\left.\frac{D}{Dt}\right|_s \delta F = -\delta\vec{a}\cdot\nabla_v F_s$$

Sometimes called the quasi-linear approximation

Vlasov Equation

$$\left.\frac{D}{Dt}\right|_s F_s = 0$$

ONLY TRUE IF $N_D >>> 1$

Outline

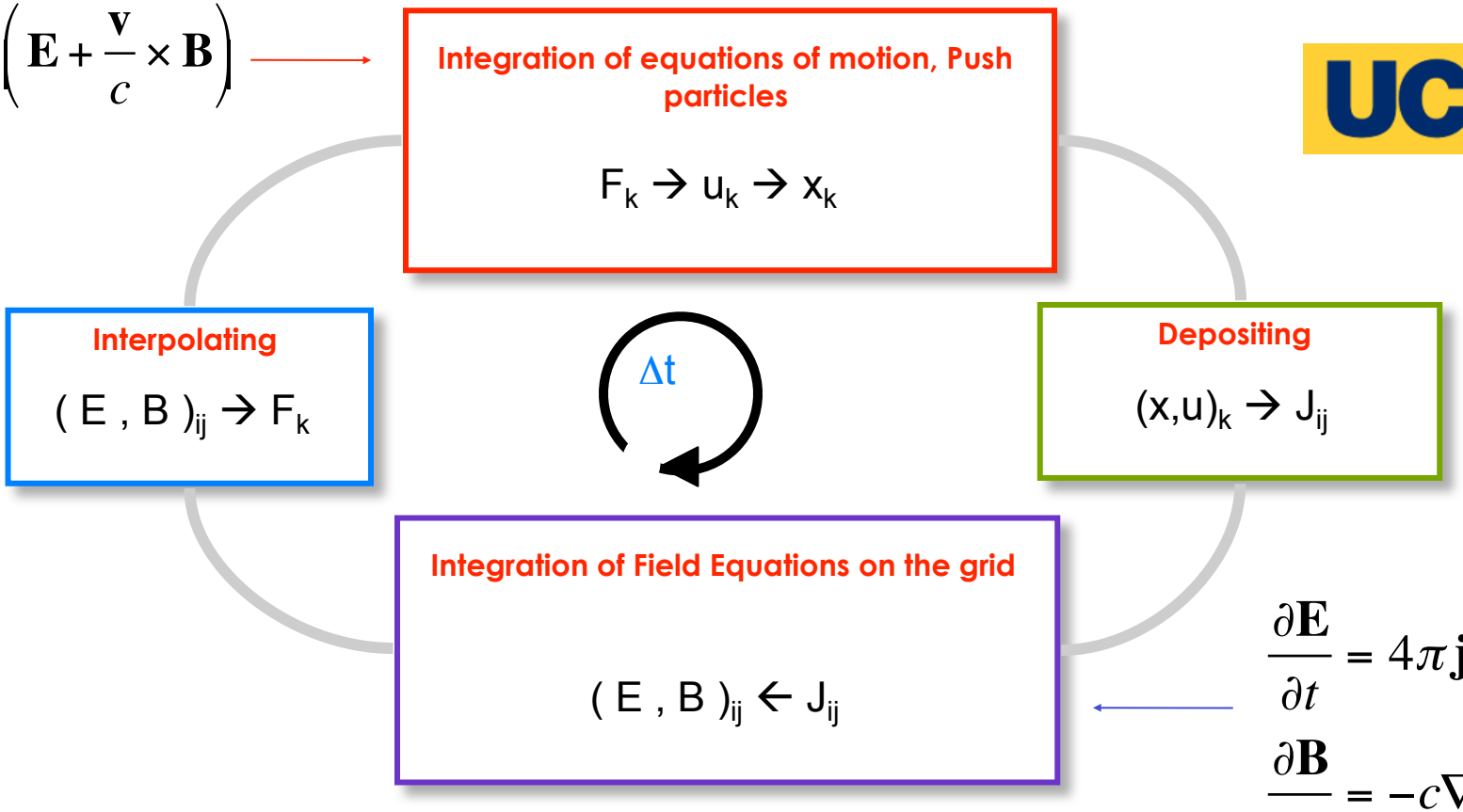
- Introduction/Motivation
- **The particle-in-cell approach**
- Code developments for HEDP modeling in HPC systems
- Conclusions and perspectives

What computational method do we use to model HEDP including discrete effects?

The Particle-in-cell method

Not all PIC codes are the same!

$$\frac{d\mathbf{p}}{dt} = q\left(\mathbf{E} + \frac{\mathbf{v}}{c} \times \mathbf{B}\right)$$



$$\frac{\partial \mathbf{E}}{\partial t} = 4\pi\mathbf{j} - c\nabla \times \mathbf{B}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -c\nabla \times \mathbf{E}$$

What is the particle-in-cell model?

Is it an efficient way of modeling the Vlasov equation?
No, it is a Klimontovich description for finite size (macro-particles)

Klimontovich eq. of macro-particles

$$\frac{D}{Dt}F = 0$$

$$F(\vec{x},\vec{v};t) = \sum_i^N S_p(\vec{x}-\vec{x}_i(t))\delta(\vec{v}-\vec{v}_i(t))$$

$$\left[\frac{D}{Dt} \equiv \partial_t + \vec{v}\cdot\nabla_x + \vec{a}\cdot\nabla_v\right]$$

$$\vec{a} \equiv \frac{d}{dt}\vec{v} = \frac{q}{m}\left(\vec{E} + \frac{\vec{v}}{c}\times\vec{B}\right)$$

Maxwell's equations

$$\frac{\partial}{\partial t}\vec{E} = \nabla\times\vec{B} - \frac{4\pi}{c}\vec{J}$$

$$\frac{\partial}{\partial t}\vec{B} = -\nabla\times\vec{E}$$

$$\vec{J}(\vec{x},t) = \int d\vec{v}\; q\vec{v}\; F(\vec{x},\vec{v},t)$$

Does it matter that we agree on what it is? I think it does.

Yes. It changes your view of what convergence tests should be done to VERIFY the implementation of the algorithm?

Yes. Today's computers are getting very powerful. We can do runs with smaller particle sizes (compared to Debye length) and with more particles. Converge to "real" conditions.

Yes. A single PIC simulation is not an ensemble average. So for some problems, perhaps we should start thinking about taking ensemble averages.

Yes. PIC simulations can be used to VALIDATE reduced models

Outline

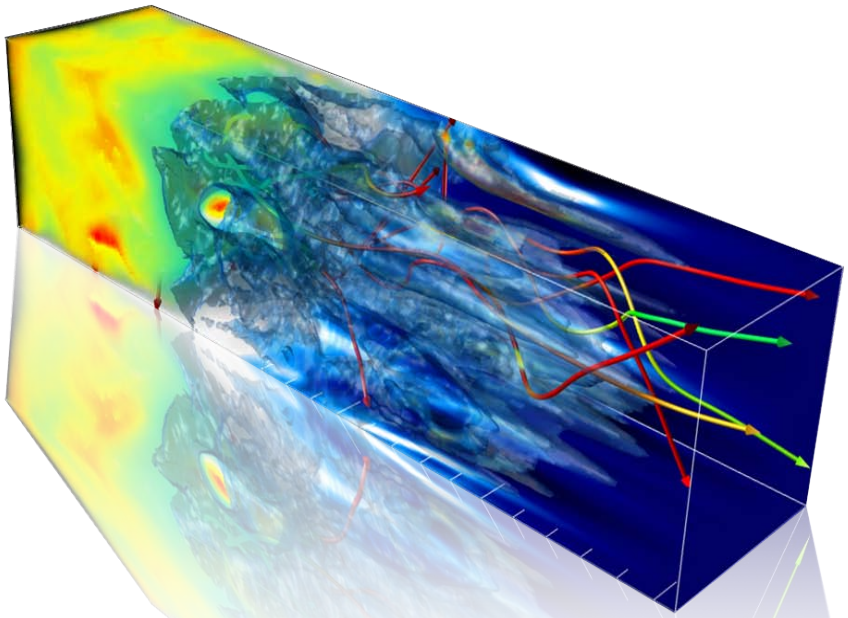
- Introduction/Motivation
- The particle-in-cell approach
- **Code developments for HEDP modeling in HPC systems**
- Conclusions and perspectives

