Enabling Breakthrough Kinetic Simulations of the Magnetosphere via Petascale Computing

Homa Karimabadi, UCSD
Kai Germaschewski, UNH

<u>Contributors</u>:
Y. Omelchenko, H.X. Vu, UCSD
W. Daughton, A. Beresnyak, LANL
M. Tatineni and A. Majumdar, SDSC
W. Mori, F. Tsung, UCLA
P. Alves, Instituto de Plasmas e Fus˜ao Nuclear, Portugal
B. Loring, Lawrence Berkeley National Lab.

Blue Waters Symposium – University of Illinois at Urbana-Champaign, May 21-22, 2013

Outline

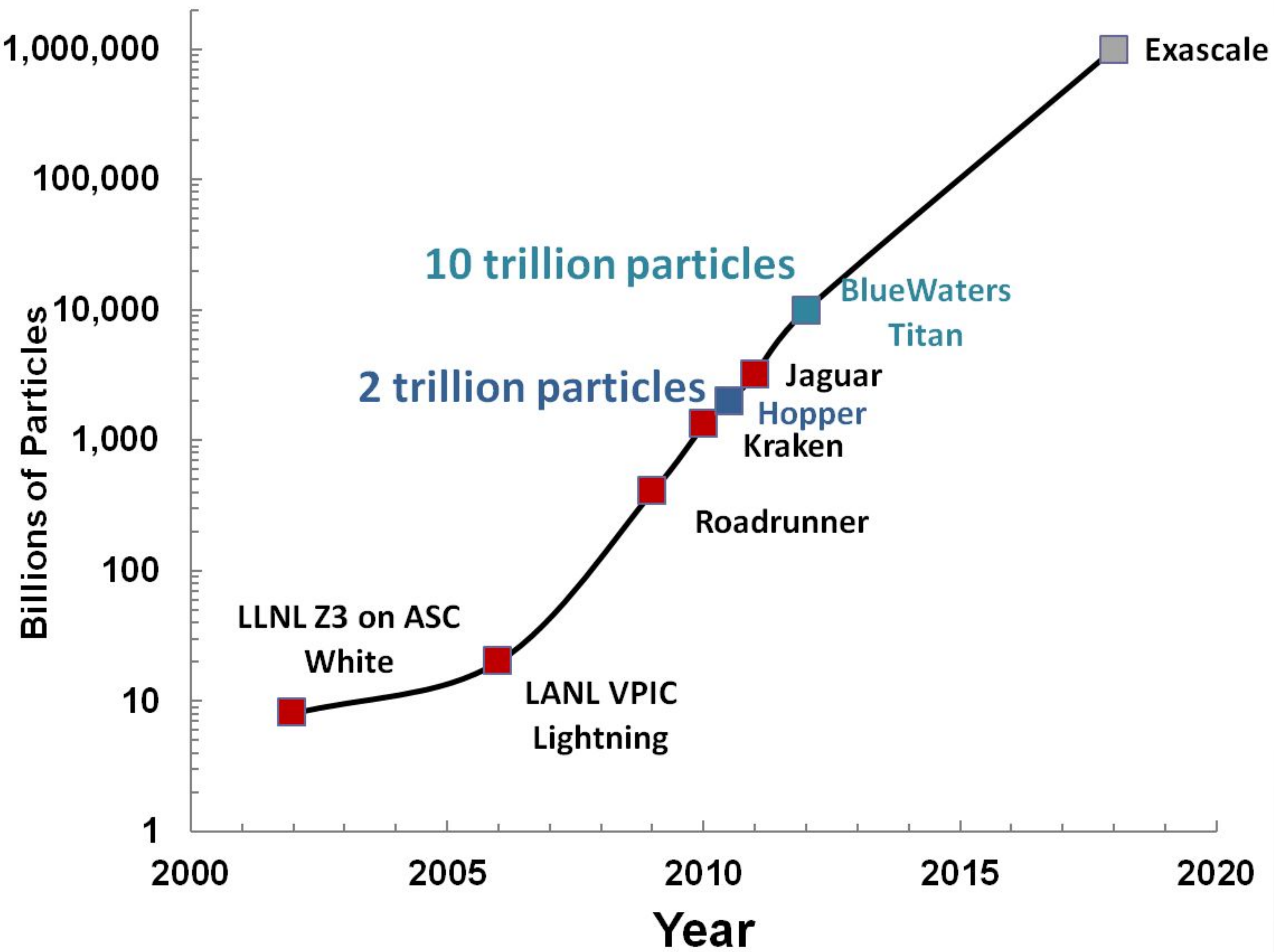
- Our Science
- Challenges and remedies (**explored as part of PRAC project**) in kinetic simulations:
  - Particle "noise"
    Remedy: Use of higher order-particles
  - Would like to skip the speed of light in many cases:
    Remedy: Semi-implicit scheme (need a scalable solver)
  - Search for performance gains
    Remedy: GPUs/Intel MIC?
- Science Results/Impact

Research Areas

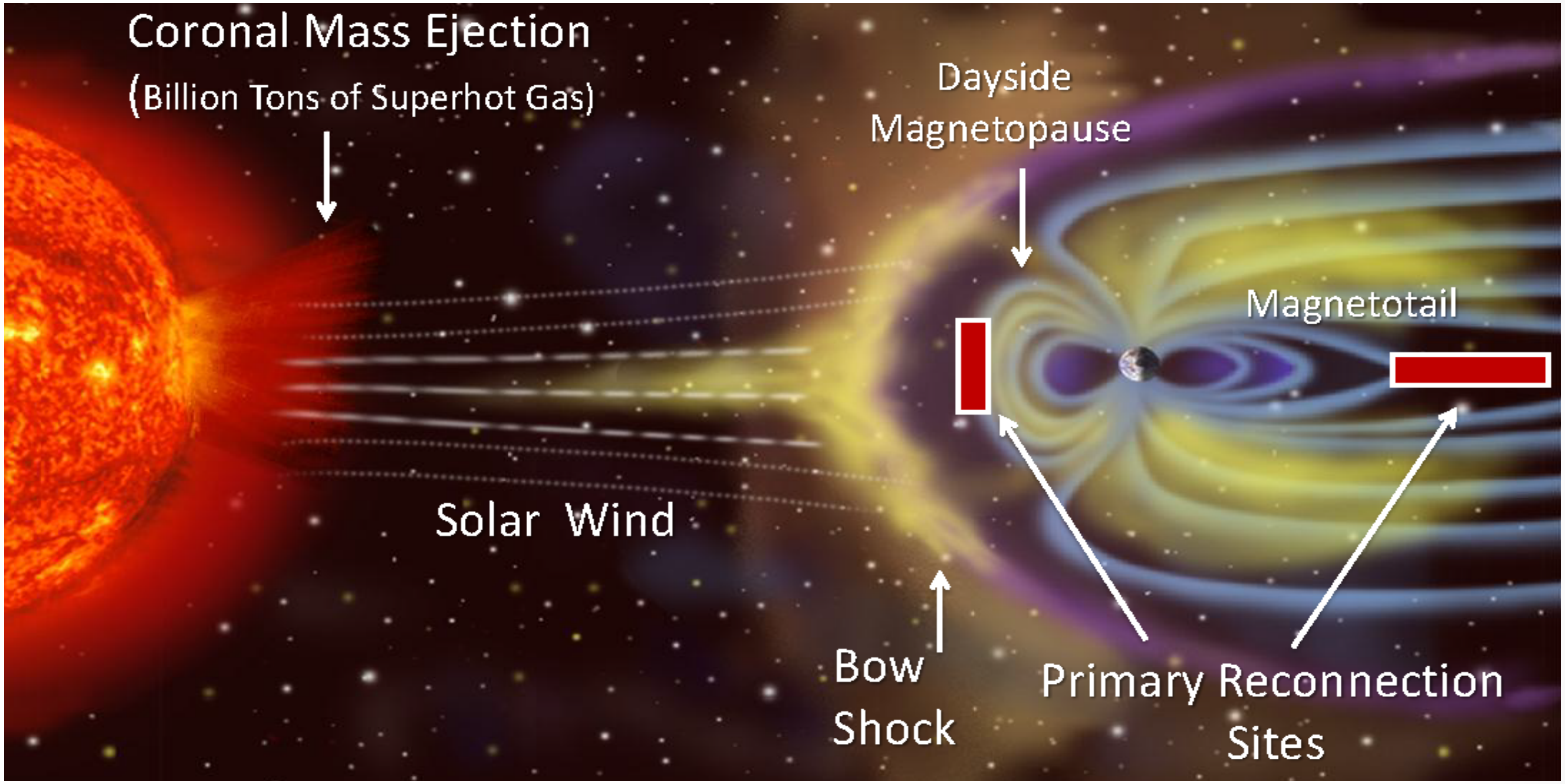
- Plasma Turbulence
- Magnetic Reconnection
- Dynamo
- Exploratory Fusion Concepts
- Space Weather

Computer Performance	
Name	**FLOPS**
yottaFLOPS	10^{24}
zettaFLOPS	10^{21}
exaFLOPS	10^{18}
petaFLOPS	10^{15}
teraFLOPS	10^{12}

Progress in Particle Simulations
(measured in terms of number of particles)



Space Weather



90 million miles or ~ 100 Suns

Goal: Develop Accurate Forecasts of Space Weather

Space weather affects our technological systems:

- Has caused over $4 billion in satellite losses

- A solar storm of the magnitude of the 1859 Solar Superstorm would cause over $2 trillion in damage today.

