Parallel Particle-In-Cell Code and its Application to Virtual Magnetosphere

Weifeng Tao

Graduate School of Systems and Information

Engineering

University of Tsukuba

November 2008

Abstract

Computer simulations play a very important role in the space weather research field. Compared to global MHD simulation, which is more inclined to treat large scale system behaviors and establish quantitative global modeling, global particle simulation provides a more accurate treatment of many local and quasi-local processes. If the global particle simulation can be modeled at quantitative level, it will clarify more clearly the coupling between macroscopic phenomena and microscopic phenomena. The result will contribute greatly to quantitative evaluation of space weather phenomena and help to form the first step toward the realization of numerical space weather prediction. But, as the large-scale global particle simulation had been named as one of the "Great Challenge" problem, which consumes great computer power and memory by their nature, implementation and analysis of the global particle simulation at quantitative level is a very challenge work. With the rapid improvement of parallel systems in recent years, especially the great improvement of Grid computing, more and more high performance compute resources are available to researchers, implementation of the global particle simulation at quantitative level is become feasible from now on.

In this research, first, the isoefficiency analysis and benchmark of two three-dimensional (3D) PIC codes are implemented on two computational grids, namely supercomputer based Grid and cluster based Grid. These two PIC codes, which are named as skeleton PIC code and parallel TRISTAN PIC code, have different physics and entirely different communication patterns. From the results we can confirm that parallel TRISTAN PIC code (without all-to-all collective communications) is more scalable than the parallel skeleton PIC code (with all-to-all collective communication). We also prove that it is feasible to implement the parallel TRISTAN PIC code at quantitative spatial and temporal level with 250 processors at efficiency of 0.70.

Second, two global particle simulations using 3D EMPM (ElectroMagnetic Particle Mode) code are implemented, they are "sash" simulation with turning IMF (Interplanetary Magnetic Field) from Northward to Southward, and "bifurcation and hysteresis of the magnetospheric structure" simulation with varying Southward IMF. The sash simulation shows that weak magnetic field band sashes form at high latitude of the cusp (in each hemisphere) for a northward IMF $\boldsymbol{B_{IMF}}$. After the IMF $\boldsymbol{B_{IMF}}$ rotated from northward to duskward, then to southward. The sashes rotate out of the pole-ward direction to the equatorial plane. In the second simulation, it shows that the solar wind magnetic field and the magnetospheric surface magnetic field gives

"subcritical" type bifurcation as the southward IMF increases and exceeds a critical value, when the southward IMF decreases the magnetopause recovers its initial state without following the same path used as the increasing step: hysteresis effect is evidenced within a certain interval of IMF $|B_z|$ values. These simulations validate the feasibility of our 3D EMPM code from the point view of physics, it should help to implement the 3D EMPM code at quantitative spatial and temporal level on the next step.

Third, the developed virtual satellite system using OpenDx virtualization toolkits shows the feasibility of compares the results with the real satellite observation. Tree types of interaction between satellite and simulation data has been mimicked, they are moving satellite in continues simulation data, moving satellite in static simulation data, and resting satellite in continues simulation data. If a quantitative level global particle simulation can be performed, the virtual satellite visualization system can be used to compare with the results of satellite observation and perform real quantitative level data assimilation to the coming new space missions.

Contents

List of Figures

List of Tables

Chapter 1: General Introduction

1.1 Computer Simulation Methods for Plasma

Plasma is frequently characterized as the fourth state of matter. It is by far the most common phase of matter in the universe, both by mass and by volume: from stars to the sun, the Earth's magnetosphere, and the terrestrial experiments attempting to harness the power of thermonuclear fusion [Gurnett, 2005]. As in most fields of science, the traditional approaches to studying the properties of plasmas have involved experiment and observation on the one hand and analytical techniques based on fundamental physical laws on the other. Progress comes from interplay between these approaches: one probes nature through experiments, and the results are used to confirm or disprove the theoretical expectations.

The explosive growth in the power of computers over the past half century has led to the development of a third approach to the study of science in general and plasma physics in particular: computational physics. The impact of this alternative approach has been particularly prominent in plasma physics. The basics laws governing plasma behavior, namely the laws of Newton and Maxwell, are well known, but the consequences of these laws for a complex system consisting of perhaps 10^{20} or more particles are frequently impossible to determine [Buchner, 2003]. Likewise, the necessary experiments may be difficult and expensive to perform, either because the plasma is located far away and must thus be probed via satellites. Given these difficulties, computer simulation of plasmas has frequently proven to be a more practical means to make progress.

Dependent on the system of equation to be solved, plasma simulation can be divided are broadly grouped into three main methods: Magnetohydrodynamics simulation, particle simulation, and hybrid simulation. In magnetohydrodynamics, the plasma is approximated as a fluid, and Navier-Stokes equation (in which the electromagnetic force is included) is used. Normally, since the phenomena under study display slow temporal variation, an approximation is used in which the displacement current term is ignored in the Maxwell equations. Particle simulation is performed by solving kinetic equations for the particles that constitute the plasma. Usually, by using the super-particle the kinetic effect can be investigated in a simulation with a much smaller number of particles than real plasma, this method is referred to as “particle-in-cell” (PIC) method. Even though, particle simulation requires much more computer memory and calculation time than in magnetohydrodynamics simulation. Further, there is an intermediate method between these two methods, named as “hybrid simulation” [Birdsall and Langdon, 1985]. In this method, heavy ions are treated as particles and light electrons are treated as fluid. Since the electrons are treated as a fluid, this method has the advantage that small scale temporal or spatial variations (such as plasma oscillation of electrons and the electron Debye length) do not need to be taken into account. Therefore, a time step and a grid can be large relative particle simulation. On the other hand, this method cannot handle the electron effect on scales equivalent to the ion Larmor radius or smaller.

Proper use of these various simulation methods is dependent mainly on the time scale and spatial scale of the plasma problem to be handled. Magnetohydrodynamics simulation is used for phenomena on spatial scales larger than that of the ion Larmor radius and on time scales longer than the ion cyclotron period. On the other hand, particle simulation is appreciable to phenomena on any scale but requires a huge amount of computation. Therefore, it is often the case that the phenomena on macroscopic scale are treated using magnetohydrodynamics simulation while particle simulation is applied to phenomena on a microscopic scale. In the case where particle emphasis is placed on phenomena on a scale comparable to the ions Larmor radius, the hybrid code is preferred.

1.2 Space Weather and Global Particle Simulation

The density of the air particles responds sensitively to the solar activity, season, longitude, latitude, local time and magnetic storm conditions. Besides emitting a continuous stream of plasma called the solar wind, the sun periodically releases billions of tons of matter, called coronal mass ejections (CME). These immense clouds of material, when directed towards the Earth, can cause the upper atmosphere to heat and expand and large magnetic storms in the magnetosphere and the upper atmosphere. Magnetic storms can produce energy more larger than that released by the atomic bomb that leveled in Hiroshima in 1945. The energetic solar outputs increase dramatically during periods of high solar activity. If magnetic activity is also triggered at the Earth, intense currents flowing through the upper atmosphere, and energy deposited by high speed particles at high latitudes dramatically increases the heating and expansion of the atmosphere in these regions. In the Northern Hemisphere, they usually occur when the IMF (Interplanetary Magnetic Field), the solar wind magnetic field is directed southward. This orientation is opposite to the Earth magnetic field on the dayside boundary of the Earth's magnetosphere (which points northward), so that the Earth magnetic field becomes interconnected with the IMF. This acts like a switch, allowing much more solar wind energy to enter the magnetosphere. In the meantime, magnetic storms produce many noticeable effects on and near the Earth, such as: aurora borealis; the northern lights; aurora australis; the southern lights; radio and television interference; wireless communication scintillation, intervention and interruption, telecommunication cable disruption; some of them are hazardous to orbiting astronauts, spacecrafts and the international space station; and current surges in power lines [Buneman 1995].

These phenomena are often referred as the “space weather” problems, in consideration of its effect on human life. Generally phenomena produced by the space plasma are complicated, and provide rich sources of nonlinear phenomena for study. For example, both magnetic storms and substorms are also diverse in generated processes, regions, characteristic length scales, characteristic time scales, and coupling processes that the causal relationship between each process remains poorly understood in many ways. In particular, macroscopic phenomena (with characteristic lengths comparable to the radius of the Earth or larger) and microscopic phenomena (with characteristic lengths of a few meters) are known to interact in cases where it is difficult to separate cause from effect clearly [Kivelson, 1995]. To understand such phenomena, computer simulation has proven extremely effective and has been used to produce results. If the coupling between macroscopic phenomena and microscopic phenomena can be clarified, it will contribute greatly to quantitative evaluation of space weather phenomena and form the first step toward the realization of numerical forecast.

Global MHD simulation model is more amendable to treat large scale properties and has been refined to establish quantitative global modeling in comparison with observations, while global particle model provides a more accurate treatment of many local and quasi-local processes. As particle model, by their nature, is greatly consume the computer resource, it is now still in the predictive modeling level instead of quantitative level. In our laboratory, a long-term effort of represent the interaction between the solar wind and the Earth's magnetosphere by using a three-dimensional EMPM (ElectroMagnetic Particle Model) has been performed [Buneman 1993]. If a global simulation can be modeled at a quantitative level, it will be great impact on the quantitative evaluation of space weather phenomena and future satellite missions. It becomes more feasible as the power and speed of parallel systems, especially Grid computing, have improved in recent years.

1.3 Grid Computing for Particle Simulation

Grid computing has been developed in middle 1990s based on distributed computing technology. It refers to an infrastructure that enables the integrated, collaborative use of high-end computers, networks, databases, and scientific instruments owned and managed by multiple organizations[Foster, 1999-a, 2001]. A computational Grid is a Grid focuses primarily on computationally-intensive operations. Here, we mainly focus on the computational Grid for our benchmark purpose. Although the computational Grid had been a popular topic for many years, there still lack of enough relatively stable and open Grid like TeraGrid for researchers. As low-cost and high performance cluster systems increase their availability and maturity in recent computing world, supercomputers are no longer the only choice for scientists to run a large-scale simulation. On the other hand, the computational Grid promise to extend our capacity to use more computational resources distributed over a wide area network. A cluster based computational Grid, like a campus Grid, seems to be a good choice for researchers who need large numbers of computer resources.

The computational Grids could be a way to satisfy the high demand of computing power for many scientific and engineering codes like PIC simulations. However, it is not clear whether a code can take advantage of the enhanced degree of parallelism or if the low network performance is a severe limiting factor. Especially, the impact of the global communications and nearest neighbor communications used in the code on the scalability of the codes in computational Grid with low and high communication performance is not clear.

Since PIC codes are extremely intensive in memory and computation by their nature, the PIC code applications are now at a threshold where they can be used for precision predictions rather than for quantitative indicators of system behaviors. We had successfully carried out the parallel High Performance FORTRAN (HPF) [Cai, 2003], OpenMP/PVM and parallel Java PIC code on various kinds of parallel architectures such as vector supercomputers, scalar supercomputers and workstation based clusters. Thus, it is natural to carry out also a parallel PIC simulation on computational Grid architectures. Two different computational Grids are used in this research, namely, a supercomputer based computational Grid and a cluster based computational Grid. In order to investigate the communication patterns on the codes in various computational Grids with different communication patterns, two PIC codes are chosen as examples of both categories of codes and therefore offer a good set for a case study that can investigate the possibility of the computational Grid for large scale computations like PIC simulations.

Before we perform the PIC code on the computational Grids, understanding its isoefficiency on computational Gird is a very important precondition. The major purpose of the isoefficiency analysis is the evaluation of a relationship between an algorithm and an parallel system on which the algorithm will be implemented [Kumar, 94]. It can test the performance of a parallel program on few processors and then predict its performance on many processors with good accuracy.

1.4 Data Assimilation and Virtual Satellite

After a global space weather particle simulation, we often compare the results with both theory and satellite observation to validate the code and adjust the original numerical model to the advanced numerical model. With this new advanced simulation, one can use the results to predict the behaviors of all system include the unperformed observation area, and then the new theory can be generalized and the new mission proposal can be submitted.

The post-processing visualization is very important in the whole research process, in this stage we often ask one question: is it possible to insert and compare, in this simulation, the experimental conditions of local measurements specific to a given space mission? This gives the basic idea of the data assimilation. If this approach could be systematically resolved, it would allow establishing a more direct link between the simulation and experimental observation. A virtual satellite post-processing visualization strategy [Giacalone, 1994], which consists of the virtual satellites inside the simulation data, can be a possible strategy to solve this question. If a global particle simulation at quantitative level can be implemented, it can not only direct comparison with the existing observation of satellite mission, but also extent the view from traditional satellite to the whole simulation area, and gives new theoretical idea from it and help to gives new proposal to the new satellite mission.

1.5 Outline

The organization of this thesis is as follows:

Chapter 2 introduces the 3D Particle-In-Cell code including the numerical algorithms, normalization, instability conditions, etc.

Chapter 3 gives a overview of the Grid Computing and describes the used computational grid environment. A Mpptest benchmark had been performed and discussed on this Grid.

Chapter 4 introduces two parallel PIC code with different physics and entirely different communication patterns. Both the theoretical isoefficiency analysis and benchmark results are compared and discussed in this chapter.

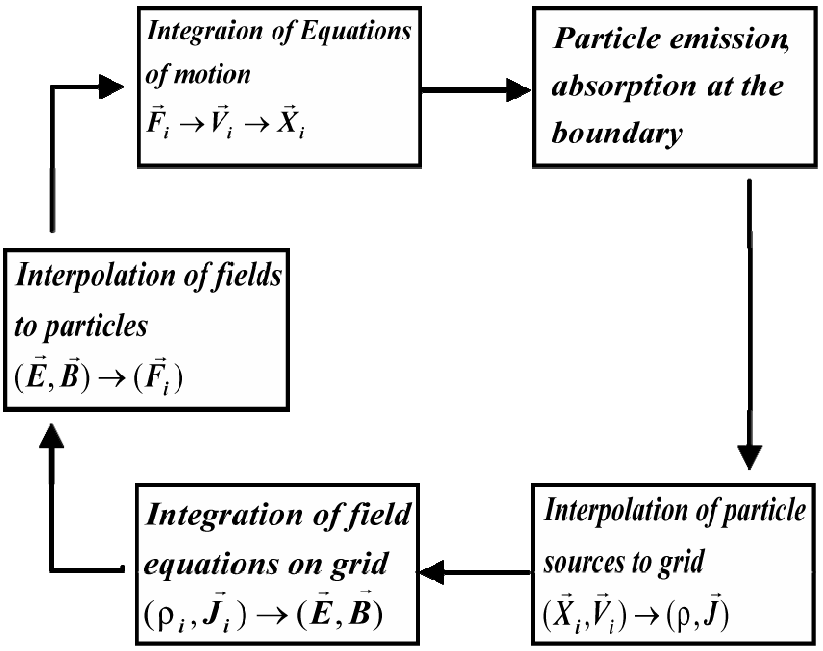
Chapter 5 gives the applications of the global 3D EMPM simulations to space weather problems. They are, the "sash" simulation with turning IMF from northward to sourthward, and "bifurcation and hystersis of the magnetospheric structure" simulations with time-varying southward IMF. Some post-processing visualization method are also presented in this chapter.

Chapter 6 describes a post-processing system, named as virtual satellite system, which can compare the simulation results directly with the satellite observation results. This system makes the data assimilation of global particle simulation more applicable.

Chapter 7 concludes this thesis and then gives several avenues for future research.

Chapter 2: Three-Dimensional Electromagnetic Particle-In-Cell Code

In this full particle code, one follows the motion of both electrons and ions in the self-consistent electric and magnetic fields obtained from a solution of Maxwell's equation. Relativistic effects are readily included by the use of Lorentz equations of motions for the particles. At this level one has introduced no approximation in the basic laws of mechanics and electromagnetism, and thus the full range of collisionless plasma physics is included in such a model.



Integraion of Equations of motion $\vec{F}_i \to \vec{V}_i \to \vec{X}_i$
Particle emission, absorption at the boundary
Interpolation of particle sources to grid $(\vec{X}_i, \vec{V}_i) \to (\rho, \vec{J})$
Integration of field equations on grid $(\rho_i, \vec{J}_i) \to (\vec{E}, \vec{B})$
Interpolation of fields to particles $(\vec{E}, \vec{B}) \to (\vec{F}_i)$

Figure 2-1. Flow schematic for the PIC code

The general flow of the PIC scheme is shown in figure 2-1. In the PIC scheme, particles are defined in continuum space in both position and velocity. Fields are defined at discrete locations in space. However, both fields and particles are defined at discrete times. Particle and field values are advanced sequentially in time, starting from initial conditions. Particle positions and velocities are offset in time by $\Delta t/2$, which allows for the leapfrog centre difference integration of the equations of motion. The particle equations of motion are advanced one time step, using fields interpolated from the discrete grid to the continuous particle locations. Next, particle boundary conditions such as absorption and emission are applied. Source terms, ρ and J, for the field equations are accumulated from the continuous particle locations to the discrete mesh locations. The fields are then advanced one time step, and the time step loop repeats.

2.1 Fundamental Equations

The particle positions and velocities obey the Newton-Lorentz equations of motion:

$$m_{e,i}\frac{d\vec{V}_{e,i}}{dt} = q_{e,i}(\vec{E} + \vec{V}_{e,i} \times \vec{B}) \tag{2.1}$$

and

$$\frac{d}{dt}\vec{r} = \vec{V} \tag{2.2}$$

Here $\vec{V}$, $\vec{r}$ are particle velocity and displacement, and subscript e, and i denote electrons and ions respectively (same as these in the thesis). The electric field $\vec{E}$ and magnetic field $\vec{B}$ have to satisfy the full set of Maxwell equations from (2.3) to (2.6):

$$\nabla \times \vec{E} = -\frac{\partial}{\partial t}\vec{B} \tag{2.3}$$

$$\nabla \times \vec{H} = \vec{J} + \frac{\partial}{\partial t}\vec{D} \tag{2.4}$$

$$\nabla \cdot \vec{B} = 0 \tag{2.5}$$

$$\nabla \cdot \vec{D} = \rho, \tag{2.6}$$

where, $\vec{J}$, $\vec{H}$ and $\vec{D}$ are the vectors of current density, magnetic field intensity and electric displacement. They can be expressed as equation (2.7), (2.8), and (2.9):

$$\vec{H} = \frac{1}{\mu_0}\vec{B} \tag{2.7}$$

$$\vec{D} = \varepsilon_0\vec{E} \tag{2.8}$$

$$\vec{J} = \sum_i n_i q_i \vec{V}_i + \sum_e n_e q_e \vec{V}_e \tag{2.9}$$

The particles are located anywhere in the computational domain, whereas the field quantities $\vec{E}$ and $\vec{B}$ are discrete in a uniform cubic grid [*Buneman,1995*].

2.1.1 Basic equations of Heaviside form

For the normalization of this code, we let:

$$\varepsilon_0 = 1 \tag{2.10}$$

$$\mu_0 = \frac{1}{c^2} \tag{2.11}$$

We get:

$$\vec{D} = \vec{E} \tag{2.12}$$

$$\vec{H} = c^2 \vec{B} \leftrightarrow \frac{\vec{H}}{c} = c\vec{B} = \vec{B}' \tag{2.13}$$

Substitute equation (2.12)-(2.13) into equation (2.3),(2.4), we get Maxwell Equations of Heaviside form:

$$\frac{\partial \vec{E}}{\partial t} = c\nabla \times \vec{B}' - \vec{J} \tag{2.14}$$

$$\frac{\partial}{\partial t}\vec{B}' = -c\nabla \times \vec{E} \tag{2.15}$$

Now Ampere's and Faraday equations are mutually symmetrical in form and the Lorentz equation of eq. (2.2) is as below:

$$m_{e,i}\frac{d\vec{V}_{e,i}}{dt} = q_{e,i}(\vec{E} + \frac{\vec{V}_{e,i}}{c} \times \vec{B}') \tag{2.16}$$

Here after $\vec{B}'$ is abbreviated by $\vec{B}$.

2.2 Spatial and Temporal Assignment of Yee Algorithm

2.2.1 Time Step chart

The time step chart adopted here is based on that of KEMPO1 [Omura and

Matsumoto, 1993]. The quantities of fields and particles are advanced in time based on the sequence shown in Figure 2-2. We define a full-integer time nΔt and a half-integer time (n +1/2)Δt. Basically the electric field $\vec{E}$ at the full-integer time and the magnetic field $\vec{B}$ at the half-integer time are integrated in time by the leap-frog method. The particle positions $\vec{r}$ at the full-integer time and the velocities $\vec{V}$ at the half-integer time are also advanced by the leap-frog method.

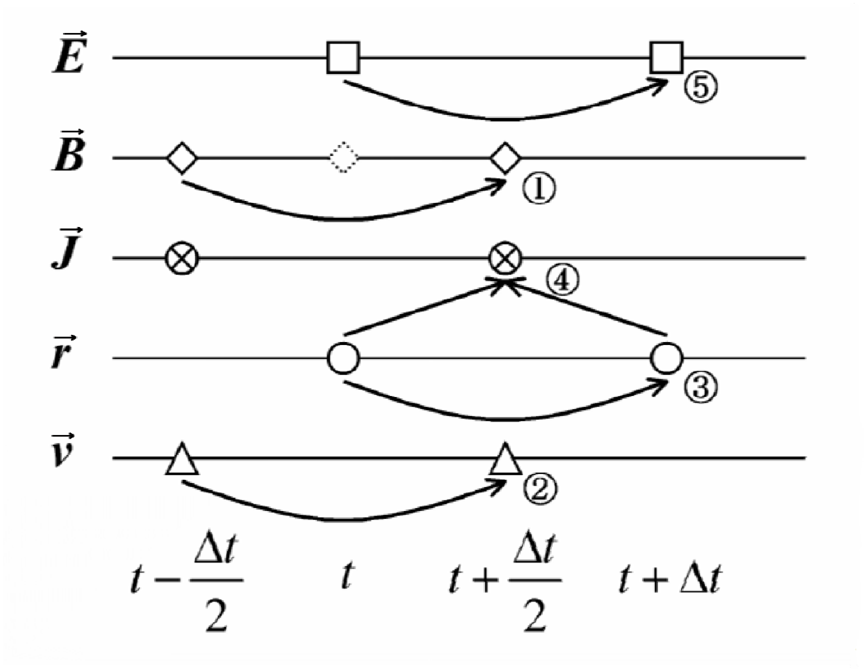


Figure 2-2. Time step chart of PIC algorithm

1. As the initial condition, we have $\vec{r}^{\,t}$,$\vec{V}^{\,t-\frac{\Delta t}{2}}$,$\vec{E}^{\,t}$ and $\vec{B}^{\,t-\frac{\Delta t}{2}}$.

2. Advance the magnetic field $\vec{B}$ from $t-\frac{\Delta t}{2}$ to $t+\frac{\Delta t}{2}$ using $\vec{E}^{\,t}$.

3. Compute $\vec{B}^{\,t}$ by $\vec{B}^{\,t}=\frac{(\vec{B}^{\,t-\frac{\Delta t}{2}}+\vec{B}^{\,t+\frac{\Delta t}{2}})}{2}$. Then advance the velocities of particles $\vec{V}$ from $t-\frac{\Delta t}{2}$ to $t+\frac{\Delta t}{2}$ using $\vec{E}^{\,t}$ and $\vec{B}^{\,t}$.

4. Advance the positions of particles $\vec{r}$ from t to $t+\Delta t$ using $\vec{V}^{t+\frac{\Delta t}{2}}$.

5. Advance the electric field $\vec{E}$ from t to $t+\Delta t$ using $\vec{B}^{t-\frac{\Delta t}{2}}$ and $\vec{B}^{t+\frac{\Delta t}{2}}$

6. Based on the charge conservation method, compute the current density $\vec{J}$ at $t+\dfrac{\Delta t}{2}$ using the position $\vec{r}^{\,t}$ and $\vec{r}^{\,t+\Delta t}$, and add it to $\vec{E}^{\,t+\Delta t}$.

2.2.2 Grid Assignment

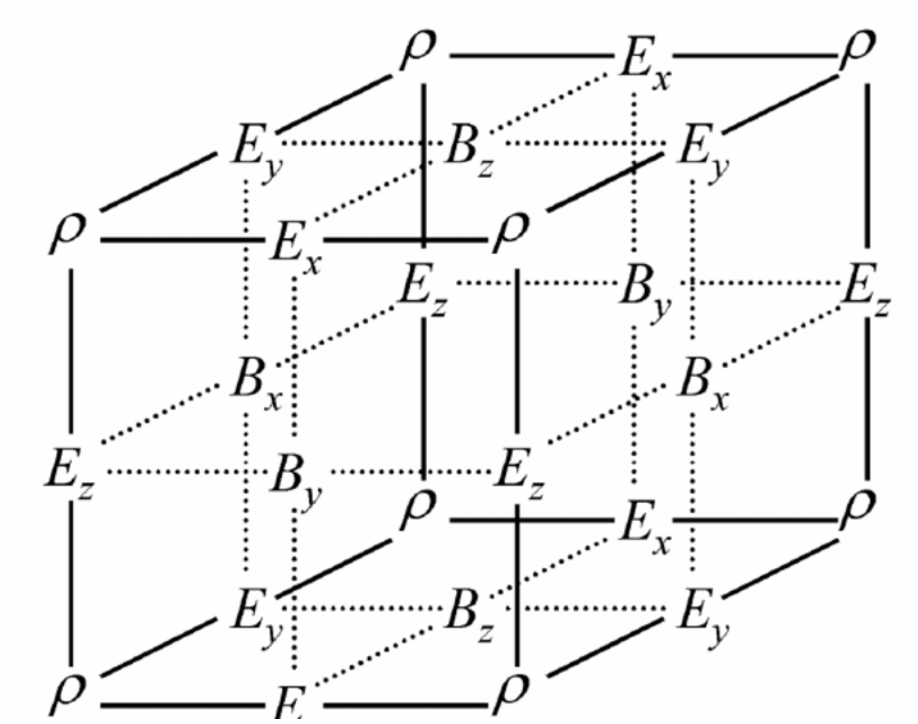


Figure 2-3. Grid assignment of the three-dimensional system

We define full-integer grids [F] at $i\Delta x, j\Delta y$, and $k\Delta z$ ($i=1,2,...,N,$ $j=1,2,...,N,$ $k=1,2,...,N$) and half-integer grids [H] at $(i+\frac{1}{2})\Delta x,(j+\frac{1}{2})\Delta y,(k+\frac{1}{2})\Delta z$. For simplicity, we express the location of the grid points with label (i,j,k) by [FFF], and the grid points with label $(i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2})$ by [HHH].

In the present three-dimensional particle code, we define the charge density ρ at [FFF]. The electrostatic potential ϕ is defined at the same grids. To realize centered deference form for the spatial derivative in the charge continuity equation, the electric field components Ex, Ey, Ez are, respectively, defined at [HFF], [FHF], [FFH]. Components of the current density Jx, Jy, and Jz are, respectively, defined at the same grids of Ex, Ey, and Ez for direct integration of the electric fields in time. To realize centered deference form for the spatial derivative in Maxwell's equations, the magnetic fields component Bx, By, Bz are respectively defined at [FHH], [HFH], [HHF]. The grid assignment of the three-dimensional system is illustrated in Figure 2-3.

2.3 Particle Update

2.3.1 Particle Position

The difference form of the equation of motion Eq.(2.2) in three-dimensional are given by:

$$\vec{X}_{i,e}^{t+\Delta t} - \vec{X}_{i,e}^{t} = \Delta t \cdot \vec{V}_{i,e}^{t+\frac{\Delta t}{2}} \tag{2.17}$$

2.3.2 Particle Velocities

The time centered finite difference version of the update equation of the particle from the Newton-Lorentz equation Eq.(2.1) is:

$$\vec{V}_{i,e}^{t+\frac{\Delta t}{2}} - \vec{V}_{i,e}^{t-\frac{\Delta t}{2}} = \frac{q_{i,e}\Delta t}{m_{i,e}}\left(\vec{E}^{t} + \frac{1}{2}\left(\vec{V}_{i,e}^{t+\frac{\Delta t}{2}} + \vec{V}_{i,e}^{t-\frac{\Delta t}{2}}\right) \times \vec{B}^{t}\right) \tag{2.18}$$

The actual computation is realized with the following method, which are called Hartree method [*Hockney and Eastwood,1981; Birdsall and Langdon, 1985*]:
By introducing the following two variables

$$\vec{V}_{i,e}^{-} = \vec{V}_{i,e}^{t-\frac{\Delta t}{2}} + \frac{q_{i,e}}{m_{i,e}}\frac{E^{t}\Delta t}{2} \tag{2.19}$$

$$\vec{V}_{i,e}^{+} = \vec{V}_{i,e}^{t+\frac{\Delta t}{2}} + \frac{q_{i,e}}{m_{i,e}}\frac{E^{t}\Delta t}{2} \tag{2.20}$$

Eq.(2.17) can be rewritten as:

$$\vec{V}_{i,e}^{+} - \vec{V}_{i,e}^{-} = \frac{q_{i,e}\Delta t}{m_{i,e}}\left(\vec{E}^{t} + \frac{1}{2}\left(\vec{V}_{i,e}^{+} + \vec{V}_{i,e}^{-}\right) \times \vec{B}^{t}\right) \tag{2.21}$$

Taking the inner product of Eq.(2.21) with $\vec{V}_{i,e}^{+} + \vec{V}_{i,e}^{-}$, from the Figure 2-4 we know that

$\vec{V}_{i,e}^{+} + \vec{V}_{i,e}^{-}$ perpendicular to $\vec{V}_{i,e}^{+} - \vec{V}_{i,e}^{-}$, so $(\vec{V}_{i,e}^{+} + \vec{V}_{i,e}^{-}) \bullet (\vec{V}_{i,e}^{+} - \vec{V}_{i,e}^{-}) = 0$, we have

$$(\vec{V}_{i,e}^{+})^2 = (\vec{V}_{i,e}^{-})^2 \tag{2.22}$$

Thus the kinetic energy of a particle is conserved in the cyclotron motion. As shown in Figure 2-4, we have

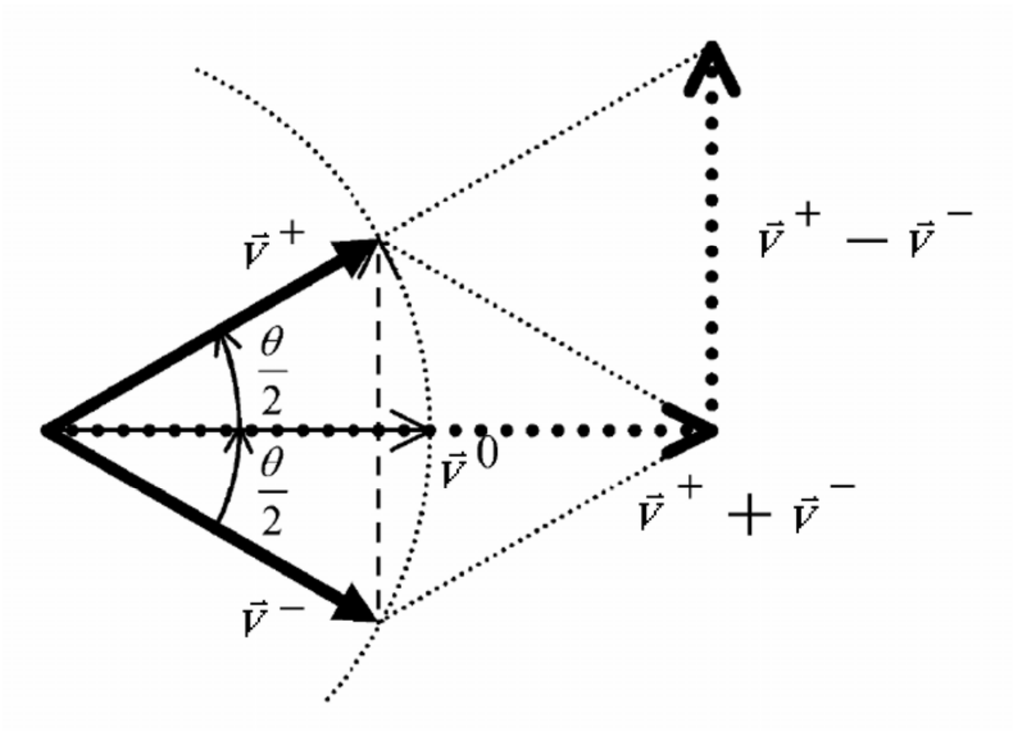


Figure 2-4. Vector relation in Hartree method

$$\tan\frac{\theta}{2} = \frac{\Delta t}{2}\frac{q_{i,e}}{m_{i,e}}\left|\vec{B}\right| \tag{2.23}$$

$$\cos^2\frac{\theta}{2} = \frac{1}{1+\tan^2\frac{\theta}{2}} = \frac{1}{1+(\frac{\Delta t}{2}\frac{q_{i,e}}{m_{i,e}}\left|\vec{B}\right|)^2} \tag{2.24}$$

$$\vec{V}_{i,e}^{+} - \vec{V}_{i,e}^{-} = 2\vec{V}_{i,e}^{0}\cos^2\frac{\theta}{2} \times \vec{B}^{t}\frac{\Delta t}{2} \tag{2.25}$$

There are three steps is needed in this method:

1). Half electric acceleration

At first, the new middle velocity $\vec{V}_{i,e}^{-}$ will be gotten from the old velocity $\vec{V}_{i,e}^{t-\frac{\Delta t}{2}}$ in half time step using Eq.(2.19).

2). Pure magnetic rotation

The new velocity $\vec{V}_{i,e}^{0}$ is generated from $\vec{V}_{i,e}^{-}$:

$$\vec{V}_{i,e}^{0} = \vec{V}_{i,e}^{-} + \tfrac{q_{i,e}}{m_{i,e}} \cdot \vec{V}_{i,e}^{-} \times \vec{B}^{t} \frac{\Delta t}{2} \tag{2.26}$$

Then, we get $\vec{V}_{i,e}^{+}$ from $\vec{V}_{i,e}^{0}$ and $\vec{V}_{i,e}^{-}$ using Eq.(2.25)

3). Another half electric acceleration

The particle will be accelerated in the other half time step. Then the new velocity $\vec{V}_{i,e}^{t+\frac{\Delta t}{2}}$ will be gotten using Eq.(2.20).

