Development of The Efficient MHD Code Using Scalar-Parallel Machine

Tatsuki Ogino and Maki Nakao
(Solar-Terrestrial Environment Laboratory, Nagoya University)

In recent years, the supercomputer which can be used for large-scale numerical computation is changing from the vector-parallel machine quickly to the scalar-parallel machine. Hitachi and FUJITSU shift to marketing of a cluster type scalar-parallel machine even in Japan, and only NEC is continuing development of the vector-parallel machine. Under such a situation, although development of the program which works by a scalar-parallel machine efficiently serves as urgent necessity, development of the efficient parallel computing program is a pending problem for ten years these days, and is in the situation which cannot be referred to as having still succeeded also in the USA and Europe. Since we found out one effective method of carrying out high-speed computation with the scalar-parallel machine, and we present the main point of the method and the result of the test calculation by the present carried out with the supercomputer containing scalar-parallel machines, such as FUJITSU PRIMEPOWER HPC2500.

1. Introduction

The environment involving the supercomputer which performs computer simulation is changing quickly. It is the shift to a scalar-parallel machine from a vector-parallel machine. In the USA and Europe, it shifted to the scalar-parallel machine from about ten years before, and since it was considered that a vector-parallel machine was a high cost overrun, it disappeared from the market temporarily. However, the Earth Simulator of Japan which is a vector-parallel machine received the honor of the supercomputer of a world maximum high speed in 2003 to 2004, and the tendency of reappraisal of a vector-parallel machine happened. Then CRAY company started the development production of the vector-parallel machine (CRAY X1E) again gaining assistance of the U.S. Government. In Japan, first, Hitachi shifted to the scalar-parallel machine (Hitachi SR8000, SR11000, called a pseudo-vector machine) from the vector-parallel machine, and FUJITSU also followed it (PRIMEPOWER HPC2500). Only NEC was continuing development of a vector-parallel machine in the 2004 fiscal year (NEC SX6, SX7and SX8).

The shift to a scalar-parallel machine from a vector-parallel machine took place rapidly because many people believed that cost performance must be good. Then, did the scalar-parallel machine have good cost performance more really than a vector-parallel machine? Although this was the greatest point in question for the user, before the practical use program which actually needs high-speed calculation of the supercomputer proved, it was actual that the shift to a scalar-parallel machine advanced quickly. Although the absolute efficiency of about 40% had come out with the vector-parallel machine by the program execution of large-scale calculation, in spite of having made efforts in the world for ten years over, it has been said with the scalar-parallel machine that only the absolute efficiency of 3-7% can reach. Isn't the wall of this low absolute efficiency exceeded? As a result of variously examining the method for calculating by the scalar-parallel machine at high speed, we found one effective solution with the possibility. Scalar-parallel type supercomputer Fujitsu PRIMEPOWER HPC2500 was renewed in the Information Technology Center of Nagoya University in 2005, while the test result by the method and present state is shown, prediction and expectation and an initial test calculation result for PRIMEPOWER HPC2500 and other supercomputers are described.

2. MHD Simulation Code.

We have carried out global computer simulation of interaction between the solar wind and the earth's magnetosphere using the three-dimensional MHD (magnetohydrodynamic) simulation code([1]-[7]). We have carried out global computer simulation of interaction between the solar wind and the earth's magnetosphere using the three-dimensional MHD simulation code. On the case in which IMF (interplanetary magnetic field) are south direction and north direction, the example of magnetic field structure of the earth's magnetosphere is shown in Figure 1 as the axis of intrinsic dipole magnetic field is tilted (The northern hemisphere is summer at the inclination angle of 30 degrees,). In addition, it is the case in which IMF rotates in the plane north and south morning and evening, and the example of the three-dimensional visualization ([5],[7], [9]) by VRML (Virtual Reality Modeling Language) in the ejection of the plasmoid to the tail direction is shown in Figure 2, after the IMF changes in south direction from the north direction. In such global model, the outside boundary is put if possible in the distance in order to reduce the boundary effect, and again, the improvement on the numerical method is carried out in order to raise spatial resolution, and it is also indispensable to carry out the efficient utilization of the world largest supercomputer. In the MHD model, MHD equation and Maxwell equation are solved using the Modified Leap-Frog method as an initial and boundary value problem[8],[9].

