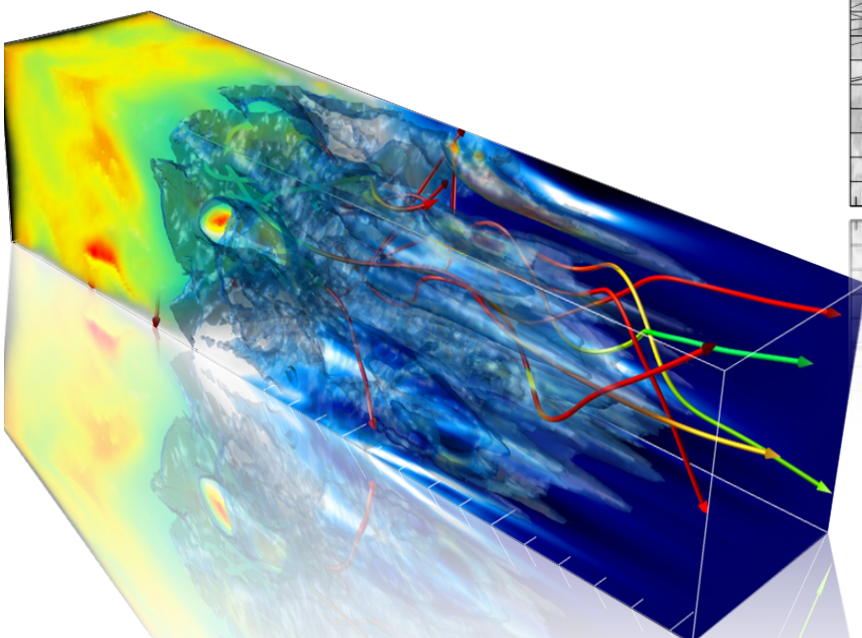
OSIRIS -- progress & scaling

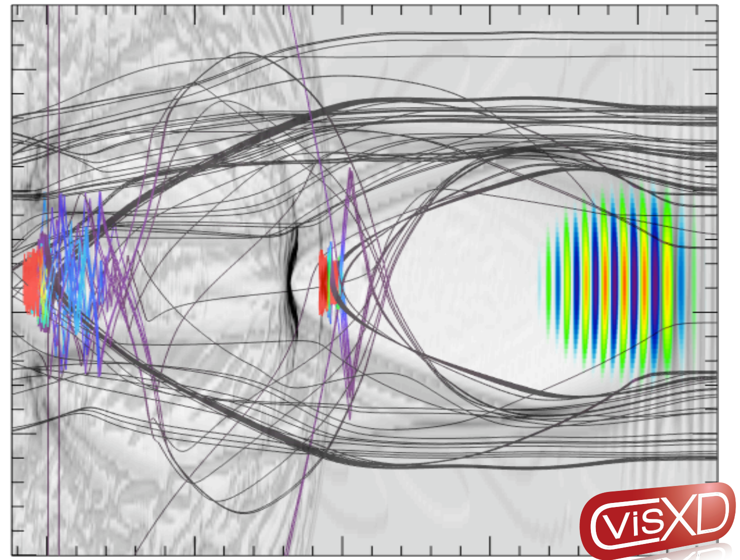
osiris framework

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



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

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



New Features Relevant to LWFA's

- Higher Order (smoother) Particle Shapes
- Bessel Beams
- Binary Collision Module
- Tunnel (ADK) and Impact Ionization
- PML absorbing BC
- Dynamic Load Balancing
- Parallel I/O w/ new diagnostics such as Particle Tracking

Single Node + Parallel Performance

OSIRIS is written in object-oriented Fortran95 and uses MPI (Message Passing Interface) for parallelism. (more about parallel scaling later)

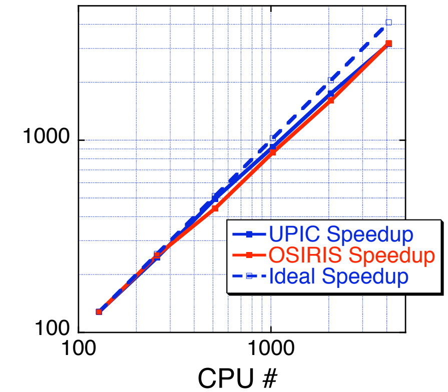
The code is very efficient in parallel. On platforms such as the BlueGene cluster @ ANL and the Atlas cluster @ LLNL, the code is ~ 90% efficient for > 4,000 CPU's on Atlas (>90% on ANL).