2.3.3 Force Interpolation

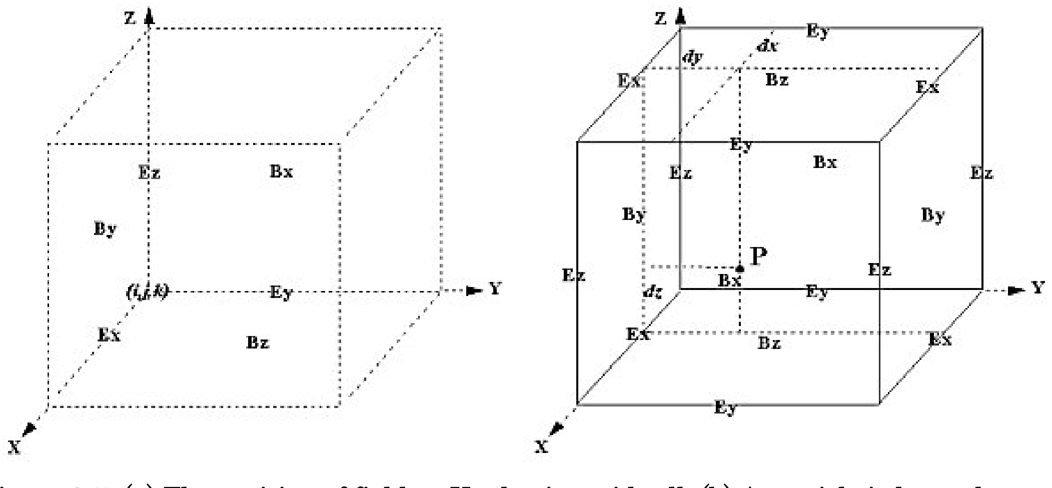


Figure 2-5. (a) The position of field on Yee lattice grid cell. (b) A particle is located at point P of Yee lattice grid cell

Electric and magnetic fields in the above equations are those at the position of particle. Since the electromagnetic fields are defined at spatial grid points, we have to interpolate them from the adjacent grid points to the position of the particle. Linear interpolation is employed for sub-grid resolution. This means that there is no stringent lower limit to the sizes of such quantities as gyro-radii or Debye lengths. It is necessary to calculate the charge density on the discrete grid points from the continuous particle positions and to calculate the force at the particles from the fields on the grids points. These calculations were called weighting, which implied some form of interpolation among the grid points nearest the particle. We use the same weighting in both density and force calculations in order to avoid a self-force. 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 $(1-dx)(1-dy)(1-dz)$, and for $(i+1, j+1, k+1)$ is $dx * dy * dz$. For the particle, position of which is (x, y, z), we define dx, dy, dz and cx, cy, cz as below:

$$\begin{cases} dx = x - \mathrm{int}(x) \\ dy = y - \mathrm{int}(y) \\ dz = z - \mathrm{int}(z) \end{cases} \tag{2.27}$$

and

$$\begin{cases} cx = 1 - dx \\ cy = 1 - dy \\ cz = 1 - dz. \end{cases} \tag{2.28}$$

In Yee lattice as shown in Figure 2-5, the interpolated force of magnetic field at (x, j, k) exerted by the x component is denoted by $F_{E_x}^{(x,j,k)}$ and expressed as following:

$$F_{E_x}^{(x,j,k)} = E_x(i, j, k) + [E_x(i+1, j, k) - E_x(i, j, k)] \cdot dx \tag{2.29}$$

Here,

$$E_x(i, j, k) = \frac{E_x(i+\frac{1}{2}, j, k) + E_x(i-\frac{1}{2}, j, k)}{2} \tag{2.30}$$

$$E_x(i+1, j, k) = \frac{E_x(i+\frac{3}{2}, j, k) + E_x(i+\frac{1}{2}, j, k)}{2}$$

(2.31)

Then we could obtain the interpolated force exerted at (x, j, k):

$$\begin{aligned} 2F_{E_x}^{(x,j,k)} &= E_x(i+\frac{1}{2}, j, k) + E_x(i-\frac{1}{2}, j, k) \\ &\quad + [E_x(i+\frac{3}{2}, j, k) - E_x(i-\frac{1}{2}, j, k)] \cdot dx \end{aligned} \tag{2.32}$$

The x components of electric fields interpretation at $(x, j+1, k)$, $(x, j, k+1)$,

$(\boldsymbol{x}, \boldsymbol{j}+1, \boldsymbol{k}+1)$ is able to be obtained as the same way:

$$\begin{aligned} 2\boldsymbol{F}_{\boldsymbol{E}_x}^{(x,j+1,k)} &= \boldsymbol{E}_x(\boldsymbol{i}+\frac{1}{2}, \boldsymbol{j}+1, \boldsymbol{k}) + \boldsymbol{E}_x(\boldsymbol{i}-\frac{1}{2}, \boldsymbol{j}+1, \boldsymbol{k}) \\ &\quad + [\boldsymbol{E}_x(\boldsymbol{i}+\frac{3}{2}, \boldsymbol{j}+1, \boldsymbol{k}) - \boldsymbol{E}_x(\boldsymbol{i}-\frac{1}{2}, \boldsymbol{j}+1, \boldsymbol{k})] \cdot \boldsymbol{dx} \end{aligned} \tag{2.33}$$

$$\begin{aligned} 2\boldsymbol{F}_{\boldsymbol{E}_x}^{(x,j,k+1)} &= \boldsymbol{E}_x(\boldsymbol{i}+\frac{1}{2}, \boldsymbol{j}, \boldsymbol{k}+1) + \boldsymbol{E}_x(\boldsymbol{i}-\frac{1}{2}, \boldsymbol{j}, \boldsymbol{k}+1) \\ &\quad + [\boldsymbol{E}_x(\boldsymbol{i}+\frac{3}{2}, \boldsymbol{j}, \boldsymbol{k}+1) - \boldsymbol{E}_x(\boldsymbol{i}-\frac{1}{2}, \boldsymbol{j}, \boldsymbol{k}+1)] \cdot \boldsymbol{dx} \end{aligned} \tag{2.34}$$

$$\begin{aligned} 2\boldsymbol{F}_{\boldsymbol{E}_x}^{(x,j+1,k+1)} &= \boldsymbol{E}_x(\boldsymbol{i}+\frac{1}{2}, \boldsymbol{j}+1, \boldsymbol{k}+1) + \boldsymbol{E}_x(\boldsymbol{i}-\frac{1}{2}, \boldsymbol{j}+1, \boldsymbol{k}+1) \\ &\quad + [\boldsymbol{E}_x(\boldsymbol{i}+\frac{3}{2}, \boldsymbol{j}+1, \boldsymbol{k}+1) - \boldsymbol{E}_x(\boldsymbol{i}-\frac{1}{2}, \boldsymbol{j}+1, \boldsymbol{k}+1)] \cdot \boldsymbol{dx} \end{aligned}, \tag{2.35}$$

respectively. Thus the interpolated forces exerted by $\boldsymbol{E}_x$ at $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{k})$, $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{k}+1)$ and $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ are:

$$\boldsymbol{F}_{\boldsymbol{E}_x}^{(x,y,k)} = \boldsymbol{F}_{\boldsymbol{E}_x}^{(x,j,k)} + [\boldsymbol{F}_{\boldsymbol{E}_x}^{(x,j+1,k)} - \boldsymbol{F}_{\boldsymbol{E}_x}^{(x,j,k)}] \cdot \boldsymbol{dy} \tag{2.36}$$

$$\boldsymbol{F}_{\boldsymbol{E}_x}^{(x,y,k+1)} = \boldsymbol{F}_{\boldsymbol{E}_x}^{(x,j,k+1)} + [\boldsymbol{F}_{\boldsymbol{E}_x}^{(x,j+1,k+1)} - \boldsymbol{F}_{\boldsymbol{E}_x}^{(x,j,k+1)}] \cdot \boldsymbol{dy} \tag{2.37}$$

$$\boldsymbol{F}_{\boldsymbol{E}_x}^{(x,y,z)} = \boldsymbol{F}_{\boldsymbol{E}_x}^{(x,y,k)} + [\boldsymbol{F}_{\boldsymbol{E}_x}^{(x,y,k+1)} - \boldsymbol{F}_{\boldsymbol{E}_x}^{(x,y,k)}] \cdot \boldsymbol{dz}, \tag{2.38}$$

respectively. The interpolation forces $\boldsymbol{F}_{\boldsymbol{E}_y}^{(x,y,z)}$, $\boldsymbol{F}_{\boldsymbol{E}_z}^{(x,y,z)}$, $\boldsymbol{F}_{\boldsymbol{B}_x}^{(x,y,z)}$, $\boldsymbol{F}_{\boldsymbol{B}_y}^{(x,y,z)}$ and $\boldsymbol{F}_{\boldsymbol{B}_z}^{(x,y,z)}$ exerted by the electric field components $\boldsymbol{E}_y$, $\boldsymbol{E}_z$, the magnetic field component $\boldsymbol{B}_x$, $\boldsymbol{B}_y$, $\boldsymbol{B}_z$ can be interpolated in the same manner, respectively.

2.4 Field Update

Two different approaches can be used to deal with the Maxwell equations. One is the Finite Difference Time Domain (FDTD) method [*Yee, 1966*]. The second is to obtain the Maxwell equations in Fourier space.

2.4.1 FDTD Method Based Field Update

For reasons of accuracy, it is desirable to employ a centered difference approximation.

This means that the spatial grids for $\vec{E}$ and $\vec{B}$ should be displaced relative to each other. This is normally done based on the Yee lattice [*Yee, 1996*], which is a fully staggered grid mesh system. The components of $\vec{E}$ and $\vec{J}$ are defined at midpoints of cell edges, while the component of $\vec{B}$ are defined at the midpoints of the cell surface (see Figure 2-3). With such a scheme, $\nabla \cdot \boldsymbol{B} = 0$ will be maintained to machine roundoff. Poisson's equation, however, requires special attention. The problem arises from the fact that the grid quantities $\vec{J}$ and ρ produced by simple interpolation schemes may not satisfy the continuity equation,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \boldsymbol{J} = 0 \tag{2.39}$$

If this equation is violated, then the straightforward integration of Maxwell's equation Eq. (2.14) and Eq. (2.15) will rapidly lead to the development of nonphysical fields which do not satisfy $\nabla \cdot \boldsymbol{E} = \frac{\rho}{\varepsilon_0}$. There are two possible methods available to solve it. In first method, one must either adopt an interpolation scheme which rigorously satisfies the continuity equation. There are several numerical techniques for solving the continuity equation locally, which allows us to avoid solving Poisson's equation at every time step. They are called "charge conservation methods". As described by [Eastwood, 1991 and Villasenor and Buneman, 1992], a charge flux of a particle can be computed from the start and end points of the particle movement, when both start and end points are located in the same cell. When a particle moves across the cell meshes, the particle movement is assigned to separate motions in each cell by the cell meshes. From superposition of charge flux, charge conservation can be realized for any particle trajectories made up of straight line segments between any start and end points. In second method, one need to make a correction to ensure that $\nabla \cdot \boldsymbol{E} = \frac{\rho}{\varepsilon_0}$ is maintained [Langdon, 1976]. This method involves adding a correction $\delta \boldsymbol{E}$ to the electric field computed from Eq. (2.14) which is determined by solving $\nabla^2(\delta\phi) = -(\frac{\rho}{\varepsilon_0} - \nabla \cdot \boldsymbol{E})$ and setting $\delta \boldsymbol{E} = -\nabla \delta\phi$.

2.4.1.1 Magnetic Field Update

The staggered grid mesh system, known in the computational electromagnetic community as Yee lattice [Yee, 1966; S. Yamada, 2001, 2002], is shown in Fig. 2.3. It ensures that the change of $\vec{B}$ flux through a cell surface equals the negative circulation of $\vec{E}$ around that surface and the change of $\vec{E}$ flux through a cell surface equals the circulation of $\vec{B}$ around that surface minus the current through it. Here $\vec{B}$ and $\vec{E}$ are in a symmetry form except subtracting the charge flux $\vec{J}$ in Ampere equation. Charge flux $\vec{J}$ is calculated and subtracted after the particles are moved later in the program. Thus magnetic fields are be able to be updated as follows.

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

$$\frac{\partial}{\partial t}\vec{B} = -c\begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ E_x & E_y & E_z \end{vmatrix} = c[\vec{i}(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}) + \vec{j}(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}) + \vec{k}(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x})] \tag{2.40}$$

In this thesis, if the values "0.5" are added to either i, j, and 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, and B_z are, respectively, updated by the negative circulation of $\vec{E}$ around Yee lattice surface as follows:

$$\begin{aligned} \frac{\partial B_x}{\partial t} &= (B_x^{t+\frac{\Delta t}{2}}(i, j+\frac{1}{2}, k+\frac{1}{2}) - B_x^{t-\frac{\Delta t}{2}}(i, j+\frac{1}{2}, k+\frac{1}{2}))/\Delta t \\ &= c[(E_y^t(i, j+\frac{1}{2}, k+1) - E_y^t(i, j+\frac{1}{2}, k))/\Delta z \\ &\quad -(E_z^t(i, j+1, k+\frac{1}{2}) - E_z^t(i, j, k+\frac{1}{2}))/\Delta y] \end{aligned} \tag{2.41}$$

To get the update form of B_y, and B_z, the same procedures are as followed:

$$\begin{aligned} \frac{\partial B_y}{\partial t} &= (B_y^{t+\frac{\Delta t}{2}}(i+\frac{1}{2}, j, k+\frac{1}{2}) - B_y^{t-\frac{\Delta t}{2}}(i+\frac{1}{2}, j, k+\frac{1}{2}))/\Delta t \\ &= c[(E_z^t(i+1, j, k+\frac{1}{2}) - E_z^t(i, j, k+\frac{1}{2}))/\Delta x \\ &\quad -(E_x^t(i+\frac{1}{2}, j, k+1) - E_x^t(i+\frac{1}{2}, j, k))/\Delta z] \end{aligned} \tag{2.42}$$

and

$$
\begin{aligned}
\frac{\partial \boldsymbol{B}_z}{\partial \boldsymbol{t}} &= (\boldsymbol{B}_z^{t+\frac{\Delta t}{2}}(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}) - \boldsymbol{B}_z^{t-\frac{\Delta t}{2}}(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}))/\Delta \boldsymbol{t} \\
&= \boldsymbol{c}[(\boldsymbol{E}_x^t(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j}+1,\boldsymbol{k}) - \boldsymbol{E}_x^t(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j},\boldsymbol{k}))/\Delta \boldsymbol{y} \\
&\quad -(\boldsymbol{E}_y^t(\boldsymbol{i}+1,\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}) - \boldsymbol{E}_y^t(\boldsymbol{i},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}))/\Delta \boldsymbol{x}]
\end{aligned} \tag{2.43}
$$

2.4.1.2 Electric Field Update

As the magnetic fields, the densities, the displacements and velocities of the particles changed, the electric fields must be updated according to Maxwell equations, too. This process are performed in the electric field push module. The vector formula is equation (38).

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{t}}\vec{\boldsymbol{E}} = \boldsymbol{c}\begin{vmatrix} \vec{\boldsymbol{i}} & \vec{\boldsymbol{j}} & \vec{\boldsymbol{k}} \\ \frac{\partial}{\partial \boldsymbol{x}} & \frac{\partial}{\partial \boldsymbol{y}} & \frac{\partial}{\partial \boldsymbol{z}} \\ \boldsymbol{B}_x & \boldsymbol{B}_y & \boldsymbol{B}_z \end{vmatrix} &= \boldsymbol{c}[\vec{\boldsymbol{i}}(\frac{\partial \boldsymbol{B}_z}{\partial \boldsymbol{y}} - \frac{\partial \boldsymbol{B}_y}{\partial \boldsymbol{z}}) + \vec{\boldsymbol{j}}(\frac{\partial \boldsymbol{B}_x}{\partial \boldsymbol{z}} - \frac{\partial \boldsymbol{B}_z}{\partial \boldsymbol{x}}) \\
&\quad + \vec{\boldsymbol{k}}(\frac{\partial \boldsymbol{B}_y}{\partial \boldsymbol{x}} - \frac{\partial \boldsymbol{B}_x}{\partial \boldsymbol{y}})]
\end{aligned} \tag{2.44}
$$

The electric fields updates components are, respectively, updated by the circulation of magnetic field around Yee lattice surface as below:

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{t}}\boldsymbol{E}_x &= (\boldsymbol{E}_x^{t+\Delta t}(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j},\boldsymbol{k}) - \boldsymbol{E}_x^t(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j},\boldsymbol{k}))/\boldsymbol{dt} \\
&= \boldsymbol{c}\{[\boldsymbol{B}_z^{t+\frac{\Delta t}{2}}(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}) - \boldsymbol{B}_z^{t+\frac{\Delta t}{2}}(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j}-\frac{1}{2},\boldsymbol{k})]/\boldsymbol{dy} \\
&\quad -[\boldsymbol{B}_y^{t+\frac{\Delta t}{2}}(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j},\boldsymbol{k}+\frac{1}{2}) - \boldsymbol{B}_y^{t+\frac{\Delta t}{2}}(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j},\boldsymbol{k}-\frac{1}{2})]/\boldsymbol{dz}\}
\end{aligned} \tag{2.45}
$$

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{t}}\boldsymbol{E}_y &= (\boldsymbol{E}_y^{t+\Delta t}(\boldsymbol{i},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}) - \boldsymbol{E}_y^t(\boldsymbol{i},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}))/\boldsymbol{dt} \\
&= \boldsymbol{c}\{[\boldsymbol{B}_x^{t+\frac{\Delta t}{2}}(\boldsymbol{i},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}+\frac{1}{2}) - \boldsymbol{B}_x^{t+\frac{\Delta t}{2}}(\boldsymbol{i},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}-\frac{1}{2})]/\boldsymbol{dz} \\
&\quad -[\boldsymbol{B}_z^{t+\frac{\Delta t}{2}}(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}) - \boldsymbol{B}_z^{t+\frac{\Delta t}{2}}(\boldsymbol{i}-\frac{1}{2},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k})]/\boldsymbol{dx}\}
\end{aligned} \tag{2.46}
$$

$$\begin{aligned}\frac{\partial}{\partial t}\boldsymbol{E}_z &= (\boldsymbol{E}_z^{t+\Delta t}(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}+\tfrac{1}{2}) - \boldsymbol{E}_z^{t}(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}+\tfrac{1}{2}))/\boldsymbol{dt} \\ &= \boldsymbol{c}\{[\boldsymbol{B}_y^{t+\frac{\Delta t}{2}}(\boldsymbol{i}+\tfrac{1}{2},\boldsymbol{j},\boldsymbol{k}+\tfrac{1}{2}) - \boldsymbol{B}_y^{t+\frac{\Delta t}{2}}(\boldsymbol{i}-\tfrac{1}{2},\boldsymbol{j},\boldsymbol{k}+\tfrac{1}{2})]/\boldsymbol{dx} \\ &\quad -[\boldsymbol{B}_x^{t+\frac{\Delta t}{2}}(\boldsymbol{i},\boldsymbol{j}+\tfrac{1}{2},\boldsymbol{k}+\tfrac{1}{2}) - \boldsymbol{B}_x^{t+\frac{\Delta t}{2}}(\boldsymbol{i},\boldsymbol{j}-\tfrac{1}{2},\boldsymbol{k}+\tfrac{1}{2})]/\boldsymbol{dy}\}\end{aligned} \tag{2.47}$$

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

2.4.1.3 Post-processing of Field Data

It is 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 component at the integer position $(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k})$. One must perform the following average:

$$\boldsymbol{E}_x(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}) = \frac{\boldsymbol{E}_x(\boldsymbol{i}+\frac{1}{2},\boldsymbol{j},\boldsymbol{k}) + \boldsymbol{E}_x(\boldsymbol{i}-\frac{1}{2},\boldsymbol{j},\boldsymbol{k})}{2} \tag{2.48}$$

$$\boldsymbol{E}_y(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}) = \frac{\boldsymbol{E}_y(\boldsymbol{i},\boldsymbol{j}+\frac{1}{2},\boldsymbol{k}) + \boldsymbol{E}_y(\boldsymbol{i},\boldsymbol{j}-\frac{1}{2},\boldsymbol{k})}{2} \tag{2.49}$$

$$\boldsymbol{E}_z(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}) = \frac{\boldsymbol{E}_z(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}+\frac{1}{2}) + \boldsymbol{E}_z(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}-\frac{1}{2})}{2} \tag{2.50}$$

$$\begin{aligned}\boldsymbol{B}_x(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}) &= \frac{1}{2}\{\frac{1}{2}[\boldsymbol{B}_x(\boldsymbol{i},\boldsymbol{j}+\tfrac{1}{2},\boldsymbol{k}+\tfrac{1}{2}) + \boldsymbol{B}_x(\boldsymbol{i},\boldsymbol{j}+\tfrac{1}{2},\boldsymbol{k}-\tfrac{1}{2})] \\ &\quad +\frac{1}{2}[\boldsymbol{B}_x(\boldsymbol{i},\boldsymbol{j}-\tfrac{1}{2},\boldsymbol{k}+\tfrac{1}{2}) + \boldsymbol{B}_x(\boldsymbol{i},\boldsymbol{j}-\tfrac{1}{2},\boldsymbol{k}-\tfrac{1}{2})]\}\end{aligned} \tag{2.51}$$

$$\begin{aligned}\boldsymbol{B}_y(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}) &= \frac{1}{2}\{\frac{1}{2}[\boldsymbol{B}_y(\boldsymbol{i}+\tfrac{1}{2},\boldsymbol{j},\boldsymbol{k}+\tfrac{1}{2}) + \boldsymbol{B}_y(\boldsymbol{i}-\tfrac{1}{2},\boldsymbol{j},\boldsymbol{k}+\tfrac{1}{2})] \\ &\quad +\frac{1}{2}[\boldsymbol{B}_y(\boldsymbol{i}+\tfrac{1}{2},\boldsymbol{j},\boldsymbol{k}-\tfrac{1}{2}) + \boldsymbol{B}_y(\boldsymbol{i}-\tfrac{1}{2},\boldsymbol{j},\boldsymbol{k}-\tfrac{1}{2})]\}\end{aligned} \tag{2.52}$$

$$\begin{aligned}\boldsymbol{B}_z(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}) &= \frac{1}{2}\{\frac{1}{2}[\boldsymbol{B}_z(\boldsymbol{i}+\tfrac{1}{2},\boldsymbol{j}+\tfrac{1}{2},\boldsymbol{k}) + \boldsymbol{B}_z(\boldsymbol{i}+\tfrac{1}{2},\boldsymbol{j}-\tfrac{1}{2},\boldsymbol{k})] \\ &\quad +\frac{1}{2}[\boldsymbol{B}_z(\boldsymbol{i}-\tfrac{1}{2},\boldsymbol{j}+\tfrac{1}{2},\boldsymbol{k}) + \boldsymbol{B}_z(\boldsymbol{i}-\tfrac{1}{2},\boldsymbol{j}-\tfrac{1}{2},\boldsymbol{k})]\}\end{aligned} \tag{2.53}$$

2.4.2 Fourier Transform Method Based Field Update

In this method, the transformation between coordinate space and Fourier space are carried out via fast Fourier transforms (FFT). To this end, it is useful to introduce the decomposition of an arbitrary vector $\vec{S}$ into its divergence free (transverse) and curl free (longitudinal) parts [*Hockney and Eastwood,1981; Birdsall and Langdon, 1985*]:

$$\boldsymbol{k} \cdot \boldsymbol{S}_T(\boldsymbol{k}) = 0, \boldsymbol{k} \times \boldsymbol{S}_L(\boldsymbol{k}) = 0 \tag{2.54}$$

From Eq. (2.5-2.6) it is clear that the magnetic field has only a transverse component. The time-dependent Maxwell equations then become

$$\frac{\partial \boldsymbol{E}_T(\boldsymbol{k})}{\partial \boldsymbol{t}} = \boldsymbol{ick} \times \boldsymbol{B}(\boldsymbol{k}) - \boldsymbol{J}_T(\boldsymbol{k}) \tag{2.55}$$

$$\frac{\partial \boldsymbol{B}(\boldsymbol{k})}{\partial \boldsymbol{t}} = -\boldsymbol{ick} \times \boldsymbol{E}_T(\boldsymbol{k}) \tag{2.56}$$

Poisson's equation determines the longitudinal part of $\boldsymbol{E}, \boldsymbol{ik} \cdot \boldsymbol{E}_L(\boldsymbol{k}) = \frac{\rho(\boldsymbol{k})}{\varepsilon_0}$. Maxwell's equations thus reduce to a set of algebraic equations for the Fourier amplitudes. This Fourier solution is quite accurate, but it has the disadvantage of being directly applicable only to cases where the system is periodic in each spatial direction. Generalizations to cases involving combinations of periodic and bounder coordinates were considered in [*Decyk, 1979*].

2.5 Numerical Stabilities and Heating Conditions

Among all of numerical simulation, the numerical stability conditions and heating conditions should be considered, which are mainly considered as following.

2.5.1 Courant condition

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

$$\Delta \boldsymbol{x}, \Delta \boldsymbol{y}, \Delta \boldsymbol{z} \geq \tilde{\boldsymbol{c}} \Delta \boldsymbol{t} \tag{2.57}$$

where $\tilde{c}$ is the light speed, and Δx, Δy, Δz are the grid size. The condition is easily derived from the numerical dispersion relation of the light mode [Cai, 2003].

Chapter 3: Computational Grid and Mpptest Benchmark

3.1 Overview of Grid Computing

The most common description of *grid computing* includes an analogy to a power grid [*Foster, 1999-a*]. When you plug an appliance or other object requiring electrical power into a receptacle, you expect that there is power of the correct voltage available, but the actual source of that power is not known. Your local utility company provides the interface into a complex network of generators and power sources and provides you with (in most cases) an acceptable quality of service for your energy demands. Rather than each house or neighborhood having to obtain and maintain its own generator of electricity, the power grid infrastructure provides a virtual generator. The generator is highly reliable and adapts to the power needs of the consumers based on their demand.

The vision of grid computing is similar. Once the proper kind of infrastructure is in place, a user will have access to a virtual computer that is reliable and adaptable to the user's needs. This virtual computer will consist of many diverse computing resources. But these individual resources will not be visible to the user, just as the consumer of electric power is unaware of how their electricity is being generated. To reach this vision, there must be standards for grid computing that will allow a secure and robust infrastructure to be built. Standards and tools such as those provided by the Globus Toolkit provide the necessary framework [*Foster, 2001*].

3.1.1 Types of Grids

Grid computing can be used in a variety of ways to address various kinds of application requirements. Often, grids are categorized by the type of solutions that they best address. The three primary types of grids are summarized below. Of course, there are no hard boundaries between these grid types and often grids may be a combination of two or more of these. However, as you consider developing applications that may run in a grid environment, remember that the type of grid environment that you will be using will affect many of your decisions.

Computational grid: A *computational grid* is focused on setting aside resources specifically for computing power. In this type of grid, most of the machines are high-performance servers.

Data grid: A *data grid* is responsible for housing and providing access to data across multiple organizations. Users are not concerned with where this data is located as long as they have access to the data. For example, you may have two universities doing life science research, each with unique data. A data grid would allow them to share their data, manage the data, and manage security issues such as who has access to what data.

Access Grid: Access Grid is a collection of resources and technologies that enables large format audio and video based collaboration between groups of people in different locations. In simple terms, it is advanced videoconferencing using big displays and with multiple simultaneous camera feeds at each node (site).

Scavenging grid: A *scavenging grid* is most commonly used with large numbers of desktop machines. Machines are scavenged for available CPU cycles and other resources. Owners of the desktop machines are usually given control over when their resources are available to participate in the grid.

Another common distributed computing model that is often associated with or confused with grid computing is *peer-to-peer computing*. In fact, some consider this another form of grid computing [*Foster, 2003*].

3.1.2 Grid Programming Models

Grid programming will require capabilities and properties beyond that of simple sequential programming or even parallel and distributed programming. Besides orchestrating simple operations over private data structures, or orchestrating multiple operations over shared or distributed data structures, a grid programmer will have to manage a computation in an environment that is typically open-ended, heterogeneous and dynamic in composition with a deepening memory and bandwidth/latency hierarchy. While it may be possible to build grid applications with current programming tools, there is a growing consensus that current tools and languages are insufficient to support the effective development of efficient grid codes.

For computational Grid, Obtaining high-performance requires a balance of computation and communication among all resources involved. Currently this is done by managing computation, communication and data locality using message-passing or remote method invocation since they require the programmer to be aware of the marshalling of arguments and their transfer from source to destination. To achieve petaflop rates on tightly or loosely coupled grid clusters of gigaflop processors, however, applications will have to allow extremely large granularity or produce over $\approx 10^{-8}$-way parallelism such that high latencies can be tolerated. In some cases, this type of parallelism, and the performance delivered by it in a heterogeneous environment, will be manageable by hand-coded applications. In general, however, it will not be. Hence, what programming models, abstractions, tools, or methodologies can be used to reduce the burden (or even enable the management of) massive amounts of parallelism and maintaining performance in a dynamic, heterogeneous environment [*LEE, 2001*]?

Programming requires the algorithmic or systematic management of data which we are calling access to state. On a uniprocessor, it is typical and convenient to think of the data as being "in the same place" as the code such that an algorithm or system performance only depends on the structure of the algorithm or system. In parallel and distributed grid environments, the issue of accessing state is even more complicated and affects an application behavior to an even greater extent. Access to state is managed in two basic ways: shared data abstractions and shared-nothing environments.

3.1.2.1 Shared Data Abstractions Models

Programming languages and tools that fall under this class provide a global shared memory model as a useful abstraction for state access, although on distributed computing architectures its implementation is distributed. This approach is called virtual shared memory or distributed shared memory (DSM). Programming languages of this category allow users to define variables that are automatically mapped into the memory of processing elements that at higher level are seen as a global memory space. When data are mapped to their processing elements, program constructs can be used to express parallel operations. Data that must be processed in parallel are defined using specific keywords. Then the compiler will partition the data and map them onto the different processors of the parallel computer so instructions that operate on these data will be executed in parallel on different processors that execute the same operation on different elements. Languages and tool kits that implement a global data space are HPF, OpenMP, and C* [Loveman, 1993 and OpenMP, 1997].

The implementation of these programming models on grids is very hard for several reasons mainly related to the different computational model between global data space computing and grid. Here, we list three of these reasons: (1) the programming models in this class abstract from several implementation issues that arise in a grid computing architecture, (2) generally they impose a tight synchronization model among parallel activities that can not be efficiently implemented on remote heterogeneous machines, and, (3) most of the applications developed with these models are based on regular patterns of computation and communication that do not match well the irregularities of grids.

3.1.2.2 Shared Nothing (Message Passing) Models

3.1.2.2.1 Two-Sided Communication

In this model processes run in disjoint address spaces and information is exchanged using message passing of one form or another. The Message Passing Interface (MPI) [*MPI, 1995*] standard defines a two-sided message passing library (matched sends and receives) that is well-suited for *shared nothing environments*. While the explicit parallelization with message passing is cumbersome it gives the user full control and is thus applicable to problems where more convenient semi-automatic programming models may fail. It also tells the programmer exactly where a potential expensive communication takes place. These two points do not hold only for single parallel machines, but even more for grid computing. A *grid-enabled* library should also be integrated into a *grid computing environments* to take care of launching the application. Problems to address are authentication, authorization and resource reservation across multiple computers on possibly different administrative domains. In the last couple of years several groups have been working on these *grid-enabled* MPI implementations MPICH-G2, PACX-MPI, Stampi, MPI_Connect, and MagPie[*Fagg, 1998, Gabriel, 1998, Foster, 1999-b, Imamura, 2000, Kielmann, 1999*].

While MPI addresses some of the challenges in grid computing, it has not addressed them all. Some issues (e.g., algorithm design, communication patterns) can only be addressed by the MPI application developer. Local and widearea networks inject significantly higher latencies and lower bandwidths, and therefore MPI applications that expect to run efficiently in grid environments must be written with respect to this disparity in communication paths. MPI has the advantage that it offers an incremental approach to grid programming. It is possible to first gain some experiences and adopt and improve an existing solution to the grid. It also precludes the user from learning a new method for interoperating.

3.1.2.2.2 One-Sided Communication

Message-passing models, whether they are point-to-point, broadcast, or associatively addressed, all have the essential attribute of explicitly marshalled arguments being sent to a matched receive that unmarshalls the arguments and decides the processing, typically based on message type. The semantics associated with each message type is usually defined statically by the application designers. One-sided message-passing models alter this paradigm by not requiring a matching receive and allowing the sender to specify the type of remote processing. Remote Procedure Call (RPC) and Remote Method Invocation (RMI) models provide the same capabilities as this, but structure the interaction between sender and receiver more as a language construct, rather than a library function call that simply transfers an uninterpreted buffer of data between points A and B. RPC and RMI models provide a simple and well understood mechanism for managing remote computations. Besides being a mechanism for managing the flow of control and data, RPC and RMI also enable some checking of argument type and arity. RPC and RMI can also be used to build higher-level models for grid programming, such as components, frameworks, and network-enabled services.

GridRPC [*Nakada, 2002*] is an RPC model and API for grids. Besides providing standard RPC semantics with asynchronous, coarse-grain, task-parallel execution, it provides a convenient, high-level abstraction whereby the many details of interacting with a grid environment can be hidden. Three very important grid capabilities that GridRPC could transparently manage for the user are: dynamic resource discovery and scheduling, security and fault tolerance.

OmniRPC [*Sato, 2001*] was specifically designed as a thread safe RPC facility for clusters and grids. OmniRPC uses OpenMP to manage thread-parallel execution while using Globus to manage grid interactions. Rather than using message-passing between machines, however, it provides RPC. OmniRPC is, in fact, a layer on top of Ninf [*Nakada, 1999*]. Hence, it uses the Ninf machinery to discover remote procedure names, associate them with remote executables, and retrieve all stub interface information at run-time. To manage multiple RPCs in a multi-threaded client, OmniRPC maintains a queue of outstanding calls that is managed by a scheduler thread. A calling thread is put on the queue and blocks until the scheduler thread initiates the appropriate remote call and receives the results.

3.2 Globus Toolkits and MPICH-G2 System

For the purpose of this research we focus here on computational Grid, which primarily handles computationally intensive PIC code simulation. Hereafter, the term "Grid" refers to the computational Grid in this thesis. One of the most widely used programming model/system for high performance computing society named as Globus+MPICH-G2 are chosen in this research.

3.2.1 Globus Toolkits

The Globus Toolkit [*GLOBUS*] is an open source software toolkit with an open architecture. It is being developed mainly by the Mathematics and Computer Science Division at Argonne National Laboratory and contributed by lots of developers world-wide. It is one of the most widely used toolkits to build the Grid.

The Globus Toolkit is a collection of software tools that provide basic services for building computational grids and appropriate grid-based applications. The toolkit is based on three main components.

a. The Resource Management Part

The first part of the Globus Toolkit provides Resource Management. The Resource Management involves the allocation and management of grid resources. It includes components like GRAM (Globus Resource Allocation Manager), DUROC (Dynamically-Updated Request Online Coallocator) and GASS (Globus Access to Secondary Storage).

b. The Information Services Part

The second part of the Globus Toolkit is Information Service, which provides information about grid resources. This area includes MDS (Monitoring and Discovery System), which in turn provides the GIIS (Grid Index Information Service) and GRIS (Grid Resource Information Service) components.

c. The Data Management Part

The third pillar of the Globus Toolkit is Data Management. The Data Management involves the ability to access and manage data in a grid environment. This includes components such as GridFTP, which is used to move files between grid-enabled storage systems.

All of these components use services provided by the GSI (Globus Security Infrastructure) security protocol at the connection layer. Since version 2 one can install each component separately in a client and a server version. Support for that is given by the grid packaging toolkit developed. The newest version is GT 4.0, form the version 3.0 it focus on web service and XML techniques. In this paper, we only dominate the Globus 2.4 which is the latest version focuses on the high performance computings.

3.2.2 MPICH-G2

The MPICH-G2 [*Karonis, 2003*] is a complete implementation of the MPI-1 standard [*MPI, 1995*] that uses service provided by the Globus Toolkit to extent the popular Argonne MPICH implementation of MPI [*Gropp, 1996*] for Grid execution. MPICH-G2 represents a complete redesign and reimplementation of the earlier MPICH-G system [15] that increase performance significantly, incorporates a number of innovations.

MPICH-G2 hides heterogeneity by using Globus Toolkit services for such purpose as authentication, authorization, executable staging, processing creating, process monitoring, process control, communication, redirection of standard input and output, and remote file access. MPICH-G2 automatically converts data in messages sent between machines of different architectures and supports multi-protocol communication by automatically selecting TCP for inter-machine messaging and vendor-supplied MPI for intra-machine messaging. MPICH-G2 alleviates the user from the cumbersome (and often undesirable) task of learning and explicitly following site-specific details by enabling the user to launch a multi-machine application with the use of a single command "mpirun".

Many groups have used MPICH-G2 for execution of both traditional parallel computing applications and nontraditional distributed computing applications, in both local-area and wide-area networks. This variety of applications persuades us that MPI can play a valuable role in Grid computing.

3.2.3 Management MPI application on Globus+MPICH-G2 System

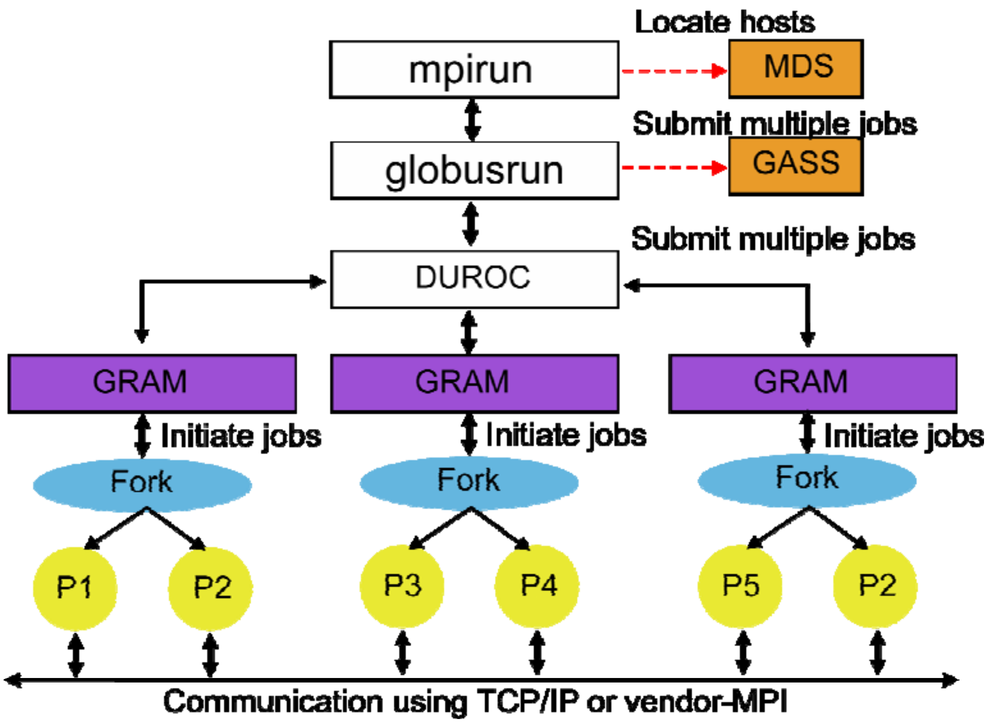


Figure 3-1. Schematic of the MPICH-G2 startup and manage different local schedulers

As illustrated in Figure 3-1, MPICH-G2 uses a range of Globus Toolkit service to address the various complex issues that arise in heterogeneous, multi-site Grid environment.

Prior to startup of an MPICH-G2 application, the user employs the Grid Security Infrastructure (GSI) [*Foster, 1998*] to obtain a proxy credential (public key) that is used to authenticate the user for each site. This step provides a single sign on capability. The user may also use the Monitor Discovery Service (MDS) [*Fitzgerald,, 1997*] to select computers on the basic of, for example, configuration, availability and network connectivity.

Once authenticated, the user uses the standard mpirun command to request the creation of an MPI computation. The MPICH-G2 implementation of this command uses the Resource Specification Language (RSL) [*Czajkowski, 1998*] to describe the job. In brief, users write RSL script that identify resources (e.g., computers) and specify requirements (e.g., number of CPUs, memory, and execution time) and parameters (e.g., location of executable files, command line arguments, and environment variables).

