Studies of 3D dynamics in the global magnetosphere using high-performance heterogeneous computing architectures

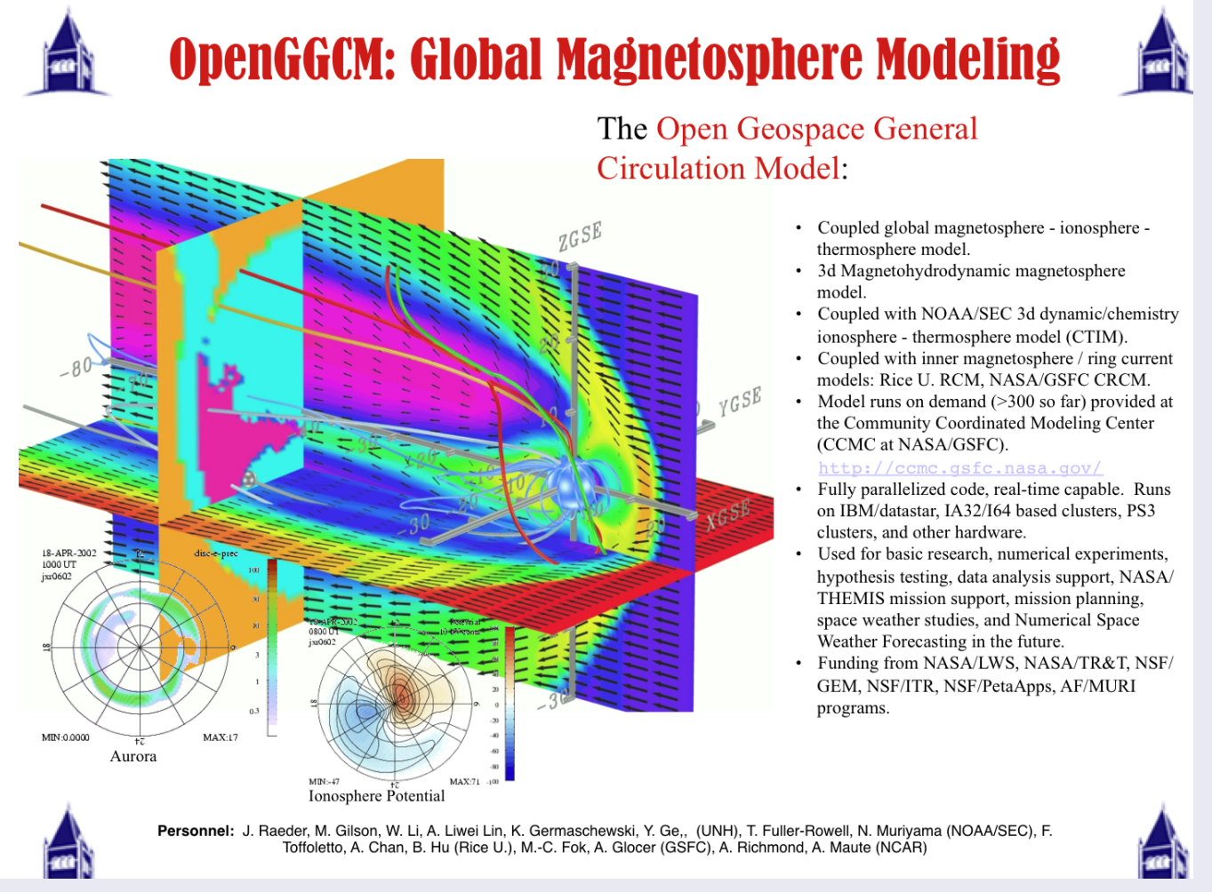
Kai Germaschewski, Kris Maynard, Narges Ahmadi, and Patrick Greene
Space Science Center, University of New Hampshire

Abstract

Magnetic reconnection and secondary instabilities of thin current sheets are of crucial importance to understanding the dynamics of Earth's global magnetosphere and space weather. We will present OpenGGCM global magnetosphere simulations of 3-d magnetopause reconnection at the Earth's day side, with a particular focus on effects of extended MHD models. We will also show results on the formation of mirror mode structures in the magnetosphere using local kinetic (particle-in-cell) simulations. We will report on progress in porting our codes to heterogeneous computing architectures using automated code generation.

Global Magnetosphere Simulations with OpenGGCM

The OpenGGCM Code



OpenGGCM: Global Magnetosphere Modeling

The Open Geospace General Circulation Model:

- Coupled global magnetosphere - ionosphere - thermosphere model.
- 3d Magnetohydrodynamic magnetosphere model.
- Coupled with NOAA/SEC 3d dynamic/chemistry ionosphere - thermosphere model (CTIM).
- Coupled with inner magnetosphere / ring current models: Rice U. RCM, NASA/GSFC CRCM.
- Model runs on demand (>300 so far) provided at the Community Coordinated Modeling Center (CCMC at NASA/GSFC). http://ccmc.gsfc.nasa.gov/
- Fully parallelized code, real-time capable. Runs on IBM/datastar, IA32/I64 based clusters, PS3 clusters, and other hardware.
- Used for basic research, numerical experiments, hypothesis testing, data analysis support, NASA/THEMIS mission support, mission planning, space weather studies, and Numerical Space Weather Forecasting in the future.
- Funding from NASA/LWS, NASA/TR&T, NSF/GEM, NSF/ITR, NSF/PetaApps, AF/MURI programs.