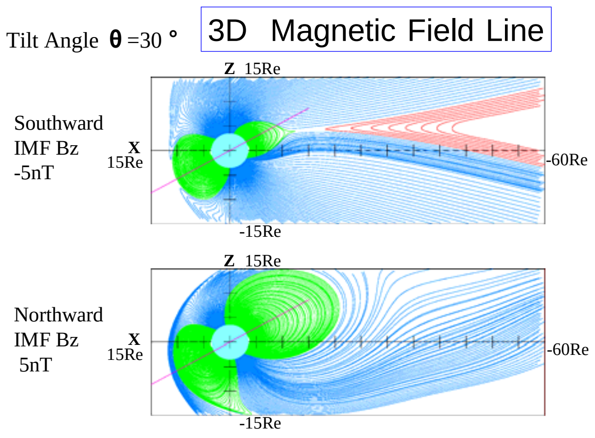


Figure 1. When Interplanetary Magnetic Field (IMF) is for south and for north, the magnetic field structure of the earth's magnetosphere is demonstrated.

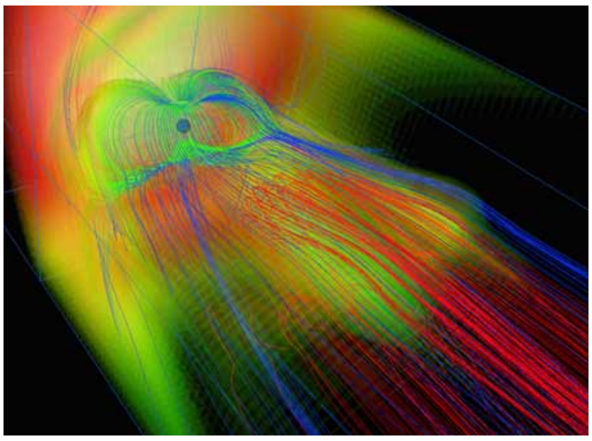


Figure 2. Generating of the tail magnetic reconnection and the global magnetospheric structure in the ejection of the plasmoid to the tail direction after IMF changes in south direction from the north direction.

＜The fundamental equation＞
MHD and Maxwell equations as a base of the MHD model

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\mathbf{v}\rho) + D\nabla^2 \rho \qquad (1)$$

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v}\cdot\nabla)\mathbf{v} - \frac{1}{\rho}\nabla p + \frac{1}{\rho}\mathbf{J}\times\mathbf{B} + \mathbf{g} + \frac{1}{\rho}\Phi \qquad (2)$$

$$\frac{\partial p}{\partial t} = -(\mathbf{v}\cdot\nabla)p - \gamma\, p\nabla\cdot\mathbf{v} + D_p\nabla^2 p \qquad (3)$$

$$\frac{\partial \mathrm{B}}{\partial t} = \nabla\times(\mathbf{v}\times\mathbf{B}) + \eta\,\nabla^2\mathbf{B} \qquad (4)$$

$$\mathbf{J} = \nabla\times(\mathbf{B} - \mathbf{B}_d) \qquad (5)$$

The structure of the parallelization Fortran MHD code using MPI (Massage Passing Interface) ([10]-[16]) which makes possible parallel computing required for a large scale simulation is shown in Figure 3 ([8], [9]). The Modified Leap-Frog method becomes the 2 step calculation as well as the Two step Lax-Wendroff method. Features of parallel computing in the MHD code using MPI are to arrange the necessary communication by parallel computing of the distributed memory in just before of the calculation of each time step. The method for effectively utilizing this consolidation communication is a very important point for making the efficient parallel computing program.

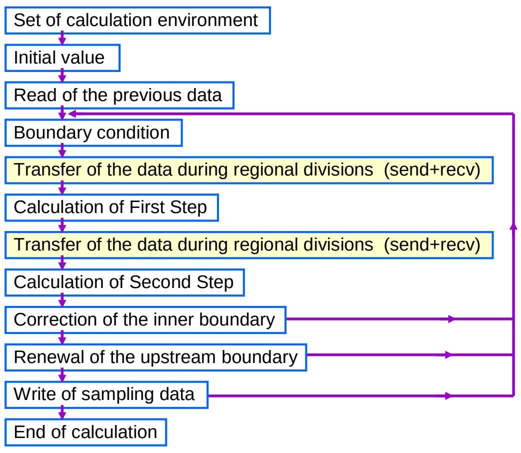

Set of calculation environment
Initial value
Read of the previous data
Boundary condition
Transfer of the data during regional divisions (send+recv)
Calculation of First Step
Transfer of the data during regional divisions (send+recv)
Calculation of Second Step
Correction of the inner boundary
Renewal of the upstream boundary
Write of sampling data
End of calculation

Figure 3. The fundamental structure of the three dimensional MHD code using MPI.

3. Parallel Computation by the Domain Decomposition Method

In parallel computation using the parallel computer of the distributed memory type, it is usual to use domain decomposition method for the three-dimensional arrangement ([8], [9]). In case of the three-dimensional model, it is possible to choose the dimension of the domain decomposition at one dimension, two dimensions and three dimensions. The computation time and the communication time in the cases can be generally estimated as following.