OSIRIS 2.0

osiris v2.0

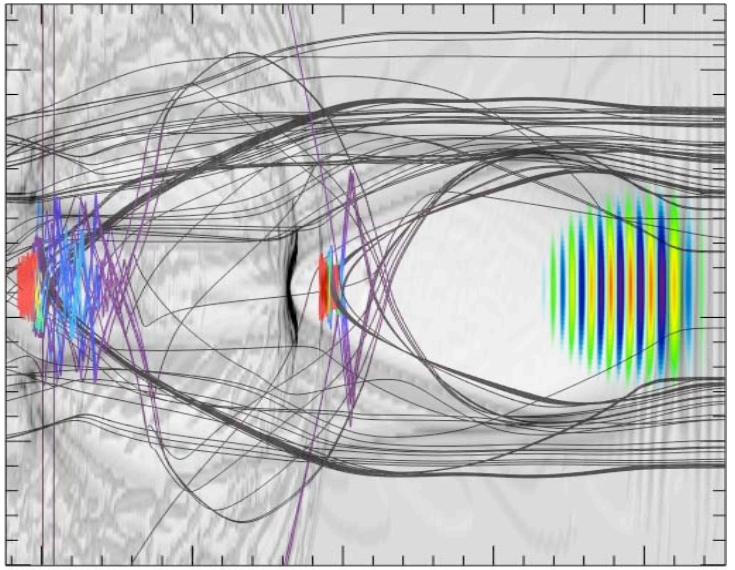
osiris framework

- Massivelly Parallel, Fully Relativistic Particle-in-Cell (PIC) Code
- Visualization and Data Analysis Infrastructure
- Developed by the osiris.consortium
  ⇒ UCLA + IST



Ricardo Fonseca: ricardo.fonseca@ist.utl.pt
Frank Tsung: tsung@physics.ucla.edu

http://cfp.ist.utl.pt/golp/epp/
http://exodus.physics.ucla.edu/



New Features in v2.0

- Optimized igh-order splines
- Binary Collision Module
- PML absorbing BC
- Tunnel (ADK) and Impact Ionization
- Dynamic Load Balancing
- Parallel I/O
- Hybrid options

Want to discuss what performance is currently possible with PIC.

This sets a bar for other kinetic approaches for HEDP

Results from a Joule Metric Excercise on Jaguar:

- ~30% of peak speed can be obtained
- dynamic load balancing can be important

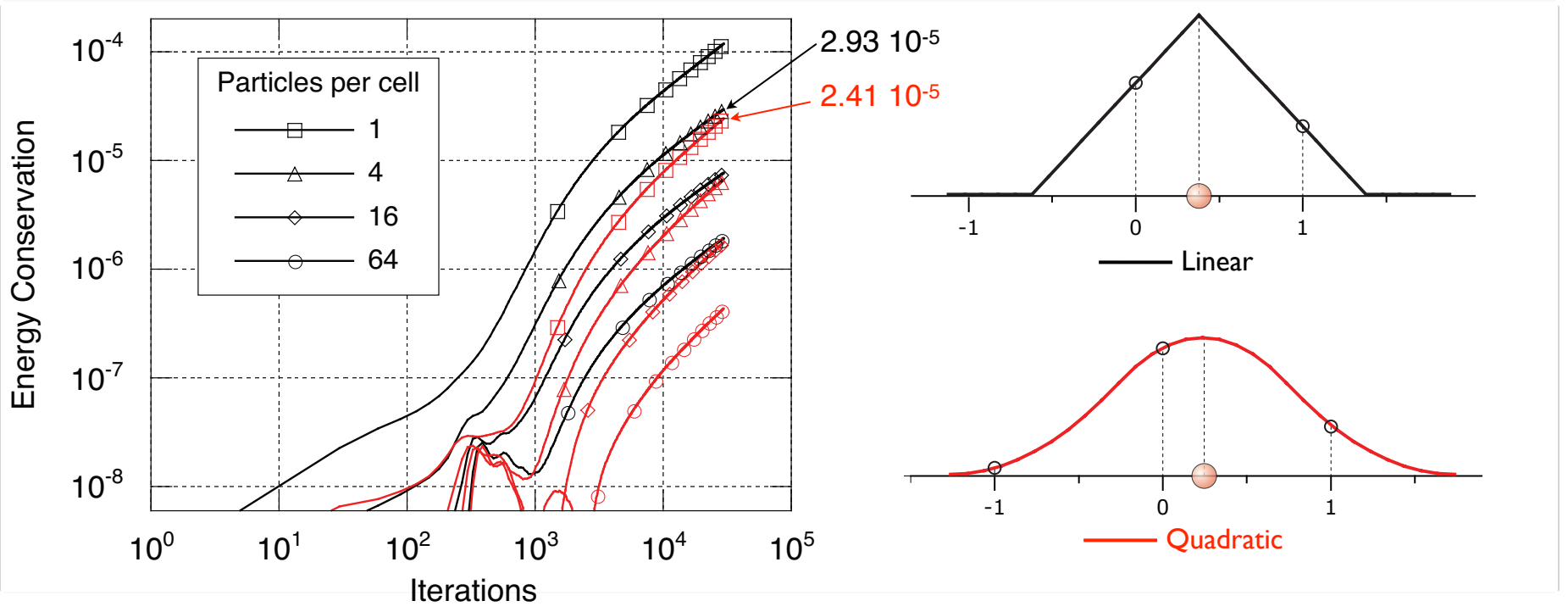
High-order particle weighting

Run simulations for up to ~ 10^7 iterations

- Stable algorithm
- Energy conservation
- Numerical noise seeded instabilities

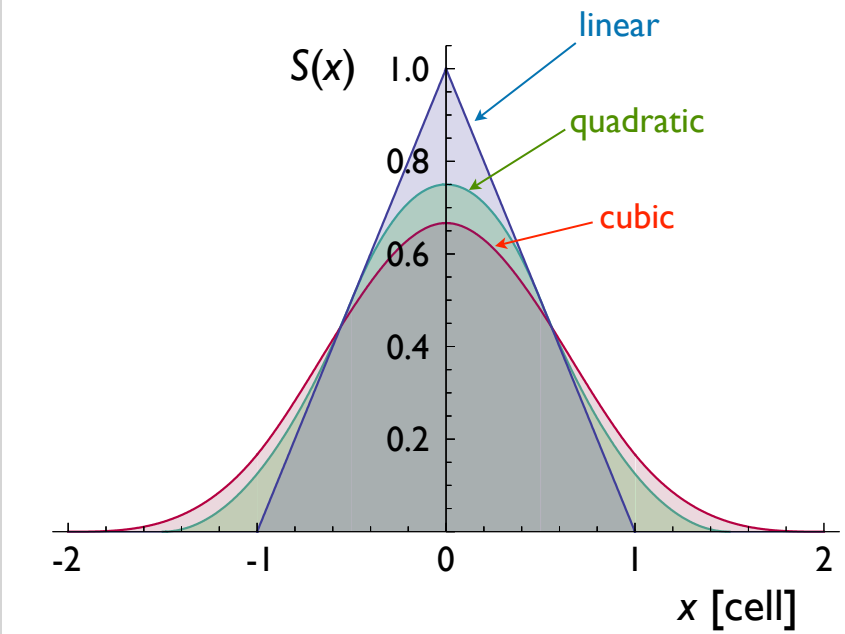
Control numerical self-heating

- Increase number of particles per cell
- Use high-order particle weighting



Calculations overhead

Order	Weights	Interpolation
linear	$W_0 = 1-\Delta$ $W_1 = \Delta$	$A_{int} = \sum_{i=0}^{1} W_i A_i$
quadratic	$W_{-1} = \frac{1}{8}(1-2\Delta)^2$ $W_0 = \frac{3}{4}-\Delta^2$ $W_1 = \frac{1}{8}(1+2\Delta)^2$	$A_{int} = \sum_{i=-1}^{1} W_i A_i$
cubic	$W_{-1} = -\frac{1}{6}(-1+\Delta)^3$ $W_0 = \frac{1}{6}\left(4-6\Delta^2+3\Delta^3\right)$ $W_1 = \frac{1}{6}\left(1+3\Delta+3\Delta^2-3\Delta^3\right)$ $W_2 = \frac{\Delta^3}{6}$	$A_{int} = \sum_{i=-1}^{2} W_i A_i$