Personnel: J. Raeder, M. Gilson, W. Li, A. Liwei Lin, K. Germaschewski, Y. Ge,, (UNH), T. Fuller-Rowell, N. Muriyama (NOAA/SEC), F. Toffoletto, A. Chan, B. Hu (Rice U.), M.-C. Fok, A. Glocer (GSFC), A. Richmond, A. Maute (NCAR)

The OpenGGCM Model

OpenGGCM facts

- OpenGGCM has been community model at CCMC since 2001. To date ∼600 runs on demand, ∼100 unique users.
- Number of papers that include OpenGGCM results approaching 100.
- Current version 4.0 delivered in 2011.
- Current version includes RCM/CRCM sub models, but these are currently not offered for runs-on-demand.
- Development was funded under various NASA and NSF grants, in particular a 2006 LWS/SC grant.
- Recent developments include modularization, checkpointing, new I/O options, numerical/XMHD enhancements, and moving dipole, numerous improvements to the RCM/CRCM coupling.

Magnetopause Reconnection

Magnetospheric plasma is rather tenuous and collisionless, but numerical models typically use resistive MHD.

Questions

- How does resistive 3D reconnection scale in a global code?
- Do we observe the plasmoid instability in a global code?

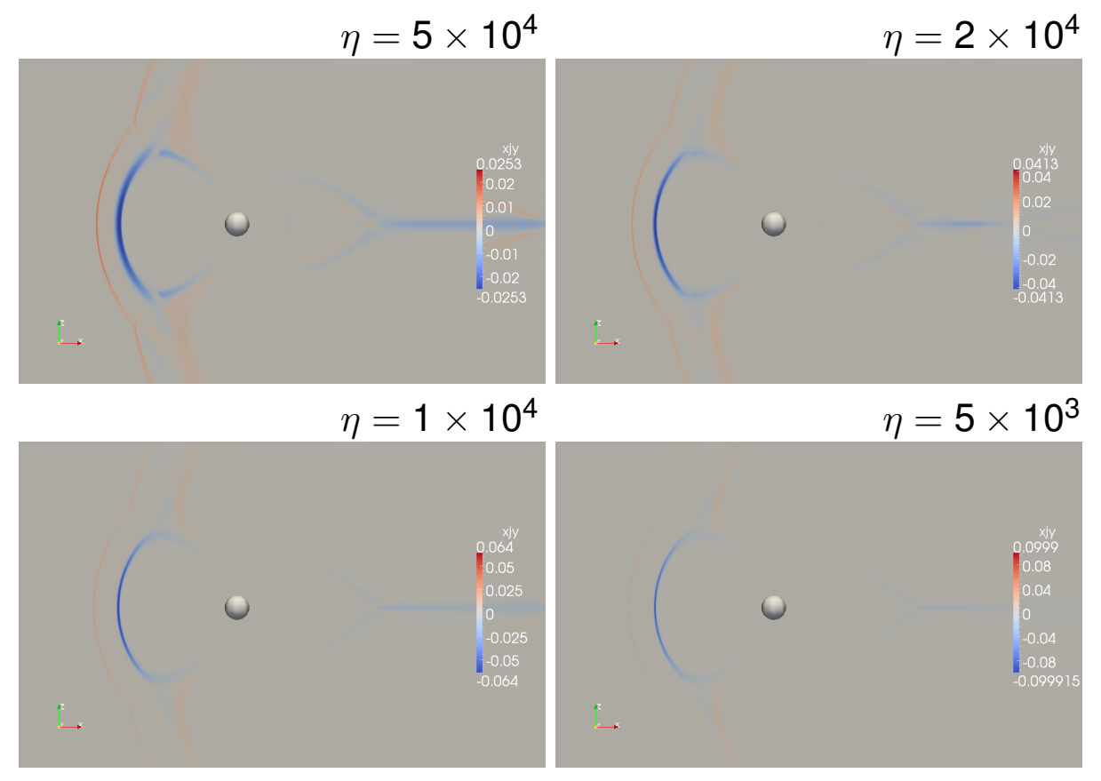
OpenGGCM runs were performed at UNH, BATSRUS runs at NASA's CCMC (Community Coordinated Modeling Center).

Parameters:

solar wind: $B_z = -5\,nT, v_x = -400\,km/s, n = 5\,cm^{-3}$

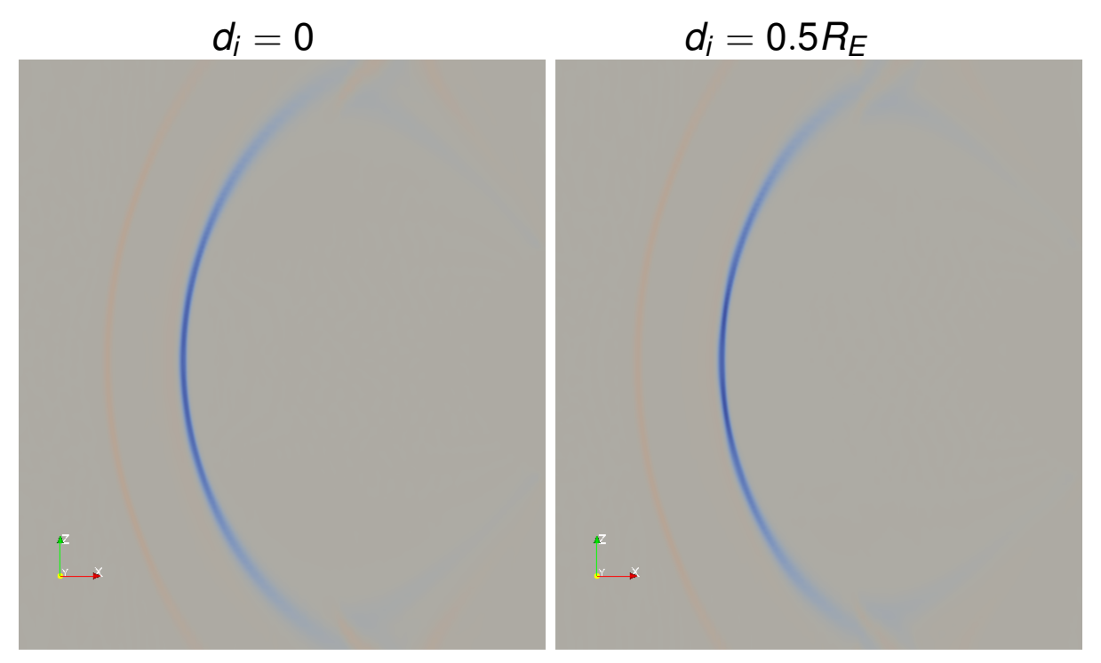
uniform resistivity: Lundquist number $S = 500 \dots 10000$

Magnetopause with varying resistivity



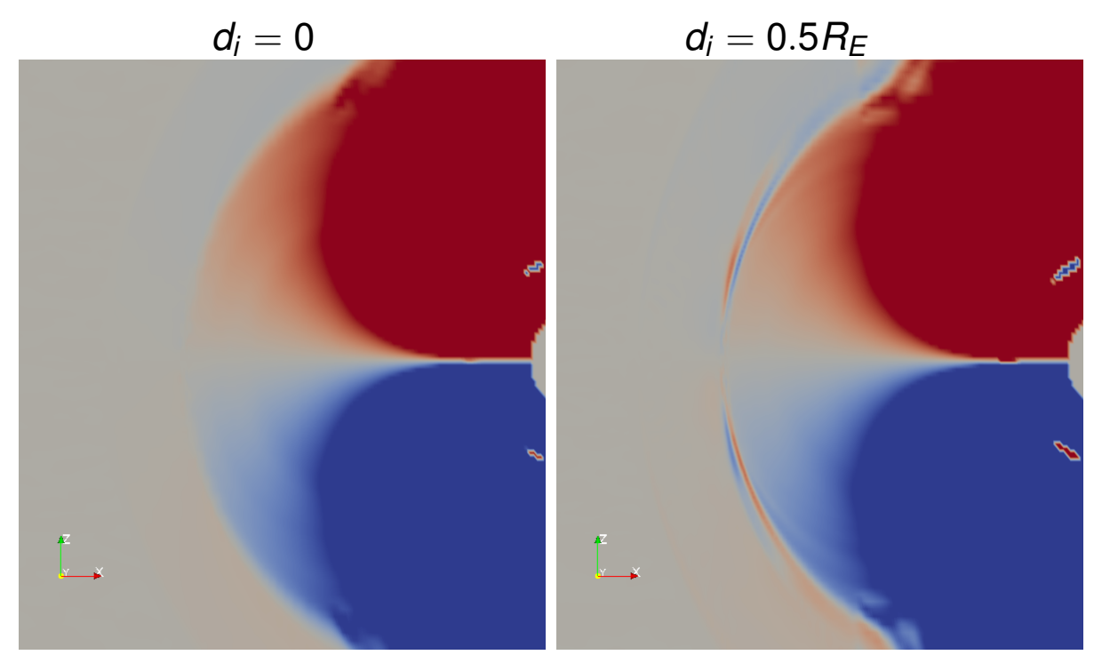
$\eta = 5 \times 10^4$, $\eta = 2 \times 10^4$, $\eta = 1 \times 10^4$, $\eta = 5 \times 10^3$