The ANL benchmark is done over a very short time period using an unmodified version of OSIRIS. A large amount of the parallel overhead comes from boundary crossing of the drifting species. The system is ~90% efficient for a plasma in thermal equilibrium.

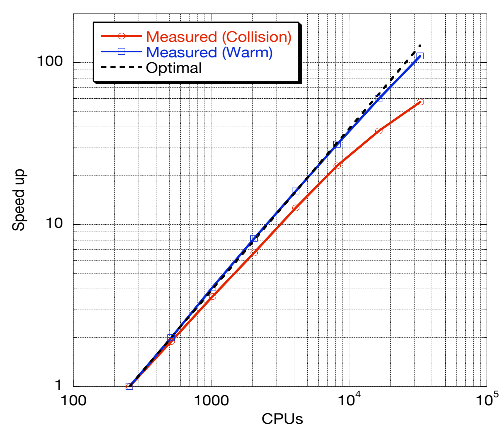
Strong scaling on LLNL Atlas (AMD)

(From Viktor Decyk)

- 512 x 512 x 256 cells
- 16 ppc (2 electron species, colliding) (2^{30}, or ~10^9 particles)
- 425 timesteps



Bluegene/P, strong scaling up to 32768 CPUs



(Warm Case:)

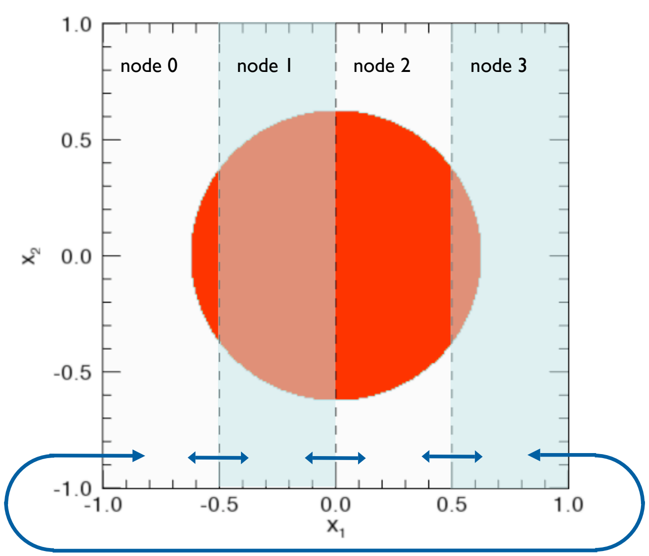
- 512 x 512 x 512 cells
- 8 ppc (2^{30}, or ~10^9 particles)
- 600 timesteps

Efficient Scaling for massively parallel systems

parallelization of FDTD EM-PIC codes

Well suited for parallelization

- Only local data required due to causality (i.e., information cannot propagate faster than c)
- Domain Decomposition should scale well for most cases (communication pattern is the same for 4 CPU's or 40,000 CPU's)



