Performance Evaluation of Kinetic Code on Scalar Processors

Takayuki Umeda

Nagoya University, Institute for Space-Earth Environmental Research

Eulerian Kinetic (Vlasov) Simulations for Space Plasma Studies

Basic equations for collisionless space plasma:

- Maxwell equations (for electromagnetic wave propagations)

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \qquad \nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t}$$

Computational load less than 0.1%

- Collisionless Boltzmann equation with electromagnetic field (known as Vlasov equation for charged particle motions)

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \frac{\partial f_s}{\partial \mathbf{x}} + \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_s}{\partial \mathbf{v}} = 0$$

f(x, y, ~~z~~, vx, vy, vz)

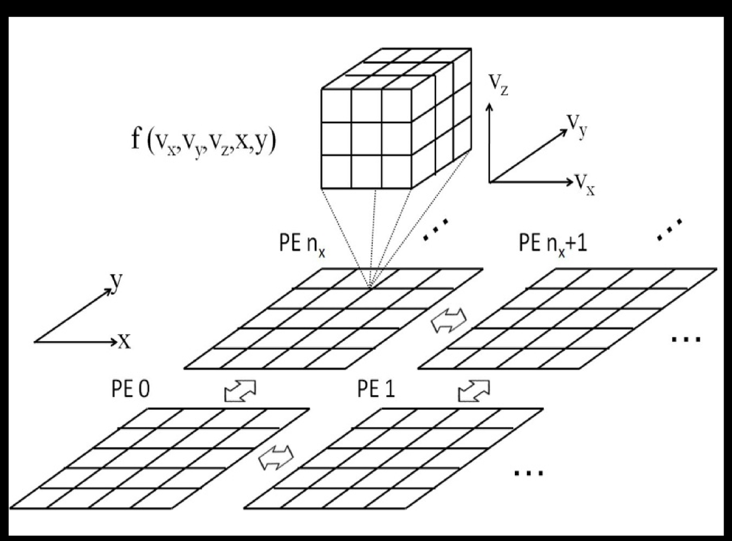
6D! ⇒ 5D

Operator splitting into three equations [Umeda et al. 2012a]

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \frac{\partial f_s}{\partial \mathbf{x}} = 0 \qquad \frac{\partial f_s}{\partial t} + \frac{q_s}{m_s}\mathbf{E} \cdot \frac{\partial f_s}{\partial \mathbf{v}} = 0 \qquad \frac{\partial f_s}{\partial t} + \frac{q_s}{m_s}(\mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_s}{\partial \mathbf{v}} = 0$$

(advection in position by v) (advection in velocity by E) (rotation by B)

Non-oscillatory, positive and conservative interpolation is used [Umeda 2009; Umeda et al. 2012]



[Umeda et al. 2012b]

40^5 ~ 4GB

40^6 ~ 160GB

- Hybrid parallelism is adopted to reduce number of processes
- Large number of dimensions (up to 6)

⇒ Requires huge memory

- Length of each loop is short: 20-40

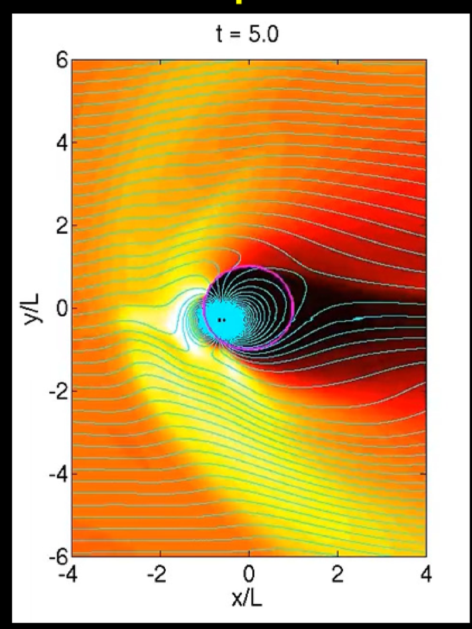
⇒ number of threads > loop length in many core environments.