	Computation time	Communication time
One-dimensional domain decomposition	$Ts=k_1 N^3/P$	$Tc=k_2 N^2 (P-1)$
Two-dimensional domain decomposition	$Ts=k_1 N^3/P$	$Tc=2k_2 N^2 (P^{1/2}-1)$
Three-dimensional domain decomposition	$Ts=k_1 N^3/P$	$Tc=3k_2 N^2 (P^{1/3}-1)$

Here, k_1 and k_2 are fixed constant coefficients, and N is variable quantity of the 1 direction in the three-dimensional arrangement, and P is a number of parallel CPUs. The sum total of computation time and communication time is needed for parallel computing. Parallel computation efficiency by the domain decomposition is shown in Figure 4. The computation time, Ts is in inverse proportion for number of CPU, P, and it shortens. Communication time, Tc is lengthened with the increase in P. However, the aspect in which the communication time is lengthened is greatly different by 1, 2, 3-dimensional domain decompositions. That is to say, the following can be understood: That the three-dimensional domain decomposition can shorten the communication time most and that the difference is large even in one dimension and two-dimensional domain decomposition. However, in this comparison, it is assumed that coefficient k_2 which decides the communication time is same, and the condition can be also realized by the contrivance of the program of the communication part. In this way, it can be anticipated that the three-dimensional domain decomposition will be the most efficient in the scalar-parallel machine, and the two-dimensional domain decomposition will be the most efficient in the vector-parallel machine with large number of CPUs.

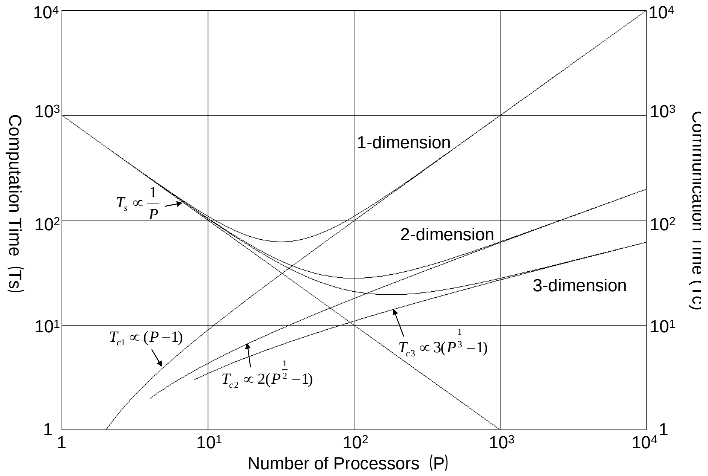


Figure 4. Parallel computing efficiency by 1, 2, 3-dimensional domain dividing method. Parallel computing time is the sum of computation time (Ts) and communication time (Tc).

Next the gist of the concrete introduction method of the three-dimensional domain decomposition will be shown in the three-dimensional global MHD code. In the case of the example of the following which performs domain decomposition in three-dimensional space (x, y, z) using MPI ([10], [14], [15]), it becomes respectively the x, y, z direction in bisection (npex=2, npey=2, npez=2) in the whole with the npe=npex*npey*npez=8 decomposition, and 8 CPU's are used. Arrangement variable which enters each CPU is divided, the arrangement of f (nb,0:nxx+1,0:nyy+1,0:nzz+1) is assigned, and nb=8 is a number of the component of MHD equation. That is to say, data of 1 plane in both sides is applied on the arrangement variable in the three-dimensional direction which exists for CPU of the neighbour originally divided into regions each. itable(-1 : npex,-1 : npey,-1 : npez) is reference table for the data transfer between CPU's with wild card. Moreover, ftemp1x, ftemp2x, etc. are the buffer arrangement of the two-dimensional boundary surface prepared for consolidation data transfer. Like this, the data transfer between the distributed memories which occur for three-dimensional domain decomposition can be simply and efficiently executed by preparing reference table and buffer arrangement.

<<The method of the three-dimensional domain decomposition.>>

```
CC MPI START
      parameter (npex=2,npey=2,npez=2)
      parameter (npe=npex*npey*npez,npexy=npex*npey)
      integer itable(-1:npex,-1:npey,-1:npez)
c
      parameter(nzz=(nz2-1)/npez+1)
      parameter(nyy=(ny2-1)/npey+1)
      parameter(nxx=(nx2-1)/npex+1)
      parameter(nxx3=nxx+2,nyy3=nyy+2,nzz3=nzz+2)
c
      dimension f(nb,0:nxx+1,0:nyy+1,0:nzz+1)
      dimension ftemp1x(nb,nyy3,nzz3),ftemp2x(nb,nyy3,nzz3)
      dimension ftemp1y(nb,nxx3,nzz3),ftemp2y(nb,nxx3,nzz3)
      dimension ftemp1z(nb,nxx3,nyy3),ftemp2z(nb,nxx3,nyy3)
c
```

