Parallel 3-D Electromagnetic Particle Code Using High Performance FORTRAN: Parallel TRISTAN

Dongsheng Cai[1], Yaoting Li[1,3], Ken-Ichi Nishikawa[2], Chijie Xiao[1,3], Xiaoyang Yan[1], and Zuying Pu[3]

[1] Institute of Information Sciences and Electronics,
The University of Tsukuba, Ibaraki 305-8573, Japan
email:{cai, ytli, cjxiao, yxy }@is.tsukuba.ac.jp
[2] National Space Science and Technology Center, 320 Sparkman Drive, SD 50,
Huntsville, AL 35805 USA email: Ken-Ichi.Nishikawa@nsstc.nasa.gov
[3] Department of geophysics, Peking University,
Beijing 100871, China
email:zypu@pku.edu.cn

Abstract. A three-dimensional full electromagnetic particle-in-cell (PIC) code, TRISTAN (Tridimensional Stanford) code, has been parallelized using High Performance Fortran (HPF) as a RPM (Real Parallel Machine). In the parallelized HPF code, the simulation domain is decomposed in one-dimension, and both the particle and field data located in each domain that we call the sub-domain are distributed on each processor. Both the particle and field data on a sub-domain are needed by the neighbor sub-domains and thus communications between the sub-domains are inevitable. Our simulation results using HPF exhibit the promising applicability of the HPF communications to a large scale scientific computing such as solar wind-magnetosphere interactions.

1 Introduction

This paper reports on parallelization of Tridimensional Stanford (TRISTAN) code (Buneman, 1993) that is a three-dimensional electromagnetic full particle code developed at Stanford University on a two-way PentiumPro PC cluster that consists of 16 distributed SMPs and other commercial parallel computers like Fujitsu VPP5000, NEC SX-6 and Hitachi SR-8000 etc. using High Performance Fortran (HPF).

In our parallel program, the simulation domain is decomposed into the sub-domains as shown in Fig. 1. The Particle-In-Cell (PIC) computation in TRISTAN to be performed on a certain sub-domain or on a certain processor where the sub-domain is distributed will typically require the data from their neighbor processors to proceed the whole PIC simulations. Here we distribute the field arrays and the particles over processors as indicated in Fig. 1. Thus the data must be transferred between processors in each time step so as to allow PIC simulation to proceed in time. These inter-processor communications in each time step need to be programmed in HPF constructs.

J. Büchner, C.T. Dum, M. Scholer (Eds.): LNP 615, pp. 25–53, 2003.

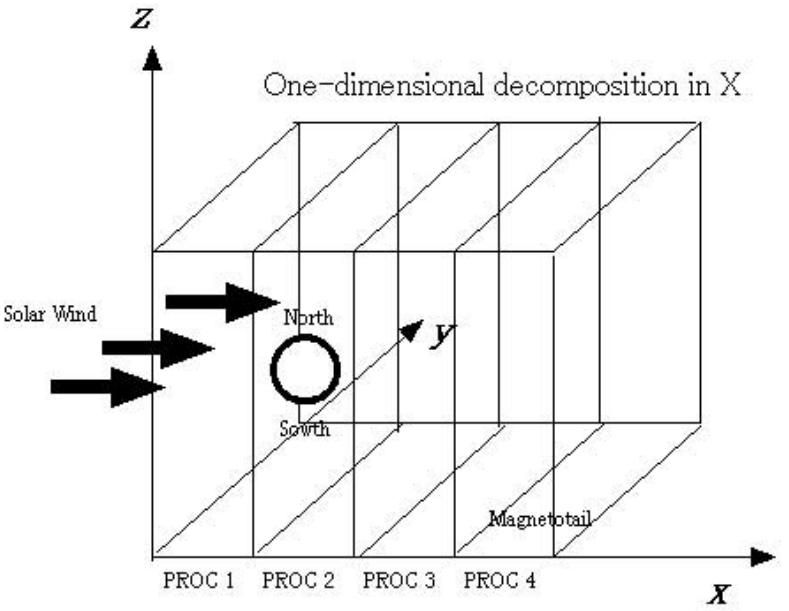

Fig. 1. Coordinate of the simulation domains and domain decomposition in x.

The amount of inter-processor communications needed for a parallel program basically depends on the algorithms and the scales of the physical problem sizes adopted in the simulations. In PIC simulations, they are the way decomposing the simulation domains, the sizes of the sub-domain boundaries, and the number of the particles in a cell, respectively.

The pgHPF compiler of Portland Group Inc. aims to realize the standard High Performance Fortran specification and can be installed on a number of parallel machines. Executable codes produced by the pgHPF compilers for PCs with IA-32 CPUs are unconstrained, and can be executed on any compatible IA-32 processor-based system regardless of whether the pgHPF compilers are installed on that system or not. From the HPF programmer's point of view, the differences between versions of the pgHPF runtime library have little effect on program developments.

In parallel programming models, usually, the SPMD (Single Program Multiple Data) models using MPI (Message Passing Interface) or PVM (Parallel Virtual Machine) are one of the most popular models. Our HPF TRISTAN code also uses the same SPMD models. The biggest advantage of HPF is its programming style. Once the simulation domain is decomposed and the data are distributed to each sub-domains or over processors using simple HPF compiler directives, other HPF programming styles are very similar to those in usual Fortrans. Of course, the biggest problem here is the performance issues comparing with those using MPI or PVM.

Actually, pgHPF is based on a RPM (PGI Proprietary Communications - Real Parallel Machine) protocol. This transport mechanism was developed by PGI to model the behavior of PVM among a homogeneous group of hosts on a network. It offers both greater programming efficiency and performance than PVM with fewer requirements. In this paper, to archive a similar high performance using HPF comparing with that using MPI or PVM in the full electromagnetic PIC simulation, some careful optimizations of inter-processor communications are proposed.

Our code is the same as TRISTAN code except for the parallelization part, which utilizes rigorous charge-conserving formulas and radiating boundary conditions (Buneman, 1993). It was written in HPF so that the code can be run on any parallel computers with the HPF compilers.

The parallelization part of our HPF TRISTAN code is same as (Liewer and Decyk, 1985) and (Decyk, 1995). We separate the communication parts from computation parts, and use both the "particle manager" and the "field manager" to localize the inter-processor communications (Decyk, 1995) as shown in Fig. 2. Thus the code can be easily converted to MPI or PVM version of TRISTAN code.

The basic controlling equations of the plasmas are Newton-Lorentz equation:

$$m_{i,e}\frac{d\boldsymbol{v}_{i,e}}{dt} = q_{i,e}(\boldsymbol{E} + \boldsymbol{v}_{i,e} \times \boldsymbol{B}), \tag{1}$$

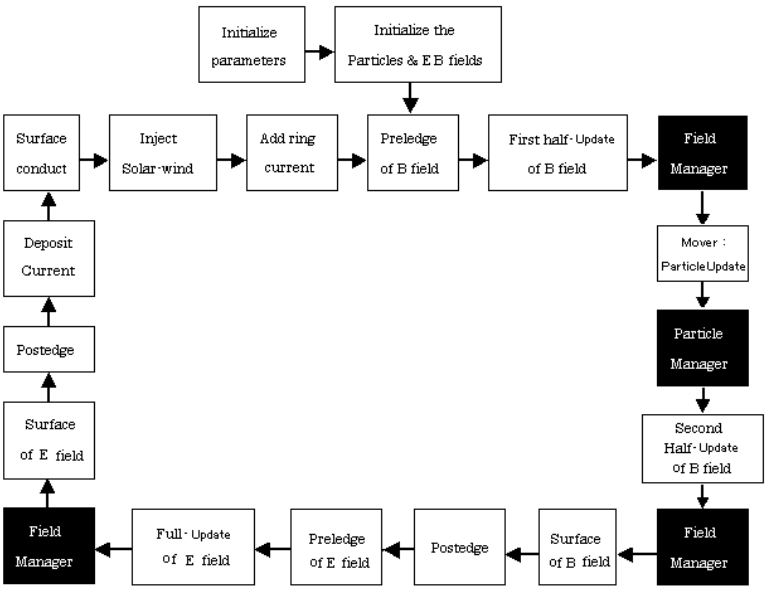

Fig. 2. The computational cycle of the HPF TRISTAN code. The black boxes represent HPF communications.

where i and e corresponds to ion and electron, respectively, and Maxwell equations:

$$\frac{\partial \boldsymbol{B}}{\partial t} = - \bigtriangledown \times \boldsymbol{E}, \tag{2}$$

$$\frac{\partial \boldsymbol{E}}{\partial t} = c^2 \bigtriangledown \times \boldsymbol{B} - \frac{1}{\varepsilon_0} \boldsymbol{J}, \tag{3}$$

Here

$$\boldsymbol{J} = \sum (n_i q_i \boldsymbol{v}_i - n_e q_e \boldsymbol{v}_e). \tag{4}$$

The coordinate and one-dimensional domain decomposition using in the simulation domain is shown in Fig. 1. For parallel benchmarking purposes, we perform the real simulations of solar wind-magnetosphere interactions using the code. For the simulation of solar wind-magnetosphere interactions, the following boundary conditions were used for the particles (Buneman, 1993): (1) Fresh particles representing the incoming solar wind (unmagnetized in our test run) are continuously injected across the yz plane at $x = x_{min}$ with a thermal velocity plus a bulk velocity in the $+x$ direction; (2) Thermal solar particle flux is also injected across the sides of our rectangular computation domain; (3) Escaping particles are arrested in a buffer zone, redistributed in those grid they escaped more uniformly by making the zone conducting in order to simulate their escape to infinity, and finally written off. We use a simple model for the ionosphere where both electrons and ions are reflected by the Earth dipole magnetic field. The effects of the Earth rotation are not included. The effect of thermal expansion of the solar-wind is also not included. Since the solar-wind and the Earth dipole magnetic field are included, some load-imbalance due to this asymmetry is expected in this HPF TRISTAN code. In Sect. 2, some basics of TRISTAN code and the ways to run it are introduced. In Sect. 3, basics of HPF TRISTAN data structure and array distributions are discussed. In Sects. 4 and 5, field and particle data domain decompositions and the way of communication between processors are discussed, respectively. In Sect. 6, unstability of the HPF communication and the way that avoids the performance degradation are discussed. In Sect. 7, benchmark and simulation results of the HPF TRISTAN code on PC cluster are discussed. Section 8 concludes the remarks of this paper.