- Multiple loops are thread-parallelized by loop collapsing of OpenMP

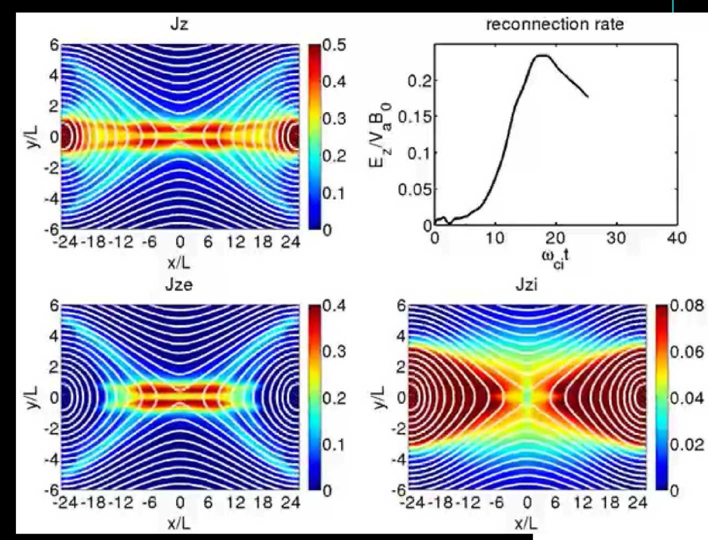
[Umeda et al. 2016]

Examples

2D position & 3D velocity (5D)



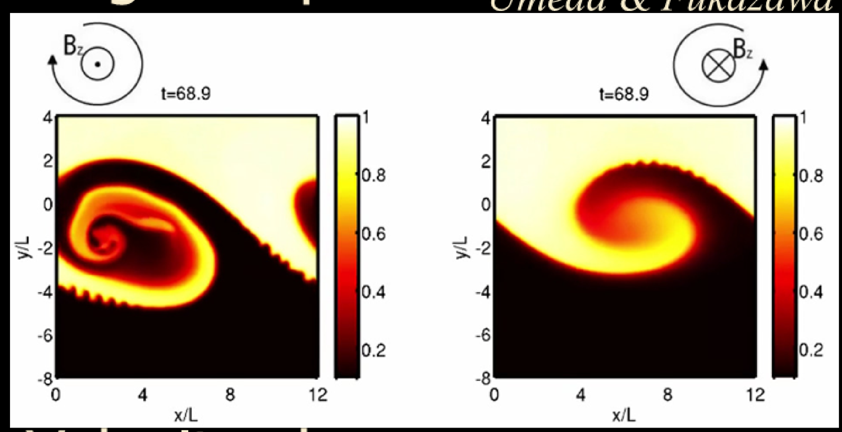
Magnetosphere



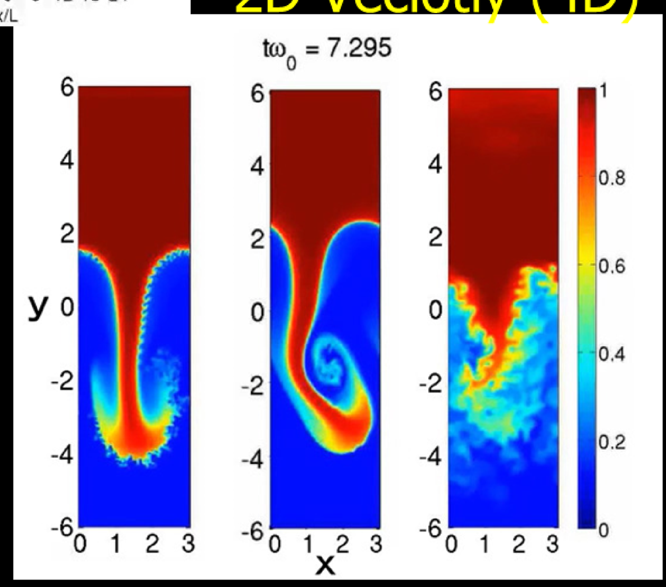
Magnetic shear (current layer) *Umeda et al. PoP 2012*

2 trillion (10^12) grids on K computer (6144 nodes) *Umeda & Fukazawa EPS 2015*

2D Position & 2D Veclotiy (4D)



Velocity shear *Umeda et al. PPCF 2014*



Density shear *Umeda et al. PoP 2017*

System Description

- Xeon Phi 7250 – 96GB DDR4
  - –Intel Compiler Ver.17.0.1
  - –Option: -ipo -ip -O3 -xMIC-AVX512
- Xeon E5-2697 v4 (dual) – 512GB DDR4
  - –Intel Compiler Ver.17.0.1
  - –Option: -ipo -ip -O3 -xCORE-AVX2
- SPARC64 (Fujitsu FX100) – 32GB HBM
  - –Option: -Kfast,ocl,simd,openmp,parallel
- Number of grids: 40*40*40*128*64*2 (~28GB)
- Elapsed time for 5 time steps is measured

©Nagoya Univ.