The program of the data transfer between distributed memories is very much simplified, as it will be shown next. To begin with, by using wild card in conversion table, itable, there is no peculiarity in the edge of the x, y, z direction in the program. There is no necessity of using conditional sentences such as if sentence. Next, by preserving the data of yz plane in the left end of the x direction of f (nb,0:nxx+1,0:nyy+1,0:nzz+1) at buffer arrangement ftemp1x at ftemp1x=f (:,is,:,:) temporarily, it is lumped together and is transferred from ftemp1x of CPU of ileftx to ftemp2x of CPU of irightx at mpi_sendrecv. By moving in the data of yz plane added to right end of the x direction of f (nb,0:nxx+1,0:nyy+1,0:nzz+1) using f (:,ie+1,:,:)=ftemp2x in continueing, the data transfer is completed. By doing like this, it is possible that the data of both sides in each direction divided three-dimensional and into regions is easily transferred the consolidation between CPU's with the distributed memory. In this method, conditional sentences such as if and do loop are completely unnecessary, as it is clear by the program. From the fact, it can be guessed that the overhead by the three-dimensional domain decomposition which becomes a problem in the comparison with one-dimensional domain decomposition will not occur almost. As a result, it is possible to equalize proportion coefficient of communication time, Tc in 1, 2 and 3-dimensional domain decomposition almost, as it is shown in Figure 4.

```
<<The data transfer between divided CPU's>>
CC MPI START
       irightx = itable(irankx+1,iranky,irankz)
       ileftx  = itable(irankx-1,iranky,irankz)
       irighty = itable(irankx,iranky+1,irankz)
       ilefty  = itable(irankx,iranky-1,irankz)
       irightz = itable(irankx,iranky,irankz+1)
       ileftz  = itable(irankx,iranky,irankz-1)
c
       ftemp1x=f(:,is,:,:)
       ftemp1y=f(:,:,js,:)
       ftemp1z=f(:,:,:,ks)
       call mpi_sendrecv(ftemp1x,nwyz,mpi_real,ileftx,200,
     &                   ftemp2x,nwyz,mpi_real,irightx,200,
     &                   mpi_comm_world,istatus,ier)
       call mpi_sendrecv(ftemp1y,nwzx,mpi_real,ilefty,210,
     &                   ftemp2y,nwzx,mpi_real,irighty,210,
     &                   mpi_comm_world,istatus,ier)
       call mpi_sendrecv(ftemp1z,nwxy,mpi_real,ileftz,220,
     &                   ftemp2z,nwxy,mpi_real,irightz,220,
     &                   mpi_comm_world,istatus,ier)
c
       f(:,ie+1,:,:)=ftemp2x
       f(:,:,je+1,:)=ftemp2y
       f(:,:,:,ke+1)=ftemp2z
CC MPI END
```

The actual three-dimensional MHD code of 1, 2 and 3-dimensional domain decomposition can be seen by the homepage shown later ([9], [17]).

4. Comparison of Computation Efficiency of MHD Code Using Domain Decomposition Method

The shell when compiling and carrying out the three-dimensional MHD code written using MPI by scalar-parallel-processing supercomputer FUJITSU PRIMEPOWER HPC2500 of the Information Technology Center is indicated to be (A) to (B) ([10], [11], [13]). (A) is use of only process parallel of 128 CPU, and (B) is a case where 32 process parallel and 4 thread parallel (automatic parallel) ([12], [13]) are used together using 128 CPU. Moreover, although the option of compile shows the thing when the greatest calculation efficiency is acquired, especially now, it is not necessary to specify these options. Although it was not used this time, the comment that it is also effective in improvement in the speed by thread parallel adding a compile option (it turning off by a default) called -Khardbarrier which enables the barrier function between threads is gotten from FUJITSU.

```
(A) Compile and execution of MPI Fortran program
    use 128 processors
    128 process parallel

mpifrt -Lt progmpi.f -o progmpi -Kfast_GP2=3,V9,largepage=2 -Z mpilist
qsub mpiex_0128th01.sh
```

```
hpc% more mpiex_0128th01.sh
#  @$-q p128 -lP 128 -eo -o progmpi128.out
#  @$-lM 8.0gb -lT 600:00:00
setenv  VPP_MBX_SIZE  1128000000
cd ./mearthd4/
mpiexec -n 128 -mode limited ./progmpi128

(B) Compile and execution of MPI Fortran program
    use 128 processors
    32 process parallel
     4 thread parallel (sheared memory)

mpifrt -Lt progmpi.f -o progmpi -Kfast_GP2=3,V9,largepage=2 -Kparallel -Z mpilist
qsub mpiex_0128th04.sh

mpiex_0128th04.sh
#  @$-q p128 -lp 4 -lP 32 -eo -o progmpi01.out
#  @$-lM 8.0gb -lT 600:00:00
setenv  VPP_MBX_SIZE  1128000000
cd ./mearthd4/
mpiexec -n 32 -mode limited ./progmpi
```