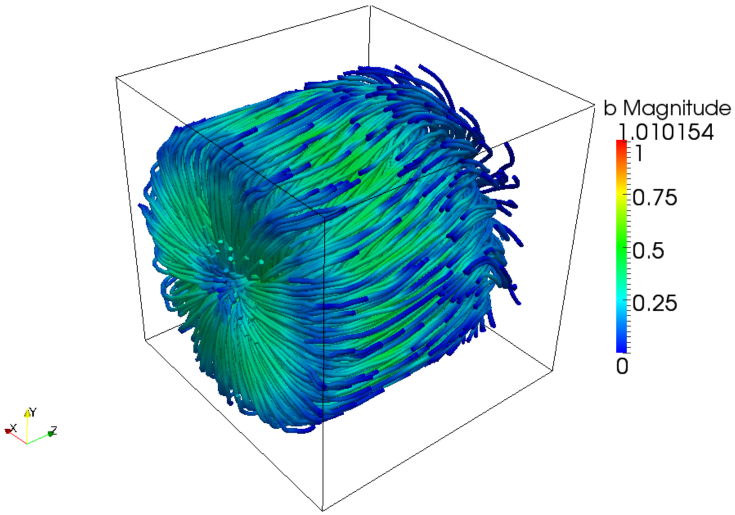
- Causes damage to sensitive electronics on orbiting spacecraft

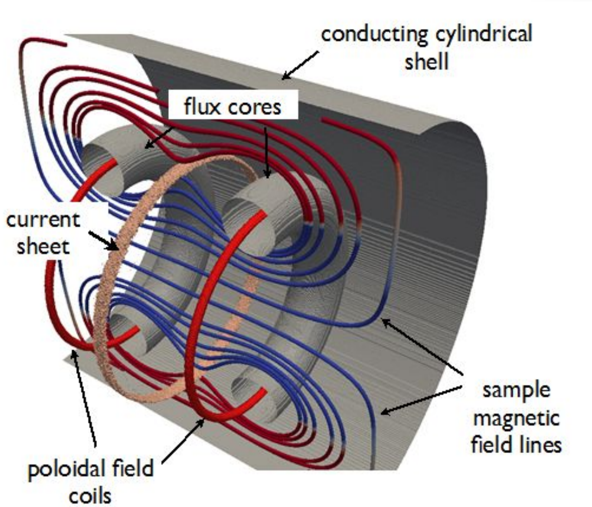
- Causes colorful auroras, often seen in the higher latitudes

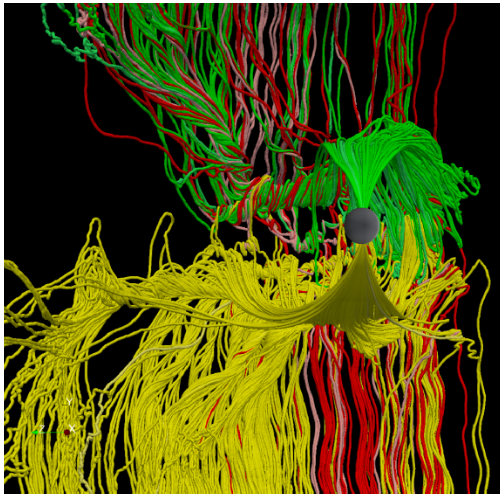
- Creates blackouts on Earth due to surges in power grids

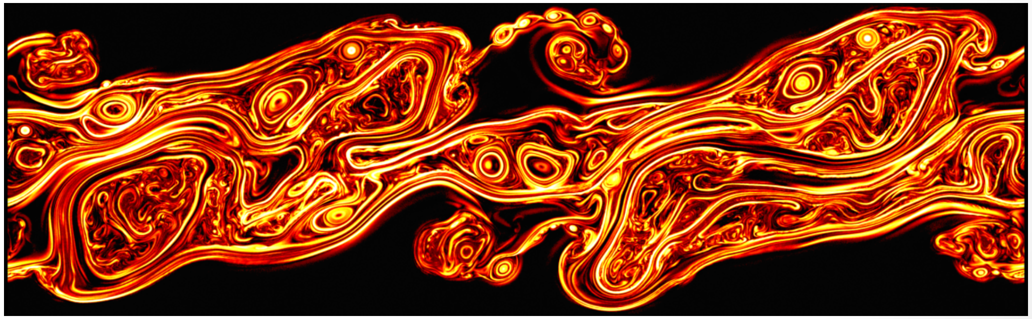
Funded by a new 5 year, multi-institutional NSF/NASA Collaborative Grant – PI A. Bhattacharjee (Princeton)

Example Simulations



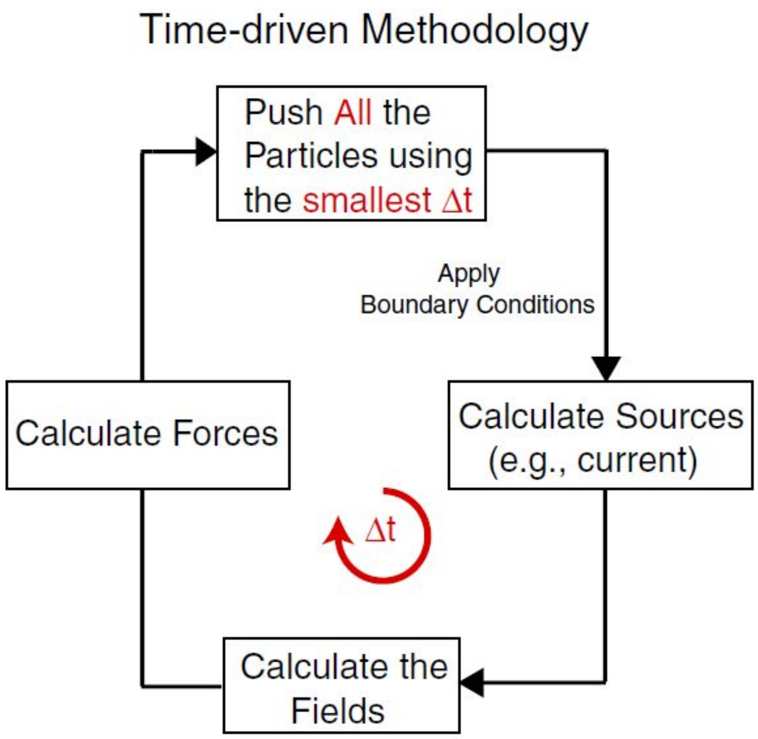


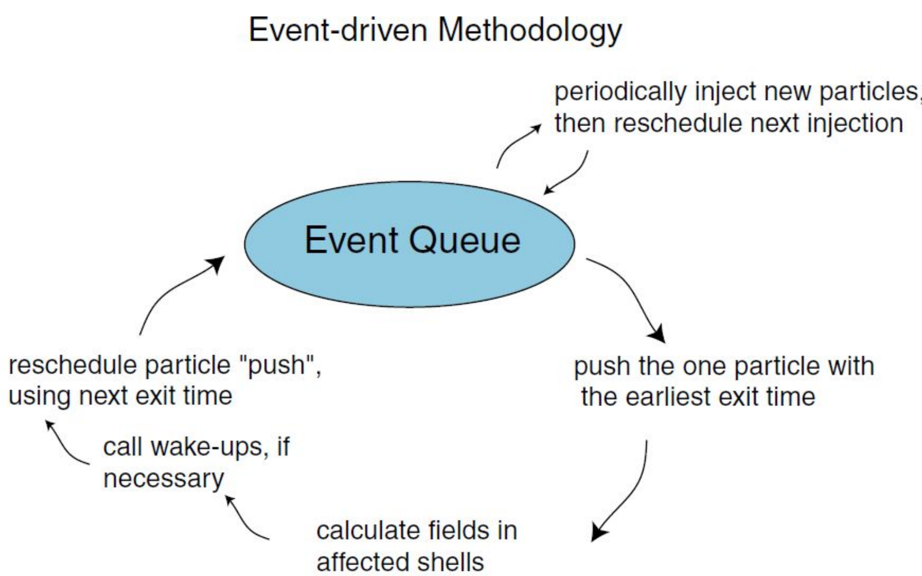




Particle-In-Cell Plasma Codes

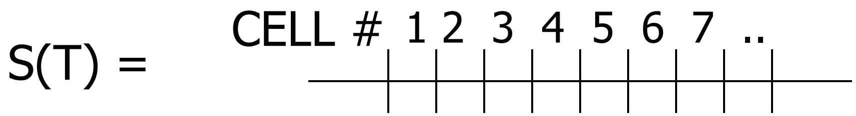
- Fully kinetic (electrons and ions are treated as particles)
- Hybrid (electron fluid, particle ions)