2 Basics of TRISTAN Code

The control equations of TRISTAN code are Maxwell and Newton-Lorentz equations only. Instead of solving Poisson equation that is solved numerically in almost all particle simulation codes, TRISTAN code solves only two curls, i.e. Ampere and Faraday equations. A rigorous charge conservation method for the current deposits is described in (Villasenor and Buneman, 1992).

The particles that are initialized as unmagnetized Maxwell distribution are updated by the leap-frog method. Throughout the code the linear interpolation is employed. The computational cycle of HPF TRISTAN code is displayed in Fig. 2. The black boxes in the figure are HPF communication subroutines and they are field and particle managers.

2.1 Fields

TRISTAN code scales such that $\epsilon_0 = 1$ and hence $\mu_0 = 1/c^2$. This also means $\mathbf{E} = \mathbf{D}$. Instead of recording components of $\mathbf{B}$ or $\mathbf{H}$, TRISTAN records bx, by, bz of $c\mathbf{B}$ (alias $\mathbf{H}/\mathbf{c}$). This makes symmetry for electric field and magnetic field ($\mathbf{E} \longleftrightarrow \mathbf{B}$) in Maxwell equations. Throughout, TRISTAN uses a rectangular cubic grid with $\delta x = \delta y = \delta z = 1$ and time discretisation with $\delta t = 1$. Before and after moving (or pushing) the particles, $\mathbf{B}$ is updated in two half steps so that it is available at the same time as $\mathbf{E}$ for the particle update.

In TRISTAN code, only two curls of Maxwell equations are solved:

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}, \tag{5}$$

$$\frac{\partial \mathbf{D}}{\partial t} = \nabla \times \mathbf{H} - \mathbf{J}. \tag{6}$$

and here

$$\mathbf{B} = \mu_0 \mathbf{H}, \tag{7}$$

$$\mathbf{D} = \epsilon_0 \mathbf{E}. \tag{8}$$

If we scale Maxwell equations using $\epsilon_0 = 1$, and substitute $\mathbf{E} = \mathbf{D}$, $\mathbf{B} = \mathbf{cB}$ into eq. (5) and (6), we obtain:

$$\frac{\partial}{\partial t}\mathbf{B} = -c\nabla \times \mathbf{E}, \tag{9}$$

$$\frac{\partial}{\partial t}\mathbf{E} = c\nabla \times \mathbf{B} - \mathbf{J}. \tag{10}$$

These two equations imply the fields symmetry($\mathbf{E} \longleftrightarrow \mathbf{B}$).

2.2 Magnetic Field Update

The staggered grid mesh system, known in the computational electromagnetic community as Yee lattice (Yee, 1966), is shown in Fig. 3. It ensures that the change of $\mathbf{B}$ flux through a cell surface equals the negative circulation of $\mathbf{E}$ around that surface and the change of $\mathbf{E}$ flux through a cell surface equals the circulation of $\mathbf{B}$ around that surface minus the current through it. Here $\mathbf{B}$ and $\mathbf{E}$ are in a symmetry form except subtracting the charge flux $\mathbf{J}$ in Ampere equation. Charge flux $\mathbf{J}$ is calculated and subtracted after the particles are moved later in the program. Thus magnetic fields are updated as follows:

The change of $\mathbf{B}$ flux can be expressed as:

$$\frac{\partial \mathbf{B}}{\partial t} = -c \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ e_x & e_y & e_z \end{vmatrix} = c[\ \mathbf{i}(\frac{\partial e_y}{\partial z} - \frac{\partial e_z}{\partial y}) + \mathbf{j}(\frac{\partial e_z}{\partial x} - \frac{\partial e_x}{\partial z}) + \mathbf{k}(\frac{\partial e_x}{\partial y} - \frac{\partial e_y}{\partial x})] \tag{11}$$

In Yee lattice, e_x, e_y, e_z, b_x, b_y, and b_z are, respectively, staggered and shifted on 0.5 from (i, j, k) and located at the positions as follows:

$$\begin{array}{c} e_x(i,j,k) \rightarrow e_x(i+.5,j,k), \\ e_y(i,j,k) \rightarrow e_y(i,j+.5,k), \\ e_z(i,j,k) \rightarrow e_z(i,j,k+.5), \end{array} \tag{12}$$

and

$$\begin{array}{c} b_x(i,j,k) \rightarrow b_x(i,j+.5,k+.5), \\ b_y(i,j,k) \rightarrow b_y(i+.5,j,k+.5), \\ b_z(i,j,k) \rightarrow b_z(i+.5,j+.5,k). \end{array} \tag{13}$$

In our simulation, we use integer grids. In both Eq. (12) and (13), i, j, k in the right-hand sides correspond to Fortran array indices notations and i, j, k in the left hand sides correspond to the real positions in the simulation domains as shown in Fig. 3. In this report, if the values "0.5" are added to either i, j, k in the array indices, then the array indices correspond to the real positions in the simulation domains.

Thus the magnetic field components b_x, b_y, b_z are, respectively, updated by the negative circulation of **E** around Yee lattice surface as follows:

$$\begin{aligned} \frac{\partial}{\partial t} b_x &= (b_x^{new}(i,j+.5,k+.5) - b_x^{old}(i,j+.5,k+.5))/\delta t \\ &= c[(e_y(i,j+.5,k+1) - e_y(i,j+.5,k))/\delta z \\ &\quad -(e_z(i,j+1,k+.5) - e_z(i,j,k+.5))/\delta y]. \end{aligned} \tag{14}$$

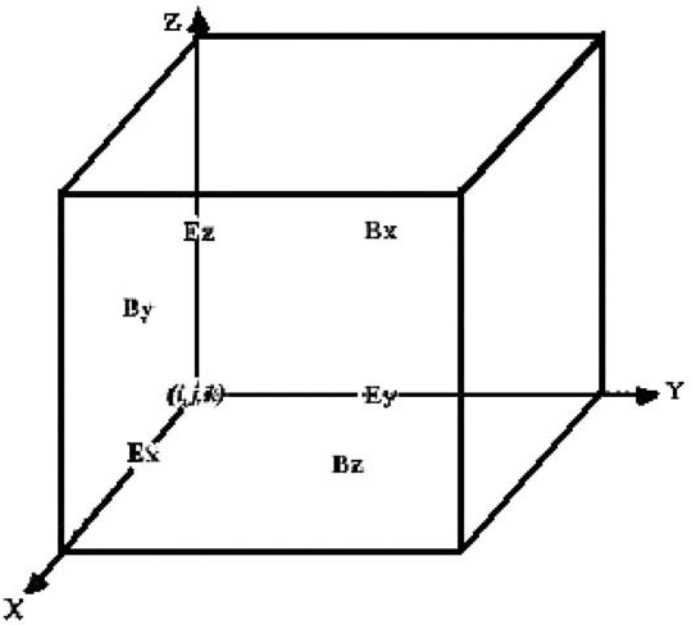

Fig. 3. The positions of field components in Yee lattice.

Here $\delta t = \delta z = \delta y = \delta x = 1$. Thus we get the update form:

$$\begin{aligned} b_x^{new}(i,j,k) &= b_x^{old}(i,j,k) \\ &\quad +c[\ e_y(i,j,k+1) - e_y(i,j,k) - e_z(i,j+1,k) + e_z(i,j,k)\ ]. \end{aligned} \tag{15}$$

To get the update form of b_y, and b_z, the same procedures are as followed:

$$\begin{aligned} \tfrac{\partial}{\partial t} b_y &= (b_y^{new}(i+.5,j,k+.5) - b_y^{old}(i+.5,j,k+.5))/\delta t \\ &= c[(e_z(i+1,j,k+.5) - e_z(i,j,k+.5))/\delta x \\ &\quad - (e_x(i+.5,j,k+1) - e_x(i+.5,j,k))/\delta z\ ], \end{aligned} \tag{16}$$

$$\begin{aligned} b_y^{new}(i,j,k) &= b_y^{old}(i,j,k) \\ &\quad +c[\ e_z(i+1,j,k) - e_z(i,j,k) - e_x(i,j,k+1) + e_x(i,j,k)\ ], \end{aligned} \tag{17}$$

$$\begin{aligned} \tfrac{\partial}{\partial t} b_z &= (b_z^{new}(i+.5,j+.5,k) - b_z^{old}(i+.5,j+.5,k))/\delta t \\ &= c[(e_x(i+.5,j+1,k) - e_x(i+.5,j,k))/\delta y \\ &\quad - (e_y(i+1,j+.5,k) - e_y(i,j+.5,k))/\delta x], \end{aligned} \tag{18}$$

$$\begin{aligned} b_z^{new}(i,j,k) &= b_z^{old}(i,j,k) \\ &\quad +c[\ e_x(i,j+1,k) - e_x(i,j,k) - e_y(i+1,j,k) + e_y(i,j,k)]. \end{aligned} \tag{19}$$

2.3 Electric Field Update

In Yee lattice, e_x, e_y, and e_z are, respectively, staggered and shifted 0.5 from (i,j,k) and located at the positions as shown in Fig. 3.