B-spline blending functions

Ops	Linear	Quadratic	Cubic
2D	219	480	890
3D	349	1180	2863

~ 3.4 ×

performance

Measured performance:

- 3D quadratic ~ **2.0 ×** slower than 3D linear
- pipeline and cache effects

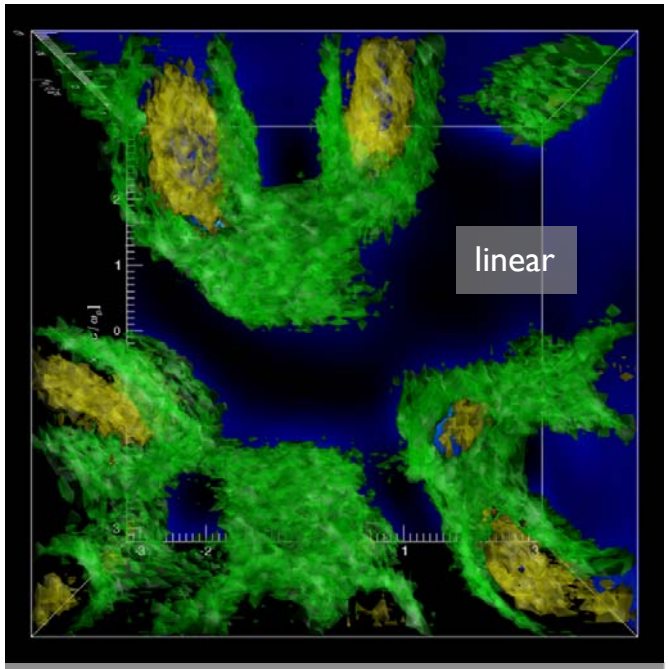
Running with high order interpolation

linear

Significantly noisier

Inferior energy conservation

- 16 × worse than quadratic

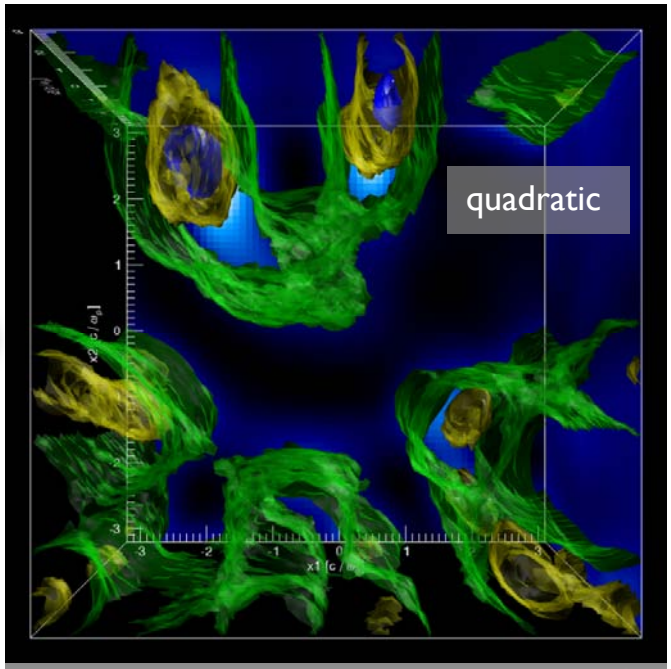


Higher Order Interpolation

quadratic

Small overhead

- advance/deposit 1.85 ×
- total simulation 1.45 ×

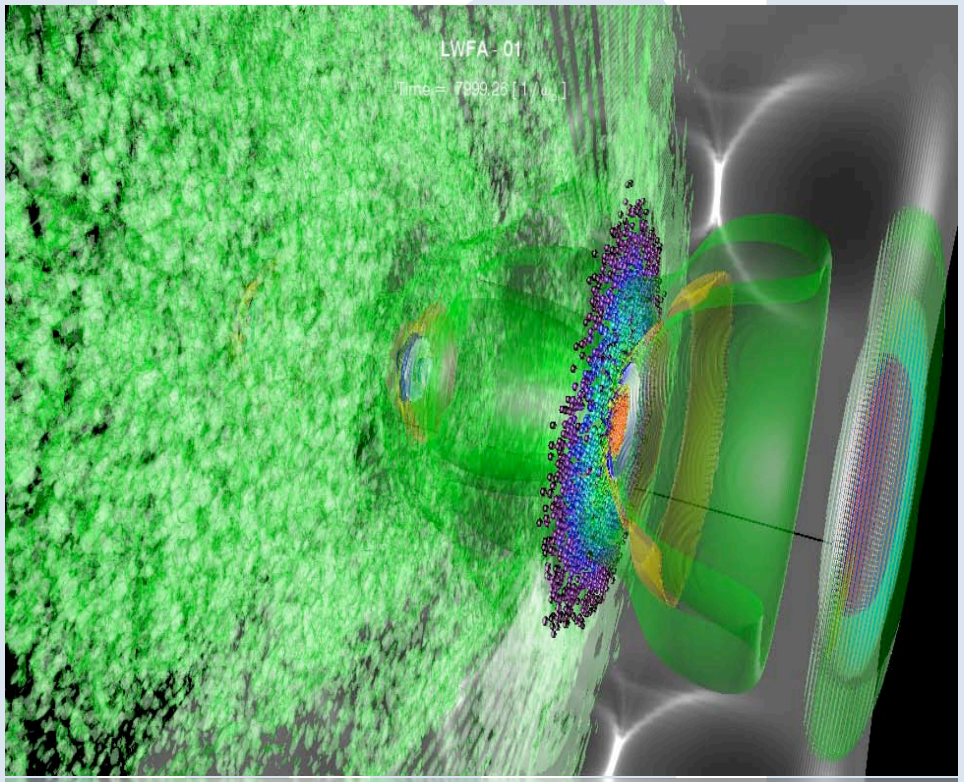


Same level of energy conservation can be achieved faster!

OASCR Software Effectiveness Metric

OASCR Joule

- Analyze/improve applications requiring high capability/capacity HPC systems
- 4 Applications selected for 2011
  - OMEN/NEMO 5, LAMMPS, OSIRIS and eSTOMP
- Q2 test problems
  - Warm plasma, baseline code performance
  - 200 TW (6 Joule) → 1.5×10^{18} cm^{-3} uniform plasma
  - 1 PW (30 Joule) → 0.5×10^{18} cm^{-3} uniform plasma
- Run in ~ 1/4 of the full machine (Jaguar)
  - 55296 cores
  - 20 hours wall clock
  - 1.1 M cpu hours
- Scale up to the full machine by Q4