First, using FUJITSU PRIMEPOWER HPS2500, the result of having compared the calculation efficiency of the three-dimensional MHD code (the one-dimensional domain decomposition technique) the case of only process parallel and at the time of using process parallel and thread parallel together is shown in Table 1. The table showed the computation time which 1 time of a time step advance takes, calculation speed (GFLOPS), and the calculation speed (GFLOPS/CPU) per 1 CPU to the total number of CPUs, process parallel number, and thread parallel number which were used. It is necessary to set a process parallel number over to two, because it is a FORTRAN program of MPI. Moreover, a thread parallel number will also be automatically chosen or less as 64 from the restrictions. In the case of 128 CPU, the case of only process parallel has the best calculation efficiency, but also in the case of combined use of process parallel and thread parallel, so much equal calculation efficiency has come out. However, if a thread parallel number becomes large like 32 and 64, the tendency for calculation efficiency to deteriorate will be seen.

Table 1. Calculation efficiency of the MHD code using thread parallel by PRIMEPOWER HPC2500

Toal number of CPUs	Process parallel number	Thread parallel number	Computation time (sec)	Calculation speed (GFLOPS)	Calculation speed/CPU (GFLOPS/CPU)
1-dimensional decomposition by f(nx2,ny2,nz2,nb)=f(522,262,262)					
4	4	-	12.572	5.98	1.494
8	8	-	6.931	10.84	1.355
16	16	-	3.283	22.88	1.430
32	32	-	1.977	38.00	1.187
64	64	-	1.108	67.81	1.060
128	128	-	0.626	120.17	0.939
128	64	2	0.692	108.77	0.850
128	32	4	0.697	107.80	0.842
128	16	8	0.637	118.07	0.922
128	8	16	0.662	113.57	0.887
128	4	32	0.752	100.07	0.782
128	2	64	0.978	76.95	0.601
256	128	2	0.496	151.45	0.592
256	64	4	0.439	171.23	0.669
256	32	8	0.429	174.94	0.683
256	16	16	0.460	163.43	0.638
256	8	32	0.577	130.45	0.510
512	128	4	0.424	177.63	0.347
512	64	8	1.452	51.80	0.101
512	32	16	0.297	253.64	0.495
512	16	32	0.316	238.37	0.466
3-dimensional decomposition by f((nb,nx2,ny2,nz2)=f(8,522,262,262)					
512	512	1	0.0747	1007.78	1.968
512	256	2	0.0947	794.75	1.552
512	128	4	0.486	154.84	0.302
512	64	8	0.628	119.77	0.234
3-dimensional decomposition by f((nb,nx2,ny2,nz2)=f(8,1024,1024,1024)					
512	512	1	2.487	916.94	1.791
1024	512	2	1.448	1575.09	1.538
3-dimensional decomposition by f((nb,nx2,ny2,nz2)=f(8,2046,2046,2046)					
512	512	1	19.694	929.13	1.815
1024	512	2	10.763	1700.12	1.660
1024	256	4	15.648	1169.36	1.142
1536	512	3	7.947	2302.60	1.499
1536	256	6	16.462	1111.54	0.724

When there is restriction which a big shared memory is required or cannot take large process parallel, generally you have to use thread parallel ([12], [13]). For example, in the case of the three-dimensional MHD code of one-dimensional domain decomposition of the direction of z of Table 1, it is necessary to take a process parallel number from restriction of an external boundary condition setup below in half of the array variable (nz2=nz+2=262) of the direction of z. Therefore, when the number of CPU becomes over 256, thread parallel will be used inevitably. Thus, in three-dimensional MHD code, when process parallel and thread parallel need to be used together, if a thread parallel number is taken about to 4 to 16, it proves that efficient calculation can be performed. Of course, although it seems that whether high efficiency will be acquired if what number of threads is used depends to a program strongly, what is necessary is likely to be just to start with first of all trying the number of threads which are not out of less than 16. Although Table 1 shows only the data which the maximum parallel computing speed was obtained, when increasing the number of threads, the variation in calculation speed appears considerably. Although this is considered to be also balance with data communications, in actual calculation, it is also likely to happen that calculation speed becomes slow to a slight degree. Moreover, there is no data in the case of 64 threads at 256CPU and 512CPU. This is because it was not able to perform by a work domain being insufficient.