The change of **E** flux through a cell surface equals the circulation of **B** around that surface minus the current through it. First, the electric field is updated by the circulation of **B** around Yee lattice surface as follows:

$$\begin{aligned} \tfrac{\partial \mathbf{E}}{\partial t} = c \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ b_x & b_y & b_z \end{vmatrix} &= c[\ \mathbf{i}(\tfrac{\partial b_z}{\partial y} - \tfrac{\partial b_y}{\partial z}) \\ &\quad + \mathbf{j}(\tfrac{\partial b_x}{\partial z} - \tfrac{\partial b_z}{\partial x}) \\ &\quad + \mathbf{k}(\tfrac{\partial b_y}{\partial x} - \tfrac{\partial b_x}{\partial y})] \end{aligned} \tag{20}$$

Thus the electric field components e_x, e_y, e_z are, respectively, updated by the circulation of **B** around Yee lattice surface as follows:

$$\begin{aligned} \tfrac{\partial}{\partial t} e_x &= (e_x^{new}(i+.5,j,k) - e_x^{old}(i+.5,j,k))/\delta t \\ &= c[(b_z(i+.5,j+.5,k) - b_z(i+.5,j-.5,k))/\delta y \\ &\quad - (b_y(i+.5,j,k+.5) - b_y(i+.5,j,k-.5))/\delta z], \end{aligned} \tag{21}$$

$$\begin{aligned} e_x^{new}(i,j,k) &= e_x^{old}(i,j,k) \\ &\quad +c[\ b_y(i,j,k-1) - b_y(i,j,k) - b_z(i,j-1,k) + b_z(i,j,k)\ ], \end{aligned} \tag{22}$$

$$\begin{aligned} \tfrac{\partial}{\partial t} e_y &= (e_y^{new}(i,j+.5,k) - e_y^{old}(i,j+.5,k))/\delta t \\ &= c[(b_x(i,j+.5,k+.5) - b_x(i,j+.5,k-.5))/\delta z \\ &\quad - (b_z(i+.5,j+.5,k) - b_z(i-.5,j+.5,k))/\delta x], \end{aligned} \tag{23}$$

$$\begin{aligned} e_y^{new}(i,j,k) &= e_y^{old}(i,j,k) \\ &\quad +c[\ b_z(i-1,j,k) - b_z(i,j,k) - b_x(i,j,k-1) + b_x(i,j,k)\ ], \end{aligned} \tag{24}$$

$$\begin{aligned}\frac{\partial}{\partial t}e_z &= (e_z^{new}(i,j,k+.5) - e_z^{old}(i,j,k+.5))/\delta t \\ &= c[\ (b_y(i+.5,j,k+.5) - b_y(i-.5,j,k+.5))/\delta x \\ &\quad - (b_x(i,j+.5,k+.5) - b_x(i,j-.5,k+.5))/\delta y],\end{aligned} \tag{25}$$

$$\begin{aligned}e_z^{new}(i,j,k) = e_z^{old}(i,j,k) \\ +c[\ b_x(i,j-1,k) - b_x(i,j,k) - b_y(i-1,j,k) + b_y(i,j,k)].\end{aligned} \tag{26}$$

After updating the electric field by the circulation of the magnetic field around that Yee lattice surface, charge flux **J** are calculated and subtracted after the particles are moved later in the program.

2.4 Particle Update

Newton-Lorentz equations are already in typical "update" form. The time centered finite difference version of the Newton-Lorentz particle update is:

$$\mathbf{v}^{new} - \mathbf{v}^{old} = \frac{q\delta t}{m} < \mathbf{E} + \frac{1}{2}(\mathbf{v}^{new} + \mathbf{v}^{old}) \times \mathbf{B} > \tag{27}$$

$$\mathbf{r}^{next} - \mathbf{r}^{present} = \delta t \mathbf{v}^{new} \tag{28}$$

This shows that position must be leap-frogged over velocities. Hatree and Boris found a good physical interpretation of the steps in this explicit procedure:

[1] Half an electric acceleration:

$$\mathbf{v}_0 \longleftarrow \mathbf{v}^{old} \tag{29}$$

or

$$\mathbf{v}_0 = \mathbf{v}^{old} + q\mathbf{E}\delta t/2m \tag{30}$$

[2] Pure magnetic rotation:

$$\mathbf{v}_1 \longleftarrow \mathbf{v}_0 \tag{31}$$

or

$$\mathbf{v}_1 - \mathbf{v}_0 = (\mathbf{v}_1 + \mathbf{v}_0) \times q\mathbf{B}\delta t/2m \tag{32}$$

[3] Another half electric acceleration:

$$\mathbf{v}^{new} \longleftarrow \mathbf{v}_1 \tag{33}$$

or

$$\mathbf{v}^{new} = \mathbf{v}_1 + q\mathbf{E}\delta t/2m \tag{34}$$

The Eq. (31) determining $\mathbf{v}_1$ from $\mathbf{v}_0$ is still implicit but its explicit form follows from: (1) dotting with $\mathbf{v_1} + \mathbf{v_0}$ to check that the magnetic field does not work and that the magnitudes of $\mathbf{v}_1$ and $\mathbf{v}_0$ are the same, (2) dotting with $\mathbf{B}$ to check that components along $\mathbf{B}$ are the same, (3) crossing with $q\mathbf{B}\delta/2m$ and substituting back, then to give

$$\mathbf{v}_1 = \mathbf{v}_0 + 2 \times \frac{\mathbf{v_0} + \mathbf{v_0} \times \mathbf{b_0}}{1 + b_0^2} \times \mathbf{b}_0 \tag{35}$$

2.5 Relativistic Generalization

In the code, the particle trajectory is integrated using a time-centered leap-frog scheme. Let

$$\mathbf{u} = \mathbf{v}\ , \qquad \gamma^2 = (1 - \frac{u^2}{c^2})^{-1} \tag{36}$$

Here γ is denoted by relativistic factor. Newton-Lorentz Eq. (27) gives:

$$\mathbf{u}^{n+\frac{1}{2}} - \mathbf{u}^{n-\frac{1}{2}} = \frac{q\delta t}{m}[\mathbf{E}^n + \frac{1}{2\gamma^n}(\mathbf{u}^{n+\frac{1}{2}} + \mathbf{u}^{n-\frac{1}{2}}) \times \mathbf{B}^n] \tag{37}$$

$$\mathbf{r}^{n+1} = \mathbf{r}^n + \mathbf{v}^{n+\frac{1}{2}}\delta t = \mathbf{r}^n + \frac{\mathbf{u}^{n+\frac{1}{2}}\delta t}{\gamma^{n+\frac{1}{2}}} \tag{38}$$

where

$$(\gamma^{n+\frac{1}{2}})^2 = 1 + (\frac{u^{n+\frac{1}{2}}}{c})^2 \tag{39}$$

2.6 Force Interpretations

In Eq. (37), **E** and **B** are interpolated from the grids to the particle positions in Yee lattice. Throughout the code, linear interpolation is employed for subgrid resolution. This means that there is no stringent lower limit to the sizes of such quantities as gyroradii or Debye lengths. For quantities recorded on the integer mesh $x = i$, $y = j$, $z = k$, this means interpolating the eight nearest entries by applying weights so-called "volume" weights (Buneman, 1993). For example, the "volume" weight for (i, j, k) is $(1 - dx)(1 - dy)(1 - dz) = cx * cy * cz$ and for $(i + 1, j + 1, k + 1)$ is $dx * dy * dz$.

In Yee lattice as shown in Fig. 4, the interpolated force at (x, j, k) exerted by the electric field components e_x is denoted by $\mathbf{F}_{e_x}^{(x,j,k)}$ and expressed as follows:

$$\mathbf{F}_{e_x}^{(x,j,k)} = e_x(i, j, k) + [e_x(i + 1, j, k) - e_x(i, j, k)]\delta x, \tag{40}$$

Here

$$e_x(i, j, k) = \frac{1}{2}\{e_x(i, j, k) + e_x(i - 1, j, k)\}, \tag{41}$$

$$e_x(i + 1, j, k) = \frac{1}{2}\{e_x(i + 1, j, k) + e_x(i, j, k)\}. \tag{42}$$

In Yee lattice, please note that the electric field and magnetic field are staggered as shown in the Fig. 4. Thus we obtain

$$\begin{aligned} 2\mathbf{F}_{e_x}^{(x,j,k)} &= e_x(i, j, k) + e_x(i - 1, j, k) \\ &+ [\ e_x(i + 1, j, k) - e_x(i - 1, j, k)]\delta x. \end{aligned} \tag{43}$$

The interpolated forces exerted by e_x at $(x, j+1, k)$, $(x, j, k+1)$, and $(x, j+1, k+1)$ are

$$\begin{aligned} 2\mathbf{F}_{e_x}^{(x,j+1,k)} &= e_x(i, j + 1, k) + e_x(i - 1, j + 1, k) \\ &+ [\ e_x(i + 1, j + 1, k) - e_x(i - 1, j + 1, k)]\delta x, \end{aligned} \tag{44}$$

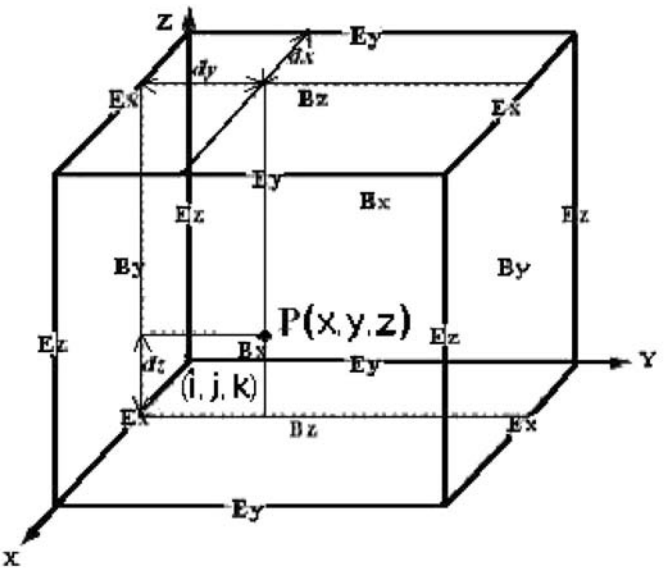

Fig. 4. The positions of field components in Yee lattice. A particle is located at the point P.