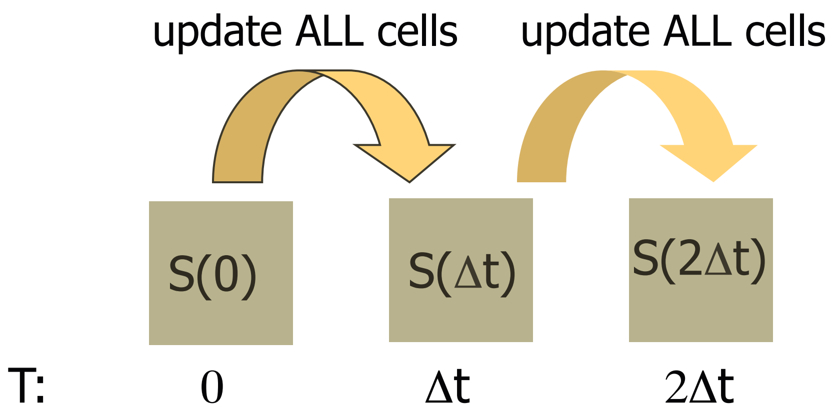
Karimabadi et al., JCP, 2005

Discrete Event vs Time Stepping



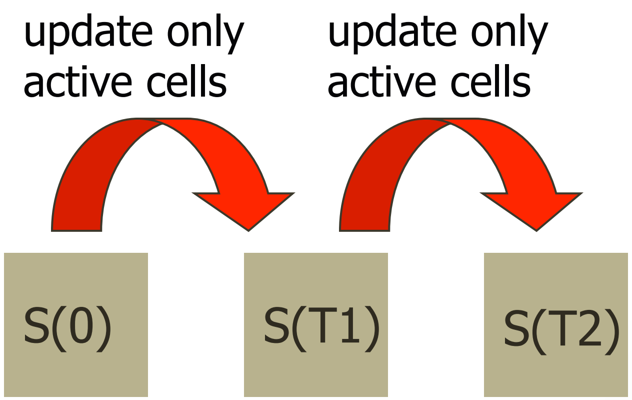
Time-driven simulation

updates the entire system

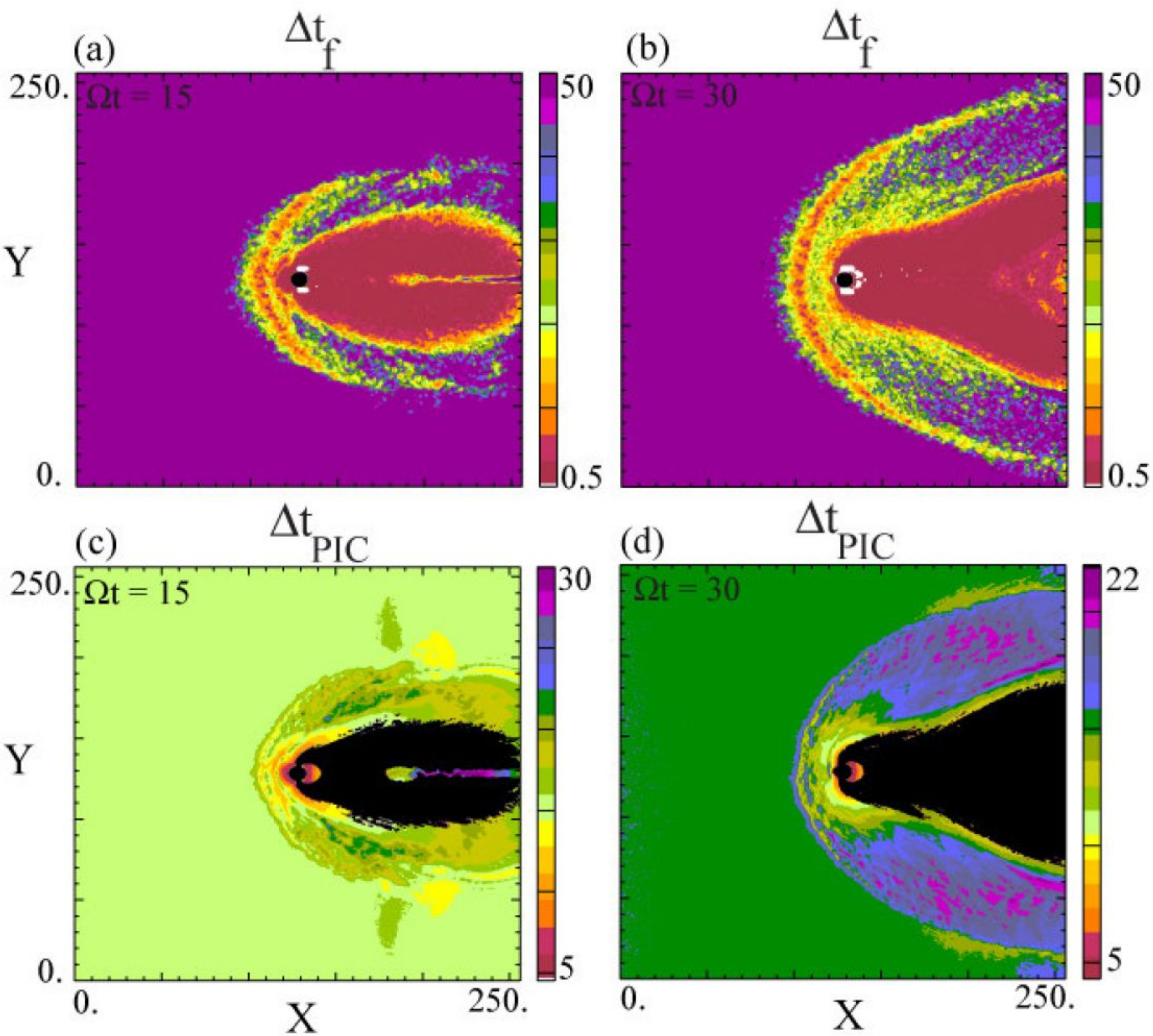


Event-driven simulation

updates active cells only



Event-Driven Simulations

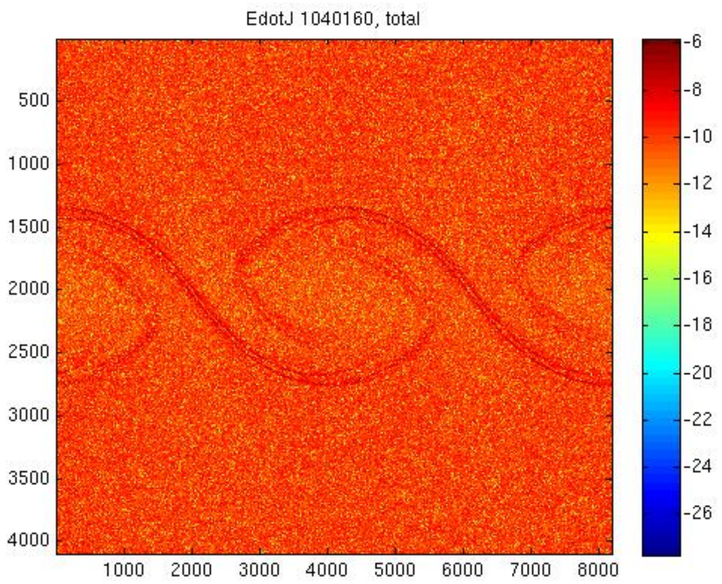


Omelchenko and Karimabadi, JCP, 2011

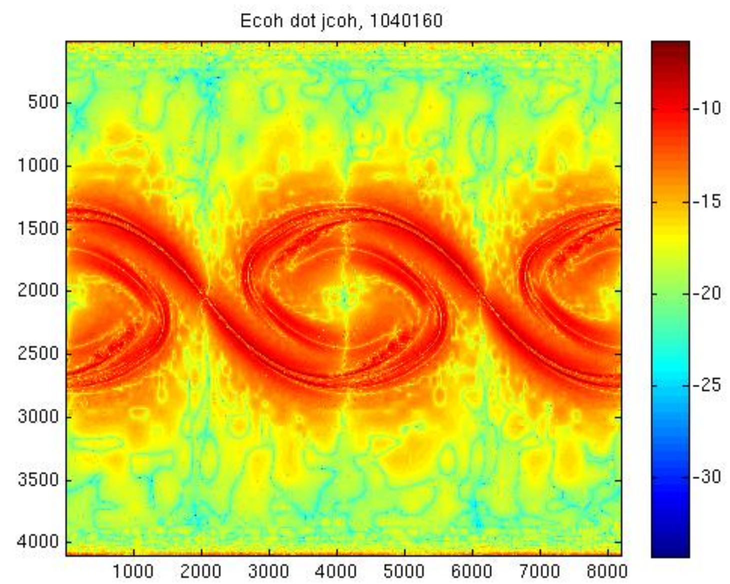
Particle "Noise"

- Discrete particle effects lead to numerical "noise" which can:
  - lead to numerical heating
  - result in poor resolution of quantities of interest such as the spectrum of turbulence, E.J, agyrotropy, etc.

Raw Data



Wavelet Filtered Data

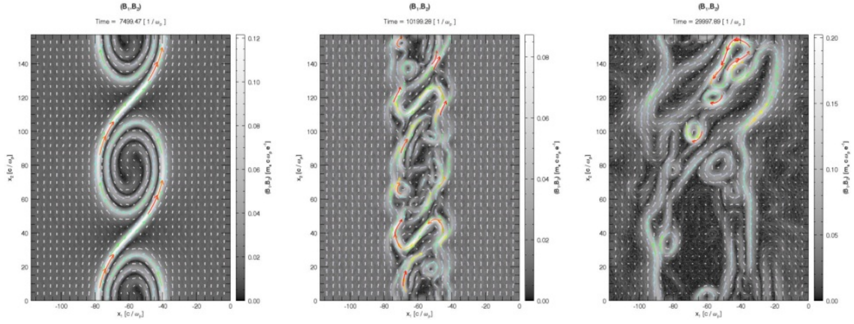


Remedies for Particle "Noise"

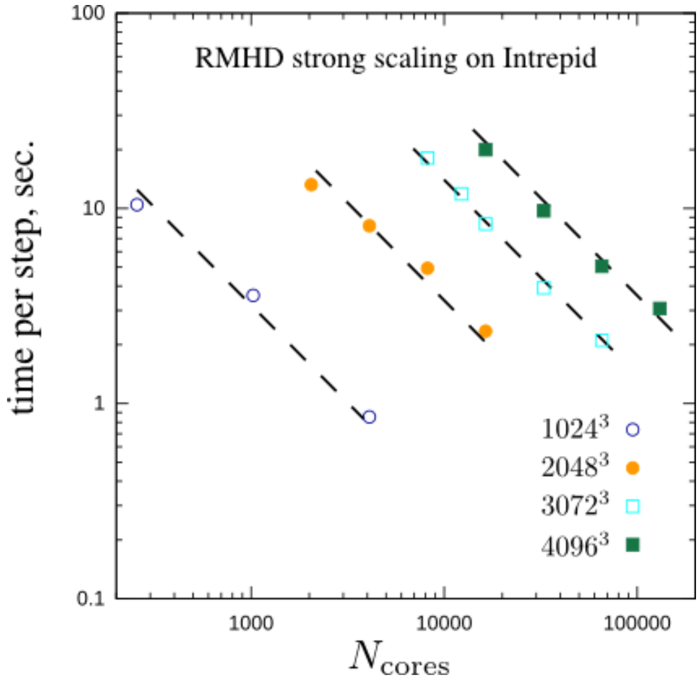
- Increase the number of particles/cell
  - noise goes down as sqrt(#particles/cell)
- Use higher order particles
  - it has extra cost + there is a limit on how "fat" the particle can be made before affecting the physics

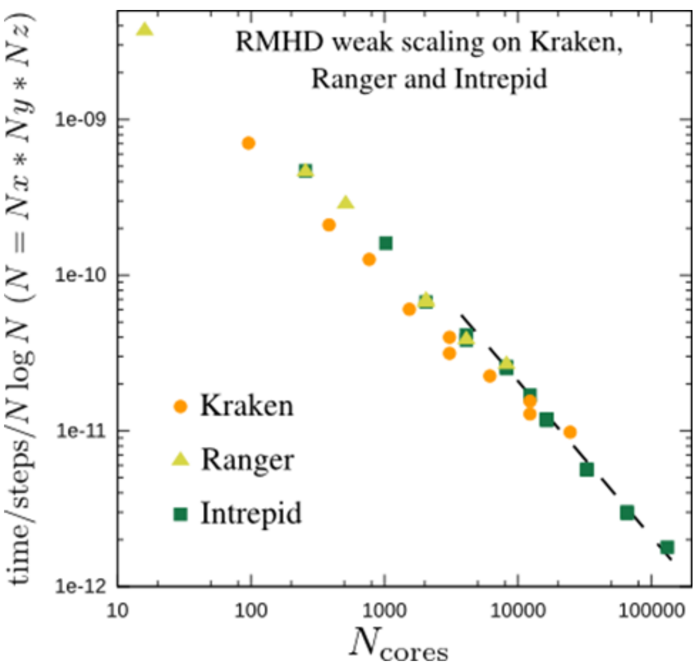
Results, using OSIRIS, Very Promising

ID	Case	Times, s	Energy drift, %
A	Linear	396	8.2×10^{-2}
B	Quadratic	561	1.9×10^{-3}
C	Cubic	852	6.8×10^{-4}
D	Linear, 256 part/cell	788	4.2×10^{-2}
E	Linear, no smoothing	394	1.05



Scalable Poisson Solver (P3DFFT library)





P3DFFT does two global transposes of a distributed array. Each transpose is sending almost the whole array over the network. If N_{cores} is large, the effective bandwidth during such global operations, when each node sends data to each other node, can drop down to several Mb/s on Kraken.

Advantages of Semi-Implicit Fully Kinetic Algorithm

Explicit PIC is constrained by CFL condition for light waves $c\Delta t < \Delta x$

There are many problems where these waves are irrelevant, but a fully kinetic description is still needed

To get around this issue, many explicit calculations are done with artificial parameters, which may influence the physics

Another approach is to employ semi-implicit differencing of Maxwell's equations for the light wave, but the rest of the algorithm remains explicit

One of the most well known formulations is from Forslund, 1985

$$\nabla^2 \phi = -4\pi\rho$$

$$\nabla^2 \mathbf{A} - \frac{1}{c^2}\frac{\partial^2 \mathbf{A}}{\partial t^2} = -\frac{4\pi}{c}\mathbf{J} + \frac{1}{c}\frac{\partial \nabla \phi}{\partial t}$$