— Communication using MPI

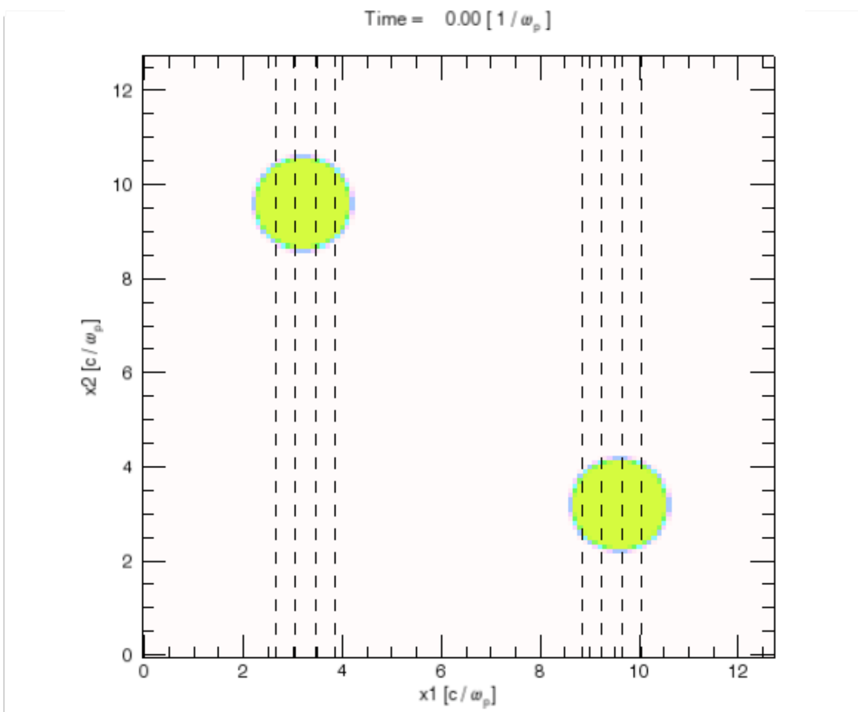
However, the "load" on each node scales linearly with the number of particles in each node. Therefore, partition the simulation box using the spatial information is not enough.

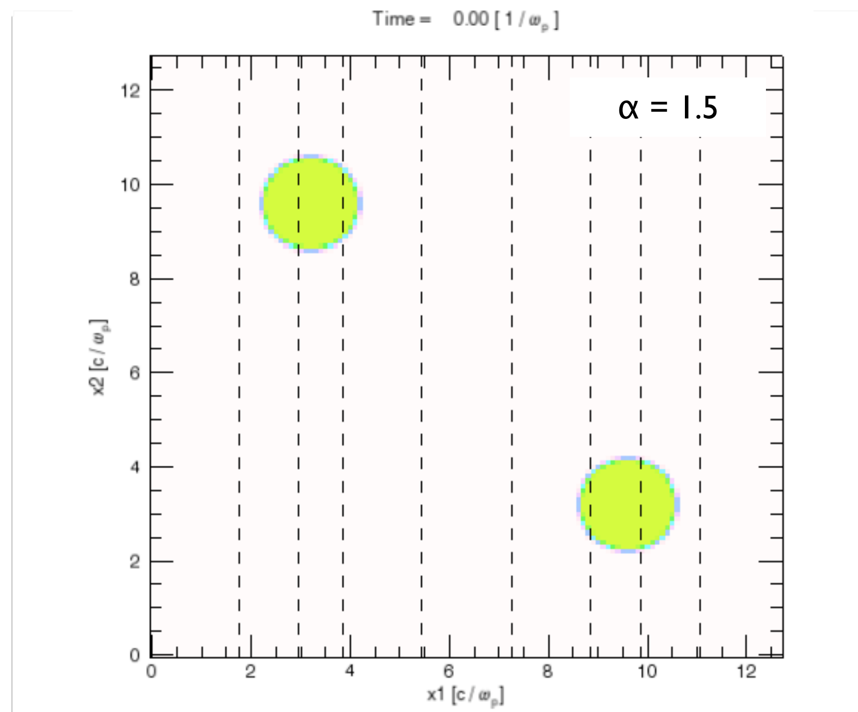
Reducing this imbalance is critical for larger supercomputers because the effects of a single computational hotspot will be greatly amplified as the # of processors increases.

Furthermore, for some problems, static load balance is not sufficient. For these systems, dynamic load balancing is critical!!!