Based on the information found in an RSL script, MPICH-G2 calls a co-allocation library distributed with the Globus Toolkit the Dynamically Updated Request Online Coallocator (DUROC) [*Czajkowski, 1999*], to schedule and start the application across the various computers specified by the user. The DUROC library itself uses the Grid Resource Allocation and Management (GRAM) [*Czajkowski, 1998*] API and protocol to start and subsequently manage a set of sub-computations, one for each computer. For each sub-computation, DUROC generates a GRAM request to a remote GRAM server, which authenticates the user, performs local authorization, and then interacts with the local scheduler to initiate the computation. DUROC and associated MPICH-G2 libraries tie the various sub-computations together into a single MPI computation.

GRAM will, if directed, use Global Access to Secondary Storage (GASS) [*Bester, 1999*] to stage executables from remote locations (indicated by URLs). GASS is also used, once an application has started, to direct standard output and error (stdout and stderr) streams to the user's terminal and provides access to files regardless of location, thus masking essentially all aspects of geographical distribution except those associated with performance. DUROC and GRAM also interact to monitor and manage the execution of application.

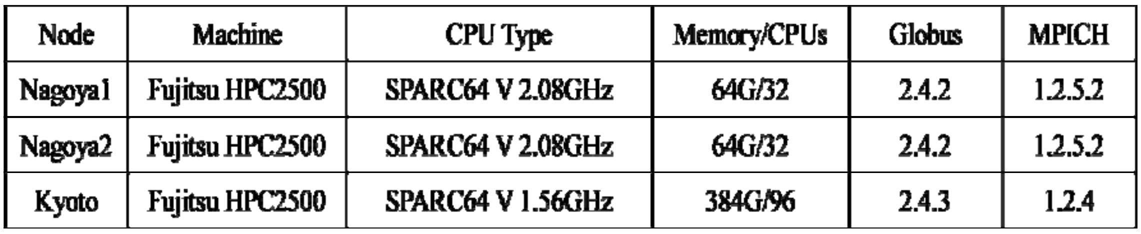
3.3 Experimental Grid Used in this Research

3.3.1 SuperSINET

SuperSINET [*SINET*] is an ultra-high-speed network intended to develop and promote Japanese academic researches by strengthening collaboration among leading academic research institutes. The e-Japan Priority Policy Program announced by the IT Strategic Headquarters in March 2001 referred to this network. The Internet backbone connects research institutes at 10 Gbps and the leading research facilities in the research institutes are directly connected at 1 Gbps. It will be merged in new Photonic-SINET project after April, 2005.

3.3.2 Supercomputer based Grid

In 2001, "a virtual" computer center composed of seven universities in Japan (i. e. University of Tokyo, Kyoto University, Nagoya University, Osaka University, Kyushu University, Tohoku University, and Hokkaido University) joined in a new project to set up a National Computational Grid connected to the SuperSINET (Sciences Internet) Network. The SuperSINET is an ultra-high-speed network intended to develop and promote Japanese academic researches by strengthening collaboration among leading academic research institutes connected at 10 Gbps and the leading research facilities in the research institutes connected directly at 1 Gbps. The Globus Toolkit (GT) is used as the middleware and configured with MPICH-G2 to run Message Passing Interface (MPI) applications on a computational Grid. In March 2004, the computer center of Nagoya University began to open their Grid nodes to researchers. Kyoto University joined this research computational Grid system. The first experimental Grid used for benchmark has one supercomputer node at Kyoto and two supercomputer nodes at Nagoya as listed in Table 3-1, and the system configuration of this supercomputer based Grid is described in Figure 3-1. The distance between Kyoto University and Nagoya University is about 200 kilometers.



Node	Machine	CPU Type	Memory/CPUs	Globus	MPICH
Nagoya1	Fujitsu HPC2500	SPARC64 V 2.08GHz	64G/32	2.4.2	1.2.5.2
Nagoya2	Fujitsu HPC2500	SPARC64 V 2.08GHz	64G/32	2.4.2	1.2.5.2
Kyoto	Fujitsu HPC2500	SPARC64 V 1.56GHz	384G/96	2.4.3	1.2.4

Table 3-1. Configuration of Supercomputer based Grid. Fujitsu Primepower HPC 2500 is the brand name of a supercomputer made by Fujitsu. It is equipped with SPARC64 processors and one node can have up to 128 SPARC64 CPUs. SPARC (Scalable Processor Architecture) is a RISC microprocessor instruction set architecture originally designed in 1985 by Sun Microsystems.

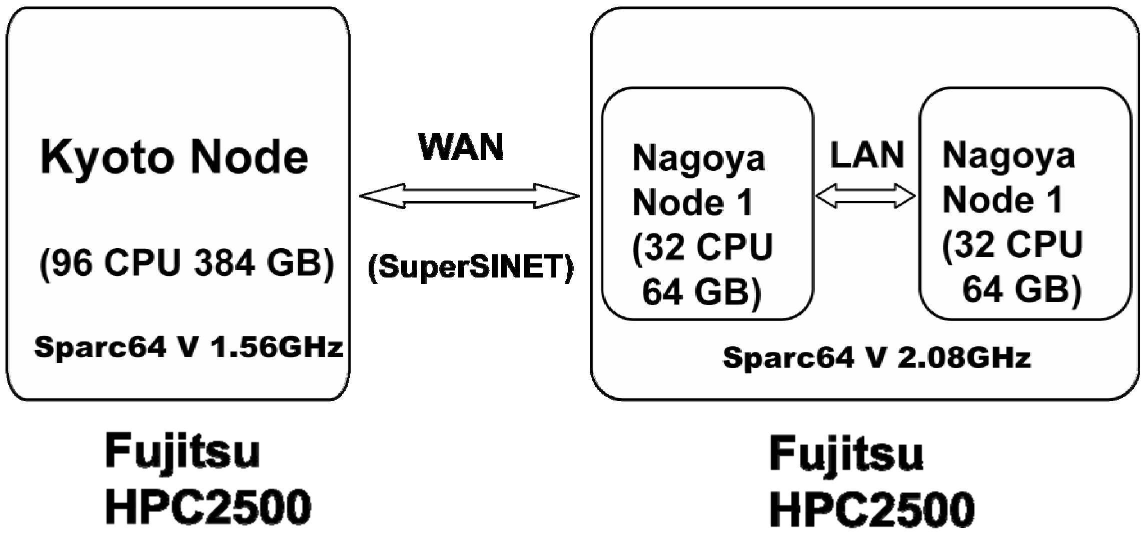


Figure 3-2. System configuration of supercomputer based Grid. Fujitsu Primepower HPC 2500 is the brand name of a supercomputer made by Fujitsu. It is equipped with SPARC64 processors and one node can have up to 128 SPARC64 CPUs. Here, Kyoto node has 96 CPUs, and both Nagoya node 1 and 2 have 32 CPUs.

3.3.3 Cluster based Grid

Name	Organization	CPU Type	Memory	Nodes/CPUs	Queue System
F32	AIST	Xeon 3.0GHz (Dual)	4GB	128/256	SGE
Sakura	AIST	Opteron 2.2GHz (Dual)	2GB	16/32	SGE
Tau	Univ. Tokyo	Xeon 2.4GHz (Dual)	2GB	175/350	SGE
Chikayama	Univ. Tokyo	Xeon 2.4GHz (Dual)	1GB	64/128	SGE

Table 3-2. Configuration of cluster based Grid. Here SGE stands for Sun Grid Engine (SGE). The Xeon is Intel's brand name for its server-class x86 microprocessors intended for multiple-processor machines. The AMD Opteron (codenamed SledgeHammer during development) is AMD brand name for the first of AMD's eighth-generation x86 processors. The term "dual" refers to a PC equipped with dual processors.

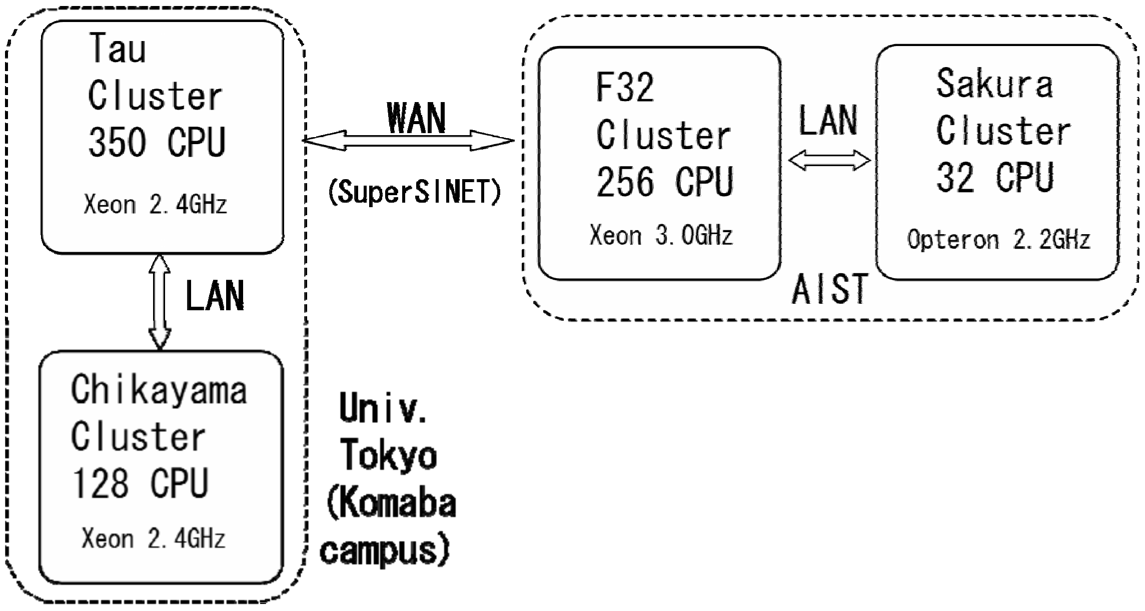


Figure 3-3. System configuration of cluster based Grid. The Xeon is Intel's brand name for its server-class x86 microprocessors intended for multiple-processor machines. The AMD Opteron (codenamed SledgeHammer during development) is AMD brand name for the first of AMD's eighth-generation x86 processors.

The second computational Grid used in our benchmarks belongs to "Grid Challenge 2006" project [*GridChallenge*], the configuration of this computational Grid is shown in Table 3-2. Four PC clusters are used in our PIC benchmark tests. They are "F32", "Sakura", "Tau" and "Chikayama" in Table 3-2; acronyms AIST and SGE respectively stand for national institute of Advanced Industrial Science and Technology of Japan (AIST), and Sun Grid Engineering (SGE). Here, "F32" and "Sakura" are at AIST with low-network latency, it can be regarded as an inter-LAN (Local Area Network) Grid; i.e., a "campus Grid." Here, "Tau" and "Chikayama" are at Komaba campus of University of Tokyo. The distance between AIST and University of Tokyo campus is about 100 Kilometers. The system configuration of this cluster based Grid is described in Figure 3-2. More than 750 processors and 1000GB memory are available in this computational Grid. All clusters are connected with the SuperSINET. For running MPI code in this Grid, GT 2.4 version is installed and configured with MPICH-G2 1.2.7.

3.4 Mpptest Benchmark on Computational Grid

MPPtest is a program that measures the performance of some of the basic MPI message passing routines in a variety of situations [*Gropp, 1999*]. In addition to the classic ping pong test, MPPtest can measure performance with many participating processes (exposing contention and scalability problems) and can adaptively choose the message sizes in order to isolate sudden changes in performance [*Gropp, 1999*]. By using MPPtest, we can investigate how the network latency in grid systems affects the basic point-to-point and collective communication patterns independent of real computing.

In the following MPPtest benchmarks, we use three different types of clusters from the cluster based Grid system described in section 3.2: (1) the so-called "inside one cluster" (cluster "F32"), (2) "two clusters connected by LAN" (between cluster "F32" and cluster "Sakura"), and (3) "two clusters connected by WAN (Wide Area Network)" (between cluster "F32" and cluster "Tau"). The RTT (Round-Trip-Time) is less than 0.4 ms (millisecond) between cluster "F32" and cluster "Sakura," and 2.1 ms between cluster "F32" and "Tau". The "MPPtest" package version 1.2 is used in the present paper.

3.4.1 Point-to-Point Benchmark Test

3.4.1.1 Round-Trip Pattern

The round-trip pattern, which also is named as *ping-pong* test, uses two processes for communication. The first process, also named the master process, sends one message of size bytes to the other process, also named slave process. Before the send function is called a timestamp is saved as $\boldsymbol{t}0$. After the slave has received the message it sends one message of same size back to the master. When having completely received the message a timestamp $\boldsymbol{t}1$ is saved. The elapsed time $\Delta\boldsymbol{t} = \boldsymbol{t}0 - \boldsymbol{t}1$ is the round-trip time. Written to the output is the 'one way value', which is estimated by $\frac{\Delta\boldsymbol{t}}{2}$. The bandwidth is accumulated for incoming and outgoing messages. This test is repeated several times with the same message size before the value of size is increased.

3.4.1.2 Head-to-Head Pattern

This pattern uses two processes for communication. After an initial synchronization the process having the master rank saves the start timestamp $\boldsymbol{t}0$. Each of both processes sends one message to the other one. After the receipt of the message the master saves the second timestamp $\boldsymbol{t}1$ and calculates the elapsed time $\Delta\boldsymbol{t} = \boldsymbol{t}0 - \boldsymbol{t}1$. The elapsed time written to output is $\Delta\boldsymbol{t}$.

3.4.1.3 Ghost Cell Pattern

In many applications data partitioning makes it necessary to introduce 'ghost' cells at the interface boundaries. These are used as a kind of cache and allow, for instance, derivatives and other operations to be performed on local variables only. The number or thickness of halo cells can be from one to tens or hundreds depending on the operations and partitioning stencil. In the example in figure 3-4 two halo cells are used. Halo cells are updated as required, which is the basis of the halo exchange test.

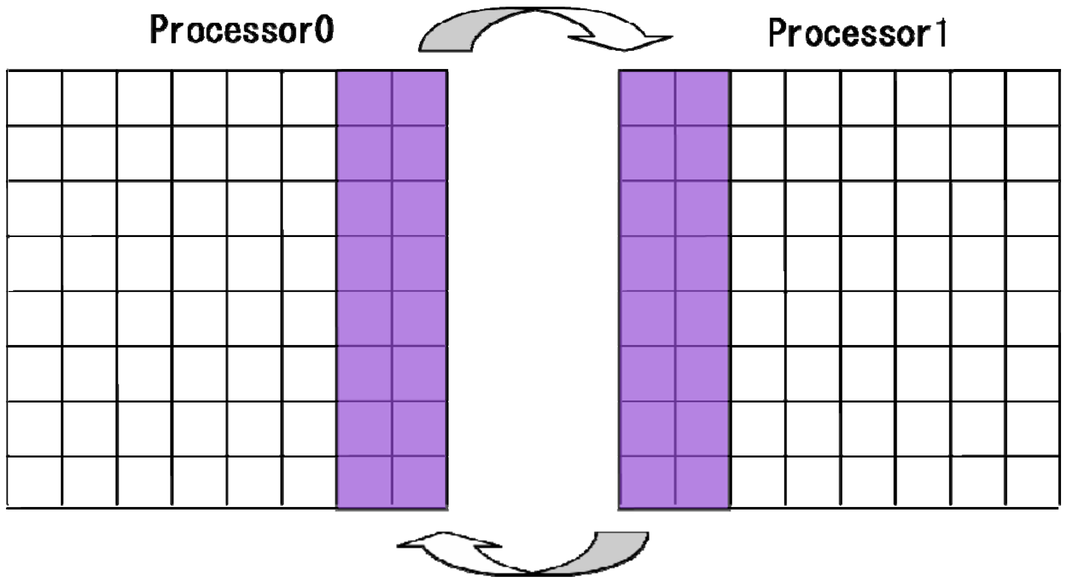


Figure 3-4. Ghost Cell Pattern

3.4.1.4 Benchmark Results

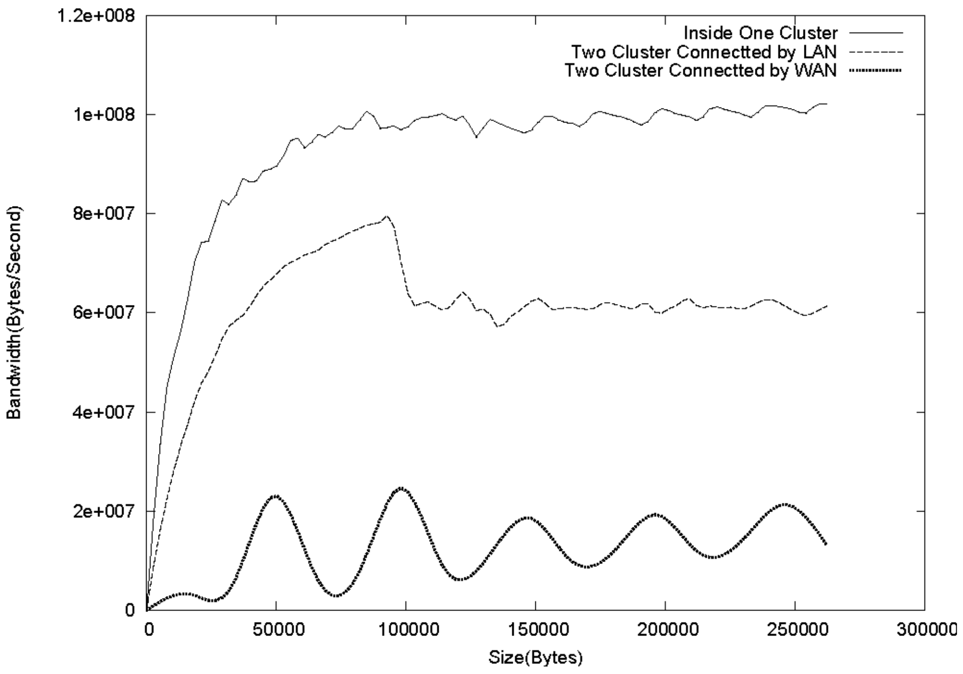


Figure 3-5. Round-Trip Benchmark

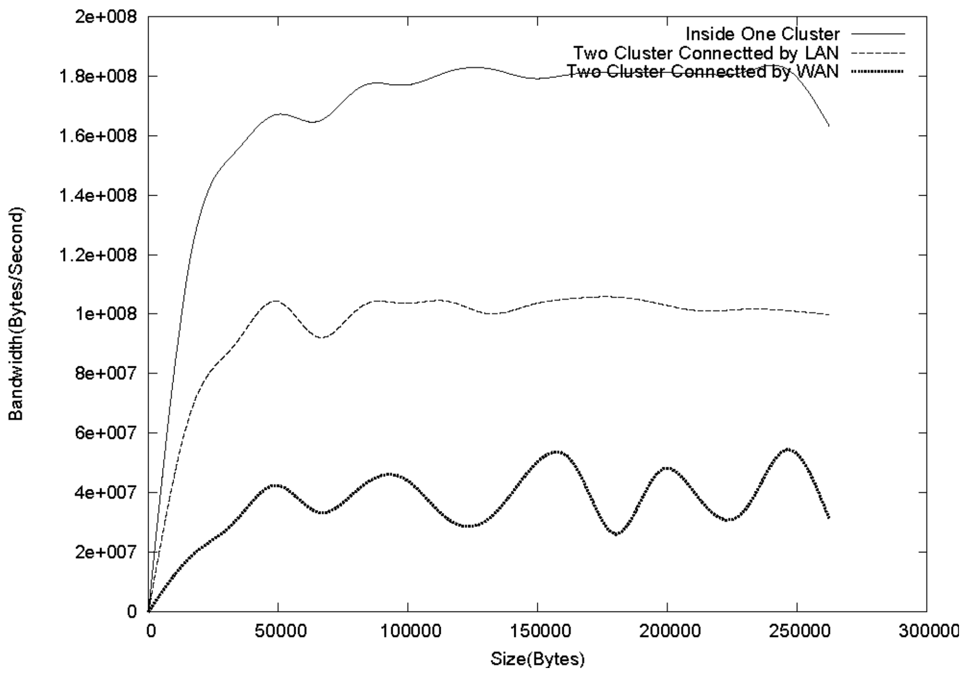


Figure 3-6. Head-to-Head Benchmark

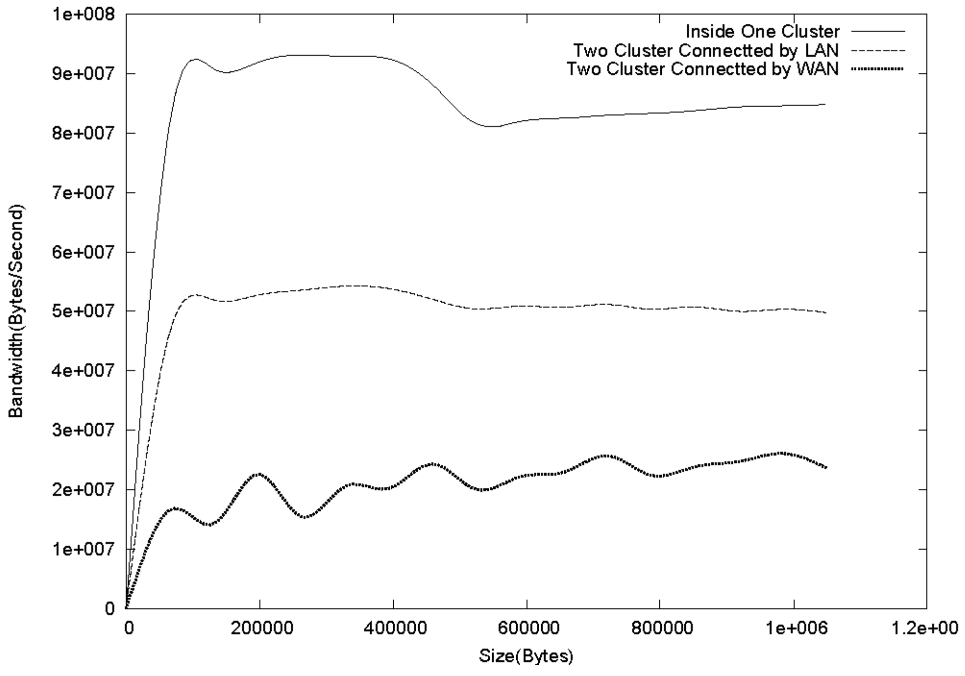


Figure 3-7. Ghost Cell Benchmark

In "round-trip" and "head-to-head"cases the results are achieved in a series of tests, measuring the available effective bandwidth and latency as a function of message size. Thus, various message sizes from 0 to 256 kbytes are used as shown in Figure 3-5, and Figure 3-6. For all cases, the bandwidth increases almost linearly as the message size increases from zero, until it reaches a peak as the message size increases beyond approximately 100 KBytes. This peak is smaller from the case (1) to (3); in other words, the maximum available bandwidth is smaller as the network latency increases.

The "ghost cell" pattern is one of the major communication patterns in following introduced parallel PIC codes. The thickness of ghost cells in this benchmark is set to be one column, and is the same in later parallel PIC code. We measure the available bandwidth versus the message size, as the message size varies from 0 to 1Mbytes in 4K Byte increments for each measurement. Figure 3-7 shows the available bandwidth for each case. For all cases, same as the above two benchmark, the bandwidth increases almost linearly as the message size increases from zero, until it reaches a peak as the message size increases beyond approximately 100 KBytes. This peak is smaller from the case (1) to (3). Sending a small message before the saturations in all three networks is more expensive than sending a long message. In the case of "inside one cluster" and "two clusters connected by LAN," the available bandwidth falls down slightly as the message size increases beyond approximately 400Kbytes. These results show that the network latency is the key factor that affects "ghost cell" pattern communication performances. Please note that the MPPtest is a pure communication test without any real computation and the ghost cell communication is the point-to-point communication between two neighboring processors. Thus, the number of processors has little effect in this communication test.

3.4.2 Collective Benchmark Test

Not all the collective communication patterns are supported by MPPtest benchmarks. Only MPI_Barrier, MPI_Bcast, MPI_Scatter and MPI_Reduce patterns are available. Here, we focus on the benchmark results of both the MPI_Barrier and MPI_Scatter. Figure 3-8 shows that, for the three types of clusters, the communication time needed for MPI_Barrier increases as the number of processors increases. This characteristic time increases almost linearly for the "inside one cluster" and "two clusters connected by LAN" cases, but increases exponentially for the "two clusters connected by WAN" case, especially when the number of processors goes beyond approximately 70. As a consequence, no scalability can be obtained for many processors to be used in the "two clusters connected by WAN" case, as the communication efficiency becomes relatively poor.

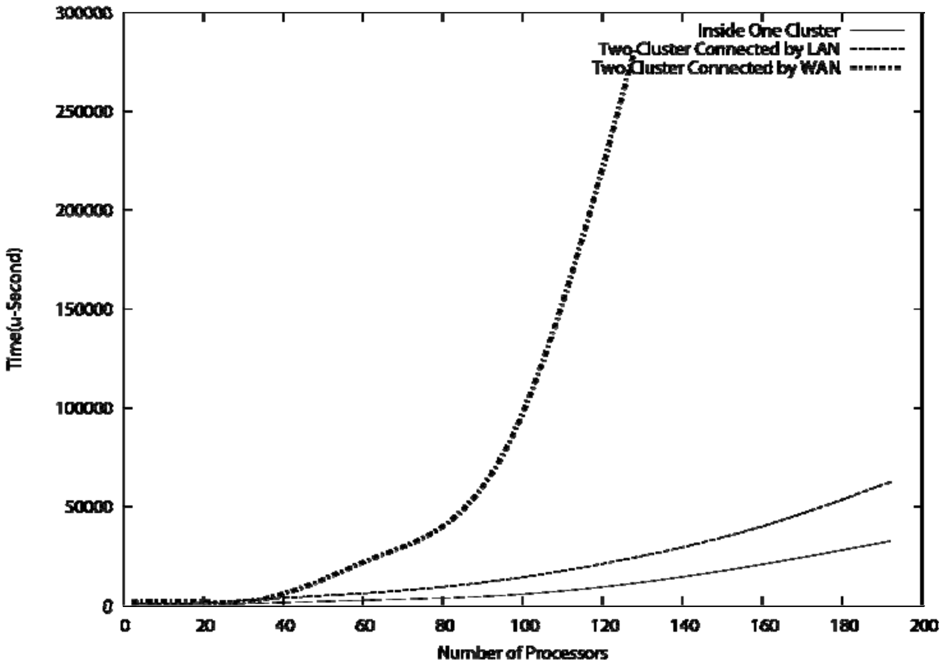


Figure 3-8. MPI_Barrier Benchmark

Figures 3-9 (a), (b), (c), and (d) show correspondingly the MPI_Scatter benchmarks with various message sizes. For (1) “inside one cluster” case and (2) “two clusters connected by LAN” case, the "MPI_Scatter" time (measured for 4k to 256k messages) increases almost linearly with the number *p* of processors. But, for (3) “two clusters connected by WAN” case, the "MPI_Scatter" time (still measured from 4k to 256k messages) increases always exponentially when the number of processors goes beyond approximately 70. However, when we increase the "MPI_Scatter" message size to and beyond 2Mbyte, Figure 3-9 (d) shows that the "MPI_Scatter" time increases near linearly as the number *p* increases for all three cases. This result shows that the "MPI_Scatter" communication time is not scalable from small to large size message in the (3) “cross-WAN” case. More precisely, the “cross-WAN” like configuration is not appropriate for high number of processors (at least p <70) managing small size messages (<256KByte).

From the above collective communication benchmarks, we confirm that not only the network latency but also the number of participating processors is the key factor that affects the communication performances. More network latency the system has more the increase of the number of processor affects the network performances. The collective communication patterns do not scale well on the system with poor communication performance like the “cross-WAN” PC cluster benchmarked in this paper.

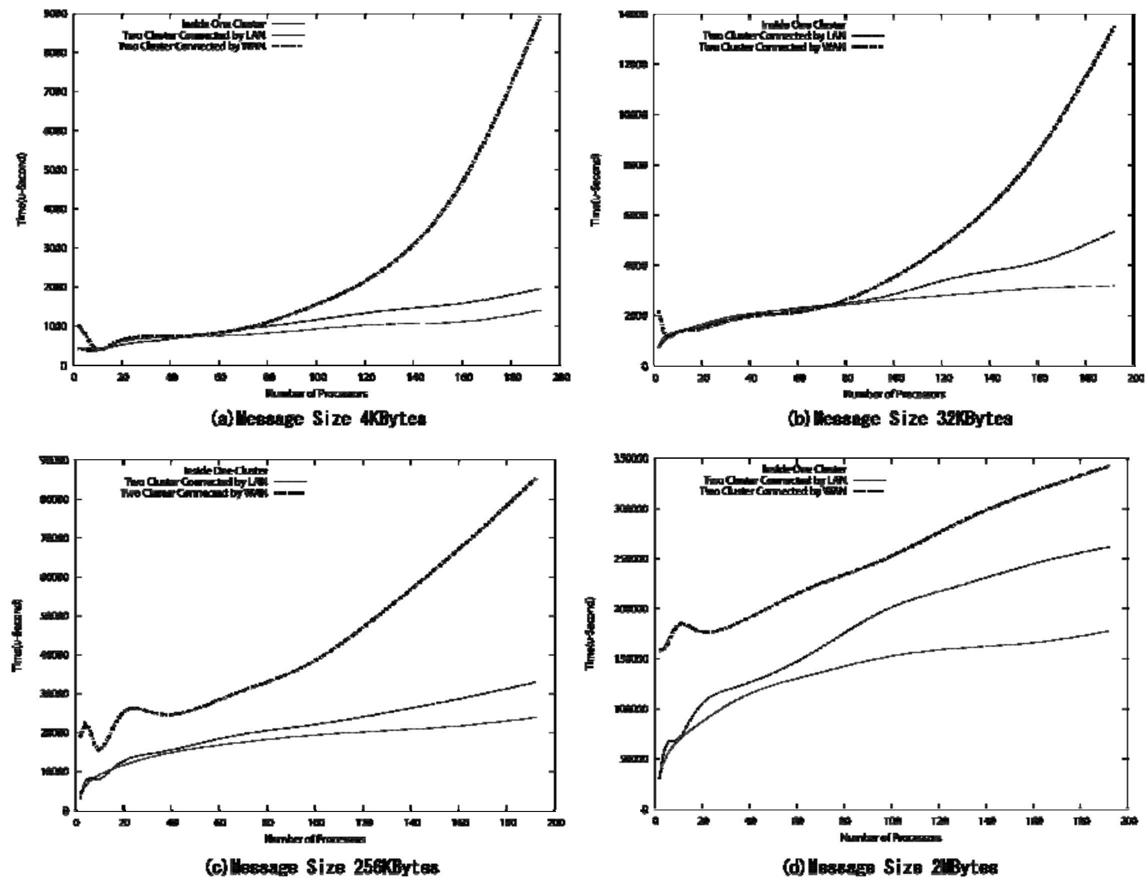


Figure 3-9. MPI_Scatter Benchmark. The message sizes are (a) 4k, (b) 64k, (c) 256k, and (d) 2M bytes

Chapter 4: Scalability Analysis and Benchmark of Parallel PIC Code on Computational Grid

4.1 General Concurrent Particle-In-Cell Algorithm

Since PIC codes are extremely intensive in memory and computation by their nature, the availability of computing resources to perform PIC codes with a large system is essential whether they can be used for precision predictions rather than for quantitative indicators of system behaviors. The general concurrent Particle-In-Cell (GCPIC) algorithm [*Liewer, 1989*] is a method for dividing such plasma particle-in-cell simulation problems among the p processors with distributed memory. In a distributed memory parallel computers, each processors has its own local memory; There is no global or shared memory.

The key feature of the GCPIC algorithm is that two distinct spatial decompositions of physical domain are used to map the problem onto the parallel processors. The two decompositions mirror the two stages of a PIC code: a primary decomposition is used to divide the particles and particles computation efficiently among the processors and a secondary decomposition is used to divide the electromagnetic field computation among the processors. The primary decomposition is designed to make the dominant portion of a PIC code, the particle push, run efficiently in parallel. For the primary decomposition, the physical domain of the simulation is divided p sub-domains such that these sub-domains have roughly equal numbers of particles. For problems with non-uniform particle densities, these sub-domains will be of unequal physical size, and, in general, will contain unequal numbers of grid points. Figure 4-1 shows such sub-domains for an eight node parallel computer for 1-dimensional decomposition with uniformly spaced grid points. In this example, the eight sub-domains have equal numbers of particles, but unequal numbers of grid points. Each processor is assigned a sub-domain and is responsible for storing and pushing the particles in this sub-domain and, in addition, for storing the electromagnetic field arrays (charge densities, electric fields, etc.) for the grid points of its assigned sub-domain. As a particle moves to a new sub-domain, the particle is passed to the processor assigned that sub-domain in this primary decomposition. The primary decompositions is also used for most of the diagnostics.

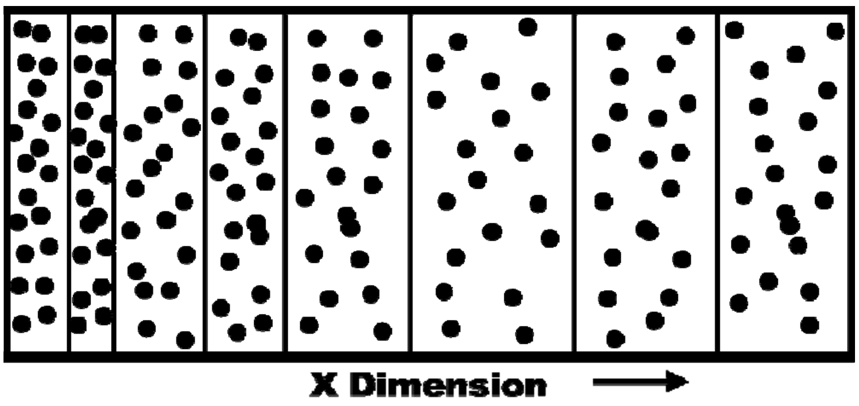


Figure 4-1. Examples of primary sub-domains created for a 8-node parallel computer using GCPIC algorithm for simulation with uniform grid spacing

The primary domain decomposition of the GCPIC algorithm, optimized for the concurrent particle computations, will not generally be the optimum decomposition for solving the electromagnetic field equation in parallel. Therefore, in the GCPIC algorithm, a second decomposition is used for the field solution. This secondary domain decomposition is chosen to make the field solve dividing the number of grid cells equally among the processors. Figure 4-2 shows the secondary decomposition for the same parallel computer. The specific decomposition chosen will be different depending on weather a fast Fourier transform (FFT) method, a finite difference method, or a multi-grid method is used.

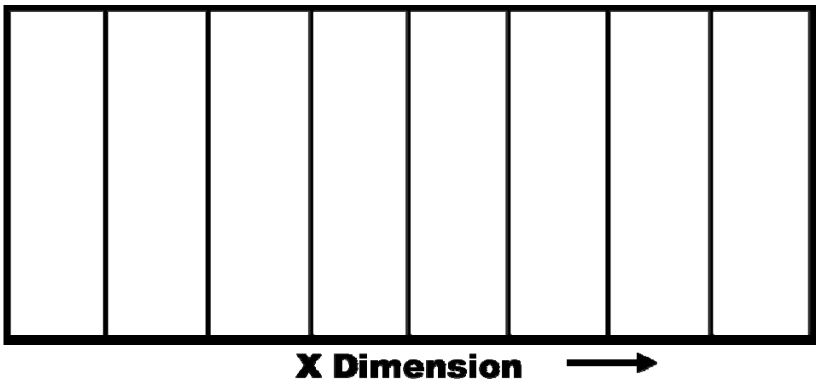


Figure 4-2. Examples of secondary sub-domains created for a 8-node parallel computer using GCPIC algorithm for simulation with uniform grid spacing

At each time step, prior to solving the field equations, the relevant grid arrays (charge, current densities) are passed among the processors and redistributed as needed in secondary decomposition. After the field equations are solved, the electromagnetic field arrays must be passed among the processors so that they are distributed as needed in the primary decomposition for the parallel particle computation. These two redistributions of grid arrays between the primary and secondary decomposition at each time step are inherent in the GCPIC algorithm and are necessary to make both the particle and field portions of a PIC code efficient on a parallel computer. Under certain circumstances, the primary and secondary decompositions will be the same and thus no redistributions are necessary. In this case, parallel code development would be simplified.