$$\begin{aligned} 2\mathbf{F}_{e_x}^{(x,j,k+1)} &= e_x(i,j,k+1) + e_x(i-1,j,k+1) \\ &+ [\, e_x(i+1,j,k+1) - e_x(i-1,j,k+1)]\delta x, \end{aligned} \tag{45}$$

and

$$\begin{aligned} 2\mathbf{F}_{e_x}^{(x,j+1,k+1)} &= e_x(i,j+1,k+1) + e_x(i-1,j+1,k+1) \\ &+ [\, e_x(i+1,j+1,k+1) - e_x(i-1,j+1,k+1)]\delta x. \end{aligned} \tag{46}$$

respectively. Thus the interpolated forces exerted by e_x at (x, y, k), $(x, y, k+1)$, and (x, y, z), are

$$\mathbf{F}_{e_x}^{(x,y,k)} = \mathbf{F}_{e_x}^{(x,j,k)} + [\mathbf{F}_{e_x}^{(x,j+1,k)} - \mathbf{F}_{e_x}^{(x,j,k)}]\delta y, \tag{47}$$

$$\mathbf{F}_{e_x}^{(x,y,k+1)} = \mathbf{F}_{e_x}^{(x,j,k+1)} + [\mathbf{F}_{e_x}^{(x,j+1,k+1)} - \mathbf{F}_{e_x}^{(x,j,k+1)}]\delta y, \tag{48}$$

and

$$\mathbf{F}_{e_x}^{(x,y,z)} = \mathbf{F}_{e_x}^{(x,y,k)} + [\mathbf{F}_{e_x}^{(x,y,k+1)} - \mathbf{F}_{e_x}^{(x,y,k)}]\delta z. \tag{49}$$

respectively. The interpolated forces $\mathbf{F}_{e_y}^{(x,y,z)}$, $\mathbf{F}_{e_z}^{(x,y,z)}$, $\mathbf{F}_{b_x}^{(x,y,z)}$, $\mathbf{F}_{b_y}^{(x,y,z)}$, and $\mathbf{F}_{b_z}^{(x,y,z)}$ exerted by the electric field components e_y, e_z, the magnetic field component b_x, b_y, and b_z can be interpolated in the same manner, respectively.

2.7 Current Deposit

As we discussed in the electric field update section, first, the change of **E** flux through a cell surface (offset grid) equals the circulation of **B** around that surface.

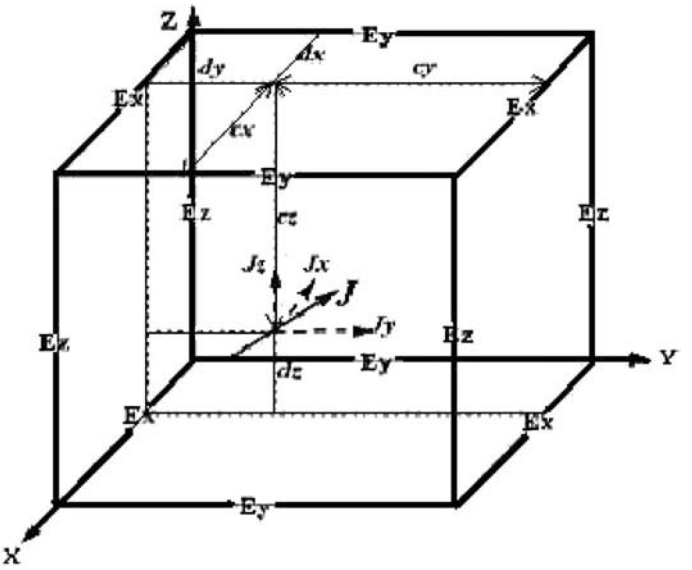

Fig. 5. The current components recorded at the point P in the Yee lattice.

Then the charge fluxes are subtracted from the **B** circulation later. TRISTAN does not employ a charge density array. Only charge fluxes, i. e., the amounts of charge flowing the faces of Yee lattice, are needed. From the Maxwell equations, one notes that Poisson equation will always be satisfied if the charge conservation condition:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{J} \tag{50}$$

is satisfied. Hence the electromagnetic field can be updated from only two curl Maxwell equations if one can enforce rigorous charge conservation numerically. A rigorous charge conservation method for current deposit is described in detail in (Villasenor and Buneman, 1992). In this scheme, one obtains the current flux through every cell surface within a time step δt by counting the amount of charge carried across the Yee lattice cell surfaces by particles as they move from $\mathbf{r^n}$ to $\mathbf{r^{n+1}}$ as shown in Fig. 5. In Yee lattice cell surfaces, the charge fluxes are subtracted from each component of **E** field as follows:

$$\begin{aligned}
e_x(i,j,k) &= e_x(i+.5,j,k) \\
&= e_x(i,j,k) - J_x * cy * cz, \\
e_x(i,j+1,k) &= e_x(i+.5,j+1,k) \\
&= e_x(i,j+1,k) - J_x * dy * cz, \\
\\
e_x(i,j,k+1) &= e_x(i+.5,j,k+1) \\
&= e_x(i,j,k+1) - J_x * cy * dz, \\
e_x(i,j+1,k+1) &= e_x(i+.5,j+1,k+1) \\
&= e_x(i,j+1,k+1) - J_x * dy * dz,
\end{aligned}$$

$$
\begin{aligned}
e_y(i,j,k) &= e_y(i,j+.5,k) \\
&= e_y(i,j,k) - J_y * cx * cz, \\
e_y(i+1,j,k) &= e_y(i+1,j+.5,k) \\
&= e_y(i+1,j,k) - J_y * dx * cz, \\
e_y(i,j,k+1) &= e_y(i,j+.5,k+1) \\
&= e_y(i,j,k+1) - J_y * cx * dz, \\
e_y(i+1,j,k+1) &= e_y(i+1,j+.5,k+1) \\
&= e_y(i+1,j,k+1) - J_y * dx * dz.
\end{aligned}
$$

and

$$
\begin{aligned}
e_z(i,j,k) &= e_z(i,j,k+.5) \\
&= e_z(i,j,k) - J_z * cy * cx, \\
e_z(i,j+1,k) &= e_z(i,j+1,k+.5) \\
&= e_z(i,j+1,k) - J_z * dy * cx, \\
e_z(i+1,j,k) &= e_z(i+1,j,k+.5) \\
&= e_z(i+1,j,k) - J_z * cy * dx, \\
e_z(i+1,j+1,k) &= e_z(i+1,j+1,k+.5) \\
&= e_z(i+1,j+1,k) - J_z * dy * dx.
\end{aligned}
$$

2.8 Position of Magnetopause

In original TRISTAN code, the Earth dipole field is located inside the simulation domain. Before running simulations, the rough size of the Earth magnetosphere should be determined and the size should be small enough to be inside the simulation domain.

As shown in Fig. 6, the Ampere equation gives

$$\mathbf{B} = \frac{\mu_0}{4\pi} \int \frac{I_0 d\mathbf{l} \times \mathbf{r}}{r^3} \tag{51}$$

where the small vector element of the ring is $d\mathbf{l} = -\mathbf{e}_\phi r_0 d\phi$, the distance from arbitrary point along the y-axis to $d\mathbf{l}$ is $r^2 = R^2 + r_0^2 - 2Rr_0^2 \cos\phi$, and the ring current is I_0. Here r_0 is the ring radius, ϕ is the ring angle. From $R^2 = r^2 + r_0^2 - 2rr_0 \cos(\alpha + \pi/2)$, we get:

$$\sin\alpha = \frac{R\cos\phi - r_0}{r}. \tag{52}$$

Thus the B_z at $(0, R, 0)$ is:

$$B_z = -\frac{\mu_0 I_0 r_0}{4\pi} \int \frac{R\cos\phi - r_0}{r^3} d\phi \tag{53}$$

when $R \gg r_0$ is assumed, it can be

$$B_z(\text{magnetopause}) \sim \frac{\mu_0 I_0 r_0^2}{2R^3} \tag{54}$$

The potential energy density of the Earth magnetic field in magnetopause is:

$$W_{magnetic} = B^2/2\mu_0 \sim B_z^2/2\mu_0 \sim \frac{\mu_0 I_0^2 r_0^4}{8R^6}. \tag{55}$$

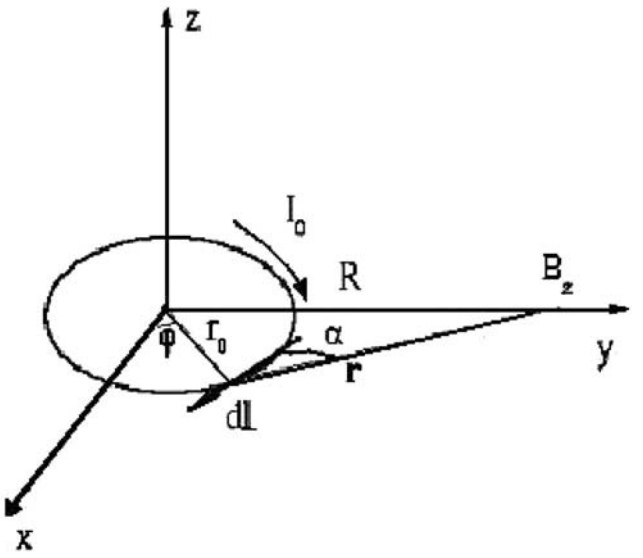

Fig. 6. The ring current to generate the dipole field of the Earth at $(0, R, 0)$.

The kinetic energy density of solar wind flow is:

$$W_{\text{solar wind}} = \frac{1}{2}\rho_{solar\ wind}V^2_{solar\ wind} \sim \frac{1}{2}(m_i D_i + m_e D_e)V^2_{\text{solar wind}}. \tag{56}$$

where D_i, D_e are the number density of ions and electron in solar wind respectively. The position of the magnetopause, R_{MP}, located where the two energy densities are equal to each other. We obtain it from Eq. (55) and Eq. (56):

$$R^6_{MP} \sim \frac{\mu_0 I_0^2 r_0^4}{4(m_i D_i + m_e D_e)V^2_{\text{solar wind}}} \tag{57}$$

In TRISTAN code, the Earth dipole filed is ramped-up linearly at an initial stage of the simulation. The half dipole field ramp-up time is $\frac{-o2}{o3} = n$, and the final ring current charge density is $o_{final} = -o3 \times n(n+1)(2n+1)/3$. The ratio -o2/o3 should be an exact integer n. The number of time steps taken for the build-up is $2n$. The final value of "o" will be $-o3n(n+1)(2n+1)/3$ in the simulation. Using Eq. (57) and the final value of "o", the rough location of the magnetopause can be estimated.