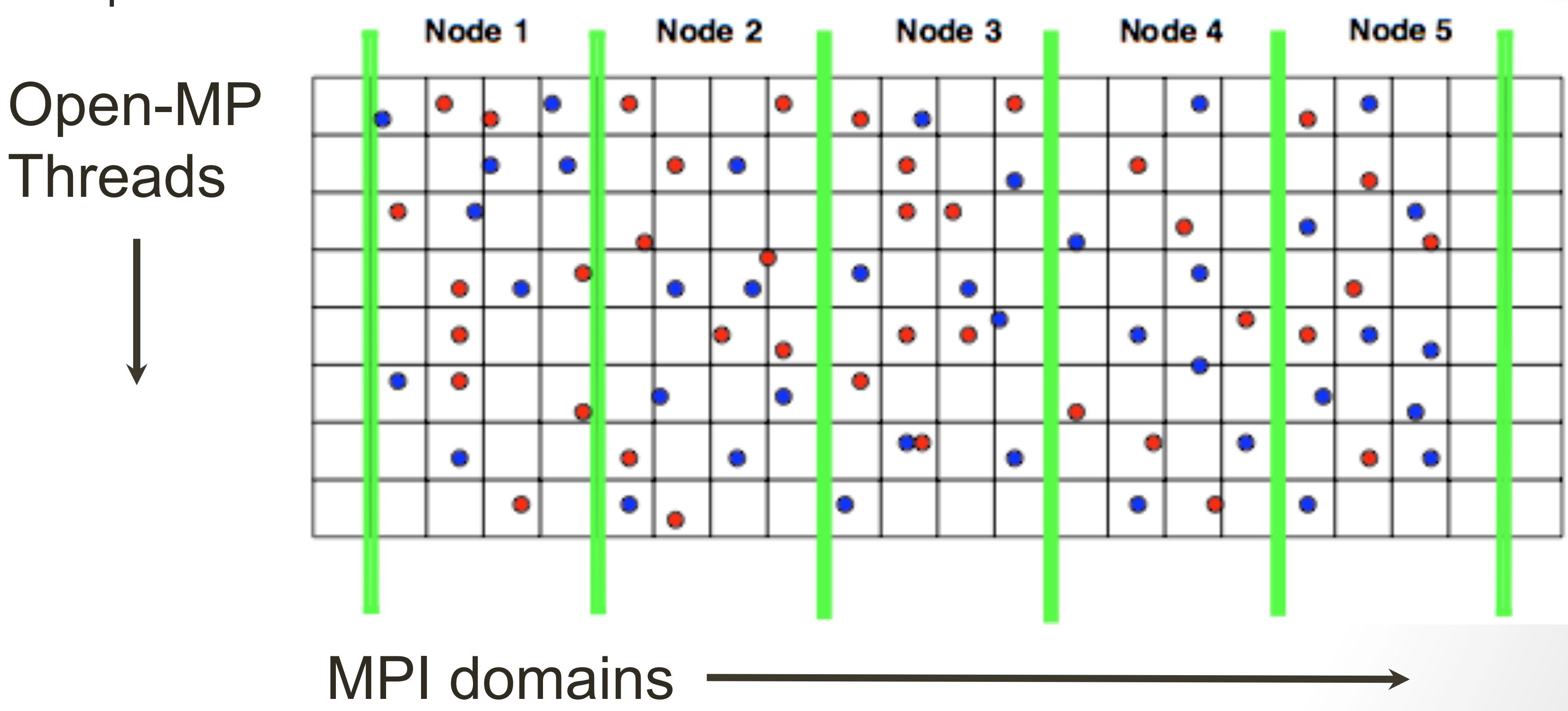
difference implicitly here

Solution requires
Requires 5 matrix
inversions 2D

FFTs allow fast direct solutions but are difficult to scale to large numbers of MPI domains

Parallel FFT relies upon transpose operations - lots of communication!

To minimize - one can limit domain decomposition to one direction (2D) or two directions (3D), and then employ threads to further parallelize



Tested this simple approach starting with old pure MPI code using semi-implicit algorithm

Open-MP threads were used in

1. Particle pusher - 90% of time
2. Particle sorting
3. FFTs in y-direction - used FFTW

Previous MPI version of code scaled to ~500 cores

New hybrid version scales linearly to ~10,000 cores!

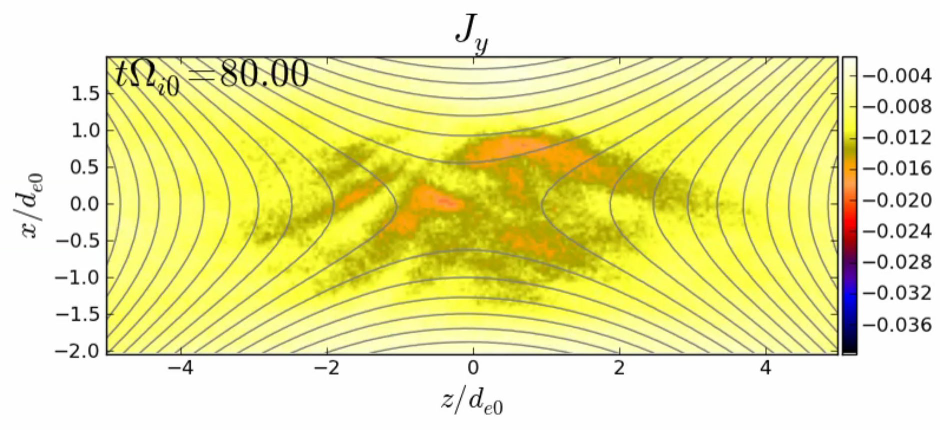
Example run on BlueWaters

512 MPI domains with 16 threads per domain = 8192 cores

2 million particles per sec per core

Implicit time step ~28x larger than possible with explicit CFL, allows us to explore previously inaccessible regimes

Uncovered new physics that was previously missed!



ω_{pe}/ω_{ce} = 16

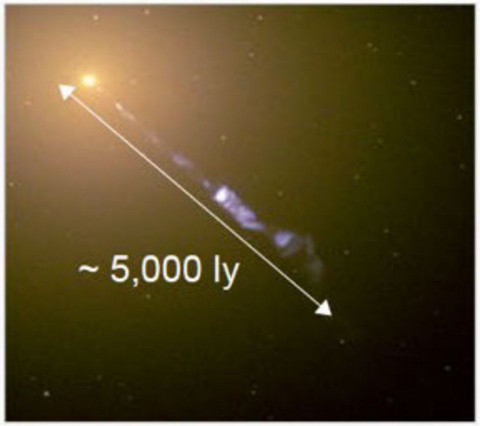
Jara-Almonte et al, 2013

Examples of New Physics Uncovered Due to Increase in Computational Power

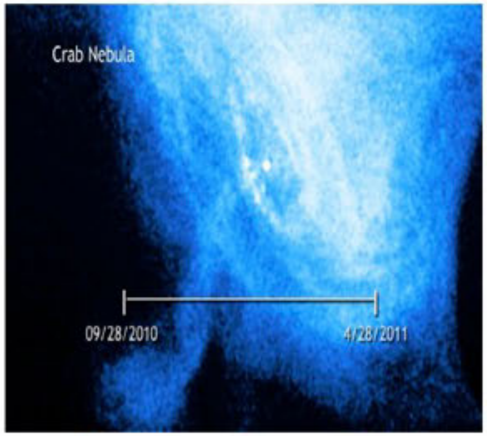
Force-free Geometry

$$\nabla \times \mathbf{B} = \Lambda \mathbf{B}$$

Astrophysical Jet



Superflares @ Crab Nebula



Solar Corona

Relativistic jets from Galaxy M87
(visible spectrum, Hubble Space Telescope)

(from Fermi Space Telescope)

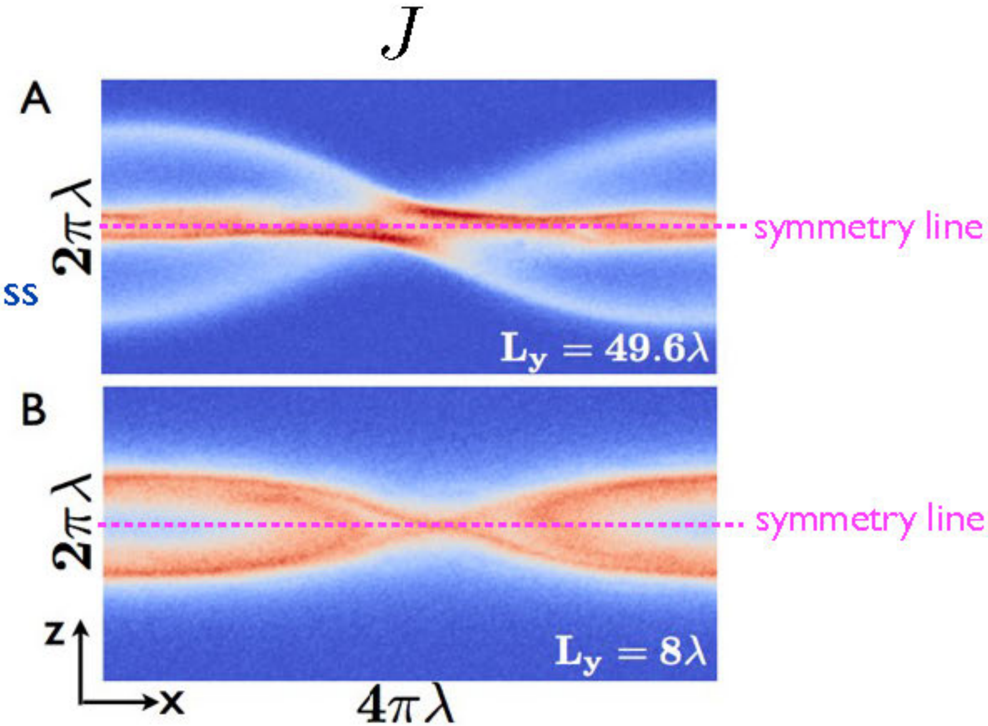
(from TRACE)