4.2 Two Three-Dimensional Particle-In-Cell Codes

4.2.1 Skeleton PIC Code

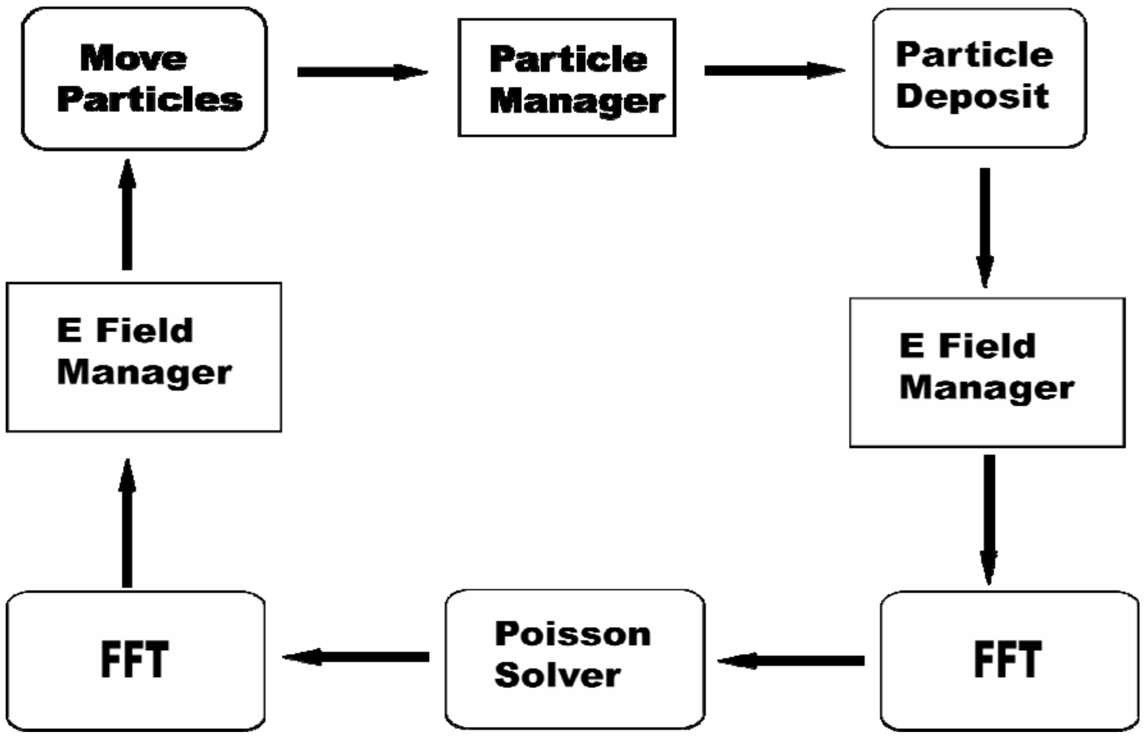


Figure 4-3. Flowchart of the skeleton PIC code.

The skeleton PIC code [*Decyk, 1995*] has been developed for parallel computers to provide a test bed where the new code can be developed and tested, and new computer architectures can be evaluated. This version of the code has been deliberately kept as simple as possible, but it includes all the essential pieces for which codes need to be developed. The code advances only the particles (Lorentz force is applied to each particle, also named herein as "particle pusher"), deposits their charge, and solves the fields using only the Poisson equation. The code is the electrostatic code, and the boundary conditions are periodic, which allows us to use an FFT. The flowchart of the skeleton PIC code is illustrated in Figure 4-3. The field and particle managers, where the electric field and particle data are, respectively, sent to the nearest neighbor processors, is characterized as "ghost cell" communication patterns. The FFT part which uses the parallel transpose algorithm is characterized by all-to-all collective communications patterns.

4.2.2 Parallel TRISTAN PIC Code

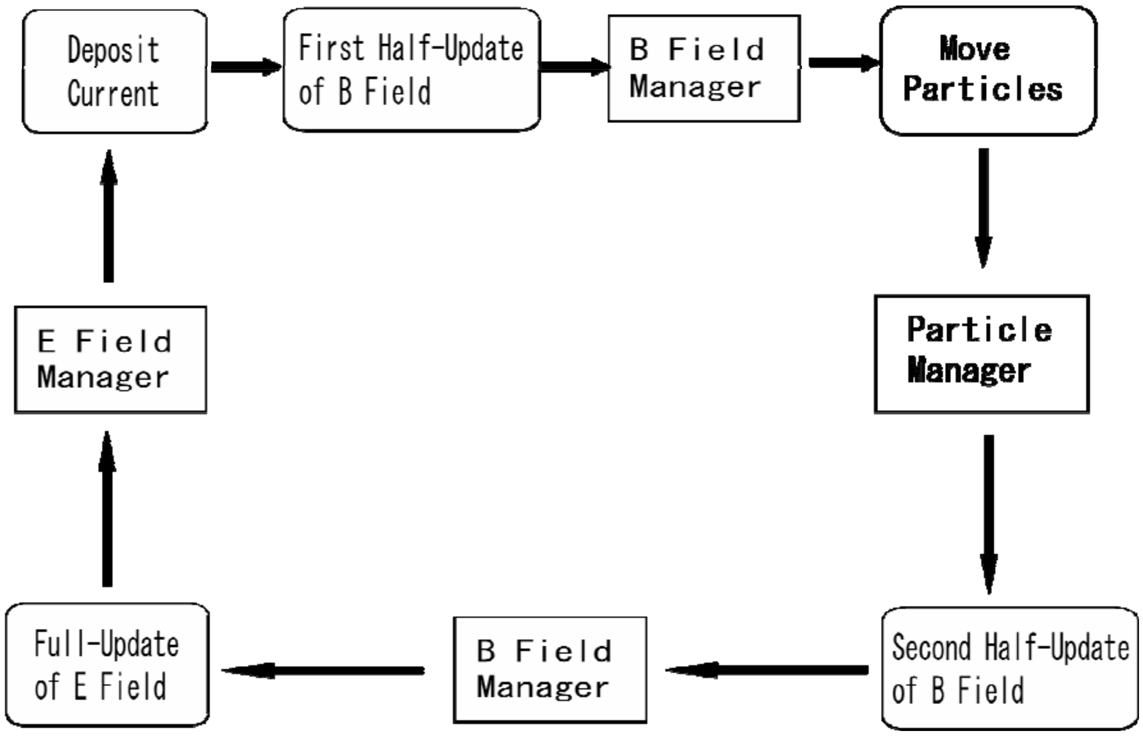


Figure 4-4. Flowchart of the parallel TRISTAN PIC code.

The TRISTAN code [*Cai, 2003*] is a three-dimensional full EM particle code. It solves the full Maxwell equations based on particle motions pushed self-consistently with the Lorentz equation. The code uses the finite difference time domain (FDTD) method, which advances a solution of Gauss' law self-consistently forward in time. A rigorous charge conservation method [*Villasenor, 1992*] for the current deposition is implemented. By injecting only charge-neutral plasma into the simulation box, it does not require the actual global Poisson solver that would be required at the beginning of the simulations. The full description can be referred in the chapter 2.This code has been parallelized with MPI, and as the Poisson solver is not needed in this code, no all-to-all collective communications are needed in contrast to the skeleton PIC code. This difference will be a key point which reveals strong impacts on the corresponding benchmarks as will be shown in the following sections. The flowchart of the TRISTAN PIC code is illustrated in Figure 4-4, where the magnetic, electric field, and particle data are, respectively, sent to the nearest neighbor processors in their managers.

4.2.3 Comparison of Two PIC Codes

The TRISTAN code is an explicit EM code, and the Courant condition is required to resolve light waves. On the other hand, the skeleton PIC code is an ES code, and is required to solve only the fastest electrostatic plasma wave. In addition, in the TRISTAN code, currents are obtained from particle motions to obtain the field, whereas in the skeleton PIC code, charges are obtained from particle positions to solve Poisson's equation.

Please also note that the Poisson solver based on FFT is only one of various methods to solve Poisson's equation. For example, Krylov solvers or multigrid solvers [*Henson, 2002*] yield good performance and scaling on parallel computers, as they do not require the global transpose communications. These two PIC codes are chosen only because they have different communication behaviors: one uses point-to-point communication pattern and another uses all-to-all collective communication pattern.

Both codes are parallelized in the same way described in GCPIC algorithm. It is important to understand how the use of collective communication patterns and point-to-point communication patterns will impact the scalability of the codes on computational Grid. In the chapter 3.4, mpptest benchmarks give us the preview performance of both communication patterns independent of real applications. In order to investigate the different communication patters in the PIC codes on various computational Grids with different communication performances, two PIC codes and two computational Grids are chosen. These two PIC codes provide a good set for a case study where we can investigate the possibility for running large-scale PIC simulations on a computational Grid.

4.3 Scalability Analysis of Two PIC codes

A sequential algorithm is usually evaluated in terms of its execution time, expressed as a function of the size of its input. The execution time of a parallel algorithm depends not only on input size but also on the number of processing elements used, and their relative computation and inter-process communication speeds. Hence, a parallel algorithm cannot be evaluated in isolation from a parallel architecture without some loss in accuracy. A parallel system is the combination of an algorithm and the parallel architecture on which it is implemented. Scalability of a parallel system is a measure of its capacity to increase speedup in proportion to the number of processors. The isoefficiency concept was first introduced by Kumar and Rao [Kumar, 94]. It is especially useful and a popular metrics in analyzing scalability of parallel algorithm-architecture combinations. There are three major applications of the isoefficiency analysis:

1. Evaluation of a relationship between an algorithm and an architecture on which the algorithm will be implemented; one can test the performance of a parallel program on few processors and then predict its performance on a larger number of processors
2. Evaluation of a degree of concurrency in a parallel algorithm
3. Evaluation of changes in hardware parameters, i.e. study of parallel systems with respect to changes in hardware parameters such as: speed of processors and speed of communication channels.

In the following section, we give the isoefficiency analysis of two parallel PIC codes on computational Grids.

4.3.1 Performance Metrics of Parallel System

4.3.1.1 Nomenclatures

Here first of all, we list some key nomenclature needed for the isoefficeincy analysis as follows:

$\boldsymbol{T_{se}}$: computer time spent for a code using the best sequential algorithm (i.e. not parallel version);

$\boldsymbol{W}$: “problem size” defined as the number of basic computational steps or computational time used in the best sequential algorithm;

$\boldsymbol{p}$: number of processors used in parallel computing;

$\boldsymbol{T_p}$: total computer time spent for a code using the parallel algorithm of a PIC code on all $\boldsymbol{p}$ processors;

$\boldsymbol{S}$: speedup

$$\boldsymbol{S} = \frac{\boldsymbol{T_{se}}}{\boldsymbol{T_p}} \tag{4.1}$$

$\boldsymbol{E}$: efficiency

$$\boldsymbol{E} = \frac{\boldsymbol{S}}{\boldsymbol{p}} \tag{4.2}$$

$\boldsymbol{T_o(W, p)}$: total overheads when the parallel algorithm is used. It is function of $\boldsymbol{W}$ and $\boldsymbol{p}$ parameters. It includes the cost spent for any communication overheads, load balancing overheads, and any extra computation overheads. Commonly, we have:

$$\boldsymbol{p \times T_p = T_{se} + T_o(W, p) = W + T_o(W, p)} \tag{4.3}$$

$\boldsymbol{B}$: is the amount of digital data per time unit that is delivered over a physical or logical link, or that is passing through a certain network node.

4.3.1.2 Overhead of Parallel Computing

Using twice as many hardware resources, one can reasonably expect a program to run twice as fast. However, in typical parallel programs, this is rarely the case, due to a variety of overheads associated with parallelism. An accurate quantification of these overheads is critical to the understanding of parallel program performance. In addition to performing essential computation (i.e., computation that would be performed by the serial program for solving the same problem instance), a parallel program may also spend time in inter-process communication, idling, and excess computation (computation not performed by the serial formulation).

Inter-process Interaction: Any nontrivial parallel system requires its processing elements to interact and communicate data (e.g., intermediate results). The time spent communicating data between processing elements is usually the most significant source of parallel processing overhead.

Idling: Processing elements in a parallel system may become idle due to many reasons such as load imbalance, synchronization, and presence of serial components in a program.

Excess Computation: The fastest known sequential algorithm for a problem may be difficult or impossible to parallelize, forcing us to use a parallel algorithm based on a poorer but easily parallelizable (that is, one with a higher degree of concurrency) sequential algorithm. The difference in computation performed by the parallel program and the best serial program is the excess computation overhead incurred by the parallel program.

4.3.1.3 Isoefficiency Definition

Based on the definition of speedup $\boldsymbol{S}$, efficiency $\boldsymbol{E}$ and overhead function $\boldsymbol{T_o(W, p)}$, we have:

$$E = \frac{S}{p} = \frac{T_{se}}{p \times T_p} = \frac{W}{W + To(W, p)} . \qquad (4.4)$$

From this formula two conclusions concerning efficiency can be obtained:

1. If problem size W is constant then increase in the number of processors p must lead to decrease in efficiency because overhead $T_o(W, p)$ increases with growing p.

2. If problem size W increases and at the same time number of processors p is constant then efficiency may increase the overhead $T_o(W, p)$ grows slower than W for fixed p.

For a desired (fixed) efficiency value E, we get W:

$$W = \frac{E}{1-E} \times T_o(W, p) . \qquad (4.5)$$

If $K = \frac{E}{1-E}$ is a constant (assuming the efficiency E is constant), we have:

$$W = K \times T_o(W, p) . \qquad (4.6)$$

Since T_o is a function of both the terms W and p, the problem size W can be explicitly expressed as $W = W(p)$. We called the function $W = W(p)$ an isoefficiency function. This function determines the growth rate of W, which keeps the efficiency E fixed as the number p of processors increases. A lower growth rate is more desirable than a higher one, since the lower rate indicates the system is highly scalable, while the higher rate means poorly scalable. Of course, the overhead function $T_o(W, p)$ may have multiple terms. If we balance the isoefficeincy W against each term of T_o and compute the respective isoefficiency functions for individual terms, the components of T_o (that requires the problem size to grow at the highest rate with respect to an increasing p) determine the overall asymptotic isoefficiency function of the concerned parallel system.

4.3.2 Scalability Analysis of Two PIC Codes

The overhead function T_o of the skeleton PIC code can be divided into three parts: (1) the "ghost cell" communication time T_{field} in the field manager and $T_{particle}$ in the particle manager; (2) the communication time of FFT transpose T_{pfft}; and (3) any other overheads T_{others}, which are usually trivial. In (1) the "ghost cell" communication, all the electric field data and particle data at the decomposition boundaries are sent to the nearest neighbor processor. In (2) the FFT transpose, the all-to-all transpose communication will be done on the common network system of the computational Grid discussed here. Thus, no hardware support for all-to-all operations like in some supercomputers is assumed. The isoefficiency function defined for the skeleton PIC code in this Grid architecture can be rewritten as:

$$W = K \times (T_{field} + T_{particle} + T_{pfft} + T_{others}) \tag{4.7}$$

Correspondingly, the overhead function T_o of the parallel TRISTAN PIC code can be divided into three parts without the FFT transpose. The isoefficiency function of TRISTAN code is:

$$W = K \times (T_{field} + T_{particle} + T_{others}) \tag{4.8}$$

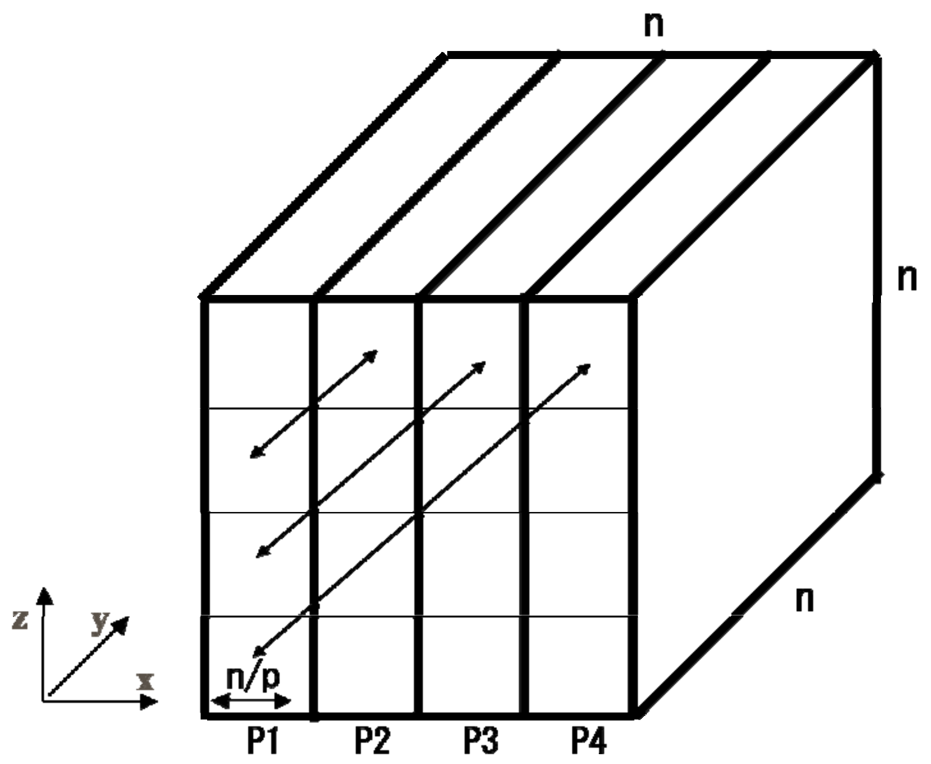


Figure 4-5. Transpose global communication used in FFT in the skeleton PIC code. Here, $\frac{n}{p} \times \frac{n}{p} \times n$ blocks are exchanged diagonally. In this figure, the grid sizes in x, y and z is approximated to be n. The computational domain is decomposed in x.

For a PIC code, T_{others} is negligible for the network of concern; however, let us note that T_{others} can not be negligible any more, if a different network is used. Here, we assume that the field size for each dimension is approximated by n, the problem size

$W \propto n^3$, and only one-dimensional domain decomposition is concerned here. Figure 4-5 shows the communication patterns that transpose the data from xyz to zyx. More precisely, in Figure 4-5, p processors send $m = \frac{n}{p} \times \frac{n}{p} \times n$ double floating-point (8 Bytes) data to the other $p-1$ processors.

The time required for the complete transfer of a message containing $m \times 8$ bytes between two processors is given approximately by $ts + tw \times m$. Here,

ts: start up time defined as the cost of starting a TCP (Transfer Communication Protocol) connection. This time value is measured as the difference between the time at which a command is sent and the time at which this command starts to be executed;

tw: per-word transfer time can be defined as;

$$tw = \frac{1}{B}. \tag{4.9}$$

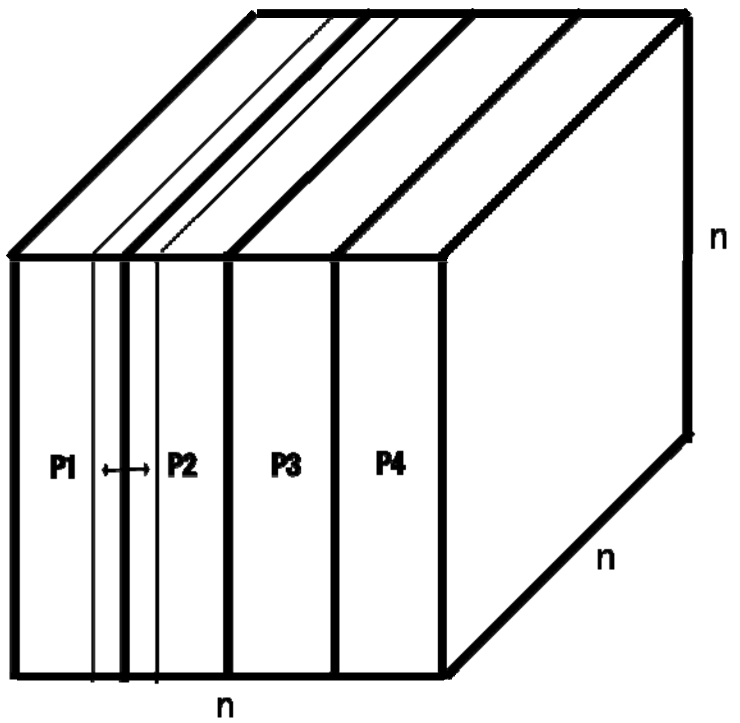


Figure 4-6. The ghost cell communication pattern for one dimensional decomposition in x. In this figure, the grid sizes in x, y, and z are approximated to be n. The computational domain is decomposed into the x direction.

For simplicity, we assume that no two messages can be concurrently on the network and define the total communication cost of T_{pfft} as:

$$T_{pfft} = (ts + tw \times m) \times (p-1) \times p \approx ts \times p^2 + tw \times n^3 \tag{4.10}$$

The isoefficiency function is defined to be $\boldsymbol{W} = \boldsymbol{W}(\boldsymbol{p})$, “$\boldsymbol{tw}$” can be determined by the network speed of the system, and the last term “ $\boldsymbol{tw} \times \boldsymbol{n}^3$” in $\boldsymbol{T}_{pfft}$ has no “ $\boldsymbol{p}$ ” dependence. Thus it can be neglected. Clearly the isoefficiency function due to the first two terms in $\boldsymbol{T}_{pfft}$ is given by:

$$\boldsymbol{W}_{Tpfft} \propto \boldsymbol{ts} \times \boldsymbol{p}^2 \tag{4.11}$$

Both the terms $\boldsymbol{T}_{field}$ and $\boldsymbol{T}_{particle}$ are the same communication patterns necessary for “ghost cell” communications. Figure 4-6 shows that each processor sends and receives one ghost cell sheet to the neighbor processors for particle-force linear interpolation and field updates etc., and each message size is approximately proportional to $\boldsymbol{n}^2$. Thus the field communication time $\boldsymbol{T}_{field}$ and $\boldsymbol{T}_{particle}$ is:

$$\boldsymbol{T}_{field}, \boldsymbol{T}_{particle} = (\boldsymbol{ts} + \boldsymbol{tw} \times \boldsymbol{m}) \times \boldsymbol{p} = \boldsymbol{ts} \times \boldsymbol{p} + \boldsymbol{tw} \times \boldsymbol{n}^2 \times \boldsymbol{p} \tag{4.12}$$

Since we know typically $\boldsymbol{W} \propto \boldsymbol{n}^3$ assuming the number of particles in cell is fixed in three dimensions, the isoefficiency function due to $\boldsymbol{T}_{field}$ and $\boldsymbol{T}_{particle}$ is:

$$\boldsymbol{W}_{Tfield}, \boldsymbol{W}_{Tparticle} \propto \boldsymbol{ts} \times \boldsymbol{p} + \boldsymbol{tw} \times \boldsymbol{n}^2 \times \boldsymbol{p} \tag{4.13}$$

In a high throughput network environment that we consider in the present benchmarks, the term $\boldsymbol{tw}$ $(10^{-8} \sim 10^{-9}\boldsymbol{s})$ is very small and negligible as compared with the terms $\boldsymbol{p}$ $(10^2 \sim 10^3 s)$, $\boldsymbol{ts}$ and $\boldsymbol{th}$ (last both of the order of $10^{-3}\boldsymbol{s}$). Thus, the last term in Eq. (4.13) $(\boldsymbol{tw} \times \boldsymbol{n}^2 \times \boldsymbol{p})$ can be ignored. Eq. (4.13) leads to the isoefficiency function of $\boldsymbol{T}_{field}$ and $\boldsymbol{T}_{particle}$:

$$\boldsymbol{W}_{Tfield}, \boldsymbol{W}_{Tparticle} \propto \boldsymbol{ts} \times \boldsymbol{p} \tag{4.14}$$

In a scalable system, comparing the FFT transpose term $(\boldsymbol{T}_{pfft})$ with the ghost cell communication terms ($\boldsymbol{T}_{field}$ and $\boldsymbol{T}_{particle}$), we confirm that the transpose communication in the skeleton PIC code may cause a worse scalability than that without the FFT field solvers.

The scalability of skeleton PIC code is mainly affected by the term "$\boldsymbol{ts} \times \boldsymbol{p}^2$" from Eq.

(4.11). Similarly, the scalability of TRISTAN PIC code in a high throughput network is mainly affected by the term "$ts \times p$" from Eq. (4.14). However, with a very low throughput network, the parallel TRISTAN code may have a poor scalability due to the term "$(tw \times n^2 \times p) \approx (tw \times p)^3$" in Eq. (4.13).

4.4 Benchmark of Two PIC codes on Computational Grids

In these benchmarks, the problem size W of grid size $64 \times 64 \times 64$ and 8 particles/cell is normalized to be 1. The problem size W range up from 1 to 2^9 (from the grid size $64 \times 64 \times 64$ to $512 \times 512 \times 512$) keeping 8 particles/cell unchanged.

4.4.1 Speedup Benchmarks

4.4.1.1 Skeleton PIC Code in Cluster enabled Grid

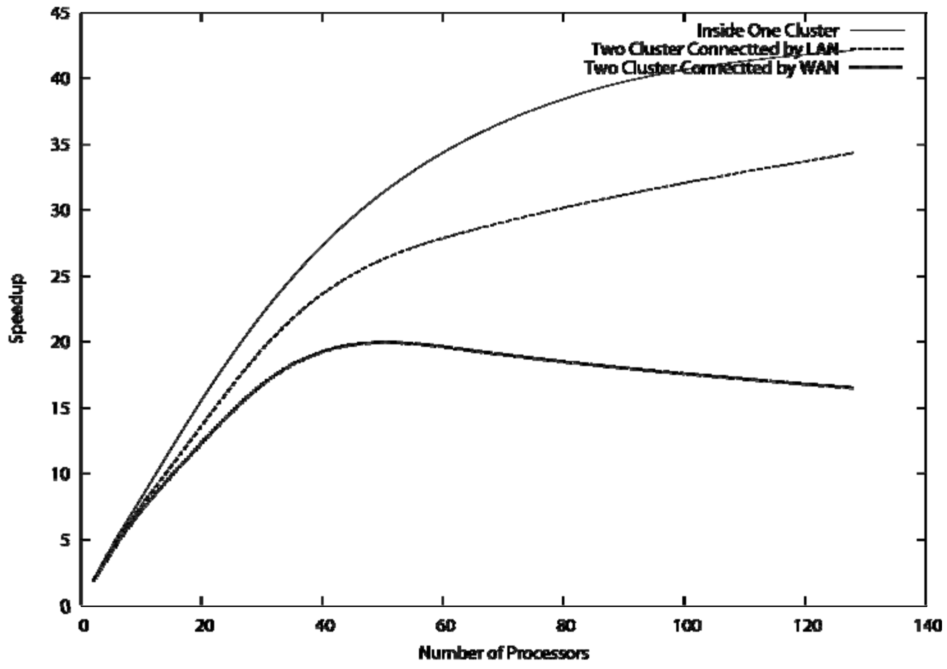


Figure 4-7. Speedups of skeleton PIC code in the cluster-based Grid. The problem size is fixed to be 16.

We first run the speedup experiment of skeleton PIC code on three cluster systems described in the subsection 3.4. In this benchmark, we use $W = 16$ (i.e. 256x128x128 meshes). The benchmark results are illustrated in Figure 4-7. It shows that the speedups of the three systems saturate, as the number p of processors increases. The speedups of both "two clusters connected by LAN" and "two clusters connected by WAN" are lower than "inside one cluster", and there is even a fall in the case of "two clusters connected by WAN" when the number p exceeds about 40. The increased network latency obviously affects the speedups.

The "ghost cell" communication times in particle and field managers can be estimated using MPPtest halo bandwidth measurement shown in Figure 3-9. For the problem size $\boldsymbol{W}$ =16, the message size of "ghost cell" communication in the field manager is approximately 384 Kbytes for each time step. The message size of "ghost cell" communication in the particle manager varies in time step, but is always slightly larger than that of the field manager. As shown in Figure 3-9, at message size = 384Kbytes, the "ghost cell" communication bandwidth has already been saturated, and is almost constant beyond this value. Thus, the ghost cell communications in this benchmark are independent of the number p, and has little effect on the speedup saturation for the case ``two cluster connected by WAN".

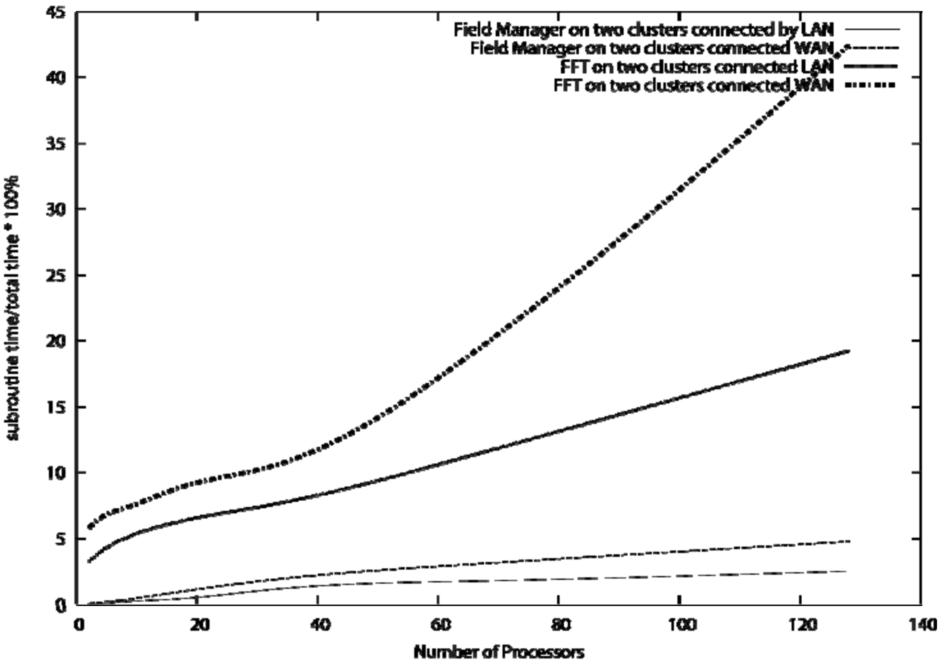


Figure 4-8. The run time percentages of the Field manager subroutine and FFT subroutine in the skeleton PIC code running on two clusters connected by LAN and WAN in the cluster-based Grid. The Field Manager subroutine adopts the ghost cell communication patterns, and FFT subroutine adopts the collective communication patterns.

The collective communication performed in the parallel transpose FFT method is "all-to-all communication." As no corresponding test can be performed in MPPtest benchmark for the cluster based Grid, an equivalent and simple way consists in performing $\boldsymbol{p}$ times MPI_Scatter operation ($\boldsymbol{p}$ is the number of processors). From Figure 4-5, for a constant problem size $\boldsymbol{W} \sim \boldsymbol{n}^3$ and using $\boldsymbol{p}$ processors, the message size needed in communication at each MPI_Scatter is $\frac{n^3}{p^2}$. For the problem size W =16, the message size for 16 processors to perform an MPI_Scatter operation of the electric field Ex is approximately 2Mbytes (shown in Figure 3-11(d)), and then repeats 16 times for one all-to-all communication. However, for 128 processors, this size is reduced to approximately 256Kbytes (shown in Figure 3-11(c)), and then repeats 128 time MPI_Scatter operations for one all-to-all communications. It means that for a constant problem size, if more processors are used in MPI_Scatter operation, smaller is the message size and more repeat times are needed. Figure 3-11 shows that "MPI_Scatter" operation does not scale well for the small size message on large number of processors for the case (3) two clusters connected by WAN, because for the smaller message size (< 256K) the "MPI_Scatter" communication time grows exponentially for large number p of processors (p> 60). This number p also corresponds to the saturation of the speedup in Figure 4-7 for the case "tow clusters connected by WAN". This is also evidenced by Figure 4-8. In the figure, only the run time percentage of the FFT subroutine in the "two clusters connected by WAN" increases rapidly when p>60. This benchmark result confirms us that the skeleton PIC code in large network latency system like the "cross-WAN" PC cluster does not scale well when the number p exceeds approximately 60.

4.4.1.2 Skeleton PIC code and TRISTAN code running across WAN

We then focus on the speedup of the problem size W =16 (i.e. 256x128x128 meshes) obtained for skeleton PIC code on both two clusters connected by WAN and supercomputer based Grid, and TRISTAN code on two clusters connected by WAN. Corresponding speedup results are plotted in Figure 4-9. The speedups of the skeleton PIC code saturate around 20 to 30 on these Grid systems as the number p exceeds around 60. The speedup of the TRISTAN PIC code is obviously higher than that of the skeleton PIC code running on the same cluster based Grid system. Based on the analysis in section 4.3, the collective communication costs caused by the FFT transform method may be the reason for the speedup saturation of the skeleton PIC code shown in Figure 4-9. This benchmark confirms that the TRISTAN PIC code scales better than that of skeleton PIC code in large network latency system like the two clusters connected by WAN.

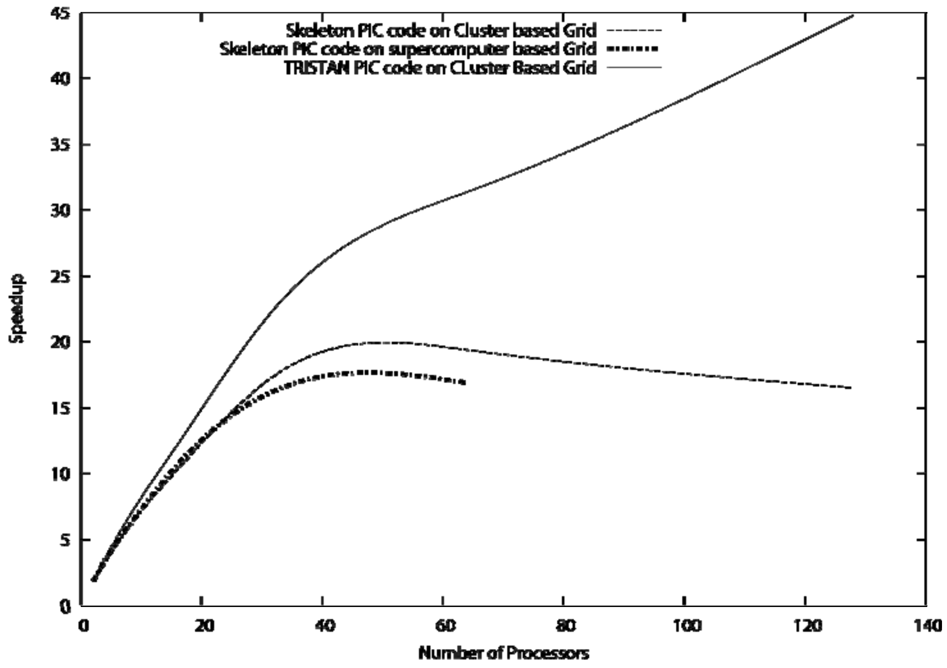


Figure 4-9. Speedups for skeleton PIC and TRISTAN code on two clusters connected by WAN in the cluster-based Grid, and the skeleton PIC code in the supercomputer-based Grid. The problem size is fixed to be 16.

4.4.2 Scalability Benchmarks

Here, we perform the isoefficiency benchmark tests and compare them with the isoefficiency analysis presented in Section 4. Three benchmark tests are performed as follows: First, a benchmark of the parallel skeleton PIC code on the supercomputer based Grid system; Second, a benchmark of the same parallel skeleton PIC code on the cluster enabled Grid system; Third, a benchmark of the parallel TRISTAN PIC code on the cluster enabled Grid system. The processor assignments to two Grid systems are listed in Tables 3, 4. The round trip time (RTT) between Kyoto node and Nagoya node is 7.3 ms on supercomputer based Grid. The RTT between "Tau" and "F32" is 2.1 ms on cluster based Grid.

Processors	2	4	8	16	32	64
Nagoya1	1	1	2	4	10	26
Nagoya2	0	1	2	4	10	26
Kyoto	1	2	4	8	12	12

Table 4-1. Processors' assignment of supercomputer-based Grid. The 'Processors' row indicates the total number of processors used. The node names 'Naogoya1', 'Naogoya2' and 'Kyoto' indicate the number of processor provided from the nodes.

Name	Organization	CPU Type	Memory	Nodes/CPUs	Queue System
F32	AIST	Xeon 3.0GHz (Dual)	4GB	128/256	SGE
Sakura	AIST	Opteron 2.2GHz (Dual)	2GB	16/32	SGE
Tau	Univ. Tokyo	Xeon 2.4GHz (Dual)	2GB	175/350	SGE
Chikayama	Univ. Tokyo	Xeon 2.4GHz (Dual)	1GB	64/128	SGE

Table 4-2. Processors' assignment of cluster-based Grid. The 'Processors' row indicates the total number of processors used. The node names 'Tau' (AIST) and 'F32' (University of Tokyo) indicate the number of processor provided from the nodes.

The benchmarks results are expressed as the problem size $\boldsymbol{W}$ versus the number $\boldsymbol{p}$ of assigned processors in Tables 4-3, 4-4, and 4-5. In Tables 4-3, 4-4, and 4-5, when increasing simultaneously the number $\boldsymbol{p}$ of processors and the sizes $\boldsymbol{W}$ of the problem, a line or curve where the efficiency is approximately kept constant can be found. This ability to maintain the efficiency $\boldsymbol{E}$ at a fixed value by simultaneously increasing the number $\boldsymbol{p}$ of processors and the size $\boldsymbol{W}$ of the problem is exhibited by many parallel systems, and we call this "scalable". The scalability of a parallel system is a measure of its capacity to increase speedup in proportion to the number of processor. For different parallel systems and programs, $\boldsymbol{W}$ must be increased at different rate with respect to $\boldsymbol{p}$ in order to maintain a fixed efficiency in each system as we discussed in Section 4.3. This rate determines the scalability of the parallel system. Lower the value of this rate is, better the performance of this parallel system is. Once the fixed efficiency in each benchmark determined from the tables, the function dictates the growth rate of $\boldsymbol{W}$ required to keep the efficiency fixed as $\boldsymbol{p}$ increases is the isoefficiency functions of the parallel programs and plotted in Figure 4-10.