2.9 Field Densities

It is most important to realize that the three components of each field vector have not been recorded at the same places in Yee lattice. To get the electric and magnetic field components at location (i, j, k), one must form the following averages;

$$e_x(i, j, k) = \frac{1}{2}\{e_x(i-1, j, k) + e_x(i, j, k)\}, \tag{58}$$

$$e_y(i,j,k) = \frac{1}{2}\{e_y(i,j-1,k) + e_y(i,j,k)\}, \tag{59}$$

$$e_z(i,j,k) = \frac{1}{2}\{e_z(i,j,k-1) + e_z(i,j,k)\}, \tag{60}$$

$$\begin{aligned} b_x(i,j,k) = \ & \tfrac{1}{2}\{\ \tfrac{1}{2}\ [\ b_x(i,j,k) + b_x(i,j,k-1)\ ] \\ & +\tfrac{1}{2}\ [\ b_x(i,j-1,k) + b_x(i,j-1,k-1)\ ]\}, \end{aligned} \tag{61}$$

$$\begin{aligned} b_y(i,j,k) = \ & \tfrac{1}{2}\{\ \tfrac{1}{2}\ [\ b_y(i,j,k) + b_y(i-1,j,k)\ ] \\ & +\tfrac{1}{2}\ [\ b_y(i,j,k-1) + b_y(i-1,j,k-1)\ ]\}, \end{aligned} \tag{62}$$

and

$$\begin{aligned} b_z(i,j,k) = \ & \tfrac{1}{2}\{\ \tfrac{1}{2}\ [\ b_z(i,j,k) + b_z(i,j-1,k)\ ] \\ & +\tfrac{1}{2}\ [\ b_z(i-1,j,k) + b_z(i-1,j-1,k)\ ]\}. \end{aligned} \tag{63}$$

2.10 Formulae and Normalization

TRISTAN code uses scales such that $\epsilon_0 = 1$ and hence $\mu_0 = 1/c^2$ (means $\mathbf{E} = \mathbf{D}$). TRISTAN also uses scales such that time step $\delta t = 1$, grid sizes $\delta x = \delta y = \delta z = 1$, electron charge to mass ratio $q_e/m_e = -1$, electron mass $m_e = 1$. Thus the normalized ion and electron cyclotron frequencies are:

$$\Omega_e = \frac{zeB}{m_e} = \frac{B}{m_e} = B, \tag{64}$$

$$\Omega_i = \frac{zeB}{m_i} = \frac{B}{m_i} = B \times \text{rmass}. \tag{65}$$

respectively. Here the mass ration rmass $= m_e/m_i$. The normalized ion and electron gyroradii are:

$$\rho_e = \frac{m_e v_e}{zeB} = \frac{m_e v_e}{B} = \frac{v_e}{B}, \tag{66}$$

$$\rho_i = \frac{m_i v_i}{zeB} = \frac{m_i v_i}{B} = \frac{v_i}{\text{rmass}B}. \tag{67}$$

respectively. The normalized electron and ion plasma frequencies are:

$$\omega_{pe} = \sqrt{\frac{n_e z^2 e^2}{\epsilon_0 m_e}} = \sqrt{\frac{n_e}{m_e}} = \sqrt{n_e}, \tag{68}$$

$$\omega_{pi} = \sqrt{\frac{n_i z^2 e^2}{\epsilon_0 m_i}} = \sqrt{\frac{n_i}{m_i}} = \sqrt{n_i \text{rmass}}. \tag{69}$$

respectively. The normalized Alfven wave speed is:

$$V_A = \sqrt{\frac{B_0^2}{\mu_0 \rho_m}} = c\sqrt{\frac{B_0^2}{\rho_m}}. \tag{70}$$

where $\rho_m = n_e m_e + n_i m_i = m_e(n_e + \frac{n_i}{\text{rmass}})$ is the mass density. The normalized sound speed is:

$$C_s = \sqrt{\frac{\gamma_e T_e + \gamma_i T_i}{m_i}} = \sqrt{\frac{\gamma p_0}{\rho_0}} \sim \sqrt{\frac{\gamma T}{m_i}}$$

$$\sim \sqrt{\frac{T_e}{m_i}} = \sqrt{\frac{\text{rmass } T_e}{m_e}} = \sqrt{\text{rmass } T_e}. \tag{71}$$

The normalized Debye length is:

$$\lambda_{De} = \sqrt{\frac{\epsilon_0 T_e}{n_e e^2}} = \sqrt{\frac{T_e}{n_e}}. \tag{72}$$

In TRISTAN code, users have to specify the solar wind velocity $vdrft$, the thermal ion velocity $vth2i$, the thermal electron velocity $vth2e$, the particle density D_{pair}, the half dipole field ramp-up time $\frac{o2}{o3} = N$, and the final ring current charge density:

$$o_{final} = -o3 \times N(N+1)(2N+1)/3. \tag{73}$$

From Eqs. (68), (69) and (72), please note that the plasma frequency is proportional to ${n_e}^{\frac{1}{2}}$ and the Debye length is proportional to ${n_e}^{-\frac{1}{2}}$. Thus, fixing all input parameters except the particle density, if we change the particle density in the simulation to reduce the statistical noises or fluctuations, all physical quantities will vary and the physical meanings of simulation results will change at the same time. Varying the value of the particle density, we have to adjust all other physical input parameters simultaneously to keep the physical problems fixed.

2.11 Stabilities and Heating Conditions

Solving Maxwell equations by the centered difference scheme in space and by leap-frog method in time, the spatial grid $\delta x, \delta y, \delta z$ and the time step δt should satisfy the following inequality, which is called Courant condition,

$$\delta x, \delta y, \delta z > c\, \delta t \tag{74}$$

where c is the light speed and $\delta x, \delta y, \delta z$ is the grid size. The condition is easily derived from the numerical dispersion relation of the light mode.

In TRISTAN code, if we consider the real physical system size of the Earth magnetosphere, it is definitely impossible to use the grid size equal to or more than one Debye length even if we use a very powerful parallel computer. You can imagine that the typical Debye length in magnetopause is roughly an order of 10 m and how many grids will be needed to simulate the whole magnetosphere. One trade-off to solve this problem is to reduce the grid size less than one Debye length. However, we have to carefully avoid nonphysical instabilities caused by the grid or numerical grid heating (Birdsall and Langdon, 1985). For a Maxwellian velocity distribution with no drift, a rough rule of thumb is that nonphysical instability has ignorable growth for $\lambda_{De}/\delta x > \frac{1}{\pi} \sim 0.3$ for linear

weighting. However, when $\lambda_{De} \sim 0.1\delta x$, the lowest and strongest aliases interact with the steep sides of a Maxwellian velocity distribution and there is little Landau damping. The result is strong numerical instability. If $\lambda_{De}/\delta x$ decreases further the instability goes away, as it should since a cold stationary plasma is inactive. Therefore, when the Debye length λ_{De} is determined, a very rough criteria to avoid nonphysical instability is to avoid the range:

$$o(10^{-1}) \leq \frac{\lambda_{De}}{\delta x} \leq 0.3.$$

Of course, there are many other types of nonphysical instabilities and we have to check them carefully in the simulation.

3 Arrays in Original TRISTAN Code

The motivation of TRISTAN, a fully three-dimensional (3D) electromagnetic (EM) particle-in-cell (PIC) code written by Oscar Buneman and other collaborators in Stanford University, is to develop a general particle-in-cell code for space plasma simulations (Buneman, 1993). In this section, we only discuss the data structure and the data distribution over processors on the HPF TRISTAN code. For the detail physics of the PIC code in general, please refer to, for examples, (Birdsall and Langdon, 1985) and (Walker, 1991).

The data structure of TRISTAN code consists of two primitive data types. The first one is the particle data as follows:

$$x(mp), y(mp), z(mp), u(mp), v(mp), w(mp),$$

where mp= total number of particles, the positions and velocities of ions and electrons are recorded at:

$$x(1:mh), y(1:mh), z(1:mh),$$

$$u(1:mh), v(1:mh), w(1:mh),$$

and

$$x(mh+1:mp), y(mh+1:mp), z(mh+1:mp),$$

$$u(mh+1:mp), v(mh+1:mp), w(mh+1:mp).$$

respectively, where $mh = mp/2$. The second one is the grided field data expressed as the triple-indexed arrays of EM (ElectroMagnetic) fields as follows:

$$ex(i,j,k), ey(i,j,k), ez(i,j,k),$$

and

$$bx(i,j,k), by(i,j,k), bz(i,j,k).$$

The original TRISTAN code uses "COMMON" block clause to save and transfer fields data between subroutines in the **MOVER** (push particles) and **DEPOSIT** (deposit current data to the field grids) subroutine calls. Meanwhile

in the subroutines that processes the surfaces and edges of the grid data, the filed data are transferred by dummy arrays in the original code. In both of these subroutines, the field arrays are treated as single-indexed. On the other hand, triple-indexed field arrays are employed in the field solver subroutines. In the code, single-indexed arrays are converted automatically to the triple-indexed arrays when they passed over two subroutines.

Converting a serial Fortran program to a HPF program, we have to stress two points that are very important for rewriting TRISTAN in HPF: [**1**] The "COMMON" statement is restricted as suggested by pgHPF user guide and there they indicated 'We strongly recommended that programmers writing new F90 code use features like "MODULE"... to avoid the use of "COMMON"...' (Koelbel, et al., 1994, Foster, 1995), in case of data overlapping, and substituted it by "MODULE" block; [**2**] To control the communications, all the arrays are treated as fixed indexes throughout the whole program. We control the communication parts using both the "field manager" and "particle manager" (Decyk, 1995).

4 Field Data Domain Decompositions

The field data are decomposed over sub-domains of that number is equal to the number of the processors used in the simulation as indicated in Fig. 1. In processing the current deposition that is so-called the scatter part of the computations, to avoid large transients or variations of currents, TRISTAN uses a "smoother" that has 27 different weights, smoothing the current deposition. In **DEPOSIT** subroutine the smoothing is performed as follows:

$$\begin{aligned} ey(i+smx+1, j+smy, k+smz+1, Np) = \\ ey(i+smx+1, j+smy, k+smz+1, Np) \\ -sv*dz*dx-ss \end{aligned} \tag{75}$$

where $smx = -1:1, smy = -1:1, smz = -1:1$,$sv = sm(smx, smy, smz, Np)*qv$ and $ss = sm(amx, amy, amz, Np)*delt$. For details, see p. 73 and p. 321 of Lecture note by Buneman (1993). (Note that one dimensional array for ex, ey, ez is used.) Therefore, the current deposition of one particle will be related to three grids in each dimension, where one of them are at the backward grid and another at the forward grids in each dimension.