Dynamic load balance in OSIRIS

- - - - - Node Boundaries





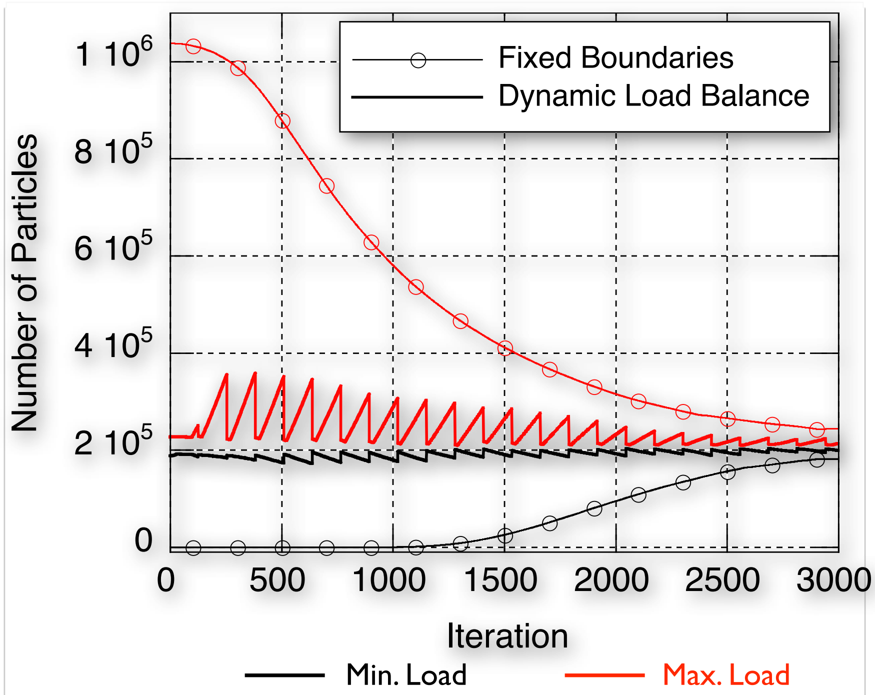
only particles

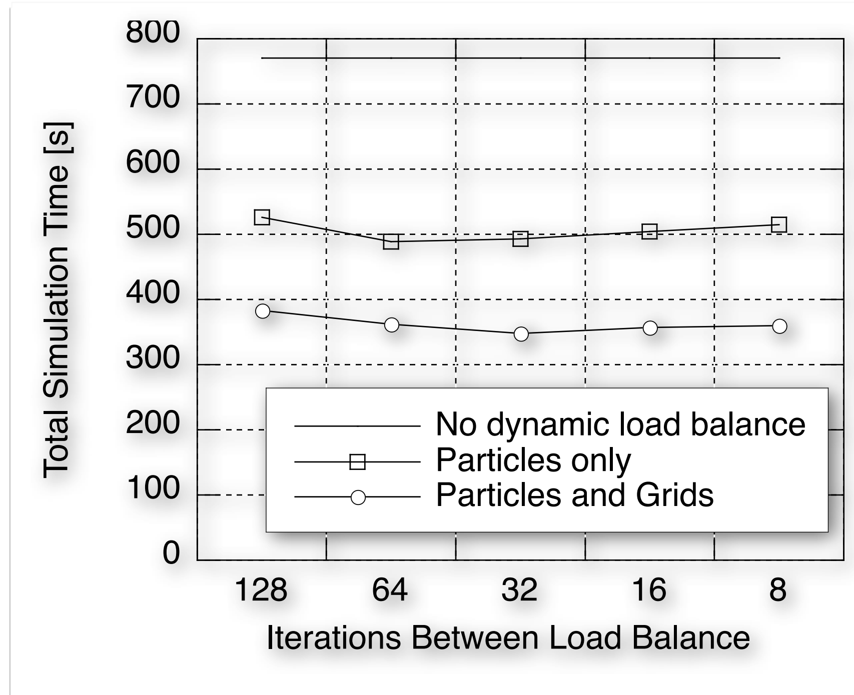
Load estimate considering particles per vertical slice

particles + cells

Cell calculations (field solver) can be considered by including an effective cell weight. Both this weight and the # of steps between load balance checks can be adjusted by the user via the input deck.