In scalar-parallel machine PRIMEPOWER HPC2500, because CPU in a node can be treated as a shared memory, it can carry out automatic parallelization of it using functions, such as thread parallel. When the three-dimensional MHD code of one-dimensional domain decomposition was used, the number of CPU was fixed to 128 from Table 1, and the efficiency of the automatic parallelization by thread parallel was shown in the graph of Fig. 5. In this case, between the number of CPU, a process parallel number, and a thread parallel number, the relation of "(process parallel number) x (thread parallel number) = (the number of CPU)" is required, and a thread parallel number is restricted to the number of CPU in a node. Although Figure 5 shows the calculation speed (Gflops) at the time of increasing a thread parallel number, and the calculation speed (Gflops/CPU) per 1CPU, because the number of CPU is fixed to 128, a difference does not appear in both graph. When thread parallel numbers are 8 and 16, calculation efficiency is increasing, but it takes for increasing a thread parallel number further like 32 and 64, and calculation efficiency deteriorates. Change of the calculation speed (Gflops) at the time of increasing the number of CPU by PRIMEPOWER HPC2500 and the calculation speed (Gflops/CPU) per 1CPU is shown in Figure 6. Calculation speed here shows a result when the maximum calculation speeds also including thread parallel use are obtained to the fixed number of CPU. That the maximum calculation speed was obtained was a case where 512CPU did not use thread parallel, but the three-dimensional MHD code of three-dimensional domain decomposition was used when it is what is called flat MPI use of only process parallel. When the number of CPUs is 1024, a thread parallel number is 2, the number of CPUs is 1536, a thread parallel number is a case of 3, and a process parallel number was set to a maximum of 512, it was the fastest. When the three-dimensional MHD code of three-dimensional domain decomposition is used, it proves that scalability is extended quite well to 1536CPU.

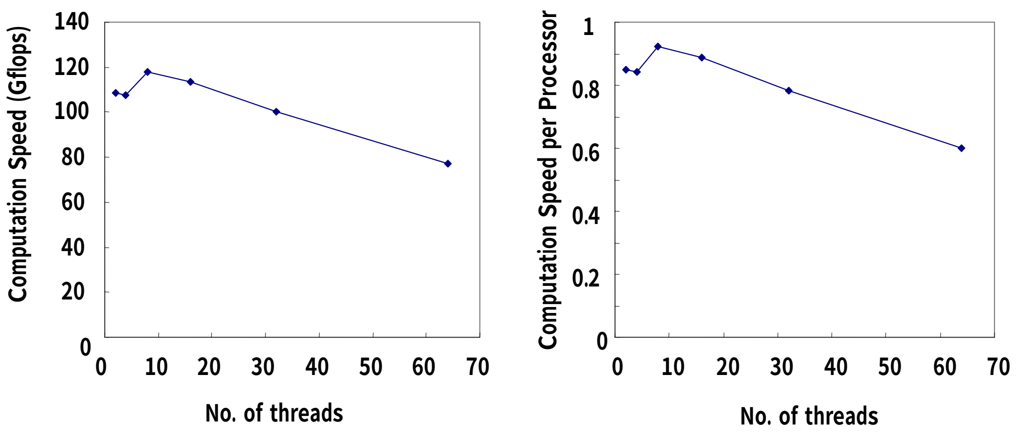


Figure 5. Calculation speed (Gflops) and calculation speed per 1CPU (Gflops/CPU) at the time of fixing the number of CPUs to 128 by PRIMEPOWER HPC2500, and increasing a thread parallel number.

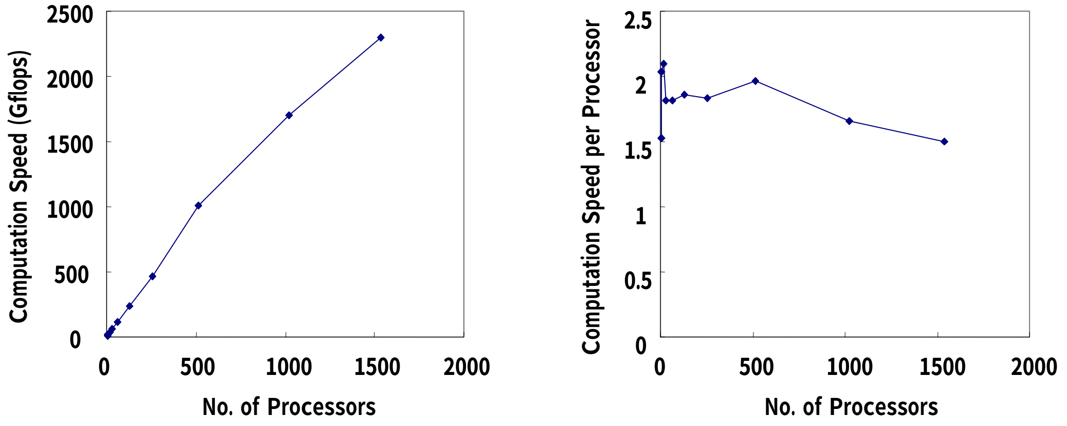


Figure 6. Calculation speed (Gflops) and calculation speed per 1CPU (Gflops/CPU) at the time of increasing the number of CPUs by PRIMEPOWER HPC2500.