Current sheet structure resistive vs Hall MHD



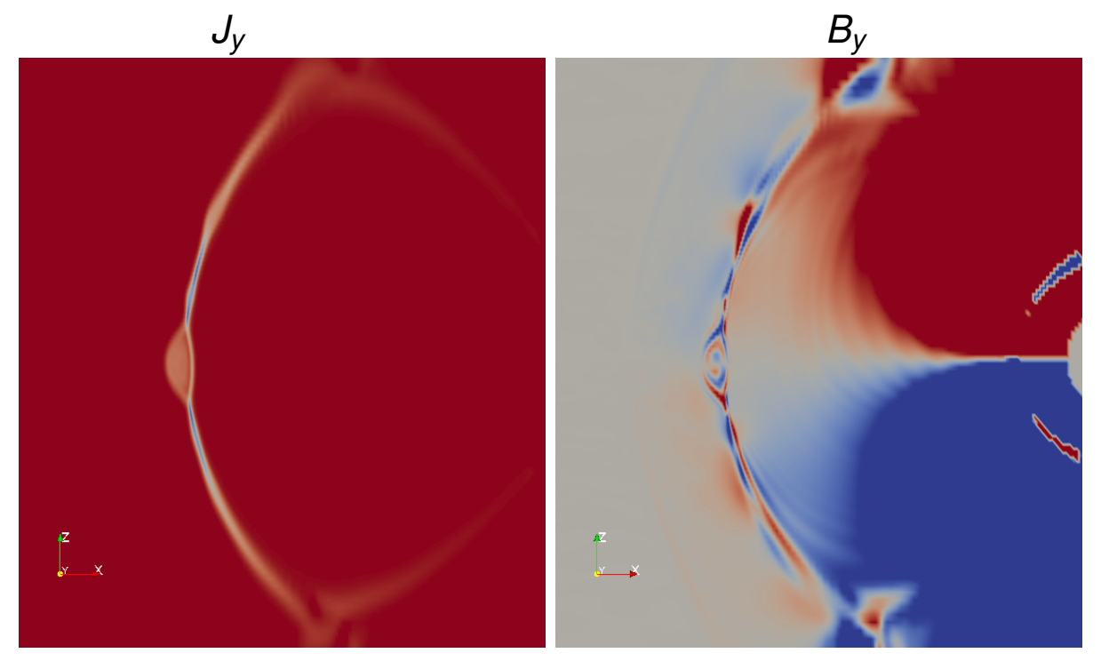
$d_i = 0$, $d_i = 0.5 R_E$

Out-of-plane magnetic field resistive vs Hall MHD



$d_i = 0$, $d_i = 0.5 R_E$

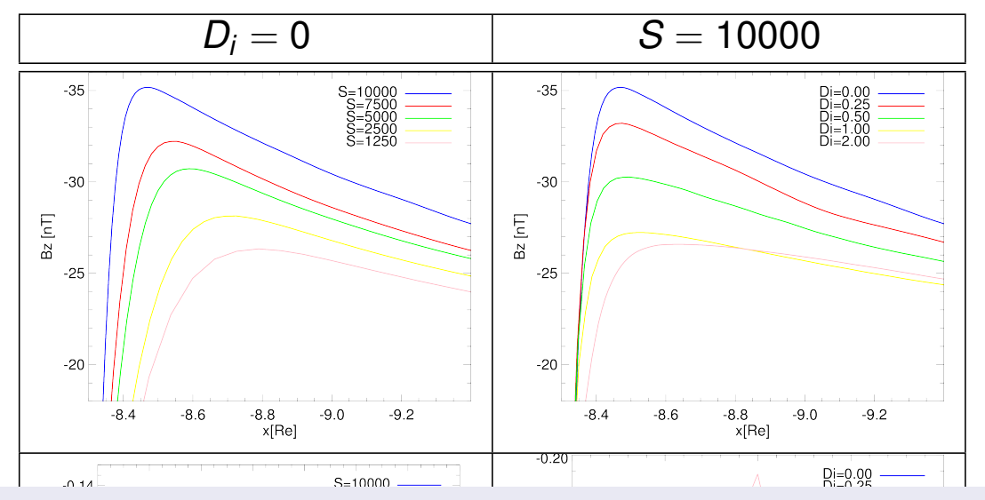
Quadrupolar out-of-plane magnetic field with Hall



J_y, B_y

Hall-OpenGGCM initial results – flux pileup

- For the purely resistive case, we essentially reproduce the results of Dorelli et al. (2004). Subsolar reconnection proceeds via a flux-pileup mechanism. Field measurements along the sun-earth line are in qualitative agreement with profiles of upstream field derived analytically.
- For the high-Lundquist number S runs, as d_i increases, we also observe that pile-up is suppressed. The simulations shown here resolve the magnetopause at resolutions of up to 104 grid cells per R_E.



$D_i = 0$, $S = 10000$

Automatic code generation

Lesson learned from porting OpenGGCM to the Cell processor: Programming heterogeneous architectures and achieving high performance is *hard*.

$\Longrightarrow$ Just let the computer do the work.

Automatic code generation lets you input your finite-difference / finite-volume equations in near symbolic form as a stencil computation, and then does all the work the generate efficient code for Cell / SSE2 / GPUs.

Example: $\partial_t \rho = -\nabla \cdot (\rho \vec{v})$

```
rhs_RHO = — Divg(ZIP(RHO, V))
```

What does code generation do?

Example: $\partial_t \rho = -\nabla \cdot (\rho \vec{v})$

```
V = 1./RHO * P
rhs_RHO = — Divg(ZIP(RHO, V))
```

```
  FLD3(r,jx,jy,jz,RHO) =
 (
  -((0.5*(RHO(x,jx+0,jy+0,jz+0)*
          (P0(x,jx+1,jy+0,jz+0) / RHO(x,jx+1,jy+0,jz+0)) +
          RHO(x,jx+1,jy+0,jz+0) *
          (P0(x,jx+0,jy+0,jz+0) / RHO(x,jx+0,jy+0,jz+0))) -
    0.5*(RHO(x,jx-1,jy+0,jz+0) *
          (P0(x,jx+0,jy+0,jz+0) / RHO(x,jx+0,jy+0,jz+0)) +
          RHO(x,jx+0,jy+0,jz+0) *
          (P0(x,jx-1,jy+0,jz+0) / RHO(x,jx-1,jy+0,jz+0)))) /
   (CRD0f(jx+1) - CRD0f(jx+0))
   +
  ...
```

Automatic code generation

Symbolic manipulation

Having the equations available in their natural (discretized) form allows for easy symbolic manipulation, e.g. find non-zero structure of the Jacobian, etc.