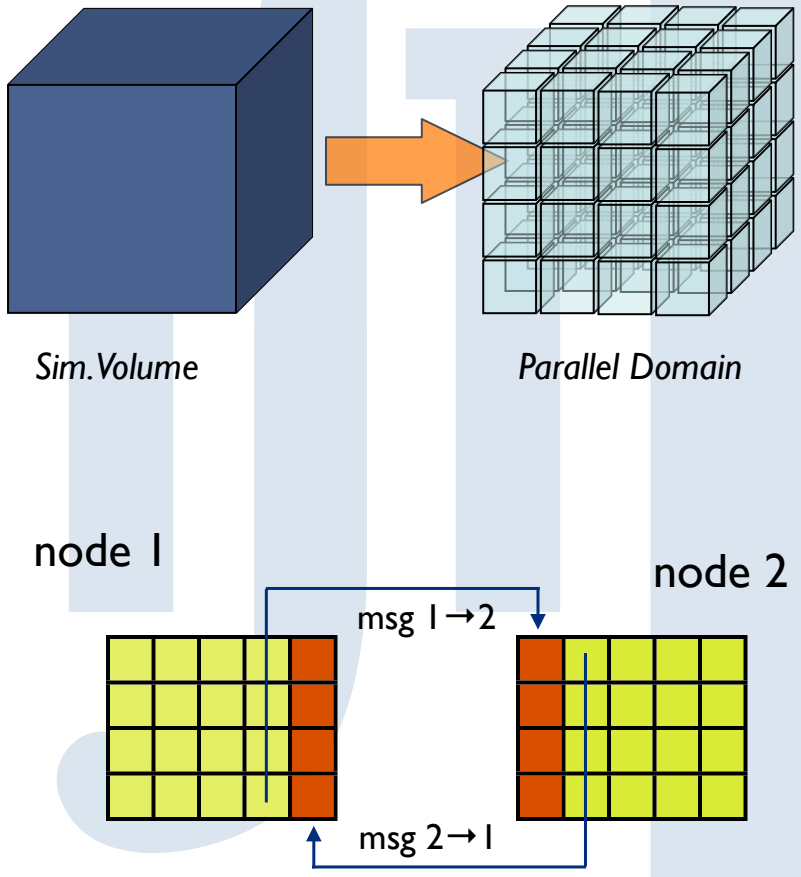


Run	Grid	Simulation Box [c/ω_0]	Particles	Iterations	Laser a_0	Ions
Warm test	6144 × 6144 × 1536	614.4 × 614.4 × 153.6	4.46×10^{11}	5600	n/a	n/a
Run 1	8064 × 480 × 480	806.4 × 1171.88 × 1171.88	3.72×10^{9}	41000	4.0	fixed
Run 2	8832 × 432 × 432	1766.4 × 2041.31 × 2041.31	6.59×10^{9}	47000	4.58	fixed
Run 3	4032 × 312 × 312	806.4 × 1171.88 × 1171.88	1.26×10^{10}	52000	4.0	moving

HPC system level parallelism

Distributed memory

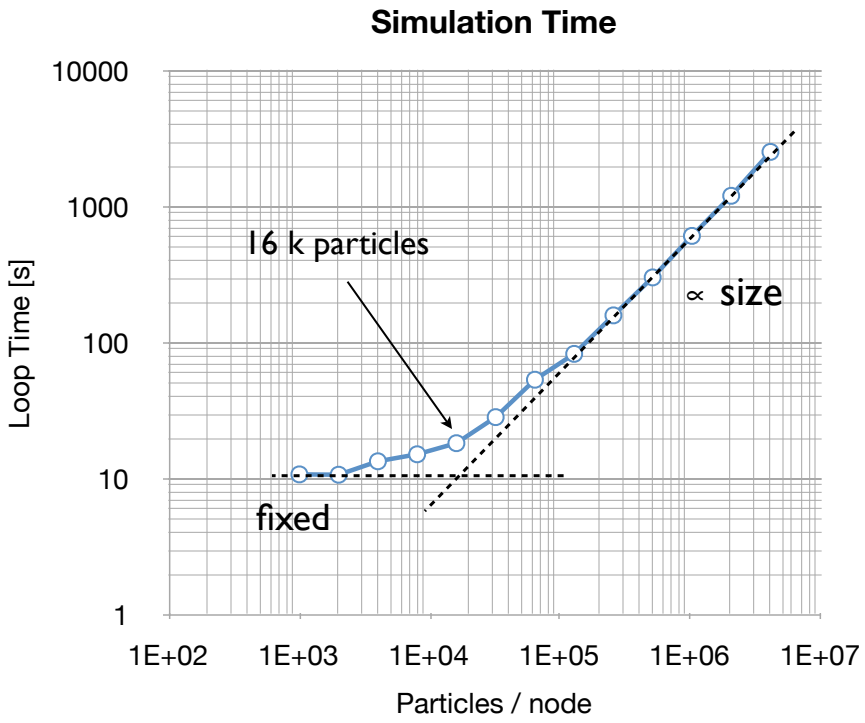
- Each process cannot directly access memory on another node:
  - Information is exchanged between nodes using network messages (MPI)
- Standard parallelization uses a spatial decomposition:
  - Each node handles a specific region of simulation space
- Works very well also on multi-core nodes
  - Message passing inside a node is very efficient
- Very efficient for uniform plasmas
- Susceptible to imbalance if particles accumulate in a subset of nodes



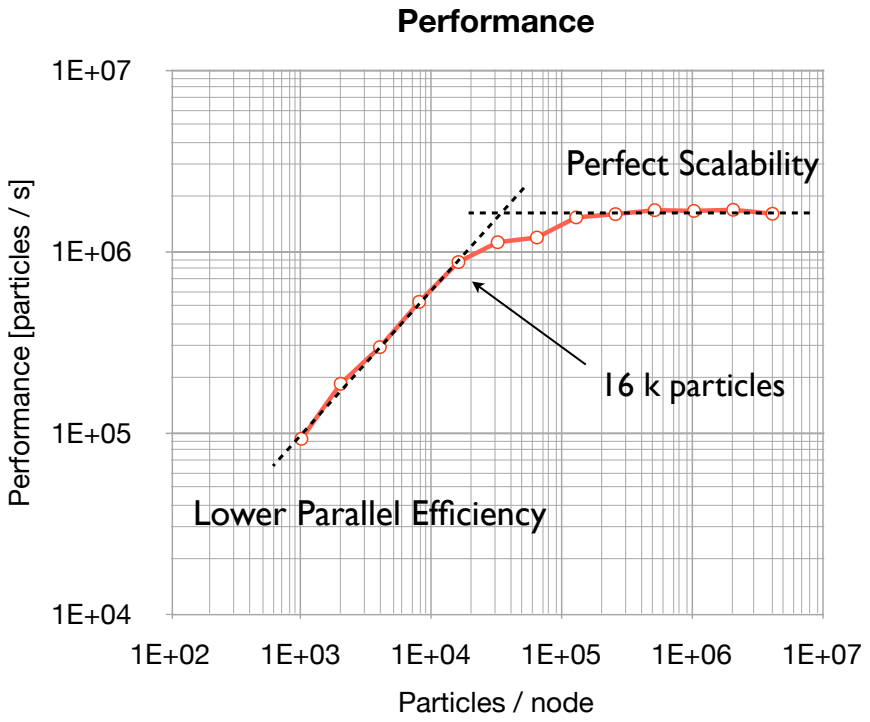
Parallel Overhead

Simulation Time

- Test problem:
  - 3D warm plasma, quadratic interpolation, 8 particles per cell
  - 216 (6^3) nodes
  - Scan the problem size starting at the smallest possible size (5^3 grid / MPI Process)