- Low beta plasmas: $0 = \mathbf{J} \times \mathbf{B} + \cancelto{\text{small}}{\nabla P}$
- Usually has guide fields: $b_g \equiv B_g / B_0$

- Laboratory plasmas
  Taylor relaxation leads to force-free geometry as well

Tearing modes with $b_g = 2.5$ $(\beta = 0.05, m_i/m_e = 25)$

- Periodic in x, y
  Conductive in z
  size: Lx x Ly x Lz
- Initial sheet half-thickness
  $\lambda = 0.5 d_i$
- Buneman stable
  $U_e < \sqrt{2} V_{the}$



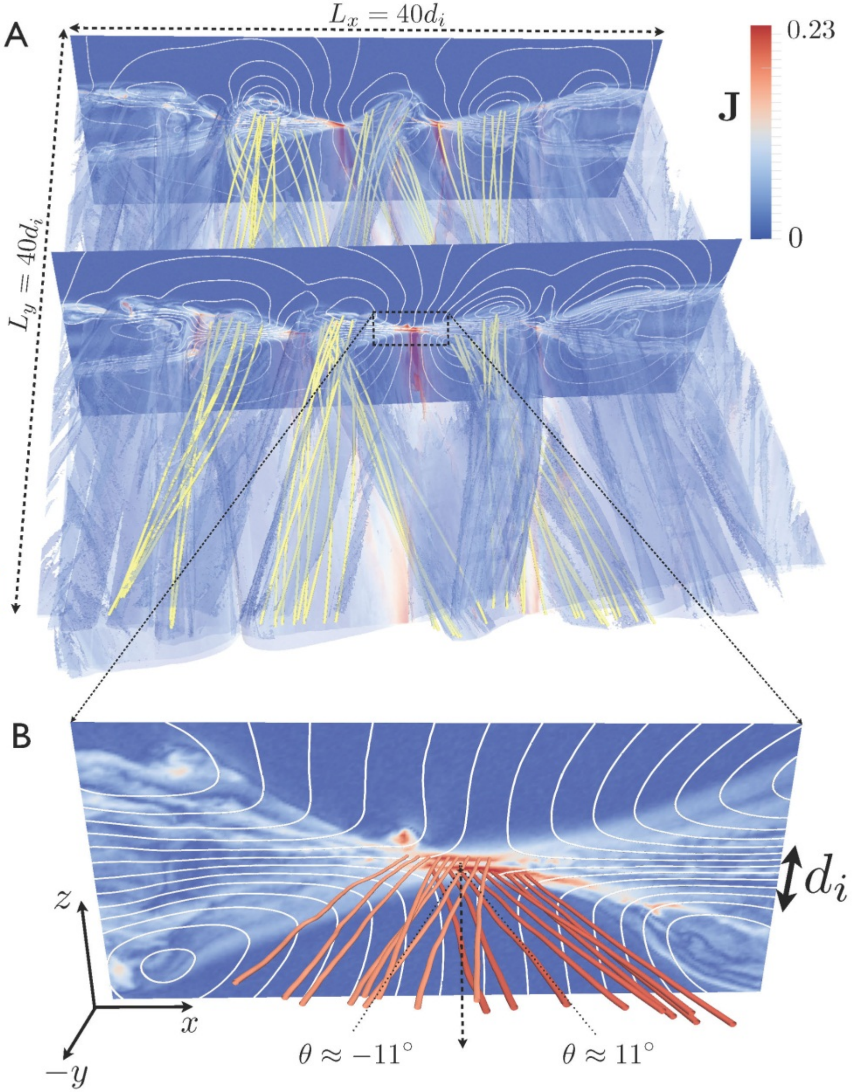
- The resonant surfaces of oblique tearing modes locate at both sides of the symmetry line.
- Oblique tearing modes dominate the 3D current sheet.
- Oblique tearing modes are suppressed when Ly is short enough.

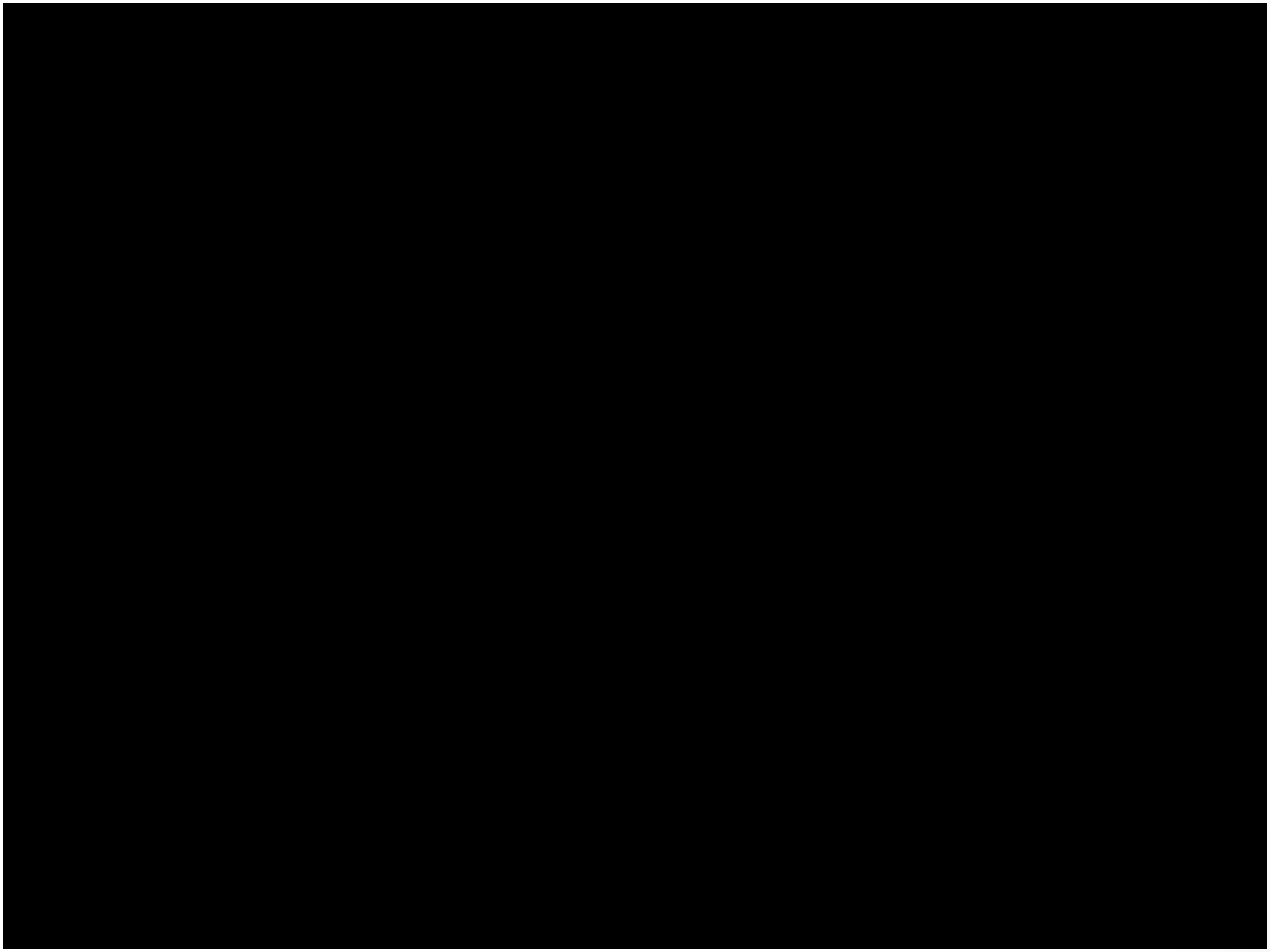
Global structure of $b_g = 2.5$ case $(\beta = 0.05, m_i/m_e = 100)$

- Oblique tearing modes sit at sides of symmetry line
  - give rise to oblique flux ropes.
  - bifurcate the electron diffusion region!