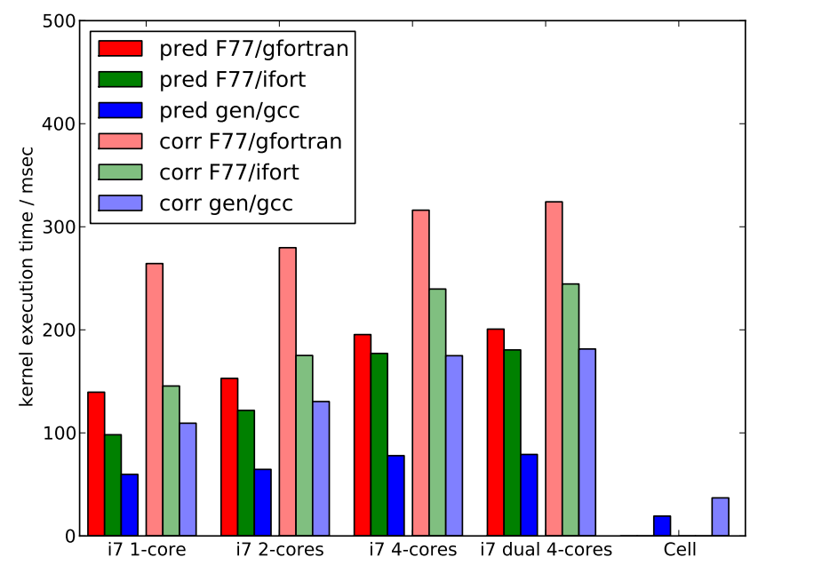
High performance

Conversion to actual computer code can automatically adapt to geometry, parameters, hardware architectures to obtain optimal performance.

Productivity

Only one version of the code needs to be maintained. Changing the underlying equations (physics) and numerics is simplified because they are abstracted out in near-mathematical form from their actual implementation.

Performance gains (SIMD, Cell)

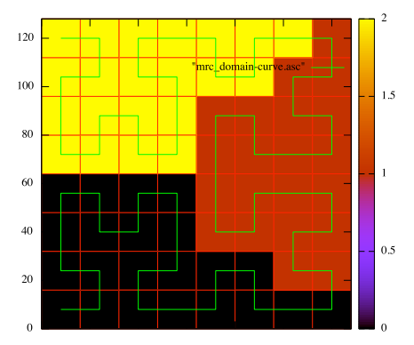


speed-up: 2.3× on multi-core, 200× (?!?) on the Cell (predictor)

Mirror mode structures in the magnetosheath

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 local domain.

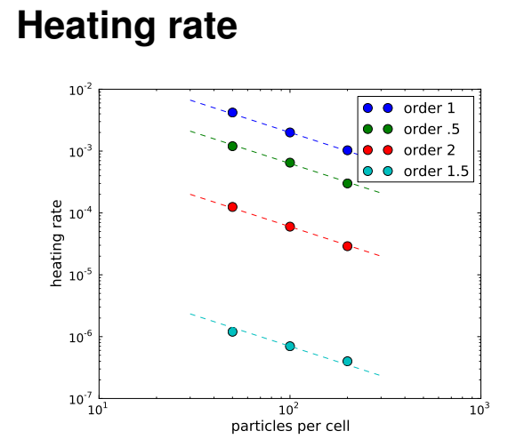
Numerical Heating: dependence on particle shape

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 ($\propto 1/N$).

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

PSC on GPUs

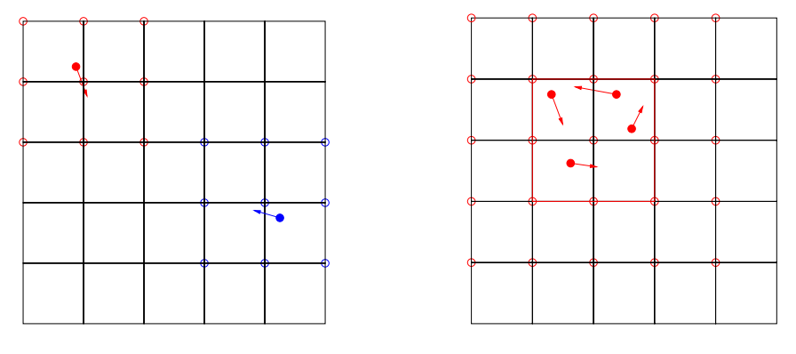
Multi-level decomposition of the problem, expose parallelism