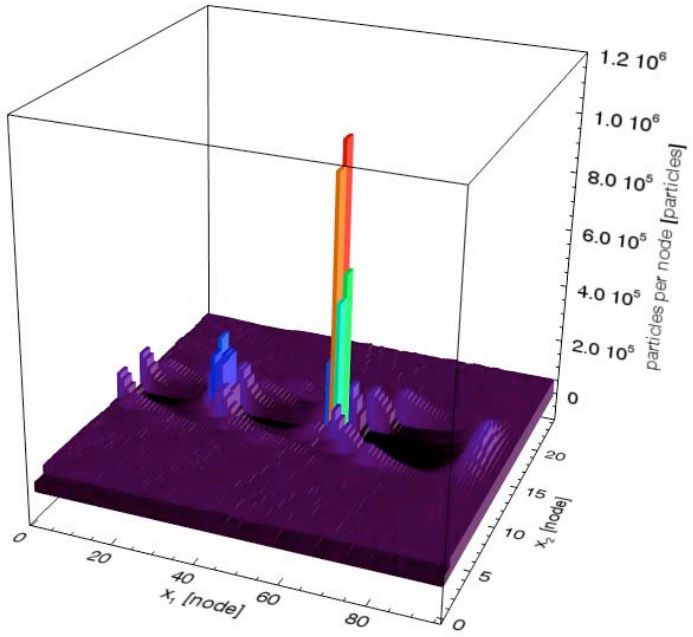
Code Performance

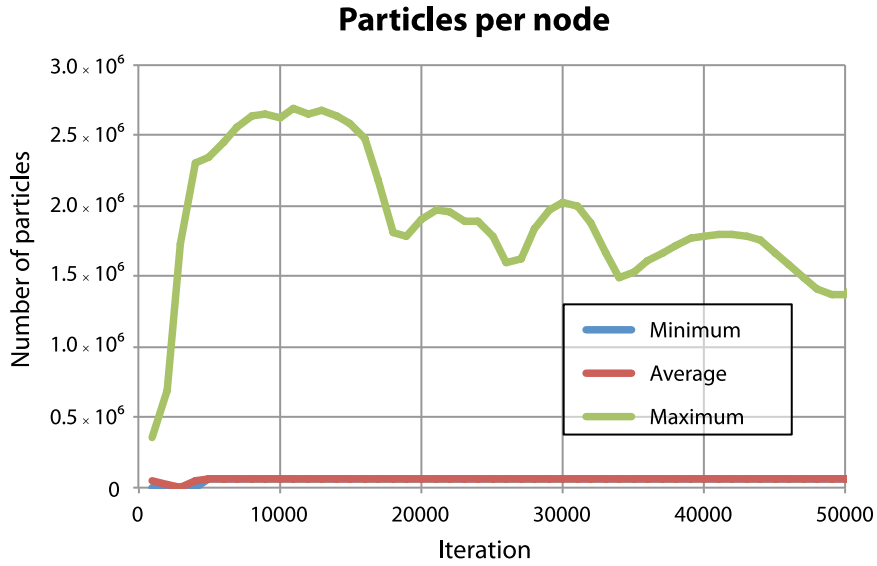


- Above 128 k particles/ 25 k cells per MPI process the performance is stable:
  - Perfect weak / strong scalability
- At 16 k particles per node the performance is ~ 50 % of peak

Load Imbalance Limitations

Run	Warm	LWFA-01	LWFA-02	LWFA-03
Partition	48 x 48 x 24	96 x 24 x 24	96 x 24 x 24	96 x 24 x 24
Min Particles/node	8.39E+06	1.20E+03	1.84E+01	6.01E+04
Max Particles/node	8.39E+06	1.11E+06	2.02E+06	1.89E+06
Avg Particles/node	8.39E+06	5.73E+04	9.69E+04	2.06E+05
Average Unbalance	**1**	**18.85**	**20.83**	**9.04**
Average Perf [Part/s]	7.62E+10	4.22E+09	3.72E+09	8.85E+09
Average Perf / core [M Part/s]	1.377	0.0764	0.0673	0.1601
Average Push Time [µs]	0.726	13.09	14.85	6.25
Push / Unbalance [µs]	0.726	0.695	0.713	0.691





LWFA-03

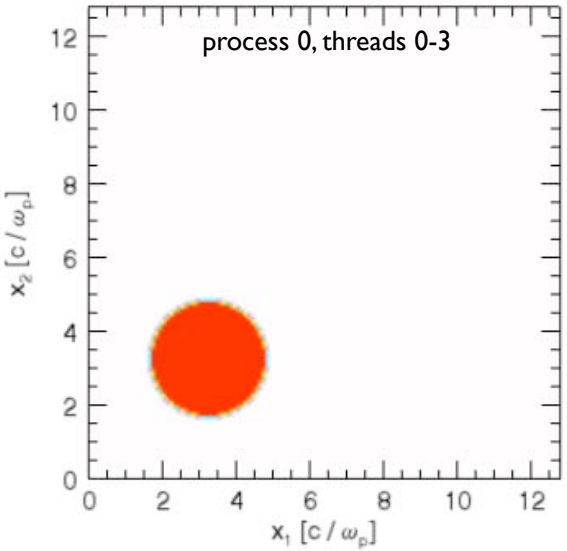
Improving Load Imbalance

Using Shared Memory

- A group of cores will share memory access / simulation region:
  - Distribute particles evenly across these cores

The workload for these cores will always be perfectly balanced

- Implement using OpenMP
  - Use 1 thread / core

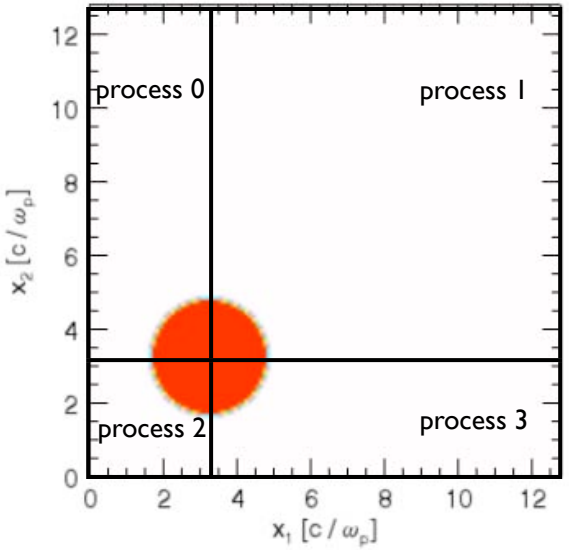


4 cores

Share particle load across processes

Dynamic Load Balance

- The code can change node boundaries dynamically to attempt to maintain a even load across nodes:
  - Determine best possible partition from current particle distribution
  - Reshape the simulation