Performance Tuning

As-is code

```
1  do n=0,nvz+1
2    do m=0,nvy+1
3      do l=0,nvx+1
4        ffi(l)=1.0d0/ff(l,m,n,i,j)
5      end do
:
30     do l=0,nvx+1
31       gfxy  = abs((gfx(l)*ffi(l))* gfy(l))                 *0.5d0
32       gfyz  = abs((gfy(l)*ffi(l))* gfz(l))                 *0.5d0
33       gfzx  = abs((gfz(l)*ffi(l))* gfx(l))                 *0.5d0
34       gfxyz = abs((gfx(l)*ffi(l))*(gfy(l)*ffi(l))*gfz(l))*inv3
35       dfx(l+l0,m   ,n   ) = dfx(l+l0,m   ,n   ) +(gfx(l)+(gfxyz-gfxy-gfzx)*sx)
36       dfx(l+l0,m   ,n+nv) = dfx(l+l0,m   ,n+nv)          -(gfxyz-gfzx)      *sx
37       dfx(l+l0,m+mv,n   ) = dfx(l+l0,m+mv,n   )          -(gfxyz-gfxy)      *sx
38       dfx(l+l0,m+mv,n+nv) = dfx(l+l0,m+mv,n+nv)          + gfxyz            *sx
39       dfy(l   ,m+m0,n   ) = dfy(l   ,m+m0,n   ) +(gfy(l)+(gfxyz-gfyz-gfxy)*sy)
40       dfy(l+lv,m+m0,n   ) = dfy(l+lv,m+m0,n   )          -(gfxyz-gfxy)      *sy
41       dfy(l   ,m+m0,n+nv) = dfy(l   ,m+m0,n+nv)          -(gfxyz-gfyz)      *sy
42       dfy(l+lv,m+m0,n+nv) = dfy(l+lv,m+m0,n+nv)          + gfxyz            *sy
43       dfz(l   ,m   ,n+n0) = dfz(l   ,m   ,n+n0) +(gfz(l)+(gfxyz-gfzx-gfyz)*sz)
44       dfz(l   ,m+mv,n+n0) = dfz(l   ,m+mv,n+n0)          -(gfxyz-gfyz)      *sz
45       dfz(l+lv,m   ,n+n0) = dfz(l+lv,m   ,n+n0)          -(gfxyz-gfzx)      *sz
46       dfz(l+lv,m+mv,n+n0) = dfz(l+lv,m+mv,n+n0)          + gfxyz            *sz
47     end do
48   end do
49 end do
```