- 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

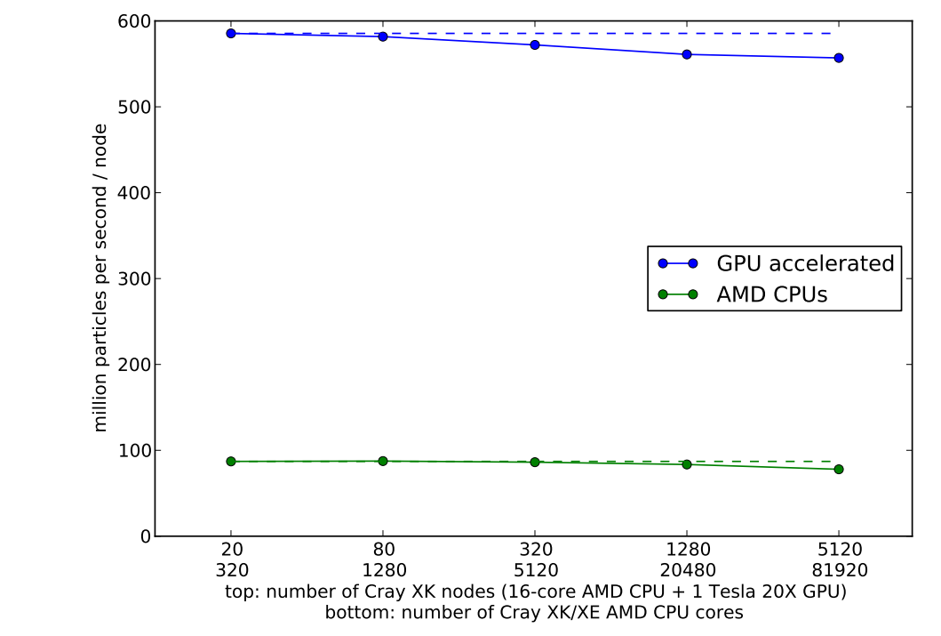
16-core AMD 6274 CPU, Nvidia Tesla M2090 / Tesla K20X

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 ≈ 45 patches that are processed on the GPU, and 15 MPI-processes that have 1 patch each that are processed on the remaining CPU cores.

PSC on GPUs – Parallel Performance

Weak scaling study on Craxy XK7 "Titan" at ORNL.



Temperatur anisotropy instabilities

Ion temperature $T_\perp > T_\parallel$ anistropy in a plasma with background magnetic field can drive kinetic instabilities:

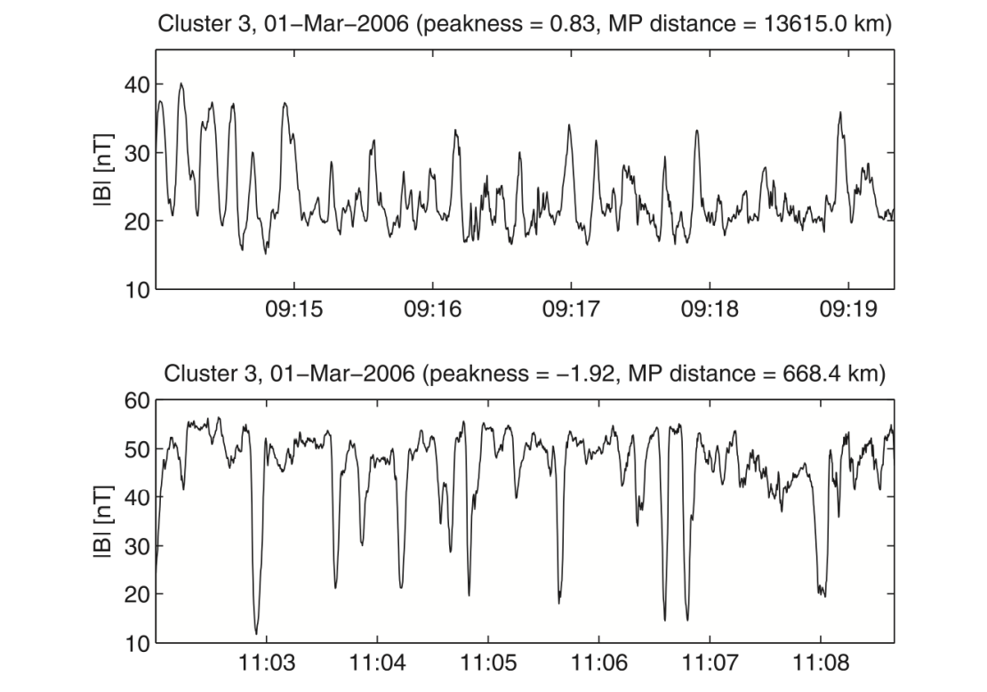
- ion cyclotron instability (propagating)
- mirror instability (standing)

Mirror mode instability condition

$$\frac{T_\perp}{T_\parallel} > 1 + 1/\beta_\perp$$

Mirror modes are observed in the solar wind, magnetosheath, and magnetosphere (near tail).