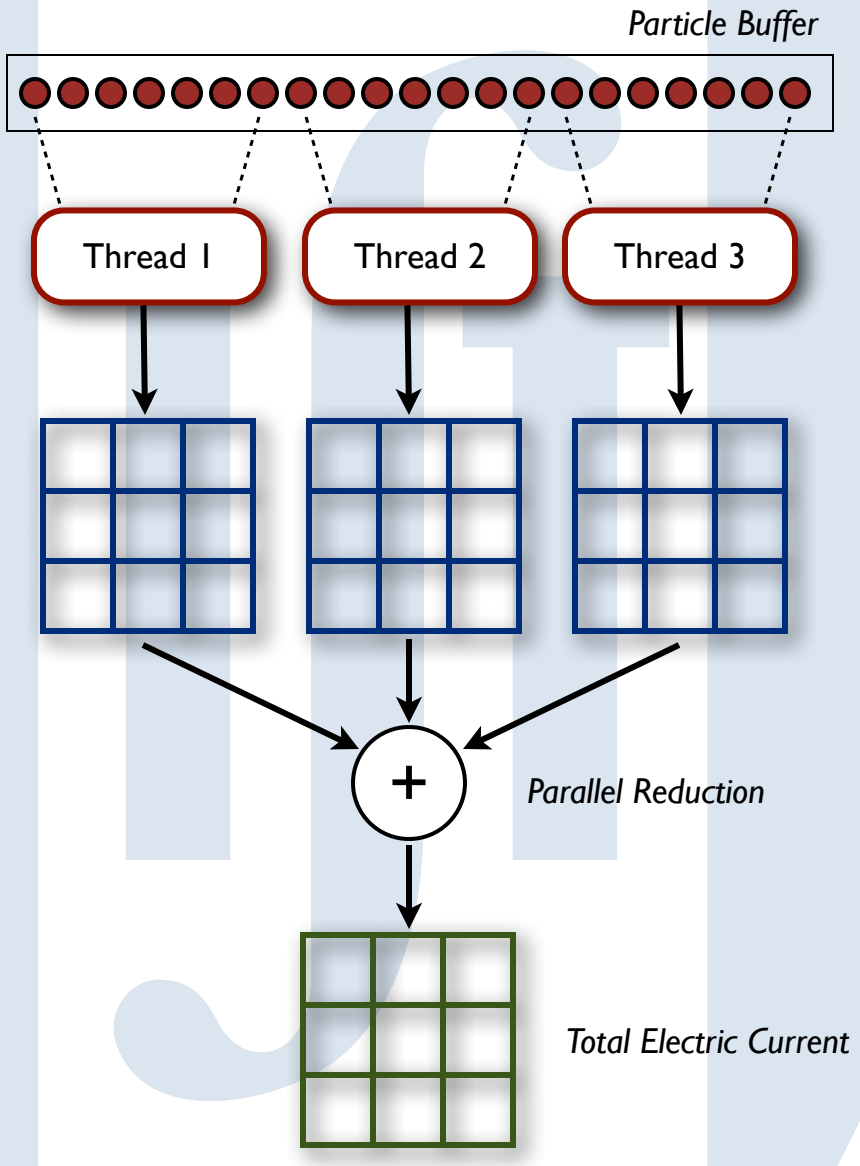
Even particle load across processes

4 cores

SMP particle pusher

Algorithm

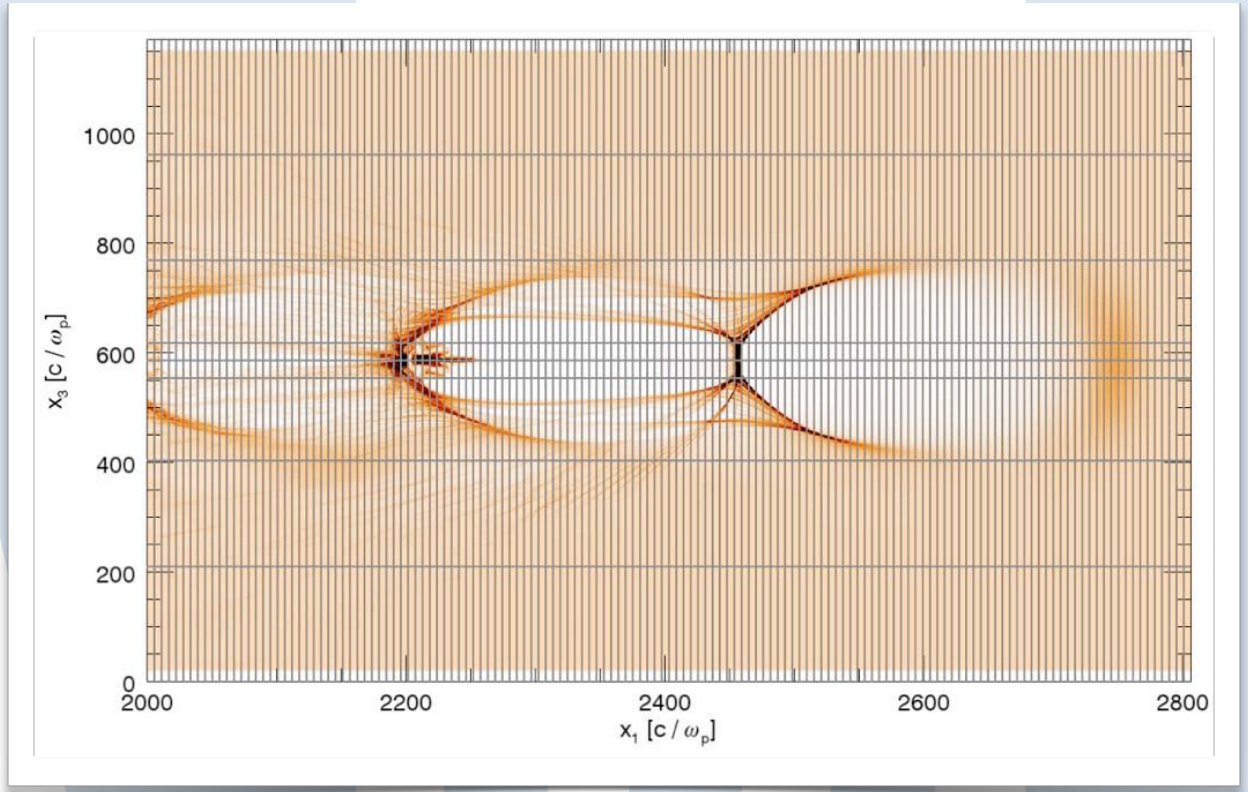
- An MPI process handles a given simulation region
- This process spawns *nt* threads to process the particles:
  - Use *nt* copies of the current grid
  - Divide particles evenly across threads
  - Each thread deposits current in only 1 of the grid copies
- Accumulate all current grids in a single current grid
  - Divide this work also over *nt* threads
- Algorithm overhead
  - Zeroing the additional grid copies
  - Reduction of multiple current grids



Dyn. Load Balance performance

No overall speedup

- Best result:
  - Dynamic load balance along x_2 and x_3
  - Use max load
  - 30% improvement in imbalance but...
  - Lead to a 5% overall slowdown!