- There dependences on array access in the loop iteration at Lines 35—46

Tuned code

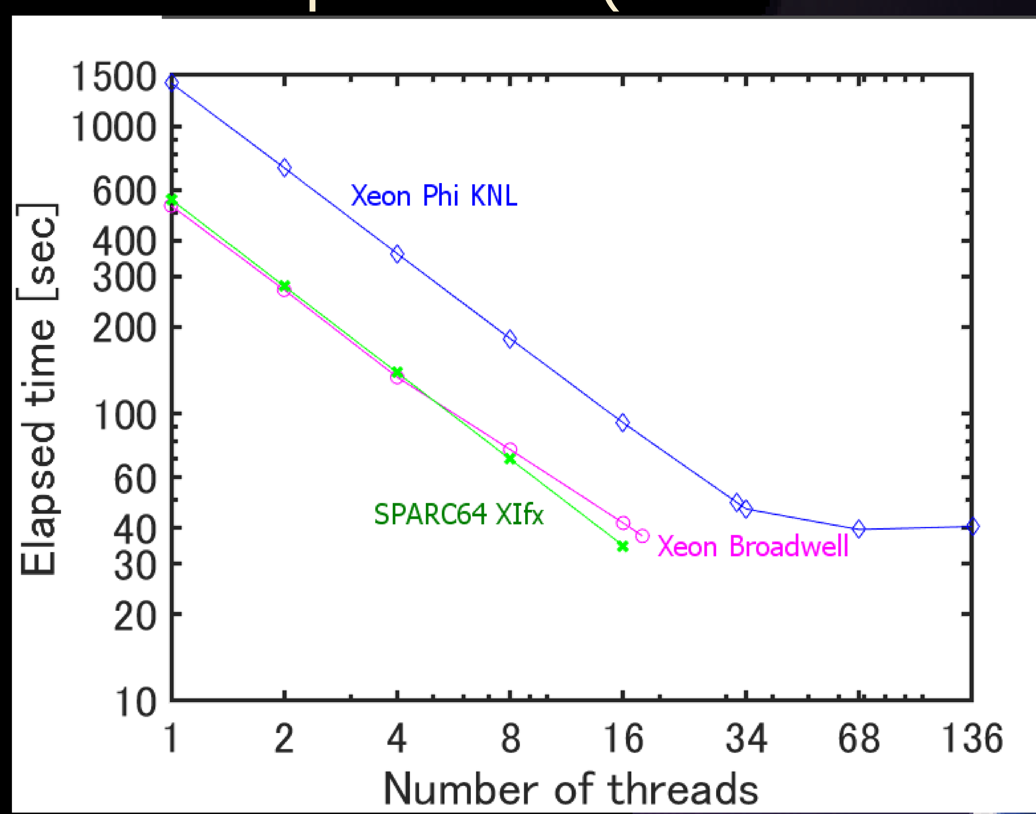
- Decomposition of most-inner loops ⇒ no loop dependence
- Reduction of temporary work arrays

```
30  do ll=nvxs-1,nvxe+1
31    ffi = 1.0d0/ff(ll,mm,nn,ii,jj)
32    gfx(ll)  = abs(gfx(ll)*ffi)
33    gfy(ll)  = abs(gfy(ll)*ffi)
34    gfz(ll)  = abs(gfz(ll)*ffi)
35  end do
36  do ll=nvxs-1,nvxe+1
37    dfx(ll+l0,mm   ,nn   ) = dfx(ll+l0,mm   ,nn   ) + ff(ll,mm,nn,ii,jj)*gfx(ll)*(gfy(ll)*(gfz(ll)*inv3-0.5d0)+(1.0d0-gfz(ll)*0.5d0))*sx
38    dfy(ll   ,mm+m0,nn   ) = dfy(ll   ,mm+m0,nn   ) + ff(ll,mm,nn,ii,jj)*gfy(ll)*(gfz(ll)*(gfx(ll)*inv3-0.5d0)+(1.0d0-gfx(ll)*0.5d0))*sy
39    dfz(ll   ,mm   ,nn+n0) = dfz(ll   ,mm   ,nn+n0) + ff(ll,mm,nn,ii,jj)*gfz(ll)*(gfx(ll)*(gfy(ll)*inv3-0.5d0)+(1.0d0-gfy(ll)*0.5d0))*sz
40  end do
41  do ll=nvxs-1,nvxe+1
42    dfx(ll+l0,mm   ,nn+nv) = dfx(ll+l0,mm   ,nn+nv) - ff(ll,mm,nn,ii,jj)*gfx(ll)*gfz(ll)*(gfy(ll)*inv3-0.5d0)*sx
43    dfy(ll+lv,mm+m0,nn   ) = dfy(ll+lv,mm+m0,nn   ) - ff(ll,mm,nn,ii,jj)*gfy(ll)*gfx(ll)*(gfz(ll)*inv3-0.5d0)*sy
44    dfz(ll   ,mm+mv,nn+n0) = dfz(ll   ,mm+mv,nn+n0) - ff(ll,mm,nn,ii,jj)*gfz(ll)*gfy(ll)*(gfx(ll)*inv3-0.5d0)*sz
45  end do
46  do ll=nvxs-1,nvxe+1
47    dfx(ll+l0,mm+mv,nn   ) = dfx(ll+l0,mm+mv,nn   ) - ff(ll,mm,nn,ii,jj)*gfx(ll)*gfy(ll)*(gfz(ll)*inv3-0.5d0)*sx
48    dfy(ll   ,mm+m0,nn+nv) = dfy(ll   ,mm+m0,nn+nv) - ff(ll,mm,nn,ii,jj)*gfy(ll)*gfz(ll)*(gfx(ll)*inv3-0.5d0)*sy
49    dfz(ll+lv,mm   ,nn+n0) = dfz(ll+lv,mm   ,nn+n0) - ff(ll,mm,nn,ii,jj)*gfz(ll)*gfx(ll)*(gfy(ll)*inv3-0.5d0)*sz
50  end do
51  do ll=nvxs-1,nvxe+1
52    dfx(ll+l0,mm+mv,nn+nv) = dfx(ll+l0,mm+mv,nn+nv) + ff(ll,mm,nn,ii,jj)*gfx(ll)*gfy(ll)*gfz(ll)*inv3*sx
53    dfy(ll+lv,mm+m0,nn+nv) = dfy(ll+lv,mm+m0,nn+nv) + ff(ll,mm,nn,ii,jj)*gfx(ll)*gfy(ll)*gfz(ll)*inv3*sy
54    dfz(ll+lv,mm+mv,nn+n0) = dfz(ll+lv,mm+mv,nn+n0) + ff(ll,mm,nn,ii,jj)*gfx(ll)*gfy(ll)*gfz(ll)*inv3*sz
55  end do
```