In the "MODULE" block, the field arrays are written in HPF directives as follows:

$$\mathbf{REAL}, \mathbf{DIMENSION}(nx, j, k, Np) :: ex, ey, ez$$

$$\mathbf{REAL}, \mathbf{DIMENSION}(nx, j, k, Np) :: bx, by, bz$$

where Np is the number of processor, $nx = i/Np + 3$ (here assuming i/Np is not necessarily equal to be integer exactly) keeping one guard cell in the left (backward) and right (forward) sides of the sub-domains in the domain-decomposition direction (i.e., in the solar-magnetotail direction). Here the indices i, j and k correspond to the numbers of field grids in x, y and z directions,

respectively. Using the HPF directive "DISTRIBUTE", we, respectively, map the sub-domains to each processor on a distributed parallel computer:

DISTRIBUTE(*,*,*,BLOCK) ONTO Np:: ex,ey,ez

DISTRIBUTE(*,*,*,BLOCK) ONTO Np:: bx,by,bz

In order to separate the communication parts from the computation parts, each sub-domain keeps extra cells, the so-called guard or ghost cells, that store the field data information in the first and last grids of that sub-domain in the decomposition direction. Fig. 1 illustrates this concept of the data mapping over the sub-domains or processors. Here the communications are required after updating the field data every time step. In the field manager (Decyk, 1995), the data sent to the neighbor processors are packed in the working arrays: $Cex(1, j, k, Np)$, $Cey(1, j, k, Np)$, and $Cez(1, j, k, Np)$, before they are sent to the neighbor sub-domains. Thus the field data communications are performed by the pgHPF $CSHIFT$ construct after the data are packed in the working arrays. The followings are the related parts of the HPF programs in the field manager (Decyk, 1995):

```
Cex(1,:,:,:)=ex(2,:,:,:)

Cex=CSHIFT(Cex,+1,4)

ex(nx-1,:,:,:)=Cex(1,:,:,:)

...
```

5 Particle Data Domain Decompositions

The particle data can be written in HPF directives as follows:

$$\mathbf{REAL}, \mathbf{DIMENSION}(m, Np) :: xe, ye, ze, xi, yi, zi$$

$$\mathbf{REAL}, \mathbf{DIMENSION}(m, Np) :: ue, ve, we, ui, vi, wi$$

where i and e, respectively, stand for ion and electron, the number m is the array size in each sub-domain. To ensure that the enough space are reserved to store the particle data due to the load-imbalance, m must be 10-30 % larger than the average number of particles. The number Np is the number of processors, and is the index used in the HPF "DISTRIBUTE" directive. As the particles move in time in the simulations, the physical position of some particles may cross the sub-domain boundaries, and move to the neighbor sub-domains. When a particle moves from one sub-domain to another, the data of the particle left the sub-domain must be sent to the appropriate neighbor processor at every time step. Before updating and sending the particle data, we have to sort the particles that should be sent to another sub-domain, and pack them in the working

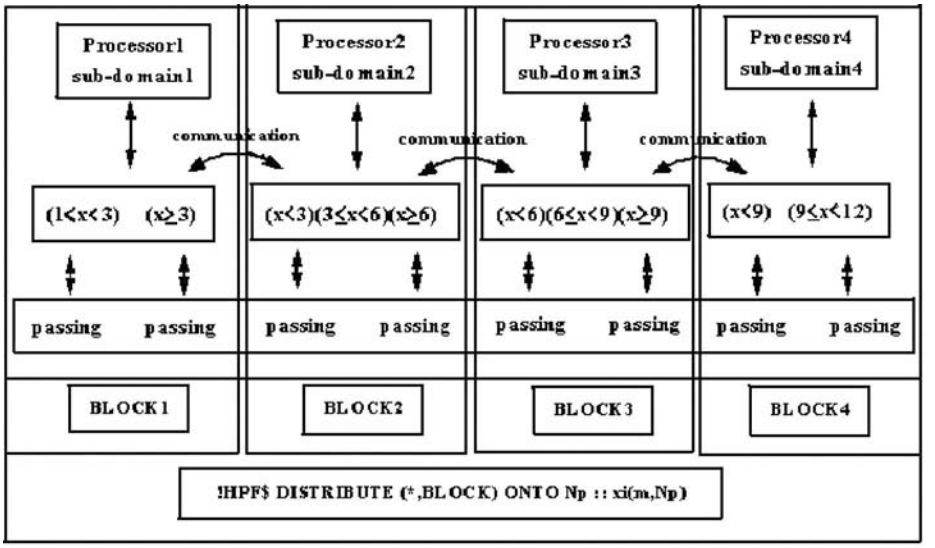

Fig. 7. Diagram of particle array decompositions and communications, with processor number $Np = 4$, grid number in decomposition direction=12.

arrays: $CRi(:, Np)$, $CLi(:, Np)$, $CRe(:, Np)$, and $CLe(:, Np)$. The number of the ions and electrons sent in right and left are denoted by the arrays $ionspsR(Np)$, $ionspsL(Np)$, $lecspsR(Np)$, and $lecspsL(Np)$, respectively. In our HPF TRISTAN code, we send both the packed arrays and their particles number arrays to the neighbor sub-domains as follows:

```
CRi=CSHIFT(CRi, -1,2)

ionspsR=CSHIFT(ionspsR, -1)

...
```

Fig. 7 shows the example of the particle data distributions and communications. After both the particle numbers and the packed working arrays are sent and received by each appropriate processors, the received particles are sorted and put into the appropriate part of the particle arrays in that sub-domain. The communications and sorting of these particles are performed in the particle manager (Decyk, 1995) as shown in Fig. 2.

6 Programming Comments on HPF Communications in PC Cluster

For benchmark purpose of HPF TRISTAN, a dual PentiumPro PC cluster consists of 16 PCs and each PC have dual 200MHz PentinumPros with 128MB EDD

DIMM memories. The PCs in the PC cluster system are hooked through 100 Base-T ethernet with 100 Base-T switching Hub. Redhat Linux version 4.1 was used as their operating systems. The pgHPF compiler version 1.7 was installed for HPF computations.

One of the most difficult problems in our HPF TRISTAN code is the communication programming, especially, the determination of the buffer sizes which is used to pack the data sent to the neighbor processors. Of course, we can define a buffer size large enough to send the particle or grid data to neighbor processors at one time. However, as shown in Fig. 8, our experience shows that when the buffer sizes become larger than some critical values, in this case 1456 bytes in our PC cluster system, the communication suddenly becomes unstable, and the communication times suddenly jump up to 5 to 8 times larger than those less than the critical value 1456 bytes. As indicated in the Fig. 8, the communication times when the buffer size go beyond 1456 bytes are not uniquely determined and rather indeterministic. In order to avoid the sudden communication slow-down, we have to carefully choose the buffer size. We have to split the particles or grids data into smaller pieces of buffers, pack the smaller data, and send the data to the neighbor processors one by one. Thus we can avoid the large slow-down of the simulations in this system due to the unstable HPF communications. In our HPF TRISTAN code, the buffer sizes can be varied and can be set without modifying the program. We can first evaluate the best buffer size and run the simulations. The best buffer size can be chosen as indicated in Fig. 8. For ex-

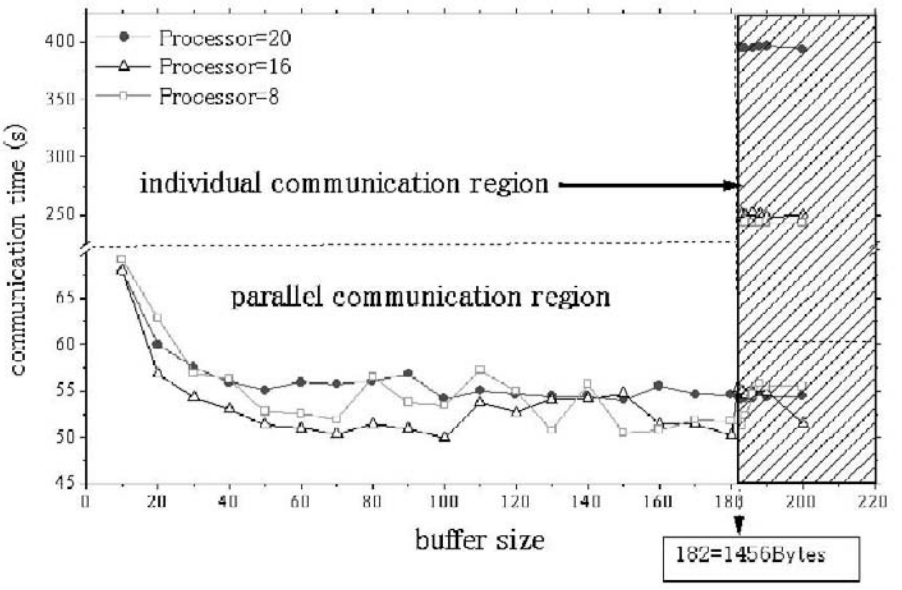

Fig. 8. Buffer sizes and CSHIFT communication times in HPF.

amples, in this figure, the best buffer sizes can be chosen between 640 to 1400 bytes.

The reasons for performance degradation in communications with this longer packets than 1456 bytes are not yet investigated in detail. One possibility of this degradation is due to MTU of the ethernet. MTU is the Maximum Transmission Unit that IP is allowed to use for a particular interface. If your MTU is set too big, in this case beyond ∼1456 bytes your packets must be fragmented, or broken up, by a switching hub along the path to the other PCs. This may result in a drastic decrease in throughput. However, we have not identified the source of this communication degradation. We would like to leave this investigation to our future research.