The isoefficiency curves in Figure 4-10(a) are respectively (i) for $\boldsymbol{E} = 0.56$ the parallel skeleton PIC code running on the supercomputer based Grid, (ii) for $\boldsymbol{E} = 0.61$ the parallel skeleton PIC code running on the cluster based Grid (two clusters connected by WAN), and (iii) for $\boldsymbol{E} = 0.70$ the parallel TRISTAN PIC code running on the same cluster based Grid. Figure 4-10 (b), (c), and (d) compares the theoretical isoefficeincy curves obtained in section 4.3 with the benchmark results of the cases (i), (ii), and (iii), respectively.

The cluster based Grids show nearly the same isoefficiency function as in supercomputer based Grid (for the skeleton PIC code benchmark). This provides that the cluster based Grid with high throughput network connection in the campus Grid and the metropolitan Grid are also good choices to run large scale PIC simulations with a good cost-performance. The isoefficiency curve of the parallel TRISTAN PIC code with

$\boldsymbol{E} = 0.70$ on a cluster based Grid increases more slowly than the isoefficiency curve of the parallel skeleton PIC code on the same cluster based Grid. This means that the parallel TRISTAN PIC code is more scalable than the skeleton code, since there is no all-to-all collective communication in the parallel TRISTAN PIC code.

E \ P W	2	4	8	16	32	64
1	0.8693	0.7221	0.5707	0.3764		
2	0.8902	0.8099	0.6731	0.4432	0.2828	
4	0.9009	0.8307	0.7085	0.5054	0.3312	
8	0.9062	0.8536	0.7178	0.5591	0.3671	
16	0.9261	0.8712	0.7642	0.6061	0.4151	0.2171
32		0.9041	0.8039	0.6738	0.4782	0.2907
64		0.9131	0.8393	0.7195	0.5587	0.3608
128			0.8513	0.7443	0.5845	0.4177
256			0.8843	0.7822	0.6382	0.4781
512				0.8103	0.6987	0.5477

Table 4-3. The benchmarked efficiency **E** *for the skeleton PIC code running on the supercomputer-based Grid varying both the problem size* **W** *and processor* **p** *. The circled values are the approximated isoefficiency values in this benchmark.*

E \ P W	2	4	8	16	32	64	128
1	0.8215	0.6643	0.6135	0.4424			
2	0.8326	0.6873	0.6553	0.5764	0.3247		
4	0.8721	0.7255	0.6934	0.6154	0.4224		
8	0.8947	0.7524	0.7153	0.6997	0.4994	0.2311	
16	0.9122	0.8288	0.7692	0.7102	0.5454	0.3035	0.1287
32	0.9236	0.8776	0.8101	0.7202	0.6221	0.341	0.1978
64		0.8976	0.8393	0.761	0.6917	0.4107	0.2572
128			0.8842	0.803	0.7464	0.4876	0.3272
256				0.8842	0.7721	0.5327	0.3619
512					0.8042	0.6018	0.4072

Table 4-4. The benchmarked efficiency **E** *for the skeleton PIC code running on the cluster-enabled Grid varying both the problem size* **W** *and processor* **p** *. The circled*

values are the approximated isoefficiency values in this benchmark.

	2	4	8	16	32	64	128
1	0.8245	0.7953	0.7074	0.6244	0.4655		
2	0.8321	0.8179	0.7568	0.6553	0.5532		
4	0.8767	0.8322	0.7935	0.6933	0.6255		
8	0.8973	0.868	0.8254	0.7452	0.667	0.4532	
16		0.893	0.8426	0.762	0.7026	0.4906	
32			0.8713	0.7888	0.7322	0.5689	
64				0.8165	0.7536	0.6268	0.4753
128					0.7744	0.6973	0.5435
256						0.7313	0.6144
512							0.6553

Table 4-5. The benchmarked efficiency E for the TRISTAN code running on the cluster-enabled Grid varying both the problem size W and processor p. The circled values are the approximated isoefficiency values in this benchmark.

From Eq. (4.5), we have $W = \frac{E}{1-E} T_0(W, p)$. If the overhead function T_0 is similar in these three benchmarks and T_0 does not asymptotically exceed the problem size W, the constant $K = \frac{E}{1-E}$ or the fixed efficiency E governs the growth of isoefficiency. In our benchmark, the overhead T_0 does not exceed the problem size W asymptotically and this is true in these three benchmarks. Therefore, we can conclude that higher the fixed efficiency E is, the slower the growth of the isoefficiency function is, in this benchmark

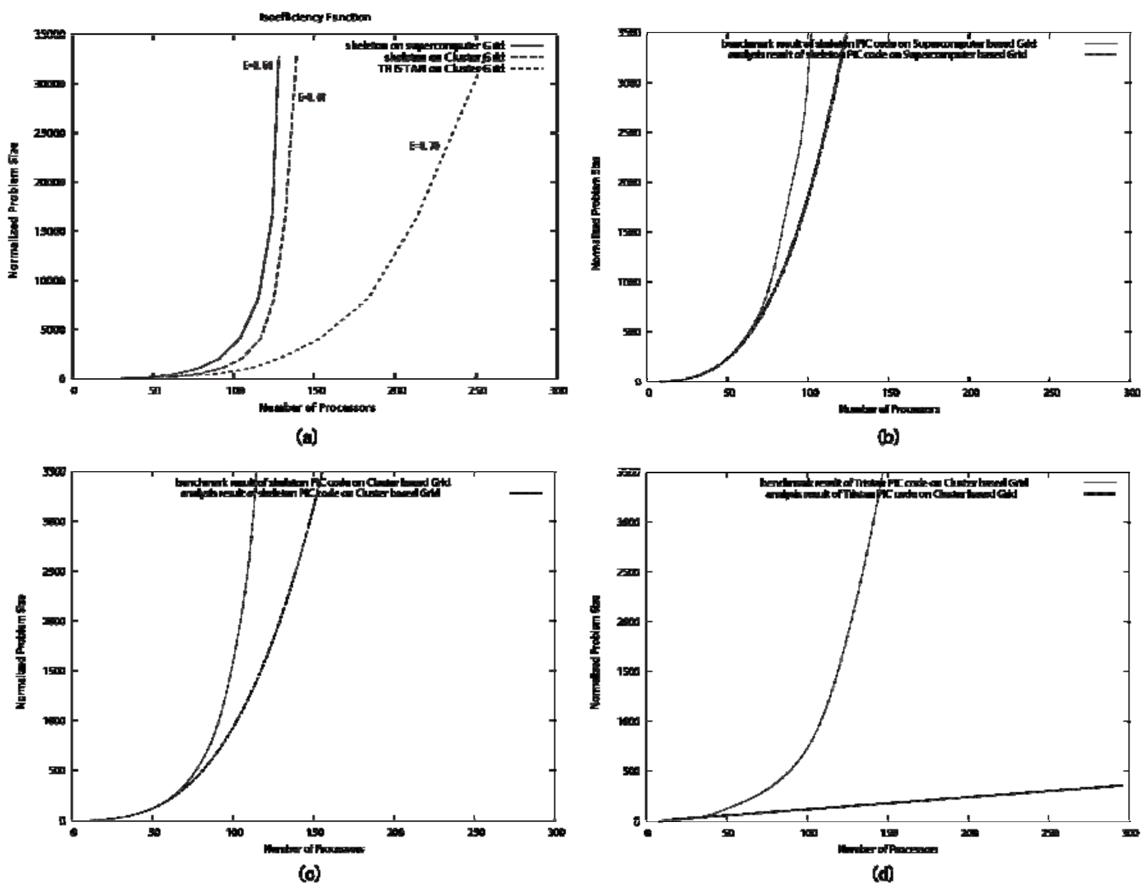


Figure 4-10. (a) Benchmarked isoefficiency functions of skeleton code in the supercomputer based-Grid, skeleton code in the cluster-based Grid and TRISTAN code in the cluster-based Grid. Benchmarked and theoretical isoefficiency functions of skeleton code (b) in the supercomputer-based Grid, (c) skeleton code in the cluster-based Grid and (d) TRISTAN code in the cluster-based Grid. Both the number p of processors and the problem size W are varied keeping the efficiency constant.

The isoefficiency curves predicted in section 4.3 are over-plotted in Figure 4-10(b), (c), and (d). The predicted isoefficeincy curves fit well for small number of processors, and deviates from the benchmark results if the number of processors exceeds about 70-80 for the skeleton PIC code, and exceeds about 40 for TRISATN code. For these benchmarks, both supercomputers and clusters are connected via WAN and the network latencies are large. Thus, for the TRISTAN benchmark, the isoefficiency curve deviates from the simple liner curve that is mainly predicted by the ghost cell communication relatively fast, and for the skeleton PIC code, the isoefficiency curves that mainly predicted by the collective communication terms fit well for larger number of processors ($p < 70$). On the large network latency systems, the predicted isoefficiency curves can only hold true within a limited number of processors.

In this benchmark, we should point out that the number of particles is limited to 8 particles per cell. If this number is changed, the experimental results will also change correspondingly. As shown in Figure 4-10, the isoefficiency curves of the parallel skeleton code increase exponentially beyond the limit $\boldsymbol{p} \approx 120$ for both Grids. It is very difficult to increase the number $\boldsymbol{p}$ of processors beyond this limit by increasing the problem size $\boldsymbol{W}$. If we increase $\boldsymbol{p}$, we have to increase W significantly to keep the efficiency constant beyond this limit.

In summary, both theoretical and benchmarks results reveal that the parallel TRISTAN PIC code without all-to-all collective communications is more scalable than the parallel skeleton PIC code with all-to-all collective communication on these computational Grid systems.

Chapter 5: Application of Global 3D EMPM Simulation to Virtual Magnetosphere

The interaction of the solar wind with the Earth's magnetic field gives rise to a number of important and intriguing phenomena, many of which are only partially understood. These include reconnection between the solar wind magnetic field and geomagnetic field lines at the dayside magnetopause (including patchy plasma transfer), reconnection in the magneto-tail, plasma convection in the magnetosphere/ionosphere, generation of field-aligned current systems, and energetic particle injection. A wide range of physical processes is involved in the solar wind-magnetosphere system, and consequently a wide array of methods has been used to study them, ranging from detailed studies of select phenomena with the assumption of a specific field geometry or boundary conditions, to fully three-dimensional simulations with MHD assumptions.

Since the strong coupling between the solar wind and the magnetosphere during the southward IMF phase of the cloud allows a significant fraction of the incident solarwind kinetic energy to enter the magnetosphere [*Goodrich, 1998*], reconnection in the magnetotail. By investigating the sources of the particles in the inner magnetosphere, one can get a better understanding of the processes that control the injection and heating of particles in the magnetosphere, changes in the solar wind conditions drive substorms and storms, and the near-Earth magnetotail region where kinetic effects are critical and particle simulation become essential [*Birn, 1996 and Pritchett, 1996*]. But these phenomena are difficult to understand because a variety of processes, regions, and scale lengths are involved.

In principle the EMPM code, which based on TRISTAN PIC code, enables us to investigate the global simulation of the solar wind interaction with an IMF (interplanetary magnetic field) including the complete particle physics. As has been shown in chapter 1, the advantage is that the basic equations of the model contain the complete physics, and the price is its large computation and memory cost. In chapter 4, we present the scalability of the TRISTAN PIC code in the computational grid. In this section, we discuss the way to use the 3DEMPM to get significant insight into the magnetic “Sash” region, and the substorms triggering mechanisms.

5.1 Global Three-Dimensional Electro-Magnetic Particle Model (3DEMPM)

Three dimensional electromagnetic Particle model is the TRISTAN PIC code used in last chapter, the details of this code had been described in chapter 2. In this section we present the simulation coordinate, normalization, boundary condition, etc..

5.1.1 The Simulation Coordinate

The simulation coordinate, xyz, used in the global 3DEMPM simulation is shown in Figure 5-1 (a), in which x axis is in the direction from Sun to the Earth, y axis is from dusk to dawn direction, z axis is from south to north direction. In the meantime, the Solar Magnetospheric (GSM) coordinate, XYZ, likes Figure 5-1 (b), where X axis is the Earth to the Sun direction, opposite direction to simulation x axis, Y axis is from dawn to dusk direction, opposite direction to simulation y anis, Z axis is from south to north direction, same as simulation z axis.

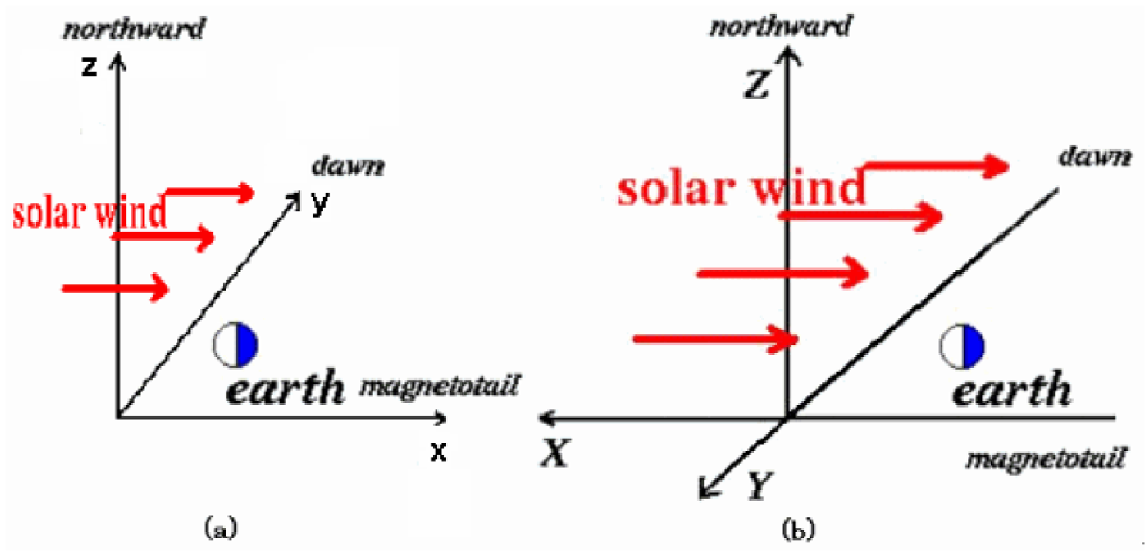


Figure 5-1 The coordinate systems used in the 3DEMPM simulation and its visualization. (a) the simulation coordinate system, xyz, used in simulation; (b) the GSM system, XYZ, used in some result visualization.

5.1.2 Normalized Physical Quantities

Here, we normalize all quantities in global 3DEMPM code as below in MKS system:

1). The grid size

$$\Delta = \Delta x, \Delta y, \Delta z = 1 \tag{5.1}$$

2). The time step

$$\Delta t = 1 \tag{5.2}$$

3). The temperature ratio

$$r_{tmp} = \frac{T_e}{T_i} \tag{5.3}$$

4). The mass ratio

$$r_{mass} = \frac{m_e}{m_i} \tag{5.4}$$

5). Number density of Ions and Electrons

$$n_i = n_e = \frac{N}{\Delta^3} \tag{5.5}$$

6). The charge

$$q = e \tag{5.6}$$

7). The electron ration of charge to mass

$$r_{qm} = \frac{q}{m_e} \tag{5.7}$$

Here after all normalization, parameters are denoted by ~:

8). Spatial

$$\tilde{x} = \frac{\vec{x}}{\Delta} \tag{5.8}$$

9). Time

$$\tilde{t} = \frac{t}{\Delta t} \tag{5.9}$$

10) Velocity

$$\tilde{v} = \frac{\vec{v}}{\frac{\Delta}{\Delta t}} \tag{5.10}$$

11). Light speed

$$\tilde{C} = \frac{C}{\frac{\Delta}{\Delta t}} \tag{5.11}$$

12). Electric and magnetic fields

$$\tilde{E} = \vec{E}\left(\frac{m_e \Delta}{e(\Delta t)^2}\right)^{-1} \tag{5.12}$$

$$\tilde{B} = \vec{B}\left(\frac{m_e \Delta}{e(\Delta t)^2}\right)^{-1} \tag{5.13}$$

13). Current density

$$\tilde{J} = \vec{J}\left(\frac{m_e \Delta}{e(\Delta t)^2}\right)^{-1} \tag{5.14}$$

5.1.3 Normalized Equations

When we use the normalized physical quantities in section 5.1.2 and substitute them into Eq. (2.14) – (2.16), we get the following normalized Newtown-Lorentz and Maxwell equations of Heaviside form as follows:

$$\frac{\partial \tilde{E}}{\partial \tilde{t}} = \tilde{C}\nabla \times \tilde{B} - \tilde{J} \tag{5.15}$$

$$\frac{\partial}{\partial \tilde{t}}\tilde{B} = -\tilde{C}\nabla \times \tilde{E} \tag{5.16}$$

$$\frac{d\tilde{V}}{d\tilde{t}} = -(\tilde{E} + \frac{\tilde{V}}{\tilde{C}} \times \tilde{B}) \tag{5.17}$$

5.1.4 Normalized Plasma Parameters

Using the normalized physical quantities, we also get the normalized plasma parameters used in our global 3DEMPM simulations as follows:

1). Frequencies

$$\tilde{\omega}_{pe} = \omega_{pe}\Delta t = \sqrt{\frac{e^2 n_e}{\varepsilon_0 m_e}}\Delta t = \sqrt{q n_e r_{qm}}\Delta t \ , \tag{5.18}$$

Here $\tilde{\omega}_{pe} \neq 1$ and

$$\tilde{\omega}_{pi} = \omega_{pi}\Delta t = \sqrt{\frac{e^2 n_i}{\varepsilon_0 m_i}}\Delta t = \sqrt{q n_i r_{qm} r_{mass}}\Delta t \tag{5.19}$$

$$\tilde{\omega}_{ce} = \omega_{ce}\Delta t = \frac{eB}{m_e}\Delta t = \tilde{B}\frac{\Delta}{\Delta t} = \tilde{B} \tag{5.20}$$

$$\tilde{\omega}_{ci} = \omega_{ci}\Delta t = \frac{eB}{m_i}\Delta t = \tilde{B}\frac{\Delta}{\Delta t}\frac{m_e}{m_i} = \tilde{B}\cdot r_{mass} \tag{5.21}$$

2). Periodicities

$$\tilde{\tau}_{pe} = \frac{2\pi}{\tilde{\omega}_{pe}}, \quad \tilde{\tau}_{pi} = \frac{2\pi}{\tilde{\omega}_{pi}}, \quad \tilde{\tau}_{ce} = \frac{2\pi}{\tilde{\omega}_{ce}}, \quad \tilde{\tau}_{ci} = \frac{2\pi}{\tilde{\omega}_{ci}} \tag{5.22}$$

3). Larmor radii

$$\tilde{\rho}_{ce} = \rho_{pe}/\Delta = \frac{v_{the}}{\omega_{pe}}/\Delta = \frac{1}{\Delta}\frac{\tilde{v}_{the}(\Delta/\Delta t)}{(\tilde{\omega}_{pe}/\Delta t)} = \frac{\tilde{v}_{the}}{\tilde{\omega}_{pe}} \tag{5.23}$$

$$\tilde{\rho}_{ci} = \rho_{pi}/\Delta = \frac{v_{thi}}{\omega_{pi}}/\Delta = \frac{1}{\Delta}\frac{\tilde{v}_{thi}(\Delta/\Delta t)}{(\tilde{\omega}_{pi}/\Delta t)} = \frac{\tilde{v}_{thi}}{\tilde{\omega}_{pi}} \tag{5.24}$$

4). Inertia length

$$\tilde{\lambda}_{ce} = \lambda_{pe}/\Delta = \frac{C}{\omega_{pe}}/\Delta = \frac{1}{\Delta}\frac{\tilde{C}(\Delta/\Delta t)}{(\tilde{\omega}_{pe}/\Delta t)} = \frac{\tilde{C}}{\tilde{\omega}_{pe}} > \Delta \tag{5.25}$$

$$\tilde{\lambda}_{ci} = \lambda_{pi}/\Delta = \frac{C}{\omega_{pi}}/\Delta = \frac{1}{\Delta}\frac{\tilde{C}(\Delta/\Delta t)}{(\tilde{\omega}_{pi}/\Delta t)} = \frac{\tilde{C}}{\tilde{\omega}_{pi}} > \Delta \tag{5.26}$$

5). Wave number

$$\tilde{k} = k\cdot\Delta \tag{5.27}$$

6). Debye length $\lambda_{De} = \dfrac{v_{the}}{\omega_{pe}}$, $\lambda_{Di} = \dfrac{v_{thi}}{\omega_{pi}}$

$$\tilde{\lambda}_{De} = \lambda_{De} / \Delta = \frac{v_{the}}{\omega_{pe}} \frac{1}{\Delta} = \frac{1}{\Delta} \frac{\tilde{v}_{the}(\Delta / \Delta t)}{(\tilde{\omega}_{pe} / \Delta t)} = \frac{\tilde{v}_{the}}{\tilde{\omega}_{pe}} \tag{5.28}$$

$$\tilde{\lambda}_{Di} = \lambda_{Di} / \Delta = \frac{v_{thi}}{\omega_{pi}} / \Delta = \frac{1}{\Delta} \frac{\tilde{v}_{thi}(\Delta / \Delta t)}{(\tilde{\omega}_{pi} / \Delta t)} = \frac{\tilde{v}_{thi}}{\tilde{\omega}_{pi}} \tag{5.29}$$

7). Pressure

$$\tilde{P} = \frac{P}{n_e m_e (\Delta / \Delta t)} \cong \frac{P}{n_e q r_{qm}} \tag{5.30}$$

8). Temperature $k_{e,i} T_{e,i} \cong m_{e,i} v_{th,e,i}^2$

$$\tilde{T}_e = \frac{kT_e}{\frac{1}{2} m_e (\Delta / \Delta t)^2} = \frac{m_e v_{th,e}^2}{\frac{1}{2} m_e (\Delta / \Delta t)^2} = 2\tilde{v}_{th,e}^2 \tag{5.31}$$

$$\tilde{T}_i = \frac{kT_i}{\frac{1}{2} m_e (\Delta / \Delta t)^2} = \frac{m_i v_{th,i}^2}{\frac{1}{2} m_e (\Delta / \Delta t)^2} = \frac{2}{r_{mass}} \tilde{v}_{th,i}^2 \tag{5.32}$$

9). Densities $\rho = n_e m_e + n_i m_i = n_0 (m_e + m_i) = n_0 m_e (1 + r_{mass})$

$$\tilde{\rho} = \frac{\rho}{(\Delta)^3} = \frac{n_0 m_e (1 + r_{mass})}{(\Delta)^3} = \rho \tag{5.33}$$

10). Alfven speed

$V_A = \dfrac{B}{\sqrt{\mu_0 \rho}}$, Where $\mu_0 = \dfrac{1}{C^2}$. Substituting them into above, we get:

$$\tilde{V}_A = \frac{V_A}{\Delta / \Delta t} = \tilde{C} \frac{\tilde{B}}{\sqrt{1 + r_{mass}}} \frac{1}{\omega_{pe}} \tag{5.34}$$

11). Lower hybrid frequency $\omega_l = \dfrac{1}{\sqrt{\dfrac{1}{\omega_{ce}\omega_{ci}} + \dfrac{1}{\omega_{pi}^2}}}$

$$\tilde{\omega}_l = \omega_l \Delta t = \frac{\Delta t}{\sqrt{(\frac{1}{\tilde{\omega}_{ce}\tilde{\omega}_{ci}} + \frac{1}{\tilde{\omega}_{pi}^2})\frac{1}{(\Delta t)^2}}} = \frac{(\Delta t)^2}{\sqrt{(\frac{1}{\tilde{\omega}_{ce}\tilde{\omega}_{ci}} + \frac{1}{\tilde{\omega}_{pi}^2})}} = \frac{\tilde{\omega}_{pi}\sqrt{\tilde{\omega}_{ce}\tilde{\omega}_{ci}}}{\sqrt{\tilde{\omega}_{pi}^2 + \tilde{\omega}_{ce}\tilde{\omega}_{ci}}} \quad (5.35)$$

12). Acoustic velocity $V_s = \sqrt{\dfrac{\gamma P_0}{\rho_0}} = \sqrt{\dfrac{\gamma k(n_i T_i + n_e T_e)}{n_i m_i + n_e m_e}} = \sqrt{\dfrac{\gamma k(T_i + T_e)}{m_i + m_e}}$

$$\tilde{V}_s = \frac{V_s}{(\Delta / \Delta t)} = \sqrt{\frac{\gamma \cdot m_i (v_{th,i}^2 + (m_e / m_i) v_{th,e}^2)}{m_i (1 + (m_e / m_i))(\Delta / \Delta t)^2}} = \sqrt{\frac{\gamma \cdot (\tilde{T}_i + \tilde{T}_e)}{2(\frac{1}{r_{mass}} + 1)}} \quad (5.36)$$

13). Plasma Beta

$$\beta = \frac{\sum nkT}{b^2 / 2\mu_0} = \frac{\sum nkT}{\frac{1}{2}\frac{B^2}{C^2}C^2} = \frac{\sum nkT}{\frac{1}{2}B^2} = \frac{(\tilde{T}_e + \tilde{T}_i)}{\tilde{B}^2}\tilde{\omega}_{pe}^2 \quad (5.37)$$

Hereafter, all of the quantities and parameters are the normalized quantities and parameters without any explicit notice [*Lembege, 2001*].

5.1.5 Position of Magnetopause

In the original TRISTAN code, the Earth dipole field is located at a grid cell 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 Figure 5-2, the Ampere equation gives:

$$\vec{B} = \frac{\mu_0}{4\pi}\int \frac{I_0 d\vec{l} \times \vec{r}}{r^3} \quad (5.38)$$

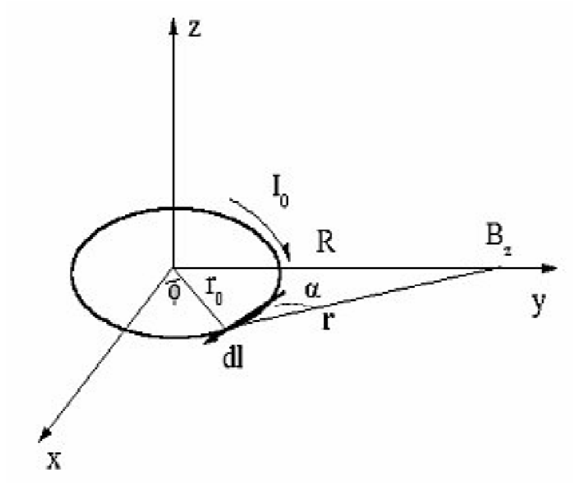


Figure 5-2. The ring current generating the dipole magnetic field of the Earth.

Where the small vector element of the ring is $\vec{dl} = -\vec{e}_\phi r_0 d\phi$, the distance from arbitrary point along the y-axis to $\vec{dl}$ is $r^2 = R^2 + r_0^2 - 2Rr_0\cos\phi$, 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:

$$\sin\alpha = \frac{R\cos\phi - r_0}{r} \tag{5.39}$$

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

$$B_z = \frac{\mu_0 I_0 r_0}{4\pi}\int \frac{R\cos\phi - r_0}{r^3} d\phi \tag{5.40}$$

When $R >> r_0$ is assumed, the B_z at the magnetopause can be obtained:

$$B_z(magnetopause) \approx \frac{\mu_0 I_0 r_0^2}{2R^3} \tag{5.41}$$

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

$$W_B^{magnetopause} = B^2/2\mu_0 \approx B_z^2/2\mu_0 \approx \frac{\mu_0 I_0^2 r_0^4}{8R^6} \tag{5.42}$$

The kinetic energy density of solar wind flow in the magnetopause is:

$$W_{soloarwind}^{magnetopause} = \frac{1}{2}\rho_{soloarwind}V_{soloarwind}^2 \approx \frac{1}{2}(m_i n_i + m_e n_e)V_{soloarwind}^2 \tag{5.43}$$

where $\boldsymbol{n}_i$, and $\boldsymbol{n}_e$ are the number density of ions and electrons in solar wind, respectively. The position of the magnetopause, $\boldsymbol{R}_{MP}$, locates where these two densities are equal to each other. We obtain:

$$\boldsymbol{R}_{MP}^6 \approx \frac{\mu_0 \boldsymbol{I}_0^2 \boldsymbol{r}_0^4}{4(\boldsymbol{m}_i \boldsymbol{n}_i + \boldsymbol{m}_e \boldsymbol{n}_e)\boldsymbol{V}_{solarwind}^2} \tag{5.44}$$

In our global 3DEMPM code, the Earth dipole field is ramped-up linearly at an initial stage of the simulation. The half dipole field ramp-up time is $-\boldsymbol{o}2/\boldsymbol{o}3 = \boldsymbol{n}$, and the final ring current charge density is $\boldsymbol{o}_{final} = -\boldsymbol{o}3 \times \boldsymbol{n}(\boldsymbol{n}+1)(2\boldsymbol{n}+1)/3$. 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 -o3 n(n+1) (2 n+1)/3 in the simulation. Using Eq. (5.42) and the final value of "o", the rough location of the magnetopause can be estimated.

5.1.6 Initial Condition

Initially at t=0, we need to place particles in $\vec{\boldsymbol{X}}$, $\vec{\boldsymbol{V}}$ in the grid with the electric and magnetic fields. The placement involves starting with prescribed densities in space $\boldsymbol{n}_0(\vec{\boldsymbol{X}})$ and in velocity $\boldsymbol{f}_0(\vec{\boldsymbol{V}})$ and then generating particle positions and velocities $(\vec{\boldsymbol{X}},\vec{\boldsymbol{V}})_i$. The initial conditions include the particles and the fields. The particles are initially distributed in full three dimensional domain region, with the initial positions in the domain and the initial velocities, the x component is the solar wind velocity plus small random, and the y and z components are small random. The positions of the particles were distributed in all the domain region. The initial velocities were initialized with a thermal velocity plus a bulk velocity in the x-direction representing the solar wind. The fields including the electric fields and the magnetic fields are initialized to the zero in all the domain. The initial electric and magnetic fields without IMF are all zero in the all domain at first. The electrons and ions are placed at the same location so that the charge conservation $\frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{t}} + \nabla \cdot \boldsymbol{J} = 0$ is satisfied.

5.1.7 Boundary Condition

The Boundary condition module is to process the electric fields, magnetic fields, and particles boundary conditions. The radiating boundary conditions are applied to the fields using the first-order Lindman approximation [*Lindman, 1975*]. We can use any other boundary conditions, for example, period boundary condition, PML boundary condition, and so on.

The particle boundary conditions are: (1) Fresh particles representing the incoming solar wind are continuously injected across the yz plane at $x = x_{\min}$ with a thermal velocity plus a bulk velocity at x-direction; (2) thermal solar particle flux is also injected across the sides of the rectangular simulation domain; and (3) escaping particles are arrested in a buffer zone and redistributed there more uniformly by making the zone conducting.

5.2 "Sash" simulations with Turning IMF from Northward to Southward

5.2.1 Magnetospheric "Sash" and "Cross-Tail S"

The magnetospheric "Sash" is the low magnetic field strength that starts from the dayside magnetopause, dayside cusp and spreads fan-like tail-ward to the near magneto-tail. The weakest field runs along the northern ridge line of the last closed field line surface and this ridge coincides with the separator line of the magnetic field, at which closed field lines and the IMF touch. Magnetosphere sashes and related phenomena have been observed and reported in some satellite observations [*Maynard, 2001*] and global 3D MHD simulations [*White, 1998 and Siscoe, 2001*].

The dawn and dusk magnetospheric "Sash" are connected by a piece, crossing through the equatorial tail, and this weak field bands on both flanks is continuous extensions of the cross tail neutral sheet. The magnetospheric "Sash" extends from one dayside cusp to the other by circling around the magnetosphere through the near tail. So the pair of conspicuous islands of exceptionally weakened field on the low-latitude flanks joins the neutral sheet through the "Cross-Tail S" [*Nishikawa, 1998*]. At some point near or behind the terminator, the sash becomes a site for reconnection of open field lines, which were previously opened by the merging on the dayside. This indicates that the significant reconnection in the magneto-tail occurs on the flanks. The source of the magneto-sheath plasma on the closed field lines behind the terminator was plasma entry through the low field connection of the sash to the central plasma sheet [*Frank, 1995; Fujimoto, 1997*].

5.2.2 Global 3DEMPM "Sash" Simulation Model

In this simulation, we used the initial conditions, and radiating boundary conditions as shown on section 5.1.6. The normalization of equations, physical quantities, and parameters are same as section 5.1, in which $\Delta = 1.0$ (in space) and $\Delta \boldsymbol{t} = 1.0$ (in time) are used. Initially, 400,000,000 (about 8 electron-ion pairs per cell) fill uniformly the entire box (215 by 145 by 145), in which the center of the current loop or the Earth is located at (80, 72.5 73) and drift with a velocity $\tilde{\boldsymbol{v}}_{sol} = 0.5\tilde{\boldsymbol{C}}$ (the normalized light speed $\tilde{\boldsymbol{C}} = 0.5$) in the x-direction, representing the solar wind with un-magnetized IMF. The injected solar wind density is 8.0 electron-ion pairs per cell, the ion and electron thermal velocity are, $\tilde{\boldsymbol{v}}_{itl} = (\tilde{\boldsymbol{T}}_i / \boldsymbol{m}_i)^{1/2} = 0.01\tilde{\boldsymbol{C}}$ and $\tilde{\boldsymbol{v}}_{etl} = (\tilde{\boldsymbol{T}}_e / \boldsymbol{m}_e)^{1/2} = 0.02\tilde{\boldsymbol{C}}$, respectively. The electron plasma frequency is $\tilde{\boldsymbol{\omega}}_{pe}\Delta \boldsymbol{t} = 0.25$. All of the normalized parameters used in this simulation are listed in Table 5-1. In this simulation, the spatial resolution is about $1\Delta = 0.5\boldsymbol{RE}$ and the time resolution is about $1.0\boldsymbol{UT} = 1980\Delta \boldsymbol{t}$.

Grids	215 x 145 x 145		Electron temperature T_e	0.13
Initial particles ($\tilde{n}_0$)	400,000,000 (8 pairs / cell)		Ion temperature T_i	0.52
Light speed ($\tilde{C}$)	0.5		Solar wind velocity	0.25
ε_0	1.0		ω_{pe}	0.25
m_i / m_e	16		$\lambda_D / \Delta x$	1.0
q_e / m_e	0.5		Plasma Beta β	0.25
$\Delta x, \Delta y, \Delta z$	1.0			
Δt	1.0			

Table 5-1. The normalized parameters used in the global 3DEMPM magnetospheric "Sash"

After a quasi-steady state is established with an un-magnetized solar wind at time step 1500, and compared with previous works by Professor Nishikawa [Nishikawa, 1992, 1996, 1997, 1998], a northward IMF ($\boldsymbol{B}_z^{IMF} = 0.2$) is switched on gradually and smoothly in time step 1501-1800(for comparison, the average Bz at the dayside magnetopause ($\Delta = 10\boldsymbol{RE}$) is nearly 2.8), after time step 1800 and until time step 3300, the northward IMF B_z^{IMF} goes through all of the domain region, and then is gradually rotated to the dawn-dusk ward IMF $B_y^{IMF} = -0.2$ in time step 3301-3600 (for simulation coordinate system) ($B_y^{IMF} = 0.2$ in the geocentric magnetospheric coordinate system (GSM) and the results for this sash simulation are all in GSM). After time step 3600 and until time step 5100, the duskward IMF B_y^{IMF} goes through all of the domain region. Then it is gradually rotated to the southward IMF B_z^{IMF} in time step 5101-5400, and after time step 5400 and until time step 7000, the southward IMF goes through the all the domain region.