Parallel performance -- exploding sphere





performance

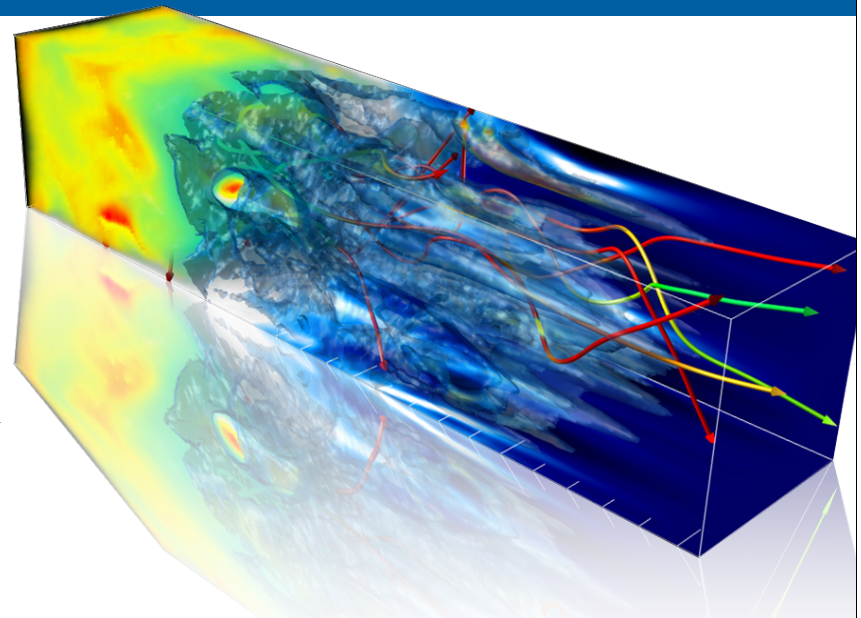
- Particles & grids ~ 50%, and better than particles only load balance
- weak dependence on $n_{load\ balance}$ between 8 and 64. $n_{load\ balance}$ ~ 64 iterations yields good results
- Performance boost ≳ 2
  - other scenarios can be as high as 4

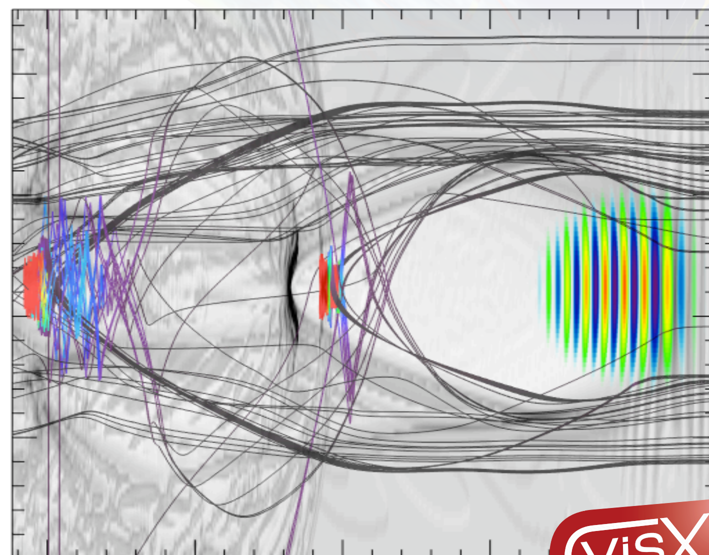
Diagnostics & Visualizations

Over the years there has been a fairly extensive set of diagnostics and visualization tools developed for the OSIRIS code.

Currently, OSIRIS outputs via the HDF5 file format, and the primary visualization/postprocessing tool is IDL (and occasionally OpenDX, Matlab and VisIT (demo later!!)).