Interesting data was obtained in the case that the size of an array is f (8,522,262,262) in three-dimensional MHD code of three-dimensional domain decomposition, when it fixed to 512CPU and process parallel and a thread parallel number were changed. Only in 512 process parallel, the maximum calculation speed obtained 1007GFLOPS. On the other hand, when the thread parallel number was increased, calculation speed fell for a while by 2 thread parallel, and calculation speed fell still more greatly by 4 and 8 thread parallel. Especially the difference of the calculation speed in the case of being parallel is as important as 64 process parallel 8 threads at 512 process parallel and 512CPU use. It was used the same four division in the latter also as x, y, and the direction of z. Although the latter was also expected that calculation speed almost comparable as the former comes out, it was different in practice. It is thought that this result means that the method which is going to make the amount of communications the minimum using the three-dimensional domain dividing method, and the method of the automatic parallelization in 8 thread parallel do not have consistency well. That is, what ordering of the array variable f(nb,nx2,ny2,nz2) in 8 thread parallel is made is likely to pose a problem. Moreover, in a big array, the tendency which f(8, 1024, 1024, 1024) and f (8, 2046, 2046, 2046) also resembled is seen. These are considered also as it has suggested that automatic parallel optimization and optimization of user parallel are necessarily incompatible. When 4 thread parallel and 6 thread parallel are especially specified by 1024CPU and 1536CPU use to the array of f (8, 2046, 2046, 2046), respectively, parallel computing efficiency does not increase but is falling rather. The comment that the compile option of -Khardbarrier which enables the barrier function between threads to calculation efficiency degradation at the time of increasing this thread parallel number will probably be effective is gotten from FUJITSU. However, the check has not been carried out yet.

The comparison of computation efficiency of the three-dimensional MHD code in the case which used 1, 2 and 3-dimensional domain decomposition method is shown in Table 2. The numerical value showed calculation speed (GF/PE) for CPU per 1. VPP5000, NEC SX6, NEC ES (Earth Simulator) are vector-parallel machine. PRIMEPOWER HPC2500 and Hitachi SR8000 are the clustery scalar-parallel machine. In the three-dimensional domain decomposition method, there are 2 kinds of codes. Order of the arrangement is x, y, z direction at first and vector component (8 pieces) at late on f (nx2,ny2,nz2,nb).   f (nb,nx2,ny2,nz2) moves the vector component (8 pieces) to the head of the arrangement in order to memorize the variable which is related to the calculation near. By the exchange of this sequence, in the scalar-parallel machine, rate of hit of the cache rises, and the improvement on the computation efficiency is expected, and in PRIMEPOWER HPC2500, the effect has remarkably appeared. In the result of vector-parallel machine NEC ES using 512 CPU, the two-dimensional domain decomposition which utilized the vectorization in the x direction is the most efficient.

Table 2. Comparison of the calculation efficiency in the three-dimensional MHD code

Computer Processing Capability by 3D MHD Code for (nx,ny,nz)=(510,254,254)