5.2.3 Simulation Results

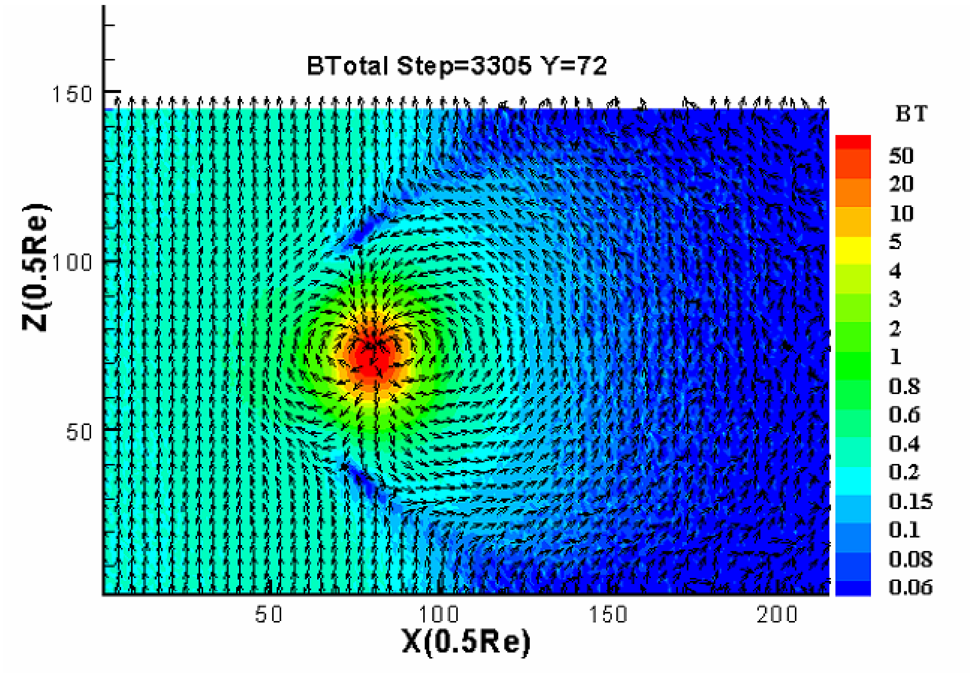


Figure 5-3. The X-Z plane of magnetic field at median plane on time step 3300

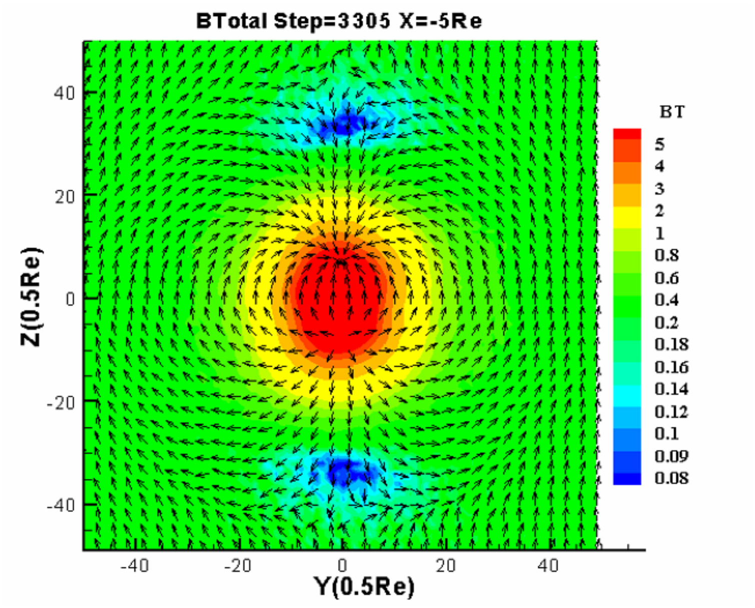


Figure 5-4. Y-Z plane of magnetic field at $x = -5\,\mathrm{Re}$ *on time step 3300*

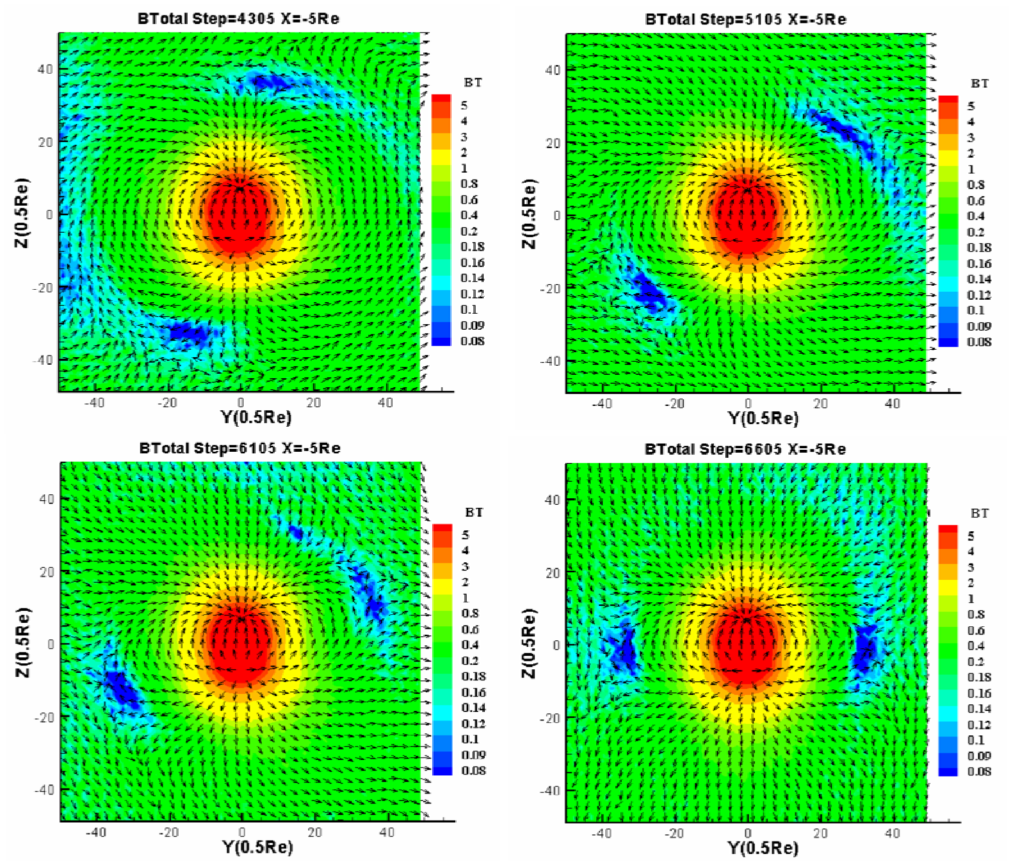


Figure 5-5. Y-Z plane of magnetic field at $x = -5\,\mathrm{Re}$ *on time step 4300, 5100, 6100, 6600, which corresponding to the time step when IMF clock angle of (45,90,135,180) goes through the* $x = -5\,\mathrm{Re}$ *plane.*

Figure 5-3, 5-4 show the X-Z plane of magnetic field at median plane and Y-Z plane of magnetic field at $x = -5\,\mathrm{Re}$ (on night side) on time step 3300, when the northward IMF goes through the whole simulation domain. It clearly present that for the northward IMF, the week magnetic field sash form poleward of the cusp at high latitudes in each hemisphere, it runs tailward along the magnetopause flanks from the dayside (near $x = 5\,\mathrm{Re}$) to tail side.

The rotation of the IMF direction from the northward to southward has an important impact on the direction of the sash region. The mapping of IMF clock angles to sash-core clock angles has been found as $(0,45,90,135,180) \rightarrow (0,24.7,54.9,77.7,90)$ [*Siscoe, 2001*]. Figure 5-5 shows how the sash clock angle depends on the clock angle of IMF of our global particle simulation, which is consistent with the results of global MHD simulation. An enlarged projection of the terrestrial magnetosphere is plotted in Figure 5-6(a) at time step 5000 within the cross sectional plane Y-Z perpendicular to the solar wind flow and located at $x = -5\,\mathrm{Re}$. The duskward IMF B_y^{IMF} prevails in the whole inner simulation domain at that time step. A "separatrix" in the magnetic field $\boldsymbol{B}$ vectors direction is clearly evidenced around the location $(y,z) = (12,9)\,\mathrm{Re}$. The configuration of $\boldsymbol{B}$ vectors found herein is in a good agreement with previous results of 3D global MHD simulation [*White, 1998, Figure 1b*]. the results from Figures 5-3 to 5-6 allows us to validate our present results.

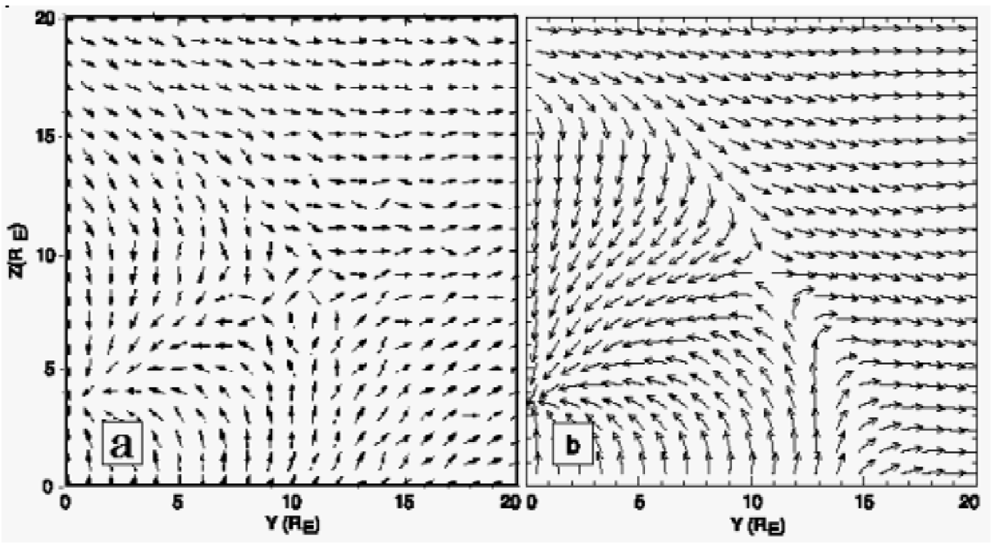


Figure 5-6 Y-Z plane of magnetic field at $x = -5\,\mathrm{Re}$ on time step 5100 with duskward IMF. (a) results of our global particle simulation, (b) global MHD simulation [White, 1998]

Figure 5-7 shows the ion and electron fluxes on time step 5100 at $x = 5\,\mathrm{Re}$ (near the dayside magnetopause) and at $x = -5\,\mathrm{Re}$ (night side). Figure 5-7 (a, b) show that a strong ion flux extends along the magnetopause at $x = 5\,\mathrm{Re}$ from $(y,z) = (0,10)\mathrm{Re}$ to

$(7,8)\,\mathrm{Re}$. the common location of $(7,8)\,\mathrm{Re}$ corresponding to the shifted region of the cusp located on the flank of the magnetopause on the basic of the magnetic field separatrix (circle in Figure 5-7). It corresponds also to the beginning of the 3D sash structure extending further to within the magnetosphere. The situation differs in the night side region (Figure 5-7 (c, d)), where both electron and ion fluxes are large in a region centered around $(y,z) = (12,13)\mathrm{Re}$ (also near the sash region). Comparing intermediary plots between $x = 5\,\mathrm{Re}$ and $x = -5\,\mathrm{Re}$ at the same time step, it confirms that these fluxes are still large near the sash region, i.e., the sash pattern acts as a magnetic "groove" along which particles are traveling from the cusp region (magnetopause flank) into the inner magnetosphere.

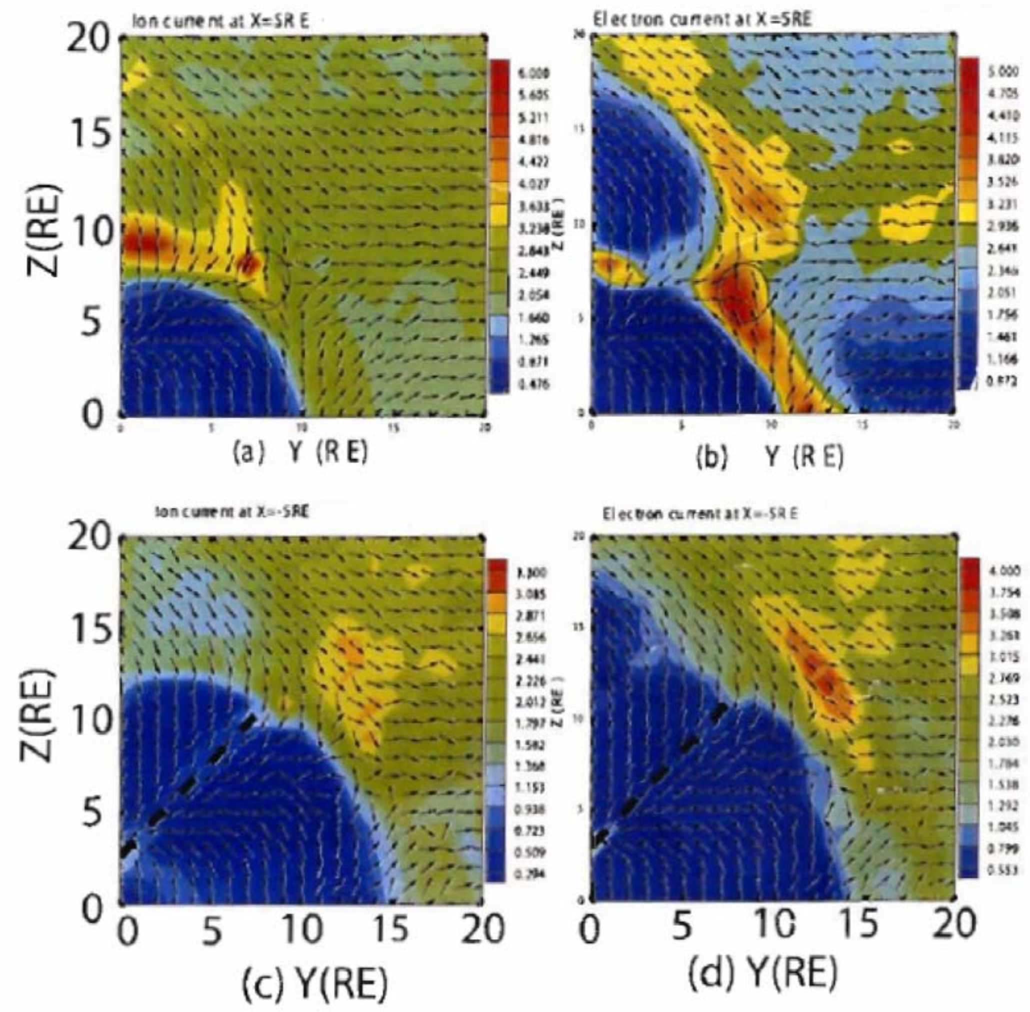


Figure 5-7. Sectional slice of the electron and ion fluxes with magnetic field vector measured at time step 5100 within the Y-Z plane located on the dayside at x=5Re for (a) ions and (b) electrons, respectively, and (c and d) similar plots shown on the night side at x=-5Re

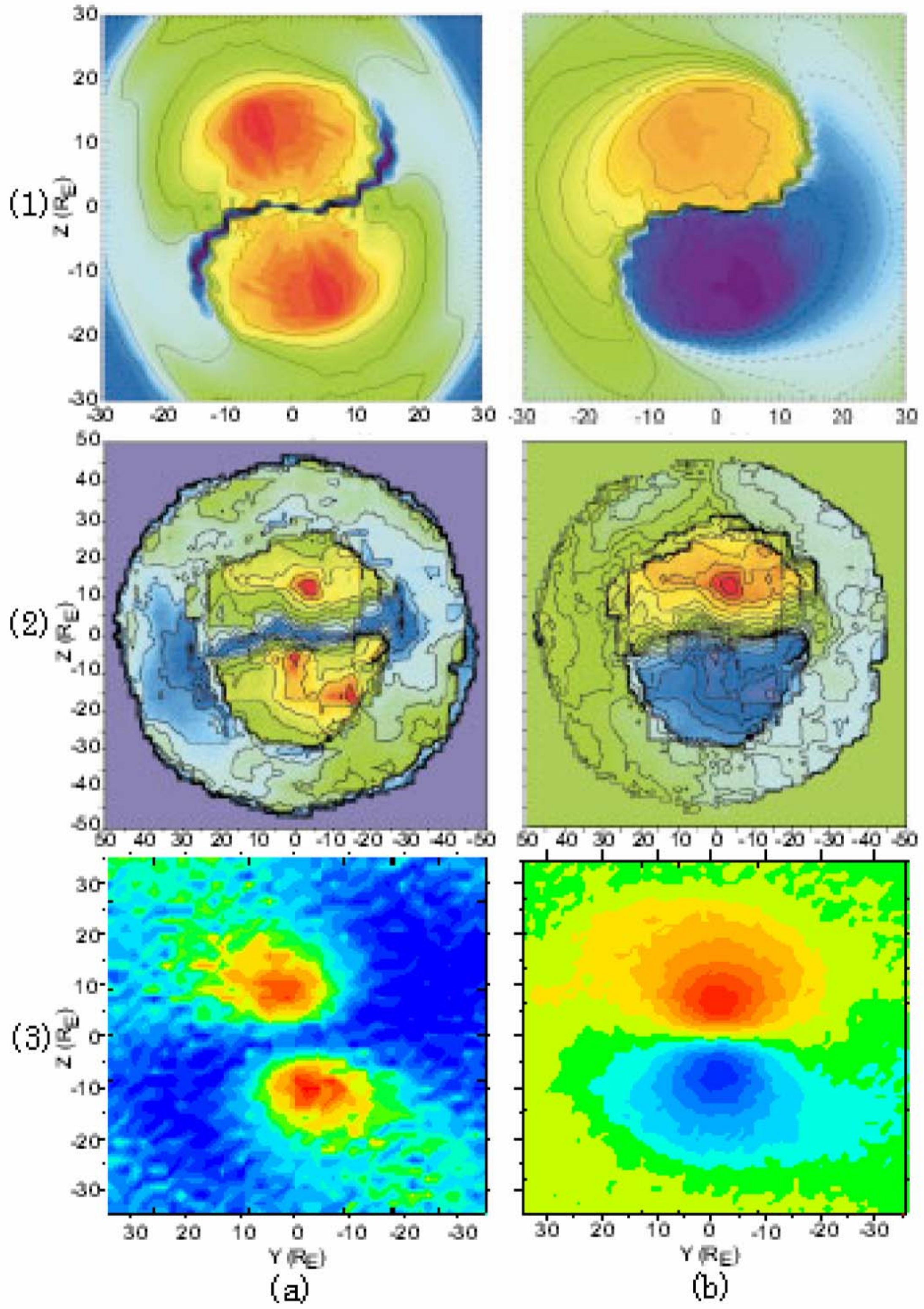


Figure 5-8. The "Cross-Tail S" as seen in B-magnitude and Bx contours from (a), (b) global 3DMHD simulation; (c), (d) from IMP 8 satellite observation; (e), (f) from global 3DEMPM simulation at time step 5100 on the night side at x=-5Re.

In order to analyze the evolution of the sash patter within the whole inner magnetosphere and to determinate the corresponding behavior of the electrons and ions, similar slices at different locations with the more distant magnetotail at the same time can also be plotted. Intermediate slice made at different x-locations clearly show that the sash pattern forms on the day side magnetopause flanks and extents to the nearby magnetotail. This sash pattern is observed in the north and south hemispheres in opposite quadrants, i.e., on the dusk and the dawn side. Sash gradually join onto neutral neutral sheet locations, and merge into each other to form a geometrical feature called the "Crosstail-S". This merge results from the fact that sash approach the neutral sheet region in the tail during their rotation. All region of weak magnetic field merge as more easily as the amplitude of the magnetic field globally decreases when moving tailward and approaching the neutral sheet. The results are illustrated in Figure 5-7, it show the similar patterns has been already evidenced in previous MDH simulation [*White, 1998*] and in IMP 8 observation [*Kaymaz, 1994a, 1994b, 1994c*].

5.3 Bifurcation and Hysteresis of the Magnetospheric Structure With a Varying Southward IMF

5.3.1 Introduction

The energy transfer from the solar wind to the Earth magnetosphere, and the causes of magnetic substorms have been analyzed for decades. Axford proposed that the reconnection of magnetic fields at the dayside magnetopause is one of the candidates to explain this huge energy transfer *[Axford, 2002 and Priest, 2000]*. However, despite many important efforts, the mechanisms of three-dimensional magnetic reconnection and the resulting energy partition during the transfer have not been well understood within a global approach of the magnetospheric dynamics.

At the present time, our understanding of three-dimensional magnetic reconnection relies principally on local observations drawn from satellites, laboratory plasma experiments, and MHD, hybrid, and particle simulations. It is commonly admitted that a necessary condition for the occurrence of the magnetic reconnection is a localized strong resistivity or a strong localized current that initiates the reconnection at the separator [*Priest, 2000*]. At the same time, whether this is a sufficient condition is a matter of current debate. The requirement to make sense of these factors within a governing hypothesis with sufficient precision to yield a convincing description of the three-dimensional magnetic reconnection has inspired the efforts of a large number of investigators. However, among the numerous attempts, only a few of the contending arguments lend themselves to the topological structure, structural stability [*Priest, 2000*], and asymptotic stability [*Guckenheim and Holmes*, 1988; *Strogatz*, 1994; *Wiggins.*, 2003] of the three-dimensional magnetic reconnection. In the present paper, we investigate the impact of a slowly varying southward interplanetary magnetic field (IMF) $|B_z|$ on the magnetosphere (more exactly in the subsolar region) in terms of bifurcation and evidence of dissipative processes [*Guckenheim and Holmes*, 1988; *Nicolis and Prigogine*, 1977; *Strogatz*, 1994; *Wiggins*, 2003] by using a global three-dimensional full-particle electromagnetic particle simulation. The consequences of magnetic reconnection expected at the subsolar region on the "overall" magnetosphere itself (changes in the field topology) will be also presented. The use of 3D-PIC simulation is justified by an extension of the present analysis on particle injection/acceleration mechanisms before and after magnetic reconnection takes place, which is under active investigation.

5.3.2 Simulation Model

The normalized parameters used in these simulations with time varying southward IMFs by our 3DEMPM code are shown in Table 5-1. We used the same normalization for equations, physical quantities and physical parameters as section 5.1. In these simulations, we run this code for 1000 time steps with every southward IMF following forward simulation from $B_z^{IMF} = -0.01$ to $B_z^{IMF} = -0.15$, and then backward simulation from $B_z^{IMF} = -0.15$ to $B_z^{IMF} = -0.01$. The applied time varying southward IMFs are shown in Figure 5-9.

Grids	215 x 95 x 95		Electron temperature T_e	0.000059
Initial particles ($\tilde{n}_0$)	32,000,000 (8 pairs / cell)		Ion temperature T_i	0.00048
Light speed ($\tilde{C}$)	0.5		Solar wind velocity	0.25
ε_0	1.0		ω_{pe}	0.088
m_i/m_e	16		$\lambda_D/\Delta x$	0.93
q_e/m_e	0.5		Plasma Beta β	3.2
$\Delta x, \Delta y, \Delta z$	1.0		IMF	Yes
Δt	1.0			

Table 5-2. The normalized parameters used in the global 3DEMPM simulation of bifurcation

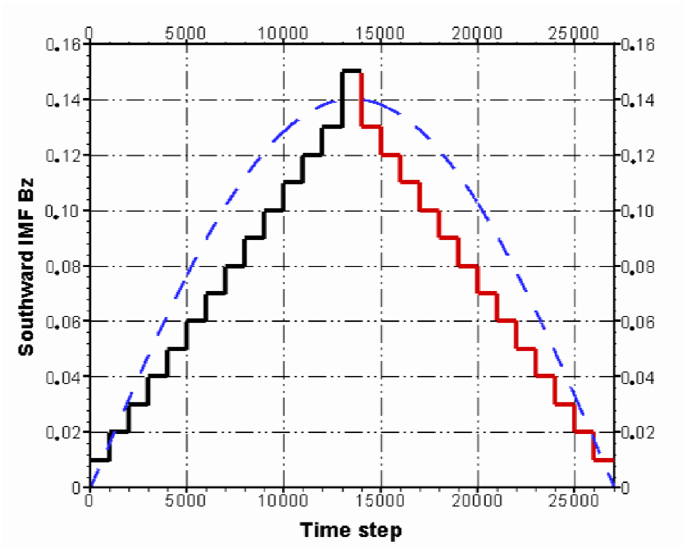


Figure 5-9. Time history of the southward IMF $|B_z|$ ***applied in this simulation.***

5.3.3 Simulation Results

In this section, with the help of 3D full electromagnetic full particle simulation code, we have investigated the asymptotic stability of the solar wind magnetic field and the structural stability of the Earth's magnetospheric surface magnetic field at the dayside magnetopause with a time-varying southward IMF.

We will focus on the magnetic field in the dayside magnetosphere and the magnetopause region in which "subcritical" bifurcation is shown to be the signature of large-scale dissipative processes. Let us first consider how magnetic reconnection on the magnetopause may originate on a magnetospheric surface of revolution when one of the main parameters herein the strength of the southward IMF $|B_z|$ is slowly varying. Herein, we focus on the changes of this magnetic field (bifurcation) during the interaction of the magnetized solar wind with the magnetosphere. Then, one finds convenient to measure the maximum distance R_{mp} of the magnetopause. Here, R_{mp} is measured from the dayside subsolar point to the geomagnetic center, and will decrease as the control parameter (the amplitude of IMF$|B_z|$) increases. Our three-dimensional global simulation is performed as the southward IMF $|B_z|$ is varying as shown in Figure 5-9. The southward IMF $|B_z|$ is once increased step by step from 0.0 to 0.15 within increment of 0.01. For each increment, we start from the last time of previous simulation and continuously extend the simulation for 1000 additional time steps so that the solar wind with new IMF prevails over the whole simulation domain and the overall solar-wind-magnetosphere interaction reaches a "near-steady-state". The process is repeated until the southward IMF $|B_z|$ increases up to 0.15. Afterwards, we begin to decrease $|B_z|$ down to zero with a decrement of 0.01 identical to the increment value. The simulation is started again and performed continuously for 1000 additional time steps for each decrement so that the solar wind with new $|B_z|$ prevails in the whole simulation domain. Varying the southward IMF$|B_z|$, the complete IMF$|B_z|-R_{mp}$ diagram can be drawn and evidences hysteresis signature (Figure 5-10). As the strength of the southward IMF $|B_z|$ increases, R_{mp} decreases (black curve), and increases as the strength of IMF $|B_z|$ decreases (red curve). The gap between the black and red curves (hysteresis) evidences that the magnetopause does not recover its initial location R_{mp} on the same way as it left it. Indeed, within the range (0.04 < $|B_z|$ < 0.08, where the slope of the curve changes noticeably respectively as IMF $|B_z|$ increases and decreases), R_{mp} is smaller (i.e. pressure of the magnetospheric field is smaller) as the IMF $|B_z|$ decreases, while R_{mp} is larger (i.e. the pressure of the magnetospheric field is larger), as the IMF $|B_z|$ increases. This indicates that some irreversible energy transfer from the solar wind to the magnetosphere takes place via some dissipative processes, over the successive increase and decrease of IMF $|B_z|$.

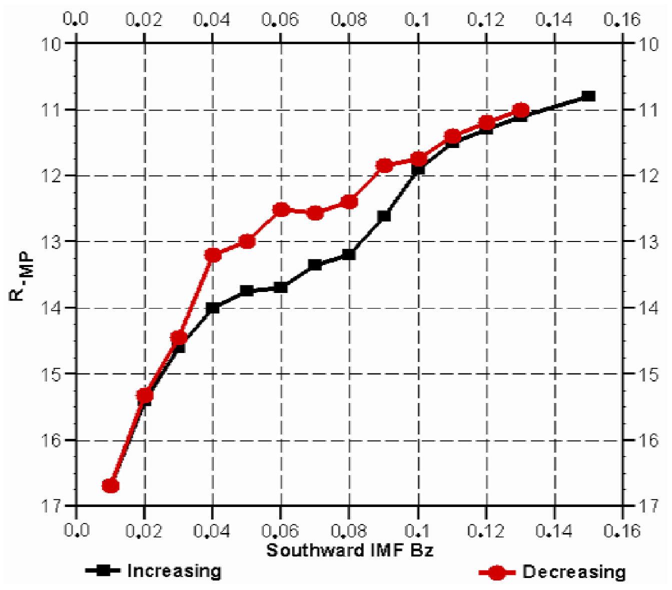


Figure 5-10. Variation of R_{mp} versus the southward IMF $|B_z|$. The hysteresis signature is evidenced by the difference between the black curve (increasing southward IMF $|B_z|$) and the red curve (decreasing southward IMF $|B_z|$)

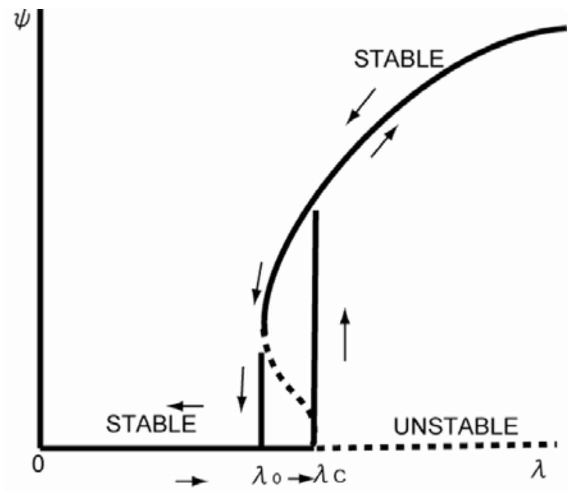


Figure 5-11. Examples of typical subcritical bifurcation. the different paths followed by the arrows illustrate the hysteresis signature as λ increases and decreases successively.

More exactly, as the strength of the southward IMF increases, Earth's magnetosphere is strongly compressed on the dayside and shrinks linearly at the initial stage ($|B_z|$= 0.01 to 0.04, where R_{mp} strongly decreases). Then, its location gradually "stabilizes" ($|B_z|$=0.04 to 0.08, where R_{mp} weakly decreases only): the shrinking almost saturates. The whole stage ($|B_z|$= 0.01 to 0.08) corresponds to the stage 0 to λ_c in the typical subcritical bifurcation as shown in Figure 5-11, although the curve in Figure 5-10 is not flat due to the nature of R_{mp}. At that stage, one can consider that the magnetosphere can be slightly deformed without bifurcation. In the final stage, the control parameter λ (IMF$|B_z|$) passes through the critical value $\lambda_c = |B_z| = 0.08$ and a catastrophic-like finite jump (up) of R_{mp} suddenly occurs. As a result, the magnetic field seems to bifurcate into a new branch of "stable" field and the signature of this bifurcation is of a subcritical-type.

After the southward IMF $|B_z|$ reached 0.15, one starts to decrease gradually the southward IMF and the magnetic field is still on the new branch of a stable field. At first stage (as $|B_z|$ varies from 0.15 to 0.08), the dayside magnetosphere progressively recovers and R_{mp} decreases linearly. However, as λ is just below λ_c, the magnetic field does not return to the original stable known magnetic field. The recovery almost "saturates", when λ_c is decreased far enough below λ_c to exceed another critical value λ_0 (= 0.04). A second finite jump of R_{mp} suddenly occurs and allows to access to the stable know magnetic field which is fully recovered.

In Figure 5-10, the finite jump is not as drastic as in Figure 5-11. The reason is that for each southward IMF value the magnetic field should reach the asymptotically stable and steady state before the critical bifurcation point λ_c. However, due to the constraints of computer power and CPU time, we have to change the southward IMF value (increment/decrement) after a reasonable number of time steps (1000): then, the perturbations caused by the changes of λ may still partially remain and the magnetic field dynamics is not in perfect steady state. We leave aside the vexing question of mechanisms underlying the magnetic reconnection itself, as well as its topological structure, and focus on the hysteresis signature and the associated phenomena.

The important point is that no artificial form of dissipation (i.e. resistivity as used in hybrid simulation) is included in our PIC simulation. However, the hysteresis signature which is clearly evidenced that indicates that some dissipation processes take place in some irreversible way. The main invoked reason for this dissipation is the energy transfer from the solar wind to the magnetosphere.

In terms of bifurcation, as the control parameter λ (i.e. the IMF $|B_z|$) increases and passes the critical value λ_c, the external (IMF) and the magnetospheric surface magnetic field bifurcate into a new mean magnetic field. The subcritical bifurcation may lead to a radical change in the topological structure of the magnetic field in such a way that the symmetry of the known magnetic field is broken. This symmetry breaking (also called hysteresis) is interpreted as being due to irreversible dissipation processes which take place to absorb just the amount of excess available energy that symmetrical known magnetic field could no longer be able to absorb. For a more comprehensive approach, let us remind that dissipative effects can support both possible subcritical and supercritical-type bifurcations. Present results indicate the key difference between both cases: dissipative effects are reversible for supercritical-type bifurcation and irreversible for subcritical type bifurcation.

The consequences on the magnetospheric field topology may be analyzed by reminding that the pressure balance between the solar wind ram pressure and the magnetospheric field pressure determines the distance R_{mp} of the dayside magnetopause (subsolar point) from the Earth. As the strength of the southward IMF increases and exceeds λ_c, the reconnection, expressed herein in terms of bifurcation, takes place at the dayside magnetopause. The magnetic field and the magnetospheric surface "phase portrait" transits to a new "stable" state. The new state is reached by large-scale dissipative processes and the excess energy contained in this new state can be absorbed.

Figure 5-12 shows two magnetic field topologies by simply visualizing the fan-surfaces (Σ-surfaces) and the spine-curves (γ-lines) of the critical points found around the dayside region in (a), (c), (e), (g) as λ increases to $\lambda = 0.1$ just after the

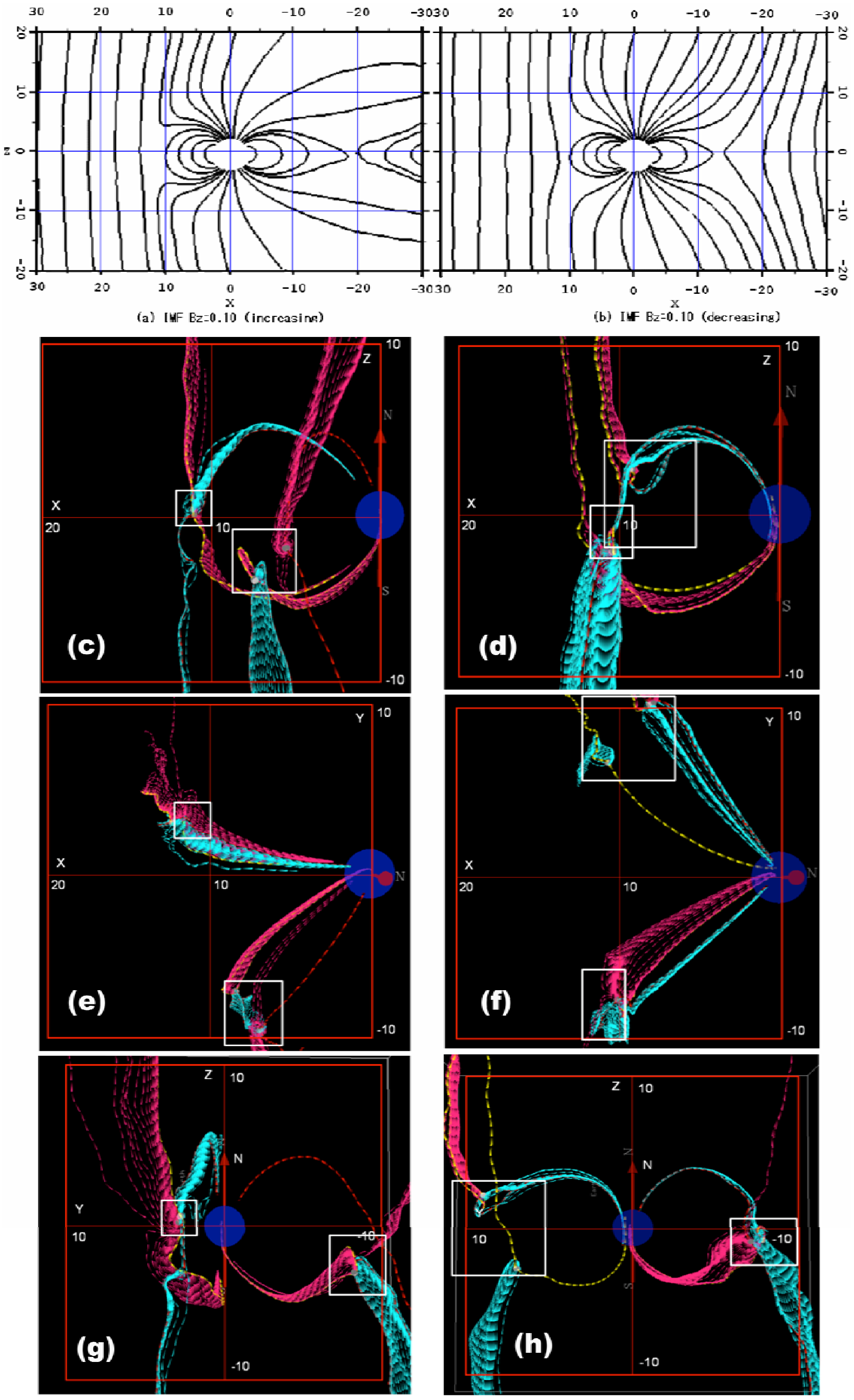


Figure 5-12. Two magnetic field topologies obtained for southward IMF $|B_z| = 0.1$ respectively for as southward IMF $|B_z|$ increases from lower values (cases (a), (c), (e), (g)) and decreases from higher values (cases (b), (d), (f), (h)). Cross-sections at the meridian plane are shown in panels (a-b). Magnetic field topologies are shown respectively at X-Z

plane in panels (c-d), at Y-X plane in panels (e-f), and at YZ plane in panels (g-h). The white dots in the figures (c-h) are critical points.

catastrophic-like jump, and (b), (d), (f), (h) as λ decreases to $\lambda = 0.1$ when $|B_z|$ does not recover yet to the original value in Figure 5-10. In Figure 5-12, the critical points are all saddle-points. The associated fan-surfaces (Σ-surfaces), and spine-curves (γ-lines) are visualized with different colors [*Cai, 2001*]. The fan-surfaces drawn in Fig. 6 cover only limited regions. Due to the insufficient numerical accuracy, it is difficult to extend these surfaces to larger region at present time. In addition, the fan-surfaces extended from other regions, especially from the magnetotail region where magnetic reconnection is also expected, are not drawn here. Thus, the magnetospheric surfaces in the visualized region are only partially covered. At (a), (c), (e), (g) increasing $\lambda = 0.1$ (black curve in Figure 5-10), in the bifurcated magnetic field, the critical point in the subsolar point is now split into four critical points, where two are located in the dawn side and other two are located in the dusk side. They are accumulated near the subsolar point on the magnetopause and form a three-dimensional “X-line”: these are the 3D null points joined by separators. For the purpose of figure clarity, the connecting line joining each critical point is not shown in the figure.