- Form double electron diffusion layers embedded in a broader ion diffusion layer !!!

- The electron diffusion regions are inherently three-dimensional.



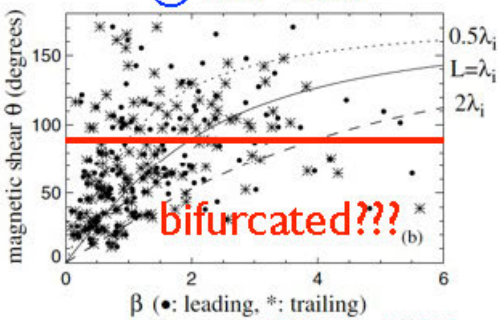


Summary

- Double/multiple electron diffusion layers embedded in a broader ion diffusion layer are observed when the guide field is large enough.

- This bifurcation-phenomenon occurs when the most unstable tearing modes become oblique and bifurcate the sheet, which requires $b_g > 1$ (i.e., shear angle 90)

@ solar wind



Phan et al. 2010

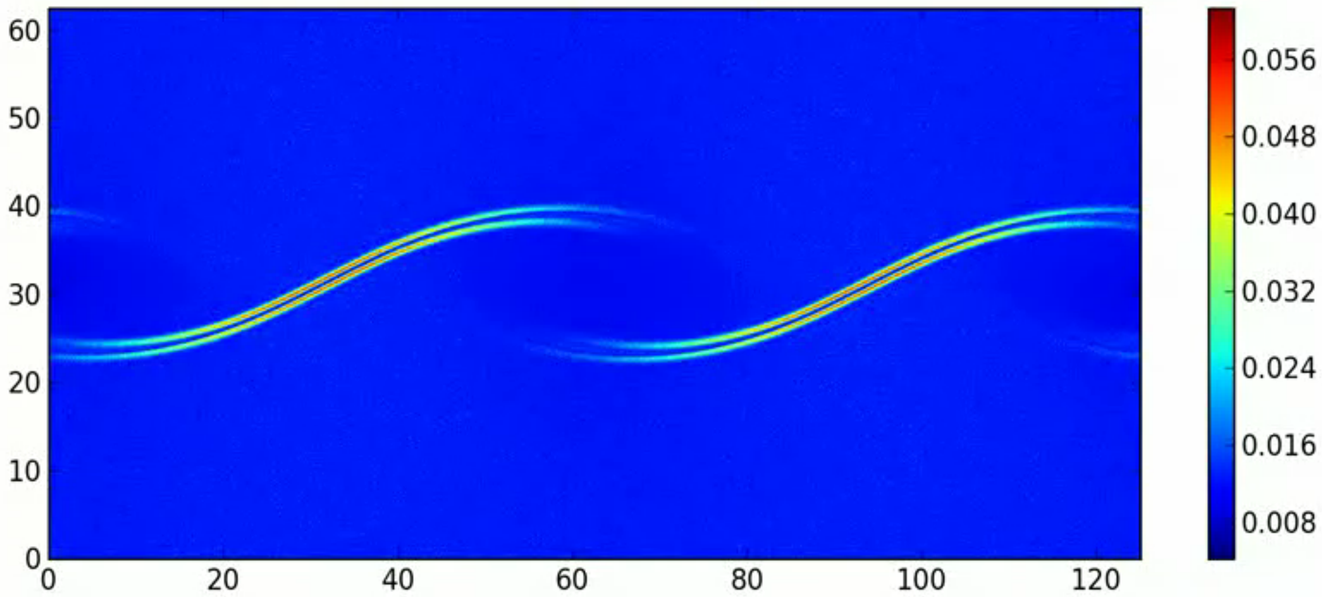
- $\partial_{\|}(P_{e\|} - P_{e\perp})$ is the dominant term that supports the non-ideal parallel electric field in three-dimensional current sheets (bifurcated or not).

- The reconnection outflow in pair plasmas with large enough guide field is opened by oblique tearing modes.

Yi-Hsin Liu et al. *Bifurcation structure of the Electron Diffusion Region in Three-Dimensional Magnetic Reconnection* (submitted)

Yi-Hsin Liu et al. *Kinetic Theory and simulation of Magnetic Reconnection in Force-Free Current Layers* (in prep.)

Kinetic Kelvin-Helmholtz Instability



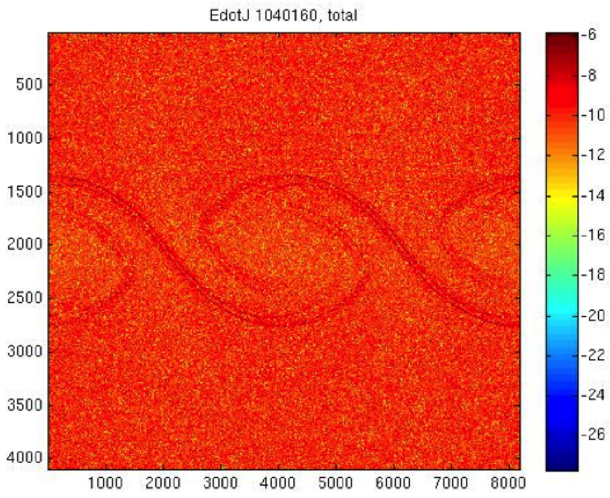
15360×7680 cells, 100 particles per cell
performed on 900 GPUs (M2090, TitanDev) in $\approx$ 24 h wallclock

Particle Noise

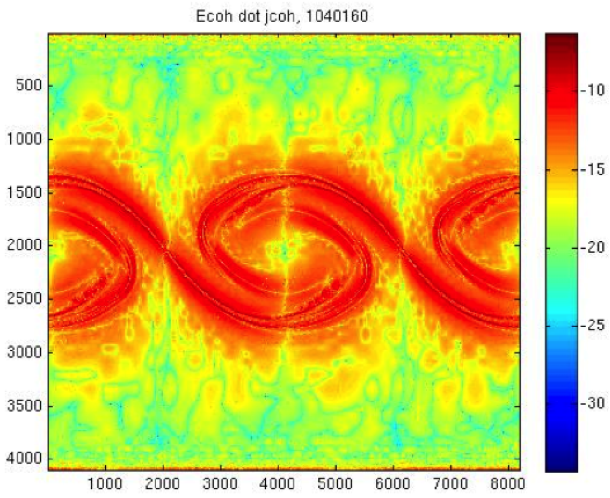
Discrete particle effects lead to numerical noise which can

- lead to numerical heating
- result in poor resolution of quantities of interest such as the spectrum of turbulence, $\mathbf{E} \cdot \mathbf{J}$, agyrotropy, etc.

Raw data



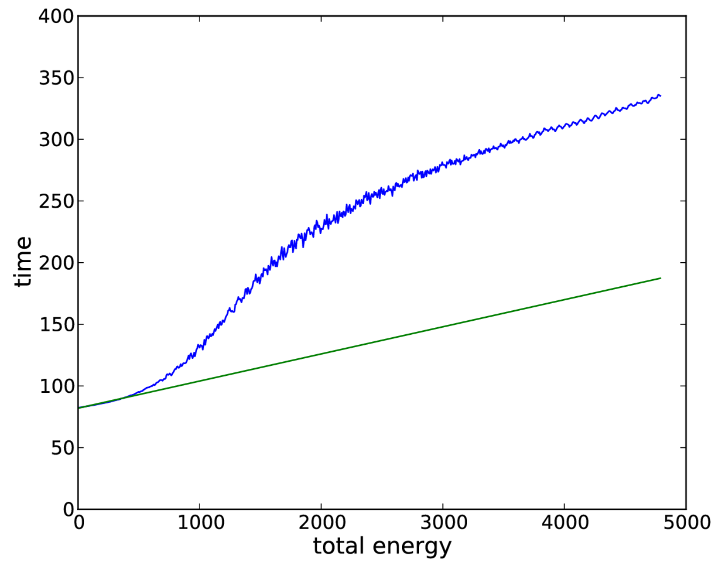
Wavelet filtered data



Particle-in-cell: Numerical Heating

While physically total energy should be conserved, particle-in-cell simulations suffer from non-physical numerical heating.