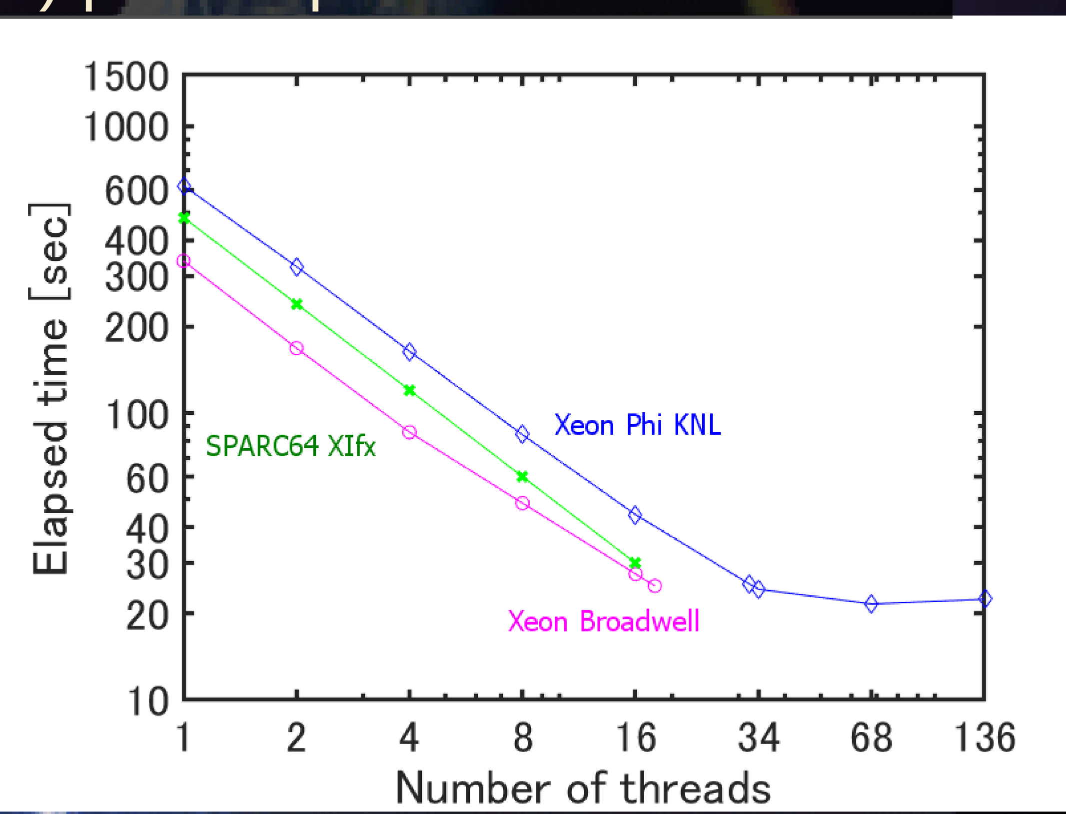
Performance Measurements

Strong scaling with $N_x \times N_y \times N_{vx} \times N_{vy} \times N_{vz} = 134 \times 70 \times 40^3$

Two processes (ion and electron) per compute node



As-is code



Tuned code

Summary

- Scales very well by thread parallelism
- Speedup by performance tuning:
  - –Xeon Phi 7250 (KNL) : 2.0 times
  - –Dual Xeon E5-2697 v4 (Broadwell) : 1.5 times
  - –SPARC64 (Fujitsu FX100) : 1.15 times
- Performance improved by reduction of temporary work arrays on Broadwell and KNL
- Bottlenecks in scheduling of instructions on SPARX64 XIfx