After $\lambda = 0.1$, we still increase λ up to 0.15. During the increase of λ, the number of critical points increased and divides into two groups (one in the dawn side and another in the dusk side), and both move towards the flanks of magnetosphere, i. e. to $\pm y$, respectively. At the same time, each group moves backward (-x) i. e. towards the Earth. The two clusters separate each other and move westward and eastward in opposite directions, and the "X-line" largely extends. This phenomenon is similar to the so-called "X-line extension" in reconnection. After λ reaches the value 0.15, we gradually decrease λ back to 0.1 (red). At (b), (d), (f), (g) the decreasing $\lambda = 0.1$ (red), the critical points are still split into two groups. One group with three critical points is located in the dawn side while another with two critical points is located in the dusk side. However, they do not return to the same positions, and remain far from the subsolar point.

In summary, when the increasing λ exceeds a certain value (black curve in Figure 5-10), a new magnetic field topology emerges after the bifurcation, and the newly bifurcated field in the branch of a stable magnetic field stays almost unchanged even if we increase the parameter further and decrease back to the same point. This means that the critical points remain split into two groups located respectively both in the dawn and dusk side region. Of course, some local modifications of the topology can be found. Two topologies of the magnetic field shown in Figure 5-12 (a), (c), (e), (g), and (b), (d), (f), (h) are “topologically-almost-equivalent”. This evidences that the two magnetic fields still remain on the same branch of the magnetic field. However, the positions of the critical points are still split into two groups and located far each other in Figure 5-12 (b), (d), (f), and (h). The two groups of the critical points do not return close enough to the subsolar point until λ still exceeds λ_0. This difference is the consequence of the decreasing λ (red curves in Figure 5-10) exceeding the critical value λ_c. Similar differences are also found in the nearby magnetotail region but are not analyzed herein. Please note that the strict argument of topological equivalence is not discussed here because the critical points are searched only in the limited region in the dayside.

5.4 Visualizations

5.4.1 Line Integral Convolution using Color Stealing Algorithm

5.4.1.1 Line Integral Convolution (LIC)

Line integral convolution (LIC) [*Cabral, 1993*] is texture based technique for visualizing vector fields and has the advantage of being able to visualize large and detailed vector fields in a reasonable display area. In each case, you start with a vector field, sampled as a discrete grid of normalized vectors. You also need an image that spatially uncorrelated (e.g., noise image), so correlations you apply to it will be more obvious. The goal is to process the image with the vector field, using line integral convolution, so you can visualize it. Note that in this technique, you will concentrate on the direction of the flow field, not its velocity; this is why the vector values at each grid point are normalized. The processed image can be calculated directly using a special convolution technique. A representative set of vector values on the vector grid are chosen. Special convolution kernels are created shaped like the local stream line at that vector by tracing local field flow forwards and backwards some user-defined distance. The resulting curve is used as a convolution kernel to convolve the underlying image. This process is repeated over the entire image using a sampling of the vectors in the vector field.

Mathematically, for each location $\boldsymbol{p}$ in the input vector field, a parametric curve $\boldsymbol{P}(\boldsymbol{p},\boldsymbol{s})$ is generated which passes through the location and follows the vector field for some distance in either direction. To create an output pixel $\boldsymbol{F}(\boldsymbol{p})$, a weighted sum of the values of the input image $\boldsymbol{F}$ along the curve is computed. The equation is:

$$F_{out}(p) = \frac{\sum_{i=0}^{l} F_{in}(p_i)h_i + \sum_{i=0}^{l'} F_{in}(p_i')h_i'}{\sum_{i=0}^{l} h_i + \sum_{i=0}^{l'} h_i'} \tag{5.45}$$

$$h_i = \int_{s_i}^{s_i+\Delta s_i} k(w)dw \tag{5.46}$$

$$s_0 = 0, s_i = s_{i-1} + \Delta s_i \tag{5.47}$$

Here $k(w)$ is weighting function. Figure 5-13(c) was rendered using LIC, that correspond to the vector field Figure 5-13(a) and noise texture Figure 5-13 (b).

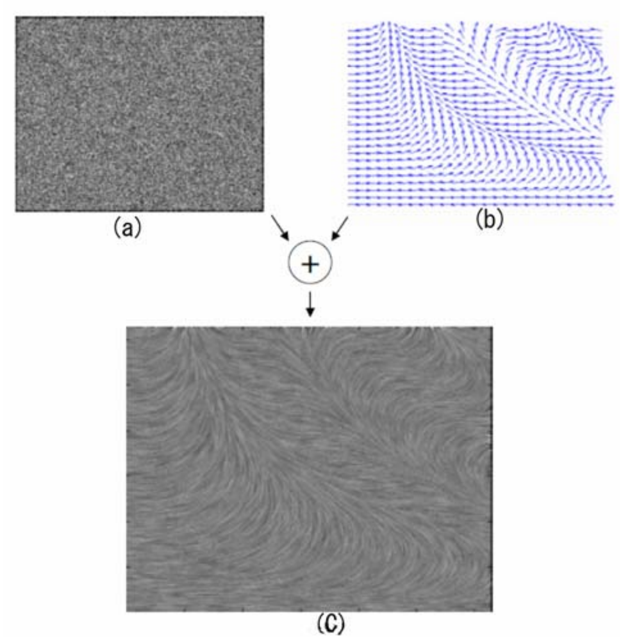


Figure 5-13. (a) a noise texture, (b) a vector field, (c)Output results using LIC algorithm

The resulting image appears stretched and squished along the directions of the distorting vector field streamlines, visualizing the flow with a minimum of display resolution. Vortices, sources, sinks and other discontinuities are clear shown in the resulting image, and the viewer can get an immediate grasp of the flow fields ``big picture''.

5.4.1.2 Available Method for Colorful Line Integral Convolution

Typically, only the grayscale noise texture is used in this algorithm. Make a colorful Line Integral Convolution can have many choices. Often, we can use random color noise based on a color table shown on Figure 5-14, or multiply the generated LIC image with the color image of vector field magnitude value shown on 5-15. The first method give a similar results as original LIC method, but lower contrast, e.g., the topology infrastructure of the vector field. The second method adds additional magnitude information, but also suffers from the loss of contrast, and it should be used as a post-processing of original LIC instead of revision.

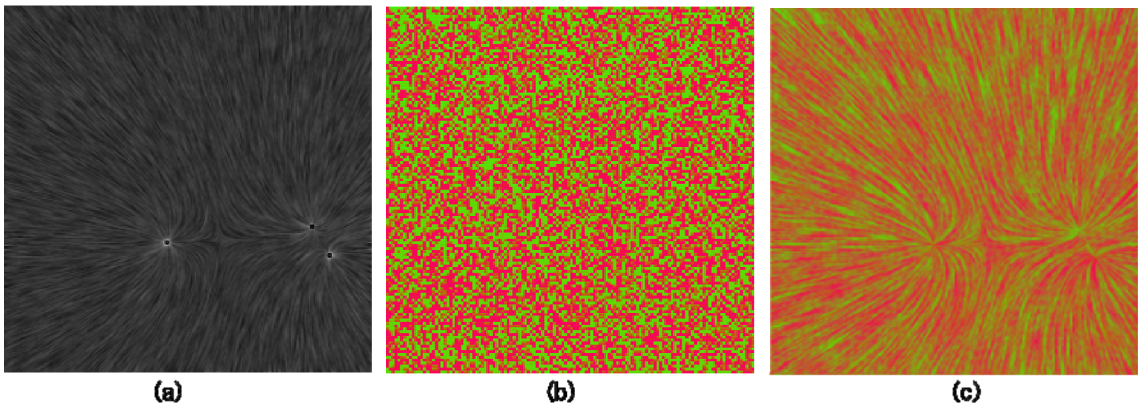


Figure 5-14. (a) Original LIC results using grayscale noise (b) A random color noise (c) LIC algorithm using random color noise texture

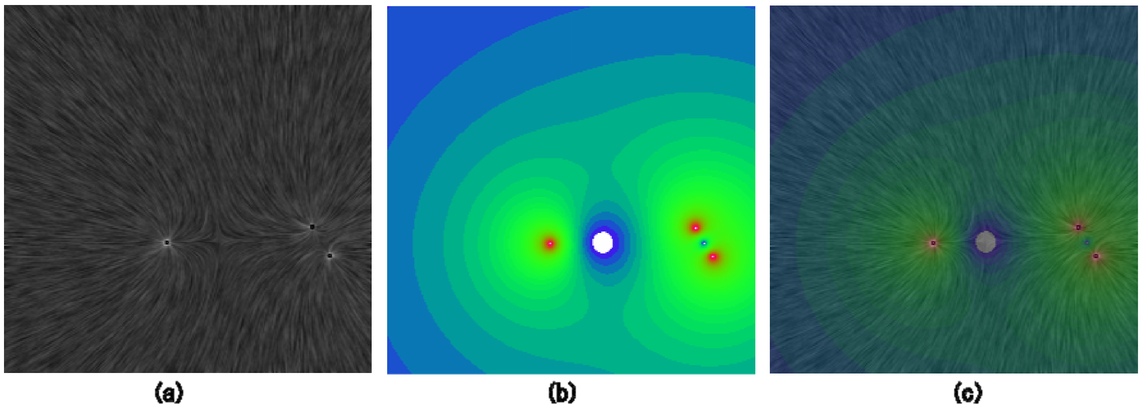


Figure 5-15. (a) Original LIC results using grayscale noise (b) Magnitude value image of vector field (c) Combine (a) with (b)

The new coming Color Stealing Algorithm [*Barnsley, 2006*] offers us a new availability to make use of the colorful texture to LIC algorithm.

5.4.1.3 Colorful Line Integral Convolution using Color Stealing Algorithm

5.4.1.3.1 The Top Function of the Iterated Function System

The following listed nomenclatures are all refer to two kooks from Dr. Barnsley[*Barnsley, 1988 and Barnsley, 2006*]. Let an iterated function system (IFS) be noted:

$$\boldsymbol{W} := \{\boldsymbol{X}; \boldsymbol{\omega}_{0,}\dots,\boldsymbol{\omega}_{N-1,}\} \tag{5.48}$$

This consists of a finite of sequence of one-to-one contraction mappings

$$\boldsymbol{\omega}_n : \boldsymbol{X} \to \boldsymbol{X}, \boldsymbol{n} = 0,1,\dots., \boldsymbol{N}-1 \tag{5.49}$$

Acting on the compact metric space $(\boldsymbol{X},\boldsymbol{d})$ with metric $\boldsymbol{d}$ so that for some $0 \le \boldsymbol{l} < 1$ we have

$$\boldsymbol{d}(\omega_n(x),\omega_n(y)) \le \boldsymbol{l} \cdot \boldsymbol{d}(x,y) \tag{5.50}$$

for all $x, y \in \boldsymbol{X}$.

Let A denote the attractor of the IFS, that is $\boldsymbol{A} \subset \boldsymbol{X}$ is the unique non-empty compact set such that

$$\boldsymbol{A} = \bigcup_n \omega_n(\boldsymbol{A}) \tag{5.51}$$

Let the associated code space be denoted by $\Omega = \Omega_{\{0,1,\ldots,N-1\}}$. This is the space of infinite sequence of symbols $\{\sigma_i\}_{i=1}^{\infty}$ belonging to the alphabet {0,1,...,N-1} with the discrete product topology. We will also write $\sigma = \sigma_1\sigma_2\sigma_3 \ldots \in \Omega$ to denote a typical element of Ω, and we will write ω_k to denote the k^{th} element of the sequence $\omega \in \Omega$. We order the element of Ω according to $\sigma < \omega$ ***iff*** $\sigma_k < \omega_k$ where k is the least index for which $\sigma_k \ne \omega_k$.

Let $\phi : \Omega \to \boldsymbol{A}$ denote the associated continuous addressing map from code space onto the attractor of the IFS. We note that the set of address of a point $x \in \boldsymbol{A}$, defined to be $\phi^{-1}(x)$, is compact and so processes a unique largest element. We denote this value by $\tau(x)$. That is, $\tau : \boldsymbol{A} \to \Omega$ is defined by

$$\tau(x) = \max\{\sigma \in \Omega : \phi(\sigma) = x\} \tag{5.52}$$

We call the $\tau(x)$ the top function of IFS.

5.4.1.3.2 Color Stealing Algorithm

A picture function is a mapping $\aleph : \boldsymbol{D}_{\aleph} \subset \boldsymbol{R}^2 \to \Im$ where $\Im$ is a color space, for example $\Im = [0,255]^3 \subset \boldsymbol{R}^3$. The domain $\boldsymbol{D}_{\aleph}$ is typically a rectangular subset of $\boldsymbol{R}^2$, we often take

$$\boldsymbol{D}_{\aleph} = \square := \{(x,y) \in \boldsymbol{R}^2 : 0 \le x, y \le 1\} \tag{5.53}$$

The domain of a picture function is an important part of its definition; for example a segment of a picture may be used define a picture function. Here we assume that given a picture function, we have some process by which we can render it to make pictures which may be printed, viewed on computer screens, etc. This is far from a simple matter in general.

Let two IFS's $\boldsymbol{W} := \{\square; \boldsymbol{\omega}_0, \boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_{N-1}\}$ and $\tilde{\boldsymbol{W}} := \{\square; \tilde{\boldsymbol{\omega}}_0, \tilde{\boldsymbol{\omega}}_1, ..., \tilde{\boldsymbol{\omega}}_{N-1}\}$ and a picture function $\tilde{\aleph}: \square \to \Im$ be given. Let $\boldsymbol{A}$ denote the attractor of IFS $\boldsymbol{W}$ and let $\tilde{\boldsymbol{A}}$ denote the attractor of IFS $\tilde{\boldsymbol{W}}$. Let $\boldsymbol{\tau}: \boldsymbol{A} \to \Omega$ denote the top function for $\boldsymbol{W}$. Let $\tilde{\boldsymbol{\phi}}: \Omega \to \tilde{\boldsymbol{A}} \subset \square$ denote the addressing function for IFS $\tilde{\boldsymbol{W}}$. Then we define a new picture function $\aleph: \boldsymbol{A} \to \Im$ by

$$\aleph = \tilde{\aleph} \circ \tilde{\boldsymbol{\phi}} \circ \boldsymbol{\tau} \tag{5.54}$$

This is the unique picture function defined by the IFS's $\boldsymbol{W}$, $\tilde{\boldsymbol{W}}$, and the picture $\tilde{\aleph}$. We say that it has been produced by fractal tops and color stealing. Colors are "stolen" from the picture $\tilde{\aleph}$ to "paint" code space. We make a code space picture, that is the function $\tilde{\aleph} \circ \tilde{\boldsymbol{\phi}}: \Omega \to \Im$, where we use fractal top of W to paint the attractor $\boldsymbol{A}$.

5.4.1.3.3 Color Stealing Algorithm for Colorful Line Integral Convolution

Start from two IFSs (the DRAWING IFS and the COLOURING IFS,), an INPUT IMAGE, a colored Red Green Blue (RGB) image, such as a photograph of a natural scene, and an OUTPUT IMAGE, which we want to render. Here we choose the following IFSs. The first IFS (DRAWING IFS) correspond to the image that we want to render, the second IFS (COLOURING IFS) acts upon rectangle that corresponds to the support of the INPUT IMAGE. We get the local IFS (LIFS) from the OUTPUT IMAGE using fractal compressing method [*Barnsley, 1993 and Fisher, 1995*], this can be extend and used as a address function as described in Eq. (5.54).

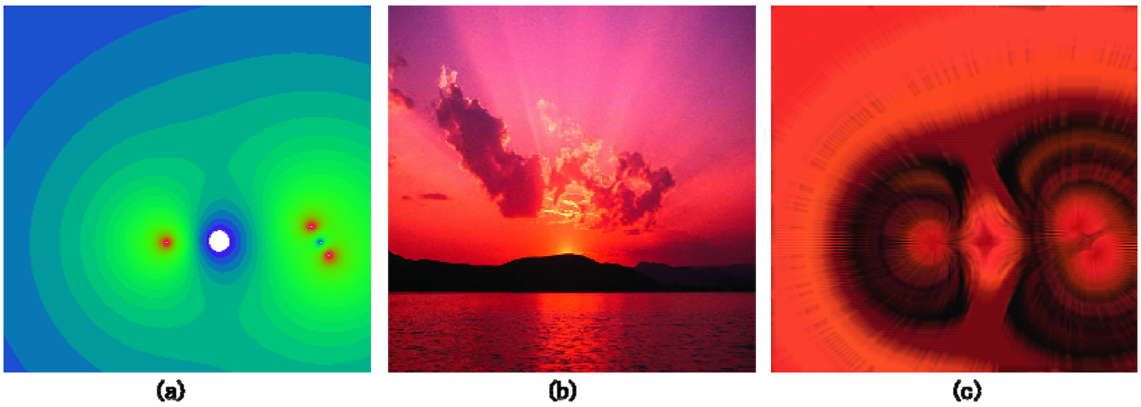


Figure 5-16. (a) image of vector field magnitude value (b) sunset image used as color palette (c) Color Stealing based LIC results

. We first get the magnitude color image of vector field and then get a fractal noise texture using the color stealing algorithm to steal color from some nature or physics image. We add the grayscale noise used in LIC algorithm by this fractal noise texture. One of the results stealing the color from a nature sunset image of vector field used in Figure 5-14 and 5-15 are shown on Figure 5-16. Compare the results of Figure 5-14, 5-15, and 5-16, we find that the color stealing based LIC method increase the topologied structure of the vector field while the global resolution decreased. This method can combine both the texture based vector field visualization and topology based visualization method. Better fractal image compression LIFS can be gotten by using better image palette, the method should offer a possible hybrid visualization method for vector field.

5.4.2 Topology Visualization of Three Dimensional Vector Field

Topological concepts are very powerful because given the critical points in a vector field and the tangent curves or surfaces connecting them; one could infer the shape of other tangent curves and hence to some extent the structure of the entire vector field [*Helman, 1990*]. Three dimensional vector fields are difficult to visualize, one simple method is to choose a set of points in the field and draws the arrows indicating the magnitude and direction of vector at each point. Unfortunately, this display way is limited to some small subset of data [*Globus, 1991*]. Three dimensional magnetic field topology in magnetosphere represents a domain of space plasma physics of great interest that is, as yet, beyond the reach of definitive theoretical analysis or numerical computations. After the computer graphics introduced as a field of study, the visualization technology based on it was closely resembled to the pictures in the vector field [*Cai, 2006*]. For three dimensional vector fields, direct visualization methods in which thousands of points and vectors or curves are displayed are inadequate.

5.4.2.1 Critical Point

Critical points or magnetic nulls are the points where the magnitude of the magnetic field vector vanishes. These points may be characterized by the behavior of nearby magnetic field curves or surfaces [*Helman, 1990*]. The set of curves or surfaces which end on the critical point is of special interest, because it defines the behavior of the magnetic field in the neighborhood of the critical points. If all of the eigenvalues of critical points in the region where we consider are hyperbolic, then the magnetic vector field topologies are uniquely determined only by the critical points, the magnetic field curves and surfaces that are originated from the critical points. The usual magnetic field configuration satisfies the hyperbolic conditions. Thus the particular sets of the critical points, curves and surfaces can be use to define a skeleton that uniquely characterizes the magnetic field.

For simplicity, we use $\vec{V} = (u, v, w)^t$ instead of $\vec{B} = (B_x, B_y, B_z)^t$ as the magnetic vector field. The magnetic vector field vanishes at the critical points, and we expend the magnetic vector field around the critical point $\vec{X}_0 = (x_0, y_0, z_0)^t$ to the first order:

$$\vec{V} = \begin{pmatrix} u(x_0, y_0, z_0) \\ v(x_0, y_0, z_0) \\ w(x_0, y_0, z_0) \end{pmatrix} + \left.\begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix}\right|_{(x_0, y_0, z_0)} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} + \mathrm{O}(2) \tag{5.55}$$

Here we know that $\begin{pmatrix} u(x_0, y_0, z_0) \\ v(x_0, y_0, z_0) \\ w(x_0, y_0, z_0) \end{pmatrix} = 0$ at the critical point $\vec{X}_0 = (x_0, y_0, z_0)^t$ and then

$$\vec{V} = \left.\frac{\partial(u, v, w)}{\partial(x, y, z)}\right|_{(x_0, y_0, z_0)} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} = \left.\begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix}\right|_{(x_0, y_0, z_0)} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} \tag{5.55}$$

$\vec{J} = \left.\begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix}\right|_{(x_0, y_0, z_0)}$ is the Jacobian matrix of the three dimensional magnetic field vector, $\vec{V} = (u, v, w)^t$. A critical point can be classified by the eigenvalues, λ_1, λ_2

and λ_3 of $\vec{J}$ with respect to the critical point $\vec{X}_0 = (x_0, y_0, z_0)^t$. Figure 5-17 shows how the eigenvalues classify the three dimensional critical points as an attracting node (7), a repelling node (6), an attracting focus (5), a repelling focus (3) and fours saddles (1), (2), (4), (8). We should note that a positive or negative real part of an eigenvalue indicates an attracting or repelling nature, respectively, and the nonzero imaginary part creates a spiral structure around the critical point. The saddle points are distinguished by the characteristic magnetic Σ-surfaces that are spanned by the two eigenvectors that have same sign of real parts of eigenvalues. Or the two outgoing or incoming eigenvectors span the Σ-surfaces.

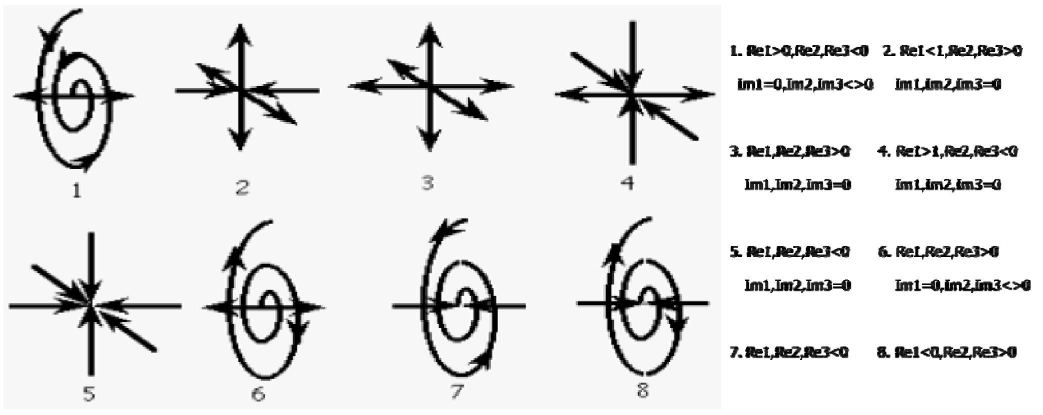


Figure 5-17. Classification of three-dimensional critical points.

In our magnetic field vector, we have the solenoidal condition:

$$\nabla \cdot \vec{B} = 0 \tag{5.56}$$

Thus the three eigenvalues, λ_1, λ_2 and λ_3 of the critical point $\vec{X}_0 = (x_0, y_0, z_0)^t$ satisfy the following condition:

$$\lambda_1 + \lambda_2 + \lambda_3 = 0 \tag{5.57}$$

So all of the eigenvalues can't be all positive or negative, therefore the four types of saddle points satisfy the condition assuming that all the critical points are hyperbolic in the region as shown in Figure 5-18. According to the notation used in [*Lau, 1990*], if two of the real parts of the eigenvalues are negative, it is called A type critical point, otherwise, called B type critical point. If the imaginary parts of the eigenvalues are not zero, the A or B type critical point are called As or Bs type critical point respectively shown in Figure 5-18. In three-dimension, both A (As) type and B (Bs) type critical points connected together form the three-dimensional X-point as shown in Figure 5-19.

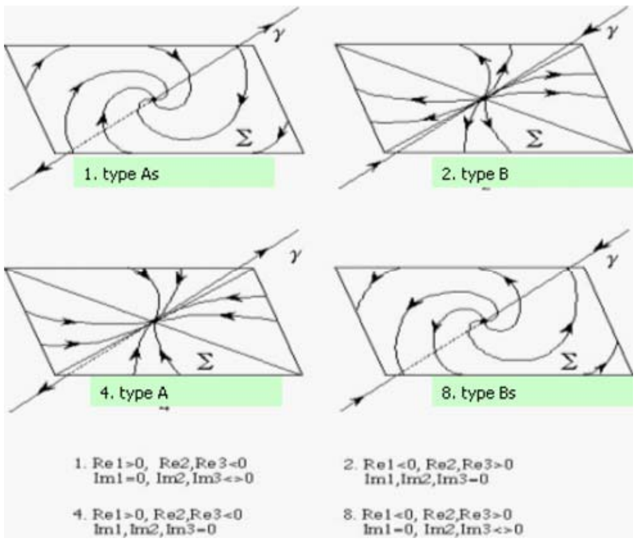


Figure 5-18. Classification of saddle points. (a) type As, (b) type B, (c) type A, (d) type Bs.

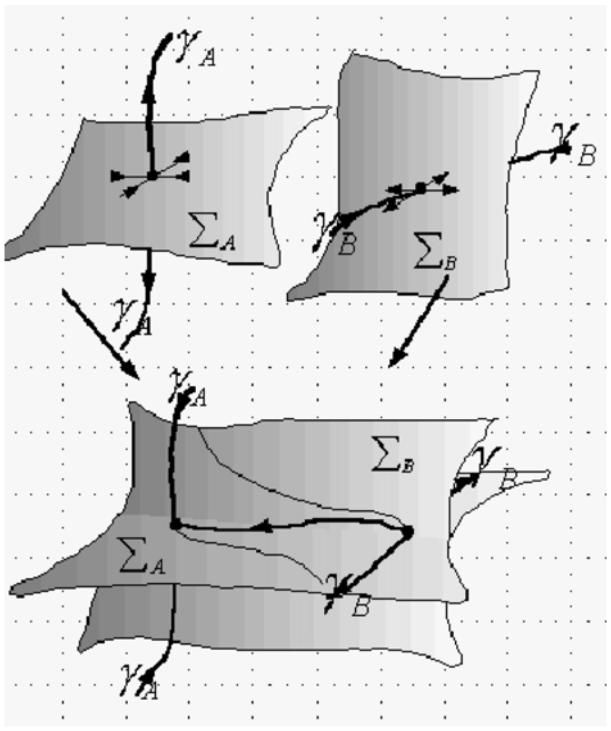


Figure 5-19. Three-dimensional X-point. Type A and type B merge and form a three-dimensional X-point.

Thus the types of the critical pints, their numbers, and the rules governing the relationship between them will characterize the full magnetic field pattern, that uniquely decide the magnetic field topology [*Cai, 2006; Helman, 1990; Forbes, 2000*], that means topological equivalence.

5.4.2.2 Flowchart of the Visualization Algorithm

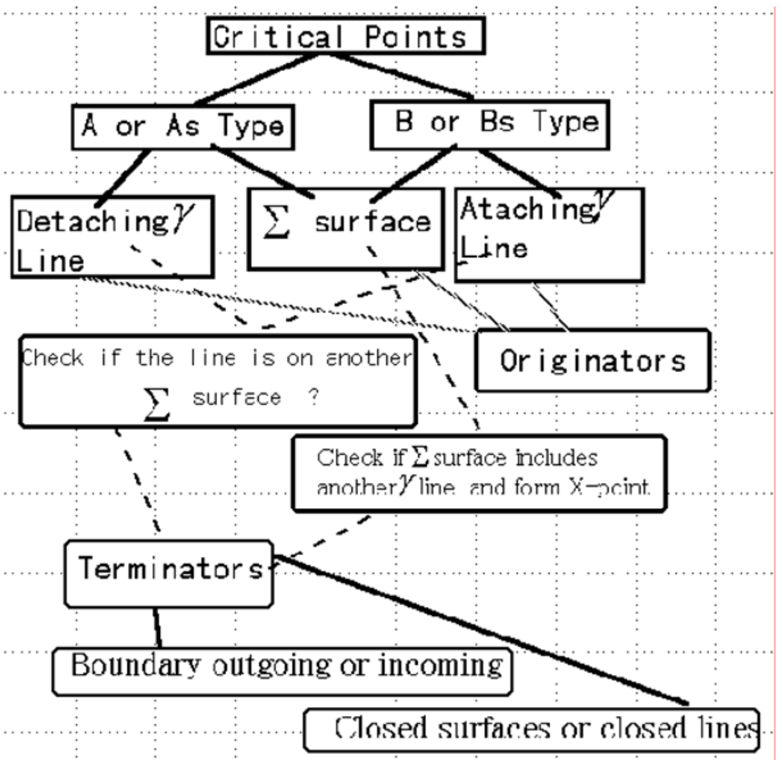

Critical Points
A or As Type
B or Bs Type
Detaching γ Line
Σ surface
Ataching γ Line
Check if the line is on another Σ surface ?
Originators
Check if Σ surface includes another γ line and form X-point
Terminators
Boundary outgoing or incoming
Closed surfaces or closed lines

Figure 5-20. The visualization flowchart used in our topology visualization of three-dimensional vector field.

The visualization method of the topology for three dimensional vector fields was introduced in [*Cai, 2006*]. The topology provides useful information on the entire magnetic field structure and also applied to magnetic reconnection. Some results using this visualization method had been shown on Figure 5-12. The algorithm of this visualization technology is shown in Figure 5-20. First of all, we made a program to find all of the critical points and then classify their types, attracting node, focus, repelling node, focus, and A type, As type and B type Bs type critical points. Then we select the A or As type, B or Bs type critical points to visualize. We could detach the γ-line and decide the Σ-surface of the critical point. Then check if the γ-line of one critical point is on the Σ-surface the another critical point, or if the Σ-surface of one critical point includes the γ-line of another critical point to form a three-dimensional X-point. The topological rules between critical points must be satisfied globally. Thus, some γ-lines and Σ-surfaces can be terminated as the boundary outgoing or incoming and satisfy the rules between them including the condition outside [*Murray, 1982*].

We accomplish the visualization project with OpenGL library after finding the critical points and their γ-lines and lines in their Σ-surfaces and then visualizing them with critical points. We make our efforts to finish this visualization project that includes the following parts, such as, the data read, listing and drawing the critical points, drawing

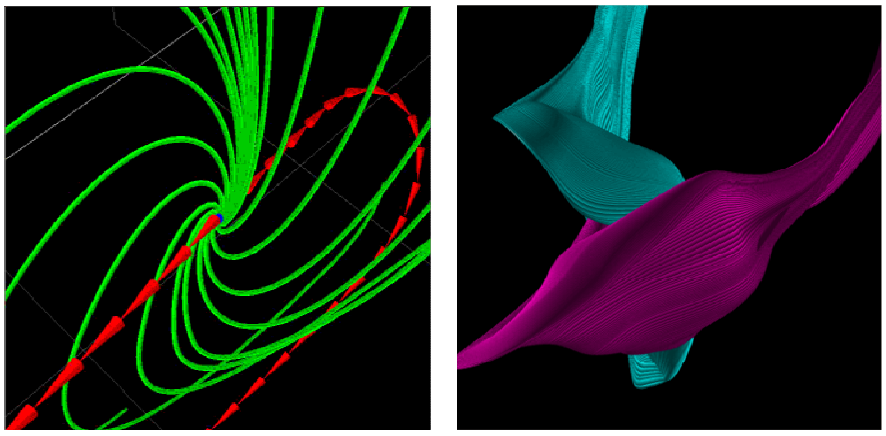


Figure 5-21. (a) The critical point and γ-line and lines of Σ-surface visualized by our visualization project. (b) Two Σ-surfaces of two critical points.

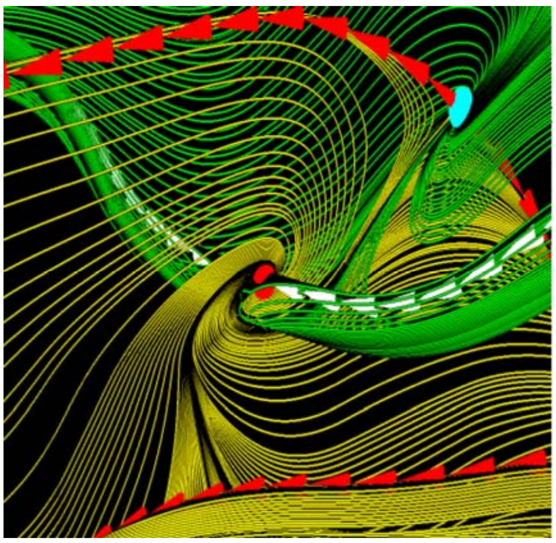


Figure 5-22. The two connected critical points.

the Earth, making the simulation domain region and three dimensional axis, visualizing the lines, etc.. We show the example of the topology structure with the critical point and its γ-line and Σ-surface. In Figure 5-21, (a) is for one critical point, in which the blue point is the critical point, the red line is its γ-line, and the green lines are in its Σ-surface, (b) is the Σ-surfaces of two critical points. Figure 5-22 shows the two connected critical points, and their γ-lines, white lines, and other lines in their Σ-surfaces. In which the γ-line of one critical point is in the Σ-surface of another critical point. In the meantime, we have checked all of the visualization source code and all of the results of very steps, for example, the critical point, the eigenvalues and eigenvector of critical point, the γ-line and the lines in the Σ-surface of the critical point, the connected critical points, and so on.

Chapter 6:
Virtual Satellite for PIC Simulation

6.1 Possibility of Data Assimilation for Global Particle Simulation

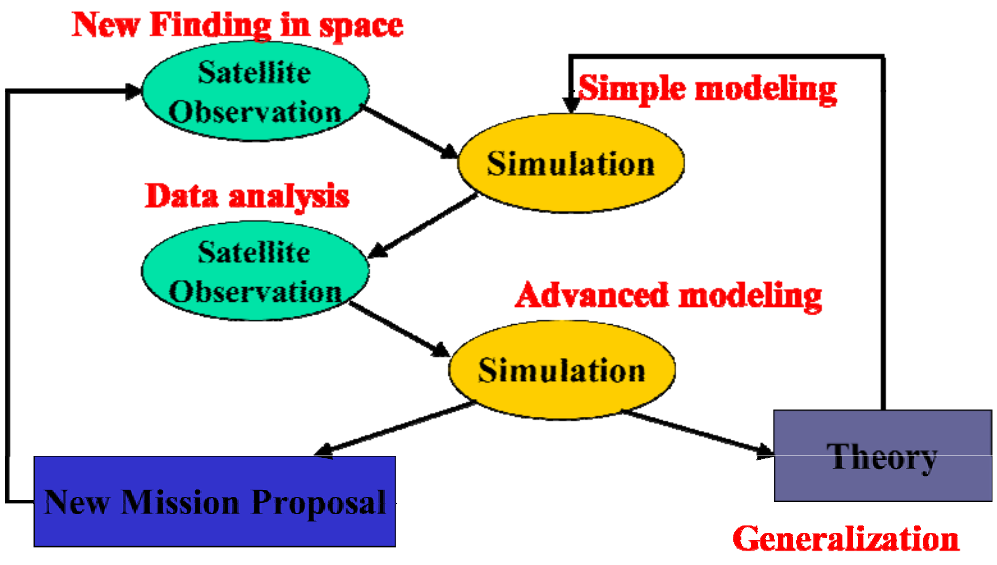


Figure 6-1 Research Flow of the Space Weather Field

As described in chapter 1, the computer simulation had been used as a new method to complement the traditional physics research method, namely, theory and experiment. In the case of large scale physical phenomena, like the interaction between IMF and Earth's magnetosphere, one must often substitute satellite observations of naturally occurring behavior for well controlled experiments. Often the research flow can be presented as shown in Figure 6-1. The basic physics laws of space physics and plasma help us to construct the simple numerical model for computer simulation, the initial physical parameter can be setup from both theory and satellite observation. After simulation, we can compare the results with both theory and satellite observation and adjust the simple numerical model to advanced numerical model. With this new advance simulation, one can use the results to predict the behaviors of all system include the unperformed observation area, and then the new theory can be generalization and the new mission proposal can be submitted.

Data assimilation is a novel versatile methodology for estimating physical variables [Evensen]. The estimation of a quantity of interest via data assimilation involves the combination of observational data with the underlying dynamical principles governing the system under observation. This method has been successfully applied to the global MHD simulation but not for global particle simulation yet as the spatial resolution are limited by their extremely needs for compute resources. On this research, first we prove the scalability of global PIC code on computational Grid from both theoretical analysis and benchmarks; Second we implement some physical simulation which can validate our code. These results make sure that implement the global particle simulation at quantitative spatial resolution which can be compared with the results of satellite observation, make use of the data assimilation for new simulation is possible now.

6.2 Virtual Satellite for PIC Simulation Data

Once a simulation is performed, another question appears: is it possible to insert, In this simulation, the experimental conditions of local measurements specific to a given space mission? Of course this question is not dedicated to full particle simulations only and can be extended to any other type of simulation. If this approach could be systematically resolved, it would allow to establish a more direct link between the simulation and experimental observation. Such procedures have been initially developed for "mimicing" the crossing of a quasi parallel shock issued from lD hybrid simulation code [*Giacalone, 1994*]. At that time, the shock crossing was restricted along the shock normal only because of the use of lD simulation results.