One example of our diagnostics relevant to the advanced accelerator community is that of the particle tracking, or the detailed study of particle orbits for a selected group of particles. Two examples of this diagnostic is shown on the right. (LWFA and relativistic shock)





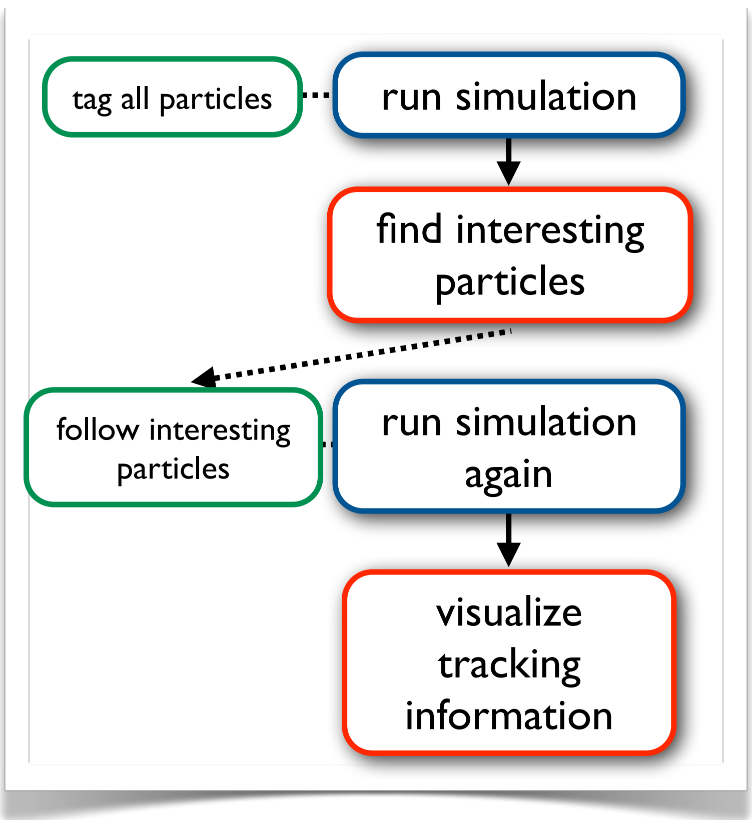
Particle tracking

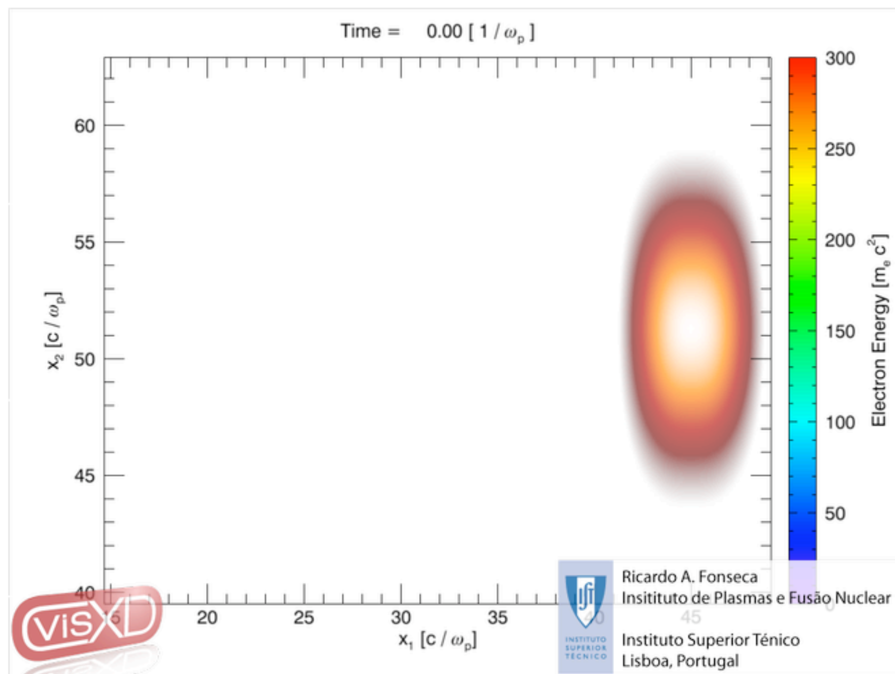
Relevant physics associated with small subset of particles

- Record detailed 7D phase-space of "interesting" particles

Technically challenging

- Subset of ~10^3 particles in ~10^9
- Storing information for every particle not feasible
  - 10^4 iter. × 10^9 part. ⇒ ~ 500 TB