Mirror mode structures in the magnetosheath



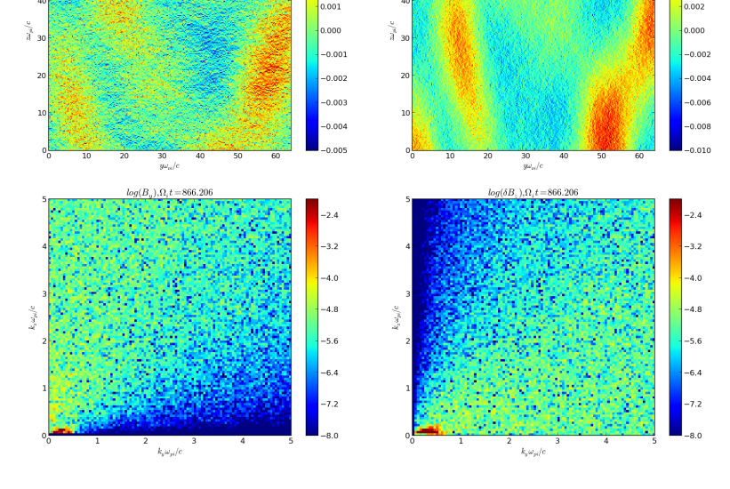
Soucek et al., 2008

Mirror mode questions

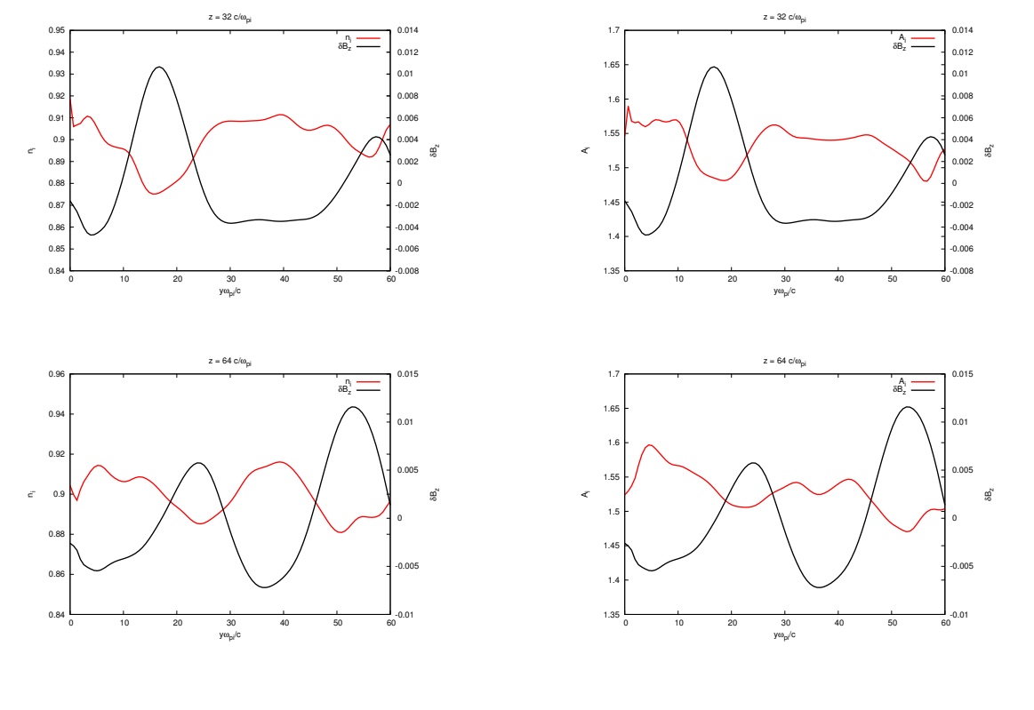
- Why is the mirror mode observed, rather than ion cyclotron waves? (depends on β, Helium stabilizes IC, IC propagate away, 2-d vs 3-d)
- How does the mirror mode evolve nonlinearly? What determines peaks vs dips? (mirror mode stable vs unstable regions)
- What role do electrons play? (isotropic in mirror structures)
- What is the spatial extent of mirror mode structures? (observations: smaller than ion gyroradius, tens of electron radii)

2-d PIC simulation

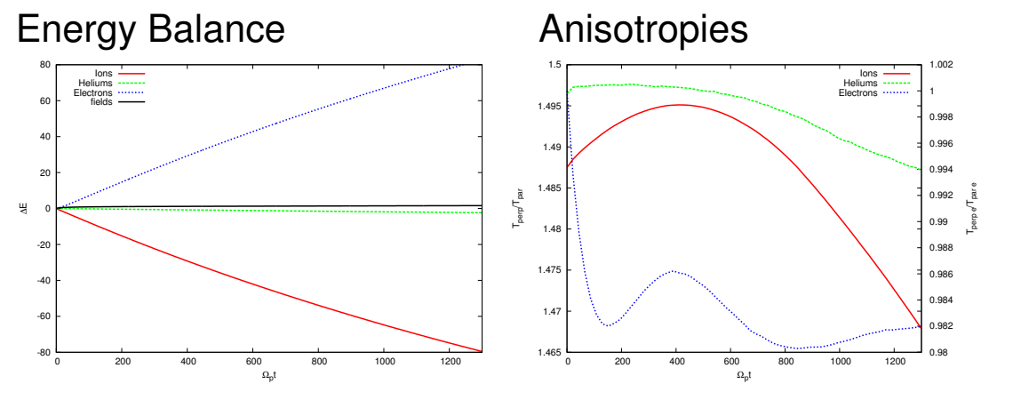
Bi-Maxwellian ions and helium are uniformly distributed in the simulation space with $T_{\perp i}/T_{\parallel i} = 1.5$. A constant background magnetic field $B_0 = 0.1$ is assumed in the z direction. Other parameters are: $\beta_i = 4.$, $\beta_e = 2.$, $\beta_h = 4.$, $T_{\perp e}/T_{\parallel e} = 1$, $T_{\perp h}/T_{\parallel h} = 1.5$, $m_h/m_i = 4.$, $m_i/m_e = 25$, $n_h = 0.1$



Anti-correlation of n_i and δB_z



Energy Balance, Evolution of Anisotropy



$T_{\perp i} = 0.03$, $T_{\parallel i} = 0.02$, $T_{\perp h} = 0.03$, $T_{\parallel h} = 0.02$, $T_{\perp e} = T_{\parallel e} = 0.01$, $m_i/m_e = 25$