Based on the original idea of Dr. Giacalone, we developed the virtual satellite system using OpenDx virtualization toolkits, which had been implemented on unix system first by IBM and later open sourced to other operation system as linux, windows and Mac. In this system, three types of interaction between satellite and simulation data has been mimicked, they are moving satellite in continues simulation data, moving satellite in static simulation data, and resting satellite in continues simulation data. The first case, moving satellite in continues simulation data, can be directly used to mimic the real satellite observation. The other two cases can be used to extent the satellite observation from both spatial and temporal direction.

In this research, some efforts have been invested in order to "mimic" the local measurements made by the four Satellites Cluster-2 mission crossing a 2D shock issued from 2D full particle simulations. In this case, any direction of the shock crossing with respect to the shock normal may be mimiced. And this kind of virtual satellite can also be extended to our 3D EMPM simulation data as presented in chapter 5. Some examples are presented in the following sections.

In the first example, the purpose was to explain the different magnetic field

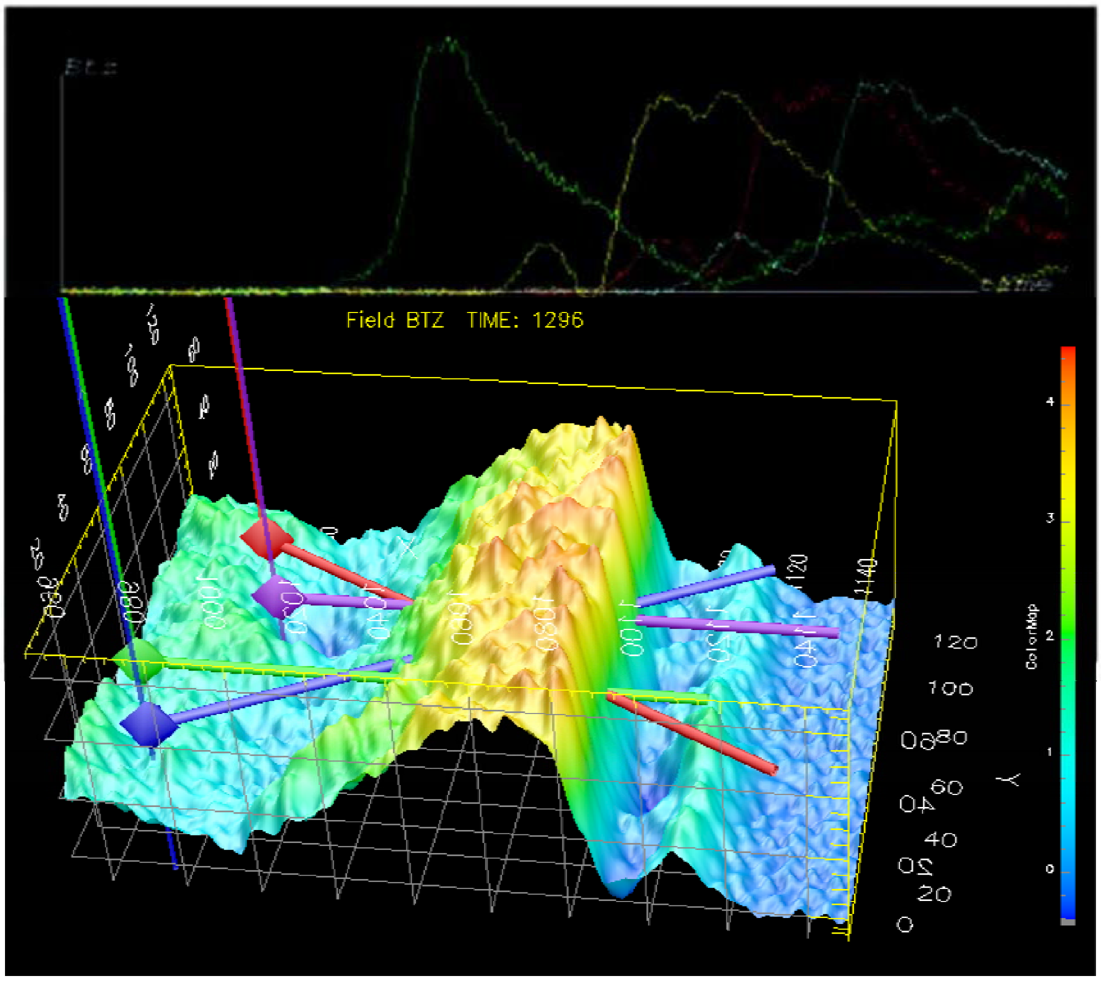


Figure 6-2. The bottom plane show the main view of the magnetic field and four crossing virtual satellites mimicking the Cluster-2 mission. Top panel show the local measurement of the magnetic field during the shock front crossing.

signatures of the measurements made by each satellite during the shock crossing. This procedure is summarized as follows. First, a 2D full particle simulation of a supercritical shock has been performed. Non-stationarity of the shock front and corresponding spatial/time scales have been retrieved. The attention is focused on the longest characteristic time scale; for doing so, we note the start and end times of a given cyclic self-reformation. Second another run is performed identical to the first one, but where field components are stored at each time step between the start and the end time of the whole selected reformation cycle. Let us focus now on the main magnetic field component. Then we have a first x-y plane containing this magnetic field data with a high time accuracy. The size of this table corresponds to the plasma box. Four virtual satellites are presented in this field and has dedicated staring location (x, y) and moving orbit, so that any type of shock crossing (inbound, outbound, oblique, normal crossing etc.) may be reproduced independent of the shock move. Measurements of local magnetic field performed by four virtual satellites during a typical shock crossing are shown in Figure 6-2. Four different signatures versus the time of the main magnetic field component are clearly evidenced by the different satellites; this illustrate quite well the field turbulence which take place over different times and spatial scales at the shock front. Complementary waves analysis are now under active investigation in order to identify these waves.

How to optimize this post-processing technique? This mimicing procedure offers different advantages and can be used in two different ways. The first one consists in reproducing experimental results with a given configuration defined by the real satellites. In other words one only inserts within the results the configuration approaching as much as possible the real conditions; the initial priority is motivated by data obtained by conditions of real space mission. The second way consists in inverting this priority. As an example, 1et us focus on the small scale turbulence along the shock front. Its analysis requires fixing at least two virtual satellites aligned along the direction parallel to the shock front and distant each other over a scale L_{sat} larger than the characteristic scale λ_{ripp}. Then these satellites should cross the shock front almost simultaneously so that any wave's activity along x-direction should be minimized in the signatures of the shock crossing. By adjusting the scale L_{sat} with respect to λ_{ripp}, one can analyze different types of waves contributing to the front nonhomogeneity; indeed λ_{ripp} may vary according the Mach regime, the shock angle and the upstream plasma β conditions. This allows to define the best satellites configurations for analyzing or identifying a certain type of components contributing to this local turbulence. Further, the next step will consist in searching among the whole set of shock crossing by CLUSTER-2, the configurations approaching that of the virtual satellites and in comparing experimental and numerical signatures. In this case, the initial priority is motivated by results obtained by the virtual satellites. Similar procedures can be also used for local measurements of particles distribution functions;

for so doing, 2D fu11 particle simulations have been made recently based on a high number of particles per grid, in order to obtain appropriate local statistic.

The second example concerns the detailed analysis of the electron fore-Shock. In this case any experimental measurement of the particle distribution function is local in space but involves particles originating from quite different regions, and which are

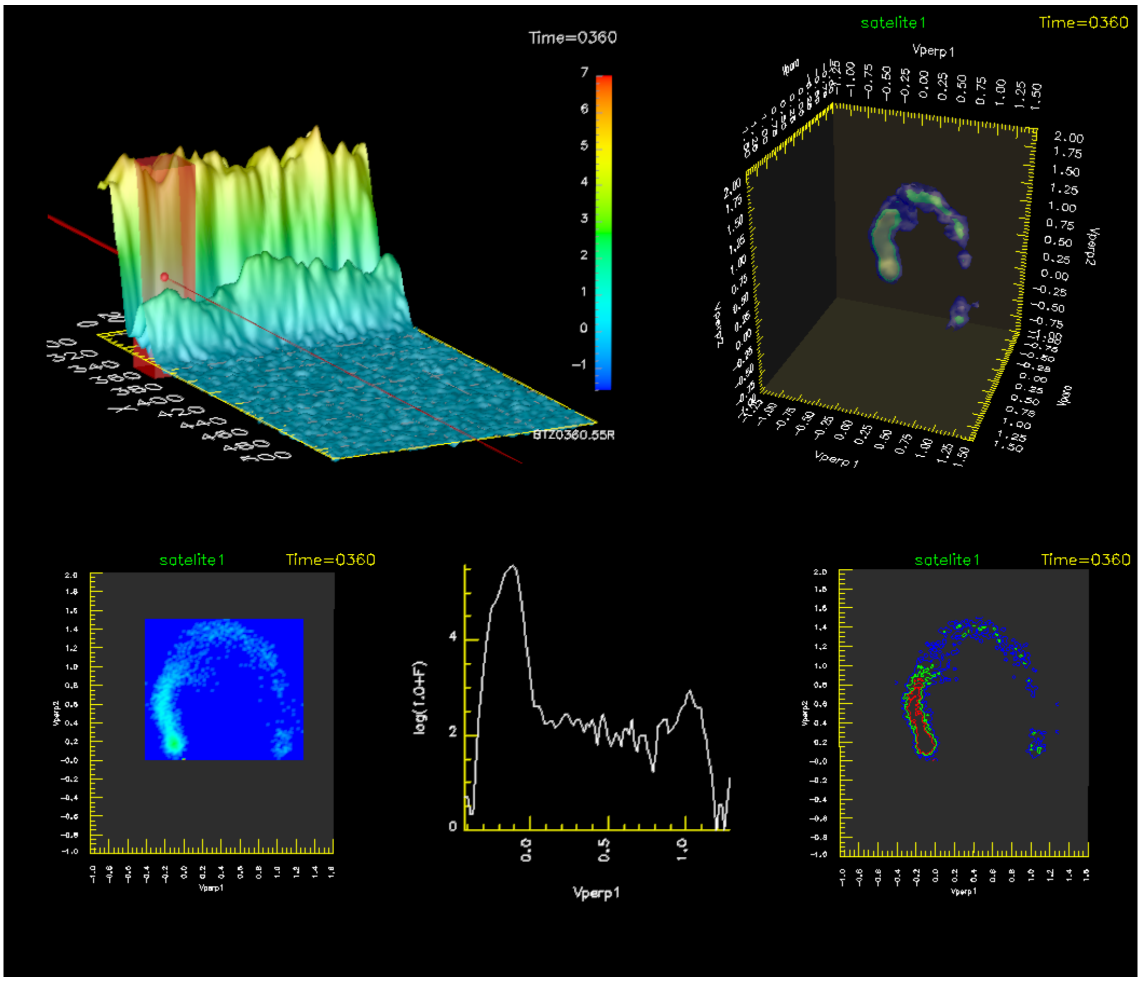


Figure 6-3. The top-left plane show the main view of the magnetic field and one crossing virtual satellites mimicing the Cluster-2 mission. Top-right panel show the isosurface present of particle distribution. The bottom panel show the distribution image and plot in one and two dimesion..

counted locally at the time and location of the measurement. These electrons have been ref1ected from different locations of the curved shock front; different ref1ection processes are involved which can be identified with different signatures in the electron distribution. In recent 2D full particle simulations of electron foreshock [*Savoini , 2001*], such as "time-Of-night-effects" which play an important role have been self-consistently included. How to mimic the real measurement performed by the satellite? In so doing, a graphical procedure has been developed which consists by super imposing a curved grid adapted to the curved bow shock and aligned to the upstream edge of the electron foreshock. Such a procedure allows keeping the size of each grid constant when moving in angle or with upstream distance from the front; this feature cannot be obtained with the use of a radial grid set (centered at the shock curvature point). The size of the grid must be chosen in appropriate way (1arge enough to have credible statistics within each grid, and small enough to scan the whole foreshock region with a minimum accuracy in angle and in distance from the front). Then, within each grid, local distribution is calculated and energy spectrum analysis is also performed. The next step consists in superimposing the trajectory of a virtual satellite to this curved grid set, mimicing the real satellite trajectory. This trajectory will select the set of successive grids within which the local electron distribution is measured; then, a direct comparison between experimental and numerical results will be possible. A refinement can be later obtained by adjusting the size of the space grid to the integration time of the real experiment (performed often over the satellite spin time period). Indeed, this feature has a full meaning since the size of each grid corresponding to the distance covered by the satellite during its full spin period is constant, whatever the angle along and the distance from the curved shock front are. The results are presented in the Figure 6-3, only one satellite of four satellites is shown here for saving space.

These examples illustrate the increasing importance of the virtual satellite post-processing stage and the efforts which need to be invested. More efforts (in developing) are necessary at a time the number of particles involved in simulations and the size of associated field components tables are drastically increasing, especially for a quantitative level 3D EMPM global simulation. This means that improved graphics techniques (interactive 2D/3D if possible) with sophisticated and real time animations facilities (as virtual reality techniques) need to been encouraged and adapted to space plasma physics problems in order to analyze the huge amounts of stored data obtained from these large scale simulations.

Chapter 7: Conclusions

There are three main works had been performed in this thesis, they can be summarized as follows:

First, Isoefficiency analysis and benchmark of two three-dimensional (3D) PIC codes with different physics and entirely different communication patterns, named as skeleton PIC code and parallel TRISTAN PIC code, had been implemented on two computational grid, namely supercomputer based Grid and cluster based Grid.

(i) The collective communication does not scale correctly on a system with poor communication performances such as the "Cross-WAN" PC cluster enabled Grid system. The MPPtest benchmarks based on MPI_Barrier and MPI_Scatter configurations confirm that the use of all-to-all collective communication in "cross-WAN" Grid configuration is not appropriate for high number of processors (at least $p > 80$) managing small size messages (< 2 Mbytes). More generally, the "MPI_Scatter" system does not scale well for small size messages exchanged on large number of processors.

(ii) Parallel TRISTAN PIC code (without all-to-all collective communications) is more scalable than the parallel skeleton PIC code (with all-to-all collective communication), in cluster based computational Grid systems where communication performances are poor. This result has required a deeper analysis of the speedup of the bandwidth and of the communication time between processors in two different communication patterns supported in present MPPtest benchmarks (MPI_Barrier and MPI_Scatter).

(iii) The speed up analysis confirms that the TRISTAN code has higher speedup since the use of FFT, which requires intensive collective communication, leads to a saturation of the speedup of the skeleton PIC code.

Based on the speedup and scalability benchmarks and analysis, we confirm that the simulation code including all-to-all collective communication scale less on the cross-WAN cluster based Grid systems, where network latency is very large. The 3D EMPM simulation code based on TRISTAN code should be a scalable code for the computational Grid.

Second, two global particle simulations using 3D EMPM code are performed, they are "Sash" simulations with Turning IMF from Northward to Southward and Bifurcation and hysteresis of the magnetospheric structure with a varying Southward IMF.

(i) 3D EMPM sash simulation show that a weak magnetic field band sash forms of the cusp at high latitude (in each hemisphere) for a northward IMF $\boldsymbol{B}_{IMF}$. After the IMF $\boldsymbol{B}_{IMF}$ rotated from northward to duskward and then to southward, this sash rotate out of the poleward direction to the equatorial plane. For the duskward IMF, on dayside ($\boldsymbol{x}-5\mathrm{Re}$), electrons and ions flux are strong along the magnetopause flanks but in different regions apart from the cusp region. When penetrating further in the night side and approaching the near magnetotail, two important processes take place. First, the "sash" approaches each other along the titled 45° axis and merge progressively to join the neutral sheet in the tail, this form a characteristic "crosstail-S". Second, both electrons and ions use this magnetospheric "sash" as a groove to penetrate the inner magnetosphere from the top cusp to the inner plasma sheet.

(ii) In the second simulation, it show that the solar wind magnetic field and the magnetospheric surface magnetic field bifurcate as the southward IMF increases and exceeds a critical value. When the parameter λ exceeds the critical value λ_c, there are no adjacent bifurcated flow that differ only infinitesimally from the unstable know magnetic field. Instead, one evidences a finite jump to a new branch of the field that may represent a radical change in the topology of the external magnetic field and in the "phase portrait" of the magnetospheric surface field. This jump and the associated bifurcation are difficult to predict precisely. This "subcritical" type (or jump) is the signature of the reconnection process which takes place at the subsolar point. As the parameter λ decreases the magnetopause recovers its initial state but not following the same path used as λ increases: hysteresis effect is evidenced within a certain interval of IMF $|B_z|$ values. This hysteresis is the signature of irreversible dissipation processes during which the topology of the magnetic field is strongly changed and does not recover its initial state (for a same value of the IMF $|B_z|$ within this interval). In other words, the field energy previously stored in the magnetosphere (since $|B_z|$ is increasing) as the parameter λ increases is not recovered as λ decreases. These properties are the consequences of the reconnection process.

Base on these two simulations results, the 3D EMPM global simulation have been physically validated.

Third, we developed the virtual satellite system using OpenDx virtualization toolkits. three types of interaction between satellite and simulation data has been mimiced, they are moving satellite in continues simulation data, moving satellite in static simulation data, and resting satellite in continues simulation data. Two examples have been invested in order to "mimic" the local measurements made by the four Satellites Cluster-2 mission crossing a 2D shock issued from 2D full particle simulations. In this case, any direction of the shock crossing with respect to the shock normal may be mimiced.

References

[Axford, 2002] W. I., Axford, Connection and reconnection, *Adv Space Res*, Vol.*29*, pp. 1025-1033, 2002.

[Barnsley, 1988] M. F. Barnsley, Fractals Everywhere, Academic Press, New York, 1988.

[Barnsley, 1993] M.F. Barnsley and L.P. Hurd, *Fractal Image Compression*, A.K. Peters, Wellesley, Mass., 1993.

[Barnsley, 2006] M. F. Barnsley, “Superfractals”, Cambridge University Press, 2006

[Bester, 1999] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, GASS A data movement and access service for wide area computing systems, pp.78-88, In Proc. IOPADS’99, ACM Press, 1999.

[Birdsall and Langdon, 1985] Birdsall, C. K., and A. B. Langdon, Plasma Physics via Computer Simulation, McGraw-Hill,New York, 1985.

[Birn, 1996] J. Birn, M. Hesse, and K. Schindler, MHD simulation of magnetotail dynamics, J. Geophys. Res, Vol.101, pp. 12939, 1996.

[Buneman 1993] Buneman, O., TRISTAN: The 3-D, E-M particle code, in Computer Space Plasma Physics, Simulation Techniques and Software, edited by H. Matsumoto and Y. Omura, P. 67, Terra Sci., Tokyo, 1993.

[Buneman 1995] Buneman, O., K. -I. Nishikawa, and T. Neubert, Solar wind-magnetosphere interaction as simulated by a 3D EM particle code, Space Plasmas: Coupling between Small and Medium Scale Processes, Geophys. Monogr. Ser., Vol. 86, edited by M. Ashour-Abdalla, T. Chang, and P. Dusenbery, pp. 347, AGU, Washington, D.C., 1995.

[Cabral, 1993] B. Cabral and C. Leedom, Imaging Vector Fields Using Line Integral Convolution, Proc. SIGGRAPH ’93, pp. 263-270, 1993.

[Cai, 2001] Cai, D. S., et al., Visualization and criticality of three-dimensional magnetic field topology in the magnetotail, *Earth Planets Space*, Vol.*53*, pp.1011-1019, 2001.

[Cai, 2003] Cai, D. S., Yaoting Li, Ken-Ichi Nishikawa, Chijie Xiao, Xiaoyang Yan and Zuying Pu, Parallel 3D Electromagnetic Particle code using High Performance Fortran: Parallel TRISTAN, Space Plasma Simulation, Springer, pp. 25-53, 2003.

[Cai, 2006] D. Cai, K. I. Nishikawa and B. Lembege, Magnetotail field topology in a three-dimensional global particle simulation, Plasma Phys. Control. Fusion, Vol. 48, Issue.12B, pp.B123–B135, 2006.

[Czajkowski, 1998] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, A resource management architecture for metacomputing systems, Lecture Notes in Computer Science, Vol.1459, pp.62-82, 1998.

[Czajkowski, 1999] K. Czajkowski, I. Foster, and C. Kesselman, Co-allocation services for computational grids, In Proc. 8th IEEE Symp. on High Performance Distributed Computing, pp.37-46, IEEE Computer Society Press, 1999.

[Decyk, 1979] Decyk, V.K. and Dawson, J.M., Computer model for bounded plasma, Journal of Computational Physics, Vol.30, pp. 407-427, 1979.

[Decyk, 1995] V. K. Decyk, Skeleton PIC Codes for Parallel Computers, Computer Physics Communications, vol. 87, Issue 1-2, pp.87-94, 1995.

[Evensen] G. Evensen, Data Assimilation: The Ensemble Kalman Filter, Springer, 2006.

[Fagg, 1998] G. E. Fagg, K. S. London, and J. J. Dongarra. MPI Connect: managing heterogeneous MPI applications interoperation and process control. In V. Alexandrov and J. Dongarra, editors, Recent Advances in Parallel Virtual Machine and Message Passing Interface, volume 1497 of Lecture Notes in Computer Science, pp. 93–96. Springer, 1998. 5th European PVM/MPI Users' Group Meeting.

[Fisher, 1995] Y. Fisher, *Fractal Image Compression, Theory and Application*, Springer-Verlag, New York, 1995.

[Fitzgerald,, 1997] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, A directory service for configuring high-performance distributed computations, In Proc. 6th IEEE Symp. on High Performance Distributed Computing, pp. 365-375, IEEE Computer Society Press, 1997.

[Foster, 1998] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, A security architecture for computational grids, Conference on Computer and Communications Security, pp.83-92, 1998.

[Foster, 1999-a] I. Foster and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, MorganKaufmann, 1999.

[Foster, 1999-b] I. Foster and N. T. Karonis. A grid-enabled MPI: Message passing in heterogeneous distributed computing systems. In Proceedings of SC 98. IEEE, Nov. 1999. http://www.supercomp.org/sc98.

[Foster, 2001] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International J. High Performance Computing Applications, Vol.15, Issue.3, pp. 200-222, 2001.

[Foster, 2003] I. Foster and A. Iamnitchi, On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing, Lecture Notes in Computer Science, Vol. 2735, pp.18-128, Springe, Berlin, 2003.

[Forbes, 2000] T. G. Forbes, Reconnection Theory In Three Dimensions, Adv. Space Res. Vol.26, No.3, pp.549-556, 2000.

[Frank, 1995] Frank, L. A., M. Ashour-Abdalla, J. Berchem, J. Raeder, W. R. Paterson, S. Kokubun, T. Yamamoto, R. P. Lepping, F. V. Coroniti, D. H. Fairfield, K. L. Ackerson, Observations of plasmas and magnetic fields in Earth's distant magnetotail: Comparison with a global MHD model, J. Geophys. Res., Vol.100, No.A10, pp. 19,177-19,190, 1995.

[Fujimoto, 1997] Fujimoto, M., Mukai, T., Matsuoka, A., Nishida, A., Terasawa, T., Seki, K., Hayakawa, H., Yamamoto, T., Kokubun, S., Lepping, R. P., Dayside reconnection field lines in the southdusk near-tail flank during an IMF By>0 dominated period, Geophys. Res. Lett Vol.24, pp. 931-934, 1997.

[Gabriel, 1998] E. Gabriel, M. Resch, T. Beisel, and R. Keller. Distributed computing in a heterogenous computing environment. In Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes in Computer Science. Springer, 1998.

[Giacalone, 1994] Giacalone J., S. J. Schwartzand, D. Burgess, Artificial space craft in hybrid simulations of the quasi paral1el Earth's bowshock: analys of time series versus spatial profi1es and a separation strategy for Cluster, Ann Geophys Vol. 12, pp.591-601, 1994

[GLOBUS] www.globus.org

[Globus, 1991]A. Globus, C. Levit, and T. Lasinski, A Tool for Visualizing the topology of three-Dimensional Vector Fields, IEEE, P. 33-40, 1991.

[Goodrich, 1998] C. C. Goodrich, J. G. Lyon, M. Wiltberger, R. E. Lopez, and K. Papadopoulos, An overview of the impact of the January 10-11, 1997 magnetic cloud on the magnetosphere via global MHD simulation, Geophys. Res. Lett., Vol. 25, pp. 2537, 1998.

[GridChallenge] www.hpcc.jp/sacsis/2006/grid-challenge

[Gropp, 1996] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, A high-performance portable implementation of the MPI passage passing interface standard, Parallel Computing, Vol. 22, pp.789-828, 1996.

[Gropp, 1999] W. Gropp, and E. Lusk, Reproducible measurements of MPI performance characteristics, Technical Report ANL/MCS-P755-0699, pp.11-18, Mathematics and Computer Science Division, Argonne National Laboratory, June 1999.

[Guckenheim and Holmes, 1988] Guckenheim, J., and P. Holmes Nonlinear oscillation, dynamical systems and bifurcation of vector fields, edited, Springer, New York,(1983).

[Gurnett, 2005] D.A. Gurnett and A. Bhattacharjee, Introduction to Plasma Physics: With Space and Laboratory Applications, Cambridge University Press, 2005.

[Helman, 1990] J. L. Helman and Lambertus Hesselink, Surface Representations of Two- and Three-Dimensional Fluid Flow Topology, IEEE, 1st Conference on Visualization,, pp. 6-13, 1990.

[Henson, 2002] V.E. Henson, and U.M. Yang, BoomerAMG: A parallel algebraic multigrid solver and preconditioner, App. Num.Math., vol. 41, Issue 1, pp. 155-177, 2002.

[Hockney and Eastwood, 1981] Hockney, R. W., and J. W. Eastwood, Computer simulation using particles, McGraw-Hill,New York, 1981.

[Imamura, 2000] T. Imamura, Y. Tsujita, H. Koide, and H. Takemiya, An architecture of Stampi: MPI library on a cluster of parallel computers. In J. Dongarra, P. Kacsuk, and N. Podhorszki, editors, Recent Advances in Parallel Virutal Machine and Message Passing Interface, volume 1908 of Lecture Notes In Computer Science, pp. 200–207. Springer, Sept. 2000.

[Buchner, 2003] J. Buchner, D. Christian and M. Scholer, Space Plasma Simulation (Lecture Notes in Physics), Springer, 2003.

[Karonis, 2003] N. Karonis, B. Toonen, and I. Foster, MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, Journal of Parallel and Distributed Computing (JPDC), Vol. 63, No. 5, pp. 551-563, May 2003.

[Kaymaz, 1994a] Z., Kaymaz, George L. Siscoe, Nikolai A. Tsyganenko and Ponald P. Lepping, Magnetotail views at 33RE: IMP 8 magnetobservations, J. Geophys. Res. Vol. 99, No.A5, pp. 8705-8730, 1994.

[Kaymaz, 1994b] Z., Kaymaz, G. L. Siscoe, J. G. Luhmann, R. P. Lepping, C. T. Russell, Interplanetary magnetic field control of magnetotail magnetic field geometry: IMP 8 observations, J. Geophys. Res., Vol.99, No.A6, pp. 11,113-11,126, June 1994.

[Kaymaz, 1994c] Z., Kaymaz, G. L. Siscoe, N. A. Tsyganenko, R. P. Lepping, Magnetotail views at 33RE: IMP 8 magnetometer observations, J. Geophys. Res., Vol.99, No.A5, pp.8705-8730, 1994.

[Kielmann, 1999] T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, and R. A. F. Bhoedjang. MagPIe: MPI's collective communication operations for clustered wide area systems. In ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (Ppopp'99), pp. 131–140. ACM, May 1999.

[Kivelson, 1995] M.G. Kivelson and C.T. Russell, Introduction to Space Physics, Cambridge University Press, 1995.

[Kumar, 94] Kumar, V. and Gupta, A., Analyzing Scalability of Parallel Algorithms and Architectures, Journal of Parallel and Distributed Computing, Vol.22, Issue.3, pp.379-391, 1994.

[Langdon, 1976] Langdon, A.B. and Lasinski, B.F., Electromagnetic and Relativistic Plasma Simulation Model's, In: Methods in Computational Physics, Vol. 16, edited by B. Alder, pp. 327-366, Academic, New York, 1976.

[Lau, 1990] Y. T. Lau and J. M. Finn, Three-Dimensional Kinetic Reconnection in the Presence of Field Nulls and Closed Field Lines, The Astrophysical Journal, Vol. 350, pp. 672-691, 1990.

[LEE, 2001] C. LEE, S. Matsuoka, D. Taila, and J. Saltz, A Grid Programming Primer, Global Grid Forum, August 2001.

[Lembege, 2001] B. Lembege, F. Simonet, Hybrid and particle simulation of an interface expansion and of collisionless shock: A comparative and quantitative study, PHYSICS OF PLASMAS, Vol.8, pp. 3967-3981, 2001.

[Liewer, 1989] P.C. Liewer, and V.K. Decyk, A General Concurrent Algorithm for Plasma Particle-in-Cell Codes, Journal of Comp. Phy., vol. 85, Issue 2, pp.302-322, 1989.

[Lindman, 1975] Lindman, E. L., Free-space boundary conditions for the time dependent wave equation, J. Comp. Phys., Vol.18, No.1, pp.66-78, 1975.

[Loveman, 1993] D. Loveman. High performance fortran. IEEE Parallel & Distributed Technology, Vol..1, Issue.1, pp. 25–43, 1993.

[Maynard, 2001] N.C Maynard, et al, Observation of the magnetospheric "sash" and its implicitions relative to solar-wind/magnetosphere coupling: a multisatellite event analysis, J. Geophys. Res., Vol.106, No.(A4), pp.6097-6122, 2001.

[MPI, 1995] Message Passing Interface Forum. MPI: A Message Passing Interface Standard, June 1995. http://www.mpi-forum.org/.

[Murray, 1982] T. Murray and J. P. David, Topology of Three-Diemnsional Separated Flows, Ann. Rev. Fluid Mech, Vol. 14, pp. 61-85, 1982.

[Nakada, 1999] H. Nakada, M. Sato, and S. Sekiguchi. Design and Implementations of Ninf: towards a Global Computing Infrastructure. Future Generation Computing Systems, Metacomputing Issue, Vol.15(5-6), pp.649–658, 1999.

[Nakada, 2002] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, and H. Casanova. GridRPC: A Remote Procedure Call API for Grid Computing. Technical report, Univ. of Tennesse, June 2002. ICL-UT-02-06.

[Nishikawa, 1992] Nishikawa, K.-I., O. Buneman, and T. Neubert, Solar Wind-Magnetosphere Interaction as Simulated by a 3-D EM Particle Code, IEEE

Transaction on plasma Science, Vol.20, Issue.6, pp.810-816, 1992.

[Nishikawa, 1996] Nishikawa, K.-I., and T. Neubert, Solar wind-magnetosphere interaction as simulated by a 3D EM particle code: A 3D reconnection at the magnetopause, Adv. Space Res., Vol.18, Issue.8, pp. 263-366, 1996.

[Nishikawa, 1996] Nishikawa, K.-I., Particle entry into the magnetosphere with a southward IMF as simulated by a 3D EM particle code, J. Geophys. Res., Vol.102, No.A8, pp. 17631, 1997.

[Nishikawa, 1998] Nishikawa, K.-I., Particle entry through reconnection grooves in the magnetopause with a dawnward IMF as simulated by a 3D EM particle code, Geophys. Res. Lett., 25, P. 1609, 1998.

[*Nicolis and Prigogine*, 1977] Nicolis, G., and I. Prigogine, *Self-organization in nonequilibrium systems*, Wiley New York, (1977).

[Omura and Matsumoto, 1993] Omura, Y., and H. Matsumoto, KEMPO1: Technical guide to one dimensional electromagnetic particle code, in Computer Space Plasma Physics, edited by H. Matsumoto and Y. Omura, pp.21-65, Terra Scientific, Tokyo, 1993.

[OpenMP, 1997] OpenMP Consortium. OpenMP C and C++ Application Program Interface, Version 1.0, 1997.

[Priest, 2000] E., Priest, and T. Forbes, *Magnetic Reconnection: MHD Theory and Applications*, Cambridge University Press, 2000.

[Pritchett, 1996] P. L. Pritchett, and V. K. Decyk, Three-dimensional stability of thin quasi-neutral current sheets, J. Geophys. Res. Vol. 101, pp. 27413, 1996.

[Sato, 2001] M. Sato, M. Hirono, Y. Tanaka, and S. Sekiguchi. OmniRPC: A Grid RPC Facility for Cluster and Global Computing in OpenMP. In WOMPAT, LNCS 2104, pp. 130–136. Springer-Verlag, 2001.

[Savoini, 2001] Savoini Ph. And B. Lembege, Two-dimensional simulations of a curved shock: Self^consistent formation of the electron foreshock, J. Geophys. Res., vol.106(A7), pp.12975-12992, 2001.

[SINET] www.sinet.ad.jp

[Siscoe, 2001] G. L. Siscoe, G. M. Erickson, B. U. O. Sonnerup, et al, Magnetospheric sash dependence on IMF direction, Geophys. Res. Lett., Vol. 28(10), pp.1921-1924, 2001.

[*Strogatz*, 1994] Strogatz, S. H., *Nonlinear dynamics and chaos*, Addison-Wesley Reading, MA, (1994).

[Villasenor, 1992] J. Villasenor, and O. Buneman, Rigorous Charge Conservation for Local Electromagnetic Field Solvers, Computer Physics Communications, vol. 69, Issue 2-3, pp. 306-316, 1992.

[White, 1998] W.W. White, G. L. Siscoe, G. M. Erickson, Z. Kaymaz, N. C. Maynard, K.

D. Siebert, B. U. O. Sonnerup, and D. R. Weimer, The magnetospheric sash and the cross-tail S, Geophys. Res. Let., Vol.25, pp. 1605-1608, 1998.

[*Wiggins*, 2003] Wiggins, S., et al., *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, Springer, (2003).

[Yee, 1966] Yee, K. S, Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media, IEEE Trans. Antennas Propogation, Vol. 14, pp. 302-307, 1966.

Publication List:

Refereed Journal Article:

[1] Weifeng Tao, Dongsheng Cai, Xiaoyang Yan, Ken-ichi Nishikawa and Bertrand Lembege: Scalability Analysis of Parallel Particle-In-Cell Codes on Computational Grid, Computer Physics Communications. (In Press)

[2] 望月　茂徳、堀江　大輔、陶　衛峰、蔡　東生：フラクタル変換を用いたコンピュータグラフィクス、情報処理学会論文誌、Vol.48 No.11、pp.3548-3556, 2007.

Refereed International Conference Article:

[1] Weifeng Tao, Dongsheng Cai : Streaming Techniques for Distributed Visualization of Space Weather Simulation in Computational Grid Environment, Proc. NICOGRAPH International 2004, pp. 68-71, 2004.

[2] Weifeng Tao, Zheng Huang and Dongsheng Cai: Effectively Visualizing Massive Time-Varying Simulation Data on SuperSINET Grid, Proc. NICOGRAPH International 2005, pp. 33-38, 2005.

[3] Weifeng Tao, Dongsheng Cai: Implementation of Parallel Particle-In-Cell Codes on SuperSINET based Grid, Proc. HPC Asia 2005, pp. 331-336, 2005.

Others:

[1] 原島　省、顔 小洋、陶 衛峰：フエリーによる海水計測データと水中画像によるサンゴ礁時系列データのGIS法 (Construction of GISs from the seawater monitoring data by ferries and from the under-water image achive of coral reef)、海洋月刊、海洋 GIS と空間解析、そのサイエンスと未来、Vol. 407, pp. 370-375, May 2004.

[2] Weifeng Tao, Dongsheng Cai: Combine the Jini and JXTA with Grid Service, Symposium on Advanced Computing System and Infrastructure IPSJ Symposium Series, No.8, pp. 175-176, 2003.

[3] Weifeng Tao, Dongsheng Cai: Grid based Space Weather Simulation and

Visualization System, Proc. HPC Asia 2004, pp. 129-130, 2004.
[4] Weifeng, Tao, Dongsheng Cai: Grid Computing of Skeleton PIC code on SuperSINET, pp. 283-284, Proc. ISSS-7, 2005.

Acknowledgement:

I would like to express my sincere and hearty appreciation to my advisior, Professor Dongsheng Cai, for attracting me to the parallel computing and visualization of space science research field. I am also grateful to him for his stimulating supervision and constructive discussion through the present research, except that I still want to thanks his support on my living during these years and advice on attitude for life

My deep appreciation also goes to Professor Betrand Lemebege (CETP/UVSQ/IPSL/France), who also gave me much valuable suggestion and discussion on the simulations, post-process of data, and analysis of the results. and give me the opportunity to study and implement the “virtual satellite” visualization system.

I also would like to thank Professor Ken. Ichi, Nishikawa (CSPAR/NSSTC/U.S.A), who give much good information and advice on space physics and parallel PIC code implementation.

Needless to say, I am grateful to all the member of the CAVELAB laboratory, Professor James B.Cole, Mrs. Obana, Dr. Mochizuki and all the other members.. Thank all their help to me in last six years.

At last, I prefer to thank my parents, my wife, Dr. Qingmin Ji and my son, for their understanding, encouragement, hearty and continual support in both my research and private life.