7 Benchmark and Simulation Results

In Table 1, the parameter $\varepsilon_{eff-grid}$ is defined as:

$$\varepsilon_{eff-grid} = \frac{Num_{total\ grid} - Num_{total\ guard\ cell}}{Num_{total\ grid}}.$$

where $Num_{total\ grid}$ is the total grid number in decomposition direction, and $Num_{total\ guard\ cell}$ is the total number of guard cell. Table 1 shows the total times, speedups and parallel efficiency vs the number of processors. The total

Table 1. Benchmark resluts with time step=100, particle number =1200,000, and grid number $=185 \times 65 \times 65$.

Procs	Time(s)	speedup S_p	efficiency $\varepsilon(\%)$	$\varepsilon_{eff-grid}(\%)$
1	4836	1.0	100.	100
2	3706	1.3	65.2	96.9
3	2416	2.0	66.7	95.4
4	1769	2.7	68.4	93.9
5	1457	3.3	66.3	92.5
6	1195	4.0	67.4	91.1
7	1034	4.7	66.8	89.8
8	937.4	5.2	64.5	88.5
9	881.9	5.5	60.9	87.2
10	762.7	6.3	63.4	86.0
11	713.0	6.8	61.7	84.9
12	652.0	7.4	61.8	83.7
13	616.8	7.8	60.3	82.6
14	575.9	8.4	60.0	81.5
15	572.5	8.4	58.3	80.4
16	516.4	9.4	58.5	79.4
17	497.2	9.7	57.2	78.4
18	470.5	10.3	57.1	77.4
19	453.5	10.7	56.1	76.4
20	426.6	11.3	56.7	75.5

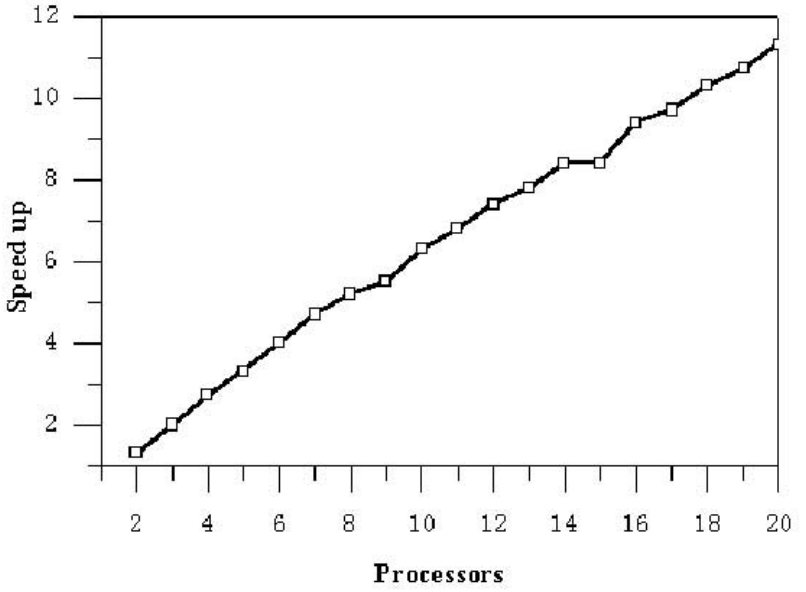

Fig. 9. Speed up vs processor number.

computation time of single processor was measured by the original version of TRISTAN code compiled by $pgf77$ compiler with the optimization level **-O2** option. Figure 9 shows the speed up vs processor number.

With fixing the problem size and increasing the processor number, the grid number in one sub-domain is reduced gradually. For example, 40 extra ghost grid cells in total must be added to each sub-domains in decomposition direction or in x for 20 processors. It is about 25 percents of the total grid number in decomposition direction in this case. Thus the communication overhead become insignificant comparing with the total PIC computation time as we increase the number of the processors. The increase of the communication overhead reduces the parallel efficiency in the table. If the communication overhead is insignificant, it is very hard to improve the parallel efficiency of the code without varying the problem size. However, even the most advanced parallel computer nowadays, it is not so easy to increase the problem sizes as we increase the number of the processors due to the large data sizes we have to store in each simulation run. Thus the optimal parallel efficiency of the scalable relation between the problem sizes and the number of processors are difficult to be measured in our simulation. However, Fig. 9 shows the high linearity of our HPF TRISTAN code and the code scales well. In addition, with the HPF compiler overhead and the load-imbalance overhead due to the Earth dipole field, the parallel efficiency around 60-65 % is affordable in this type of large scale simulations.

PIC simulations exploring the solar wind-magnetosphere interaction with this HPF code were accomplished on the PentiumPro PC cluster (Cai and Lu, 1999). After measuring the communication efficiencies via different processor

Table 2. Simulation parameters of case 1.

grids	205x165x165
initial ion-electron pairs	500,000,000
Light speed	0.5
ϵ_0	1.0
m_i / m_e	16
q_e / m_e	1.0
Δx	1
Δt	1
electron temperature T_e	0.004
ions temperature T_i	0.00025
solar-wind speed	0.25
IMF	no
ω_{pe}	0.2
$\lambda_D/\Delta x$	0.4
plasma parameters	g$\sim$2.8

numbers, the CFSHIFT function and its communication have been optimized by splitting a large data into many small size ones. So that a high performance communication is achieved. We also have run the code on Hitachi supercomputer SR2201 and Fujitsu VPP 5000 etc..

In case 1, we use a 205 by 165 by 165 grid and 50,000,000 plasma particles. The other parameters are listed in table 2: (1) the center of the loop current is located at (100Δ, 82.5Δ, 82.5Δ); (2) the solar-wind drift velocity is 0.5c, where c is the light speed; (3) $m_i/m_e = 16$; (4) use 8 particles per grid cell in average; (5) $T_e = 0.004$ and $T_i = 0.00025$; (6) the plasma parameter is about 2.8. Figure 10 (a) and (b) show the ion density profiles on the XZ and XY planes at time step 1500 respectively. It is clear that the complete configuration of magnetosphere, including bow shocks, magnetopause, magnetosheath, magnetotail, plasma sheet, and polar cusp are generated. In this case, more basic kinetic behaviors of space plasma in the magnetosphere have been investigated including a time-varying IMF. We use a northward IMF ($B_z = 0.01$) for an initial condition. At time step 500, the IMF is switched into southward ($B_z = -0.02$) at the sunward boundary of the domain. It is shown that after the arrival of a southwad IMF, due to the reconnection at dayside magnetopause the convection pattern across the entire polar cap begins to change in a few minutes. In contrast, the response of the equatorward motion of the open-closed field-line boundary that depends on the local time is delayed about 20 minutes relative to the onset of the reconnection at the dayside magnetopause. This time delay is considered as the

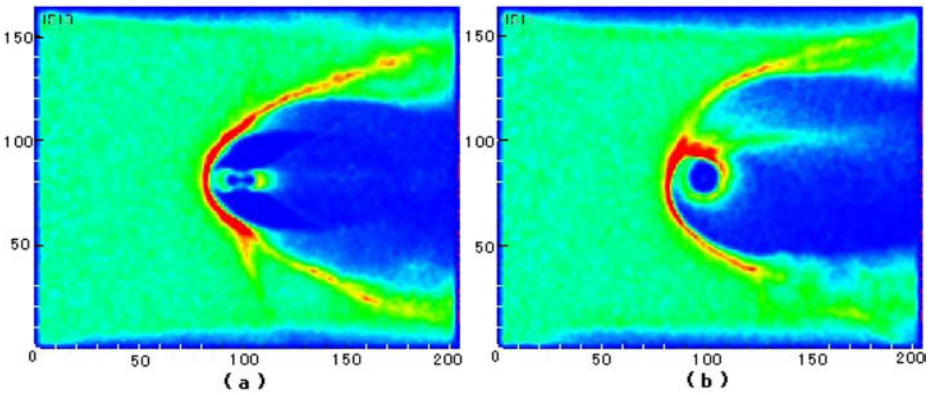

Fig. 10. The ion density profiles at (a) XZ and (b) XY planes of case 1.

Table 3. Simulation parameters of case 2.

grids	185x125x125
initial ion-electron pairs	24,000,000
Light speed	0.5
ϵ_0	1.0
m_i/m_e	16
q_e/m_e	1.0
Δx	1
Δt	1
electron temperature T_e	0.4
ions temperature T_i	0.1
solar wind speed	0.25
IMF	yes
ω_{pe}	0.2
$\lambda_D/\Delta x$	0.4
plasma parameter	g∼3

time required to convect the newly merged flux from the dayside magnetopause to the nightside inner magnetosphere.

In case 2, we use a 185 by 125 by 125 grid and 24,000,000 total particles. The other parameters are list in Table 3. It is shown in Fig. 11 that when the southward IMF arrived, the magnetic field of magnetosphere is modified. Some structures are formed. The nature of these structures is still under investigation.

We investigate the relationship between the IMF and the particle flux in polar region in case 3, in which we used a 85 by 105 by 105 grid and 3,500,000 paired particles. The other parameters are listed in Table 4. It is assumed that the cusp region is located at $25 < x < 35$, $45 < y < 60$, $56 < z < 60$. The IMF is initially zero, and switch on three times, i.e., $B_z = -0.01$ during time step 100-120, $B_z = -0.02$ during time step 600-620 and $B_z = -0.005$ during time step 900-920. The ion density profile in cusp region via time step was shown

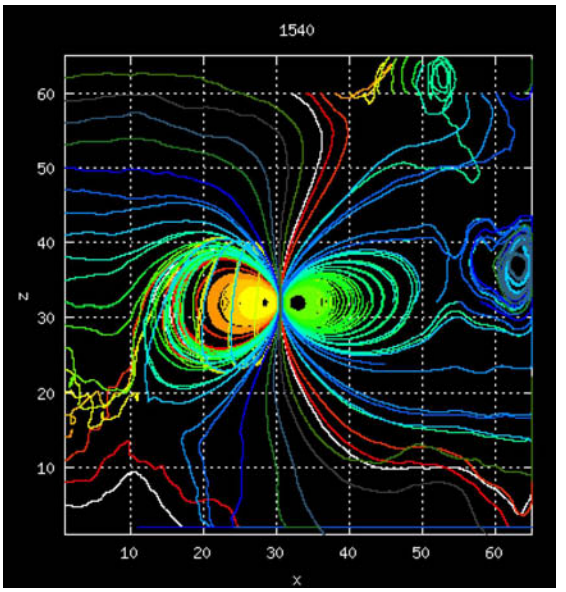

Fig. 11. Magnetic field lines at time step 1540 of case 2.