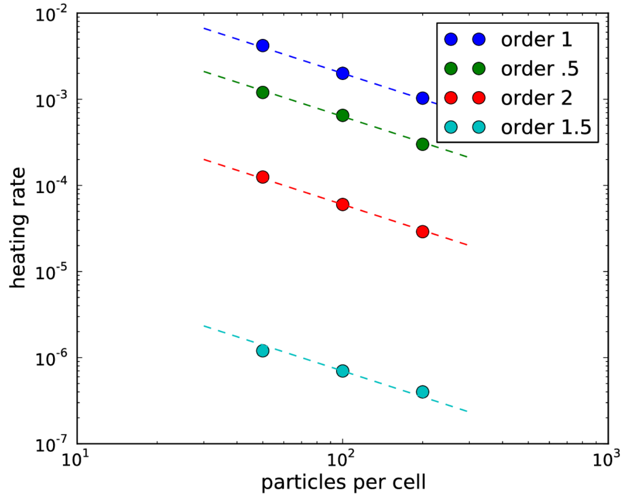
- **Finite Grid Instability.** Aliasing of unresolved grid modes gives rise to a numerical instability if the Debye length is not resolved.
- **Stochastic heating.** Particle noise leads to errors in the electromagnetic fields that heat the plasma linearly ($\propto 1/N$).



Numerical Heating: dependence on particle shape

Remedies: Use more particles, or use higher order particles.

Heating rate



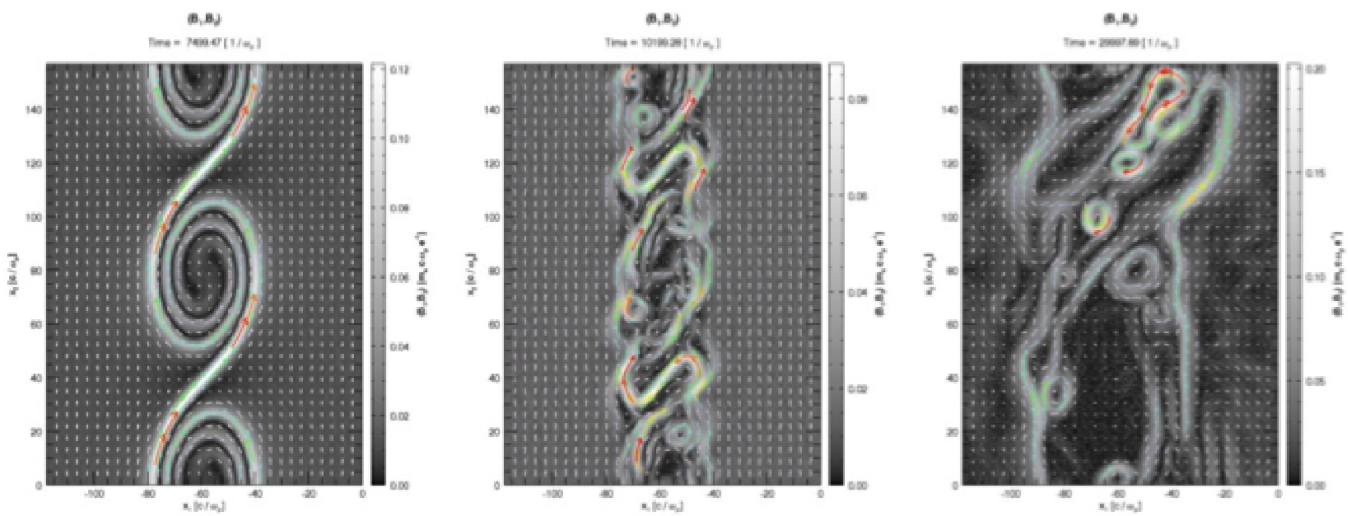
Performance
(16 core AMD Opteron / Nvidia K20X)

pusher	performance
order 2/1.5	23 M/sec
order 1	59 M/sec
order 1 (single)	78 M/sec
order 1 (SSE2)	94 M/sec
order 1 (CUDA)	824 M/sec

Numerical Heating: OSIRIS results

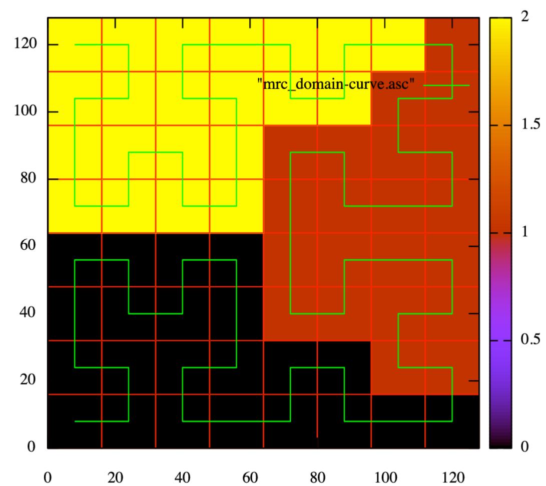
Results, using OSIRIS, Very Promising

ID	Case	Times, s	Energy drift, %
A	Linear	396	8.2×10^{-2}
B	Quadratic	561	1.9×10^{-3}
C	Cubic	852	6.8×10^{-4}
D	Linear, 256 part/cell	788	4.2×10^{-2}
E	Linear, no smoothing	394	1.05



Plasma Simulation Code (PSC)

- 1D, 2D, 3D configuration space
- relativistic, electromagnetic
- boost frame, moving window, PMLs, collisions, ionization...
- modular architecture: switching from legacy Fortran particle pusher to GPU pusher can be done on the command line.
- support for modern hardware (GPUs, Intel MIC)



Color indicates the MPI process responsible for the corresponding part of the domain.

PSC on GPUs

Multi-level decomposition of the problem, expose parallelism

- At the top-level, decompose spatial domain into *patches*. Each MPI process gets assigned one or more patches. Patches communicate via ghost cells / particle exchange.
- (Hybrid level can be introduced here: Each MPI process will distribute patches onto a set of cores or GPUs using OpenMP / threads)
- GPU: Each patch gets further divided into *blocks* (a.k.a. supercells) of multiple cells. These blocks are handled (in parallel) by threadblocks.
- Particles in a block are processed in parallel by threads in the threadblock (GPU) / by SIMD instructions (CPU/MIC).

PSC on GPUs

Particle-in-cell algorithm

for timestep $n = 0,1,2,...$:

for each particle m:

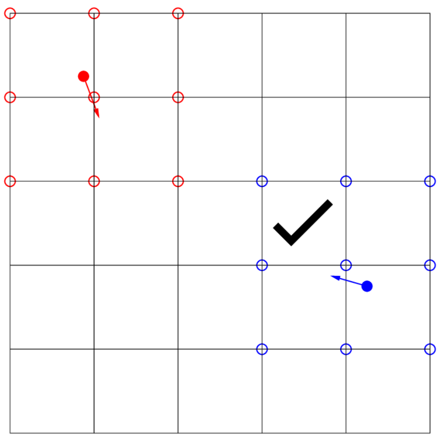
advance momentum: $\vec{p}_m^{\,n} \to \vec{p}_m^{\,n+1}$

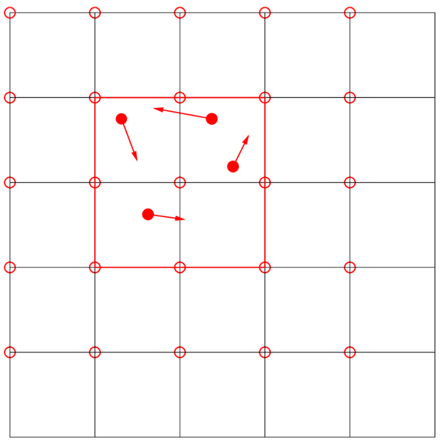
(using interpolated $\vec{E}^{n+1/2}$, $\vec{B}^{n+1/2}$)

advance position: $\vec{x}_m^{\,n+1/2} \to \vec{x}_m^{\,n+3/2}$

deposit current density contribution $\vec{j}_m^{\,n+1}$ onto mesh.

advance fields: $\vec{E}^{n+1/2}, \vec{B}^{n+1/2} \to \vec{E}^{n+3/2}, \vec{B}^{n+3/2}$ using $\vec{j}^{\,n+1}$.





PSC on GPUs – TitanDev/BlueWaters Performance

16-core AMD 6274 CPU, Nvidia Tesla M2090 / Tesla K20X

Kernel	Performance [particles/sec]
2D push & V-B current, CPU (AMD)	130×10^6
2D push & V-B current, GPU (M2090)	565×10^6
2D push & V-B current, GPU (K20X)	710×10^6

For best performance, need to use GPU and CPU simultaneously. Patch-based load balancing enables us to do that: On each node, we have 1 MPI-process that has $\approx$ 30 patches that are processed on the GPU, and 15 MPI-processes that have 1 patch each that are processed on the remaining CPU cores.