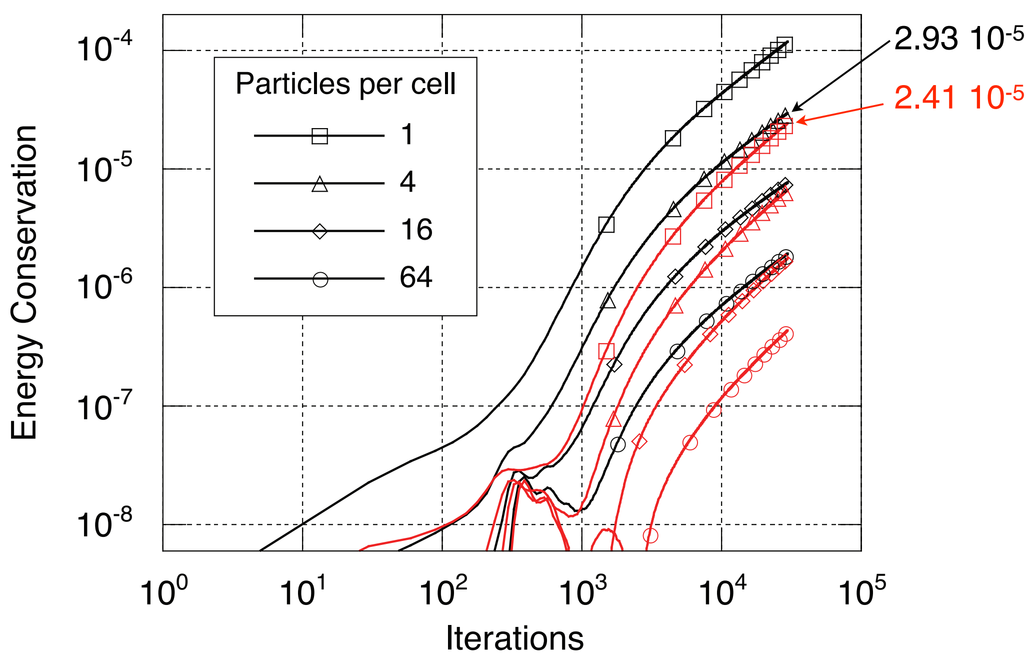


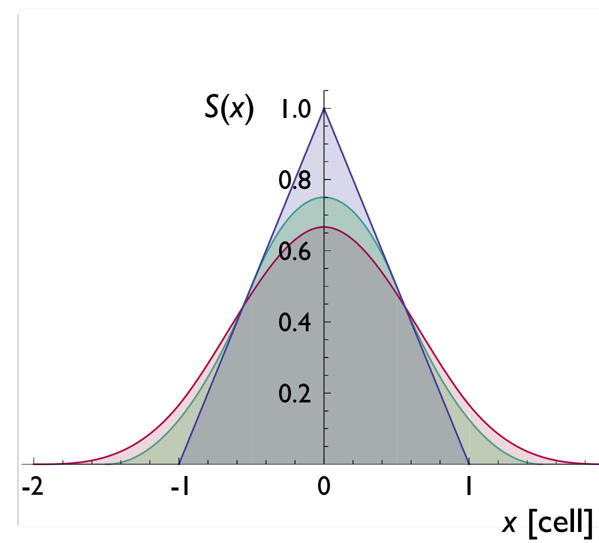
High-order particle weighting

(The benefits of higher order particle shapes will be addressed later by Estelle and I will not say too much here, other than to say that it is implemented in OSIRIS.)

Control discrete particle noise & numerical self-heating

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





Ops	Linear	Quadratic	Cubic
1D	4	15	27
2D	9	37	69
3D	18	83	187

~ 4.6 ×

performance

Measured performance:

- 3D quadratic ~ **1.85 ×** slower than 3D linear

Future Directions -- What's next?

- Parallel I/O with HDF5 -- Underway
- Optimization (e.g. modified particle pushers) for future architectures, e.g., GPU's, Cell processors, etc...