Table 4. Simulation parameters of case 3.

grids	85x105x105
initial ion-electron pairs	3,500,000
light speed	0.5
ϵ_0	1.0
m_i/m_e	16
q_e/m_e	0.5
Δx	1
Δt	1
ion temperature T_i	5.9×10^{-6}
electron temperature T_e	4.8×10^{-5}
solar wind speed	0.25
IMF	yes
ω_{pe}	0.088
$\lambda_D/\Delta x$	0.93
plasma parameter	g∼3.2

in Fig. 12. The mean ion velocities in cusp region via time step was shown in Fig. 13. The total thermal energy in cusp region via time step was shown in Fig. 14. Total particle energy in cusp region via time step was shown in Fig. 15.

It is clearly shown that the first and second switch-on of the southward IMF can cause particle flowing into cusp regions after some time steps. Many particles are entered into cusp regions. It seems that not all southward IMFs can cause

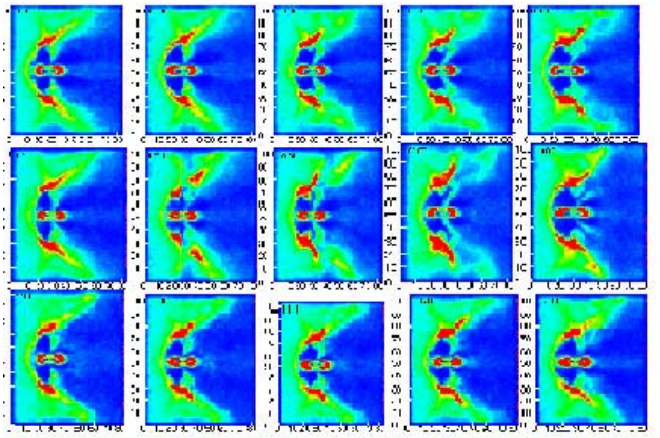

Fig. 12. The ions density profile via time step in the cusp region of case 3.

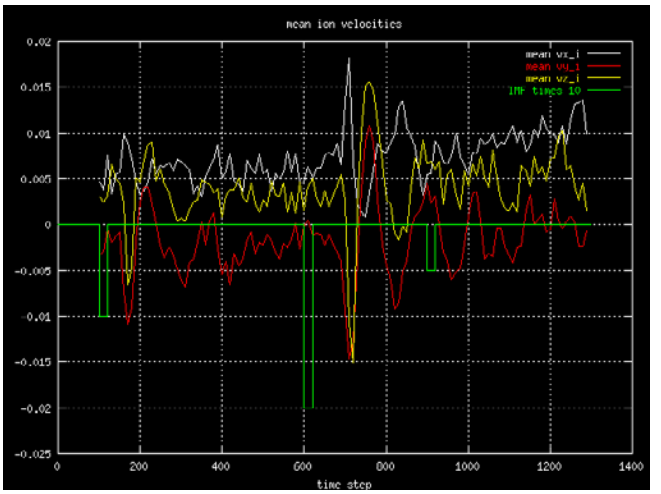

Fig. 13. Mean ion velocity and IMF via time step in cusp region of case 3.

the particle-entry into cusp regions. Only those IMFs that are strong enough can cause particle-entry into cusp regions.

The figures show that in the first two IMFs switch-on, the total particle number and the mean ion velocities decreased from the local maximum to the local minimum after the southward IMF switch-on. The intense southward IMF could push the particles into cusp region to magnetotail-ward directly. This may be related to substorm or magnetic reconnection.

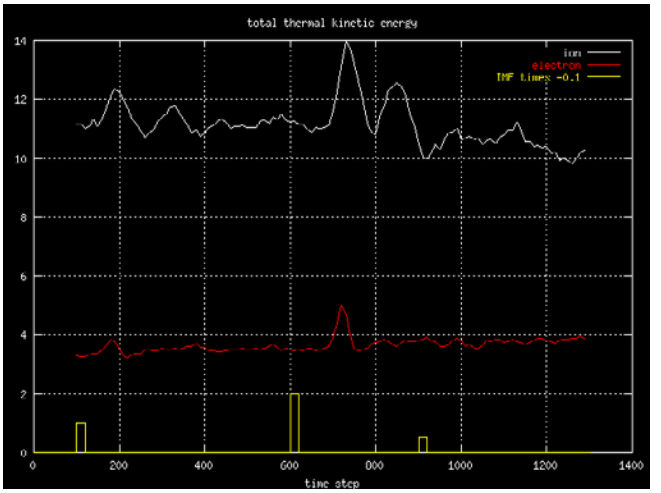

Fig. 14. Total thermal kinetic energy and IMF via time step in cusp region of case 3.

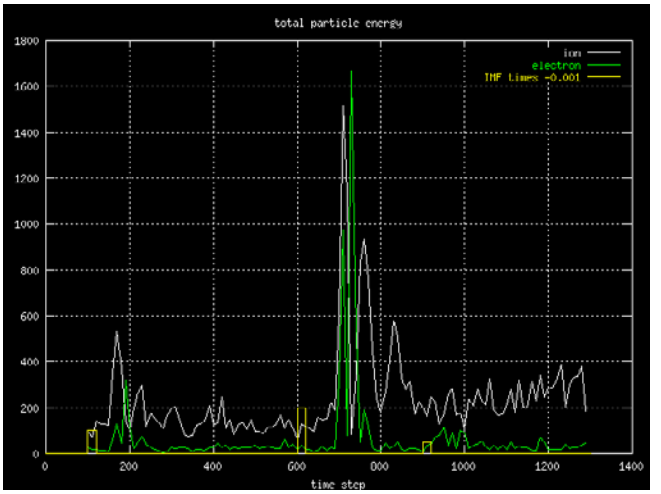

Fig. 15. Total particle energy and IMF via time step in cusp region of case 3.

8 Concluding Remarks

In the present paper, we have successfully parallelized the three-dimensional full electromagnetic and full particle code using HPF. The code is originally the same as the TRISTAN code and the code is for the space plasma simulations. As shown in Fig. 9 and Table 1, fixing the problem size, our HPF TRISTAN code has a high linearity and scales well. However, our HPF code introduces about 70% overhead and the reason for this overhead is not yet investigated. We have also parallelized the three-dimensional skeleton-PIC code introduced by V. K. Decyk (Decyk, 1995) in the same parallel algorithm (Liewer and Decyk,

1985, Decyk, 1995) using HPF. The HPF three-dimensional skeleton-PIC code introduces about 20% overheads (Cai, et al., 1999). One possibility to explain the larger overheads in our HPF TRISTAN code over the HPF skeleton-PIC code is that there are more complicated data structures in HPF TRISTAN code than those in the skeleton-PIC code. Our PCs in the cluster have no enough memory and this may degrade the performance of the PCs. Another possibility is the load-imbalance originated in the TRISTAN code as we discussed previously. Our HPF TRISTAN code has the Earth dipole filed which generates and simulates the Earth magnetosphere in one of sub-domains, and this may cause a large load-imbalance. We would like to leave the detailed investigation to our future work.

The parallelization algorithm we used in our HPF TRISTAN code is basically the same as (Liewer and Decyk, 1985) and (Decyk, 1995). We separate the communication parts from the computation parts. Thus the code can easily be converted to MPI or PVM code by replacing the HPF "CSHIFT" constructs to appropriate message passing interfaces. Our experiences show that the utilization of HPF "FORALL" or "DO INDEPENDENT" constructs in the data-parallel manner without separating the communication parts from the computation parts results in almost no gain of speedups or very poor speedups.

We have also compared the HPF skeleton-PIC code with the MPI or PVM skeleton-PIC code. The HPF code degradation of the total CPU time over the MPI or PVM code is only 10-15 % (Cai, et al., 1999) in this case. Thus we expect that we should be able to enjoy the easier HPF programming with a very small performance degradation even in the more complicate codes like the TRISTAN code.

Acknowledgment

The authors thank Professor Viktor K. Decyk for his help using the skeleton PIC codes. The authors also thank Dr. Bertrand Lembege for invaluable comments regarding to this work.

References

1. Birdsall C. K. and A.B.Langdon, Plasma Physics via Computer Simulation, McGraw-Hill, New York, (1985)
2. Buneman O., TRISTAN. In: Matsumoto H., and Omura, Y. (eds.): Computer Space Plasma Physics: Simulation Techniques and Software. Terra Scientific, Tokyo, (1993) 67-84
3. Cai, D., Q. M. Lu, and Y. T. Li, Scalability in Particle-in-Cell code using both PVM and OpenMP on PC Cluster, *Proceedings of 3rd Workshop on Advanced Parallel Processing Technologies* (1999) 69-73
4. Decyk, V. K.: Skeleton PIC Codes for Parallel Computers, Comput. Physcs. Comm. **87**, (1995) 87-94
5. Foster, I.: Designing and Building Parallel Programs, Addison-Wesley, (1995)
6. Koelbel, C.H. et al., The High Performance Fortran Handbook, The MIT press,(1994)

7. Liewer, P. C. and V. K. Decyk: A General Concurrent Algorithm for Plasma Particle-in-Cell Simulation Codes, J. Comput. Phys. **85**, (1985) 302-322
8. Homepage of Portland Group Inc.: http://www.pgroup.com
9. Villasenor, J., and O. Buneman: Rigorous Charge Conservation for Local Electromagnetic Field Solvers, Comput. Physcs. Comm. **69**, (1992) 306-316
10. Walker D. W., Particle-in-cell Plasma Simulation Codes on the Connection Machine, Computing Systems in Engineering, **1** (1991) 307-319
11. Yee, K. S, Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media, IEEE Trans. Antennas Propagat. **14**, (1966) 302-307