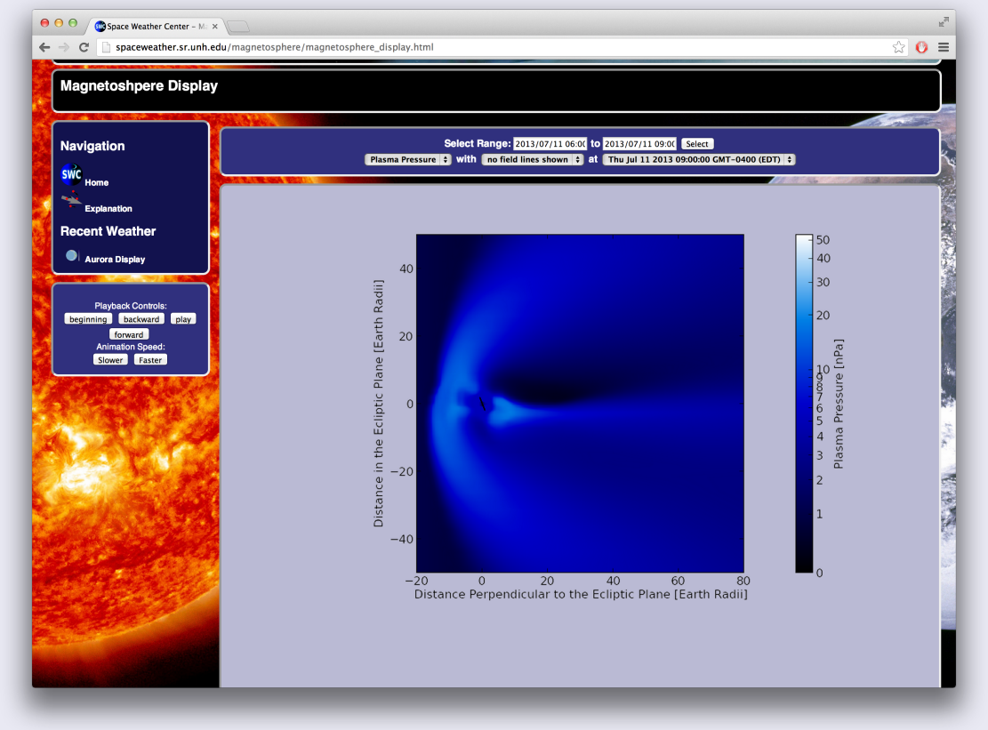
Education

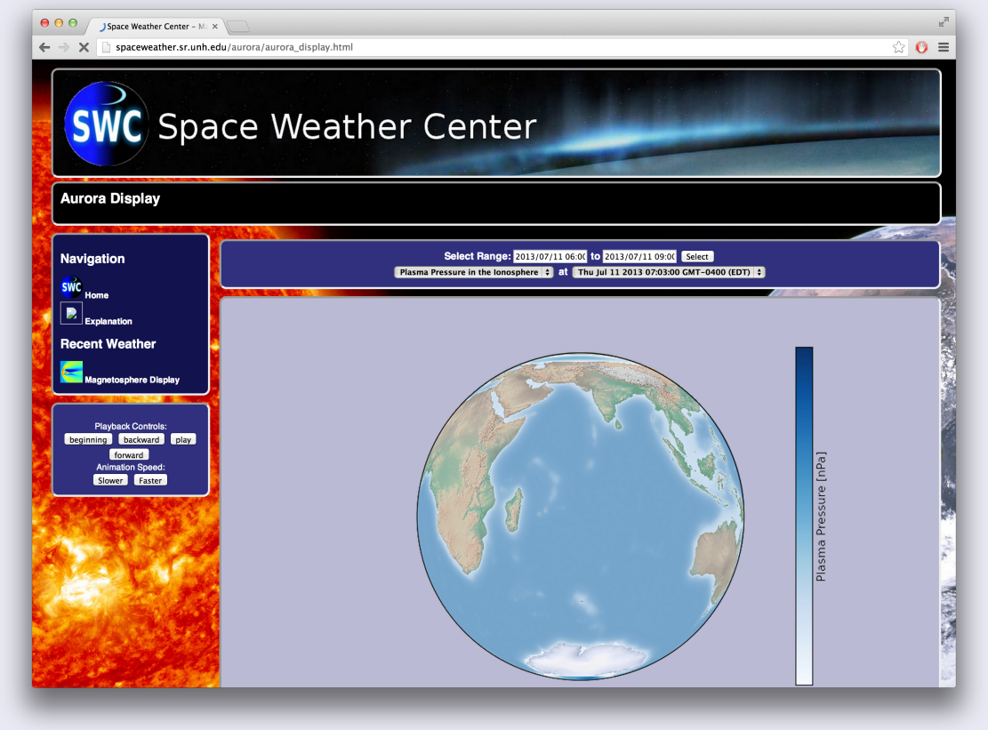
New graduate class: **"Introduction to High-Performance Computing"**

Topics include: Elementary Numerical Methods, Algorithm Development and Optimization, Parallel Programming Techniques, Distributed Processing over Multiple CPUs, Code Management and Interfacing to Fortran/C Programming Libraries, Data Visualization, Source Control.

Outreach: Spaceweather info and now-casting

Show live simulation results using current solar wind data `http://spaceweather.sr.unh.edu`





kai.germaschewski@unh.edu