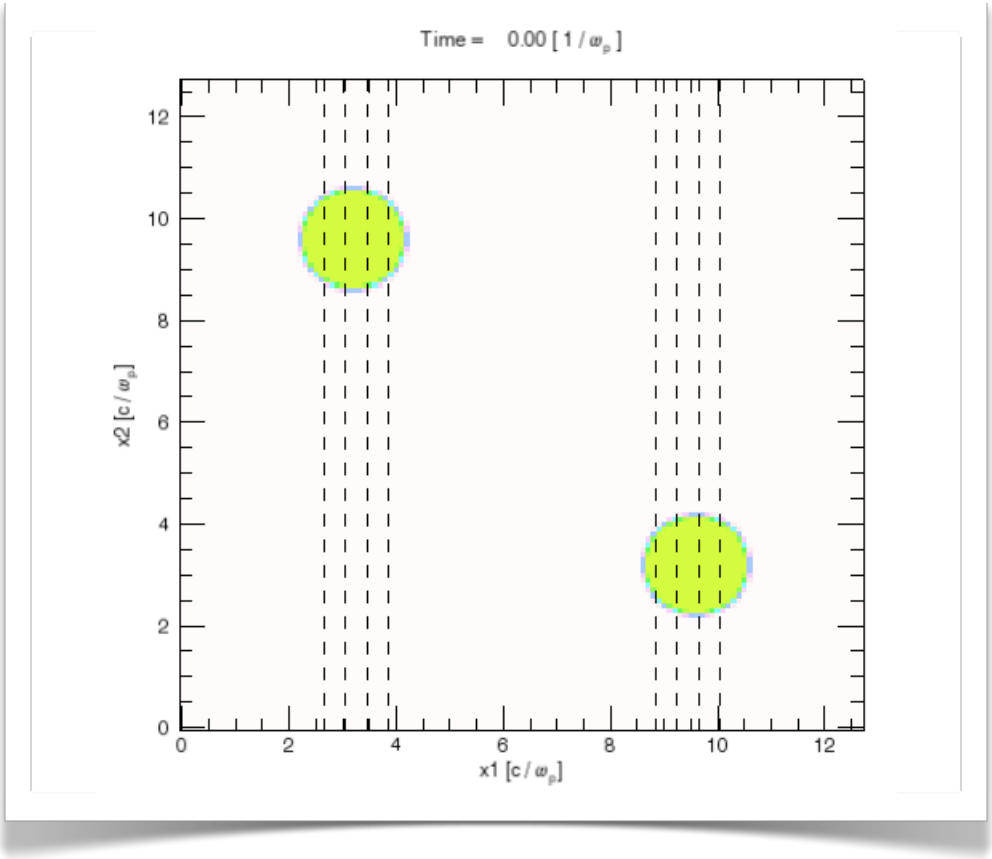


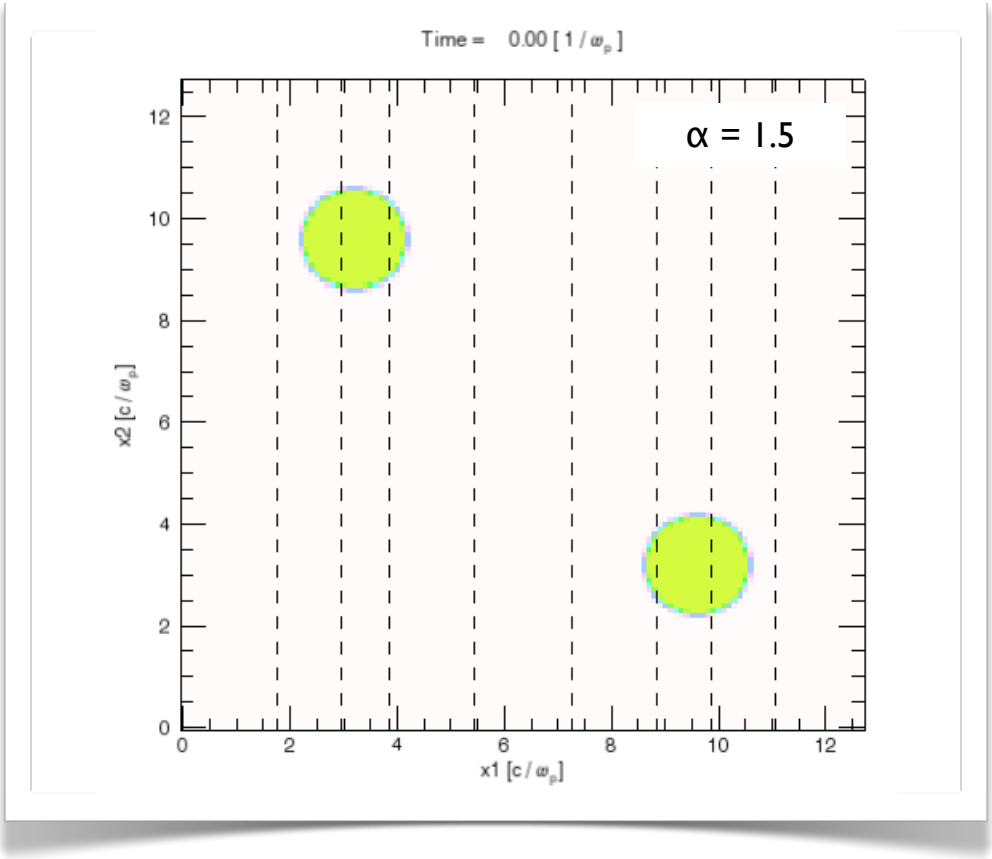
- The ASCR problems are very difficult to load balance
  - Very small problem size per node
  - When choosing partitions with less nodes along propagation direction imbalance degrades significantly
- Not enough room to dynamic load balance along propagation direction

- Dynamic load balancing in the transverse directions does not yield big improvements
  - Very different distributions from slice to slice
  - Dynamic load balance in just 1 direction does not improve imbalance
  - Using max node load helps to highlight the hot spots for the algorithm

Dynamic load balance

– – – – – Node Boundaries





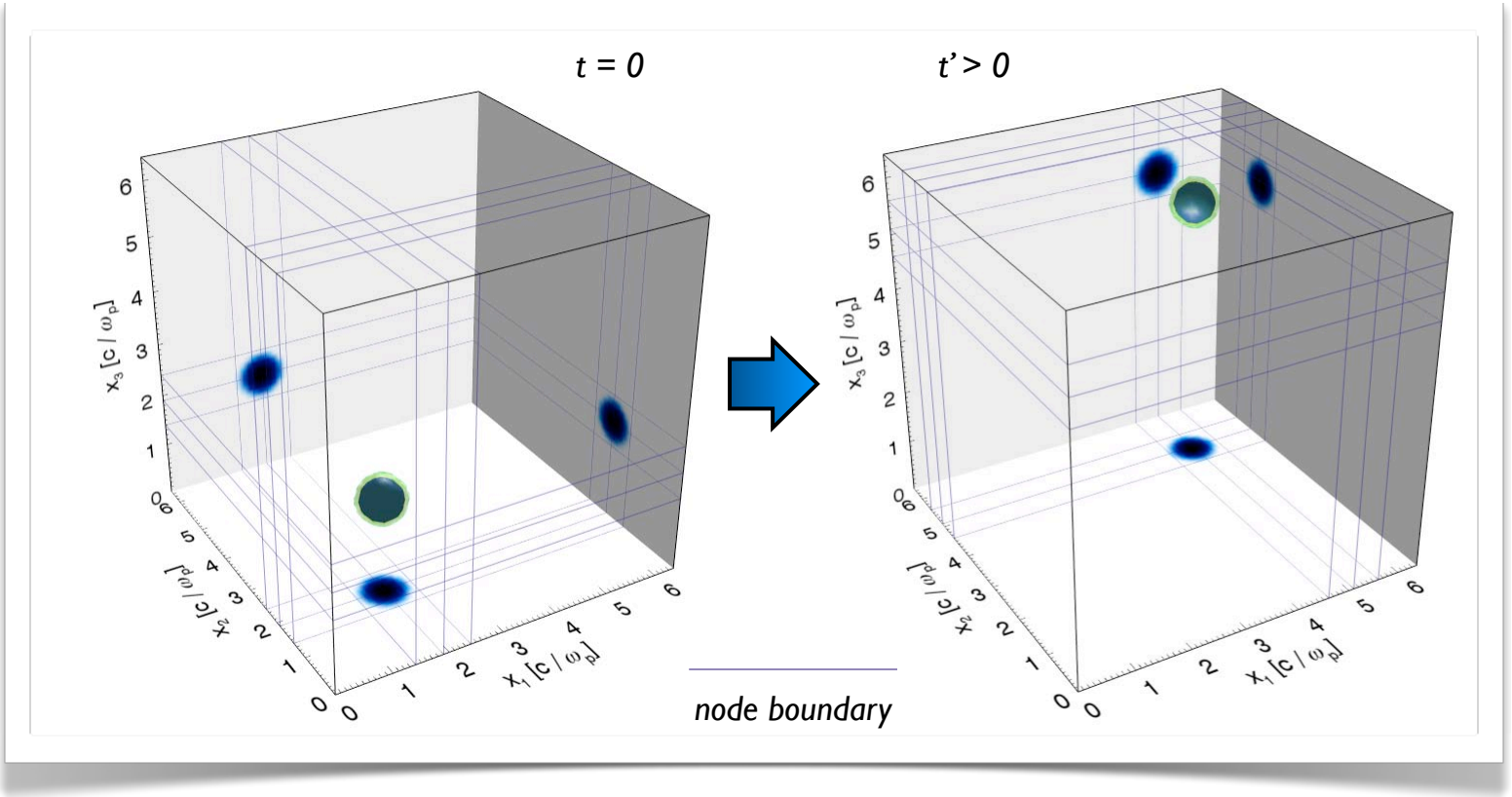
only particles

Load estimate considering particles per vertical slice

particles + cells

Cell calculations (field solver) can be considered by including an effective cell weight

Multi-dimensional dynamic load balance



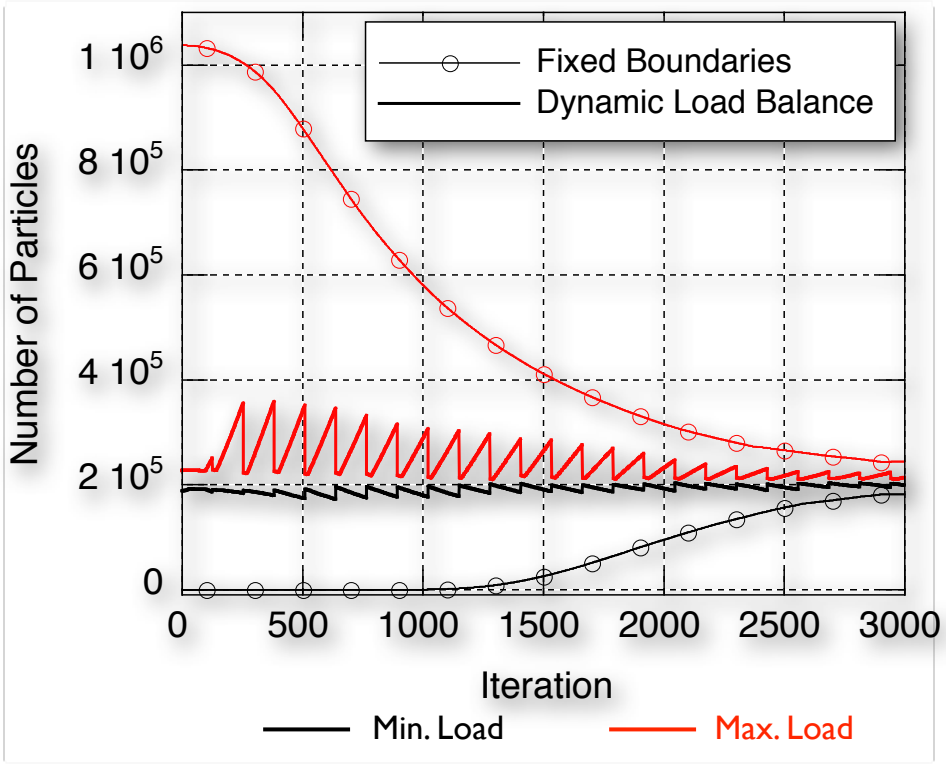
Best Partition

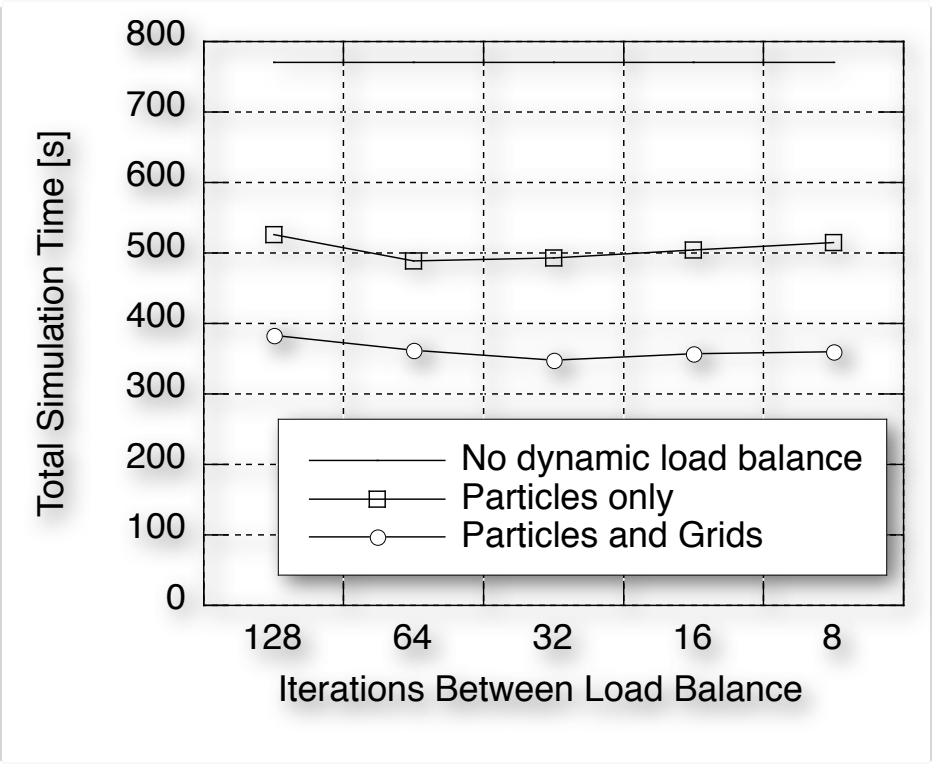
- Difficult task:
  - Single solution exists only for 1D parallel partitions
  - Improving load balance in one direction might result in worse results in another direction

Multidimensional Partition

- Assume separable load function (i.e. load is a product of $fx(x) \times fy(y) \times fz(z)$)
  - Each partition direction becomes independent
- Accounting for transverse directions
  - Use max / sum value across perpendicular partition

Parallel performance





load

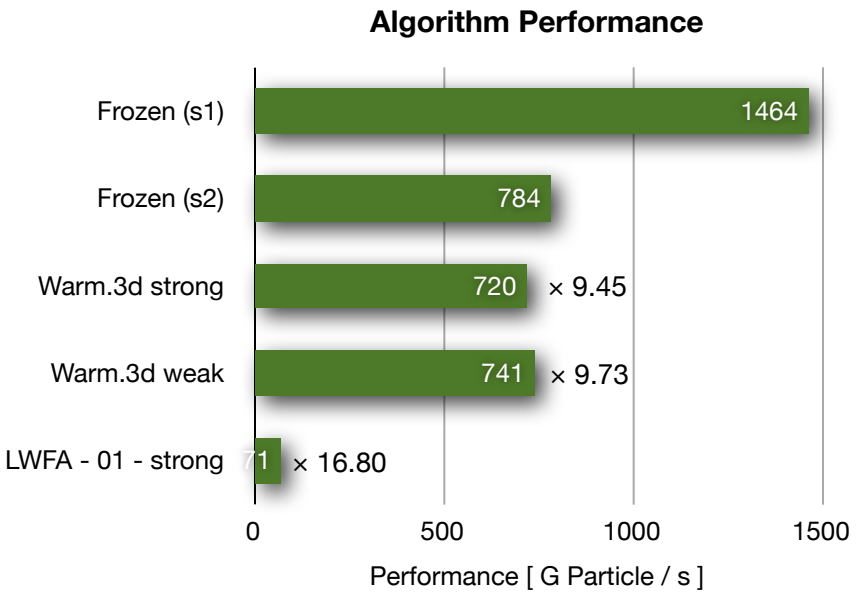
- Algorithm maintains the best possible balance
- Repartitioning the simulation is a time consuming task:
  - Don't do it too often!

performance

- Particles & grids ~ 50% better than only particles
- $n_{\text{load balance}}$ ~ 64 iterations yields good results
- Performance boost ≳ 2
  - other scenarios can be as high as 4

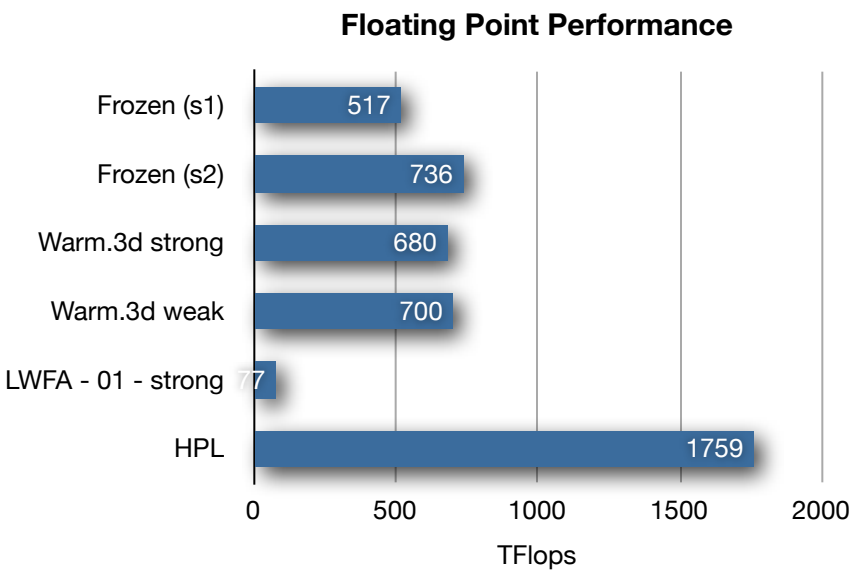
OASCR Full System Tests

PIC Performance



- 4× the number of cores
- Frozen tests
  - Asymptotic limit for code performance
- Very large particle simulations
  - up to 1.86×10^{12} Particles
- Outstanding performance
  - up to 1.46×10^{12} Particles / s
- Excellent Speedup from Q2
  - 9.45× to 16.80×

Machine Efficiency



- Very high floating point / parallel efficiency for the full system
  - 0.74 PFlops peak performance
  - 32% of R_{peak} (42% of R_{max})
- Quadratic Interpolation improves memory / operation ratio
  - Linear interpolation has higher performance but lower FLOP count

What was unthinkable is now possible

UCLA

10^12 particles/10^6 time eteps

- Today: 50-300 ns/part/step
- Today: 10^12 particles can fit on a machine ~.1PByte
- 3D EM PIC simulation for of 10^12 particles/10^6 time steps ~10^11 seconds or ~100 days on 250,000 cores
- GPU/Many core can improve speed by 50-100?
- Load balancing is an issue

HEDP

- Make cell sizes less than to much less than a Debye length: Collision term becomes "correct".
- 100s of interacting speckles in 3D (1000s in 2D).
- Shocks
- and much more.....