	CPU Number	VPP5000 (GF/PE)	PRIMEPOWER HPC2500 (1.3GHz)	PRIMEPOWER HPC2500 (2.08GHz)	NEC SX6 (GF/PE)	NEC ES (GF/PE)	Hitachi SR8000 (GF/PE)	Hitachi SR11000/j1 (GF/PE)
1D Domain	2	7.08	-----	-----	6.36	6.66		
Decomposition by	4	7.02	0.031	0.039(1.494)	5.83	6.60		0.182
f(nx2,ny2,nz2,nb)	8	6.45	0.030	0.037(1.355)	5.51	6.50	0.016	
	16	6.18	0.028	0.046(1.430)	5.44	6.49		
	32	(7.49)		0.042(1.187)		6.39		
	64	(6.90)		0.040(1.060)		6.37		
	128			0.039(0.907)		(2.11)		
	256			0.016(0.683, 2 thread)				
	512			0.003(0.347, 4 thread)				
2D Domain	4	7.51	0.199	1.529	6.34	6.63		0.775
Decomposition by	8	6.88	0.191	1.451	6.28	6.47		
f(nx2,ny2,nz2,nb)	16	6.49	0.200	1.575	6.23	6.45		
	32			1.395		6.47		
	64			1.421		6.32		
	128			1.409		6.27		
	256			1.396		6.05		
	512			0.868		5.62		
f(2048,1024,1024,8)	1024	(MPI)				7.18(7.36 TF)		
f(1024,1024,1024,8)	512	(HPF/JA)				6.47(3.31 TF)		
3D Domain	8	7.14	0.207	1.558	6.24	6.38	0.253	0.869
Decomposition by	16	6.77	0.202	1.593	6.34	6.33		
f(nx2,ny2,nz2,nb)	32			1.527		6.25		
	64			1.534		5.61		
	128			1.518		5.57		
	256			1.513		5.38		
	512			0.923		3.94		
3D Domain	8	2.91	1.438	2.038	1.13	4.11	0.268	2.221
Decomposition by	16	2.63	1.416	2.099	4.53	4.06		
f(nb,nx2,ny2,nz2)	32			1.820		4.11		
	64			1.813		4.17		
	128			1.857		4.12		
	256			1.831		4.12		
	512			1.968		3.70		
f(8,1024,1024,1024)	512	(MPI)		1.791(0.917 TF)				
f(8,1024,1024,1024)	1024	(MPI, 2 thread)		1.538(1.575 TF)				
f(8,2048,2048,2048)	512	(MPI)		1.815(0.929 TF)				
f(8,2048,2048,2048)	1024	(MPI, 2 thread)		1.660(1.700 TF)				
f(8,2048,2048,2048)	1024	(MPI, 4 thread)		1.142(1.169 TF)				
f(8,2048,2048,2048)	1536	(MPI, 3 thread)		1.499(2.303 TF)				
f(8,2048,2048,2048)	1536	(MPI, 6 thread)		0.724(1.112 TF)				

In order for a scalar-parallel machine to realize high efficiency, it has been proposed that it is important to perform to raise rate of hit of the cache and three-dimensional domain decomposition, and to reduce the amount of communications. The importance of that can be clearly known by the result which is 512 CPU of RIMEPOWER HPC2500 in Table 2. The calculation result of the 2 or 3-dimensional domain decomposition by PRIMEPOWER HPC2500 uses only process parallel altogether to 512CPU. In the one-dimensional domain decomposition, the very bad result has come out by HPC2500. This is because grid size (nx,ny,nz)=(510,254,254) which includes a boundary in three-dimensional MHD code was taken to the power of 2 (512=510+2,256=254+2). However, if grid size was changed into (nx,ny,nz)=(520,260,260), efficiency will have been greatly improved like the numerical value shown in a parenthesis. Namely, when the size of the array of the domain decomposition direction is taken to the power of 2 and the number of CPU is also taken to the power of 2, it is thought that the phenomenon in which competition of memory access arose and calculation speed deteriorated extremely occurred ([12], [13]). Therefore, even when high efficiency is not acquired first, if making even a small change of changing some numbers of CPU, and sizes of an array, it proved that it is calculable at high speed without a big problem for almost all the conventional MHD code .

When the number of CPUs is increased, using the three-dimensional MHD code(3Da) of the one(1D), two(2D) and three-dimensional domain decomposition using the conventional array f(nx2, ny2, nz2, nb) and the three-dimensional MHD code (3Db) of the three-dimensional domain decomposition which changed an order of the array and f (nb, nx2, ny2, nz2), scalar-parallel machine PRIMEPOWER HPC2500 is shown in Figure 7, and the vector-parallel machine ES (Earth Simulator) is shown in Figure 8 for how the tendency of calculation speed change is seen. However, the size of an array uses (nx, ny, nz) = (510,254,254) except for one-dimensional domain decomposition, and ss Table 2 explained only in one-dimensional domain decomposition, in order to avoid extreme degradation from the size of an array including a boundary becoming a the power of 2, (nx, ny, nz) = (520,260,260) was used. Because the difference in the size of the array is converted, it can measure directly the calculation speed (Gflops) between four kinds of MHD codes, and the calculation speed (Gflops/CPU) per 1CPU even in Figure 7. First, in the case of scalar-parallel machine PRIMEPOWER HPC2500 of Figure 7, since calculation speed is quick from 1D in 2D, the effect which reduced the amount of communications can be seen from having changed to two-dimensional domain decomposition from one dimension. However, change of remarkable calculation speed is not seen between 2D and 3Da. This is considered that the hit ratio of cash is because it is not good similarly. Here, in order to improve the hit ratio of cash, in the three-dimensional domain decomposition (3Db) which changed an order of the array, calculation speed is greatly improved to 512 CPU, and scalability is seen be very good. Thus, in the scalar-parallel machine, it was checked to 512 CPU that very good parallel computing efficiency can be acquired by devising an order of an array in order to improve the hit ratio of cash, and using the three-dimensional domain dividing method.

On the other hand, in the vector-parallel machine Earth Simulator of Figure 8, two-dimensional domain decomposition (2D) keeps the merit of scalability the best to 512CPUs as expected. Because this uses do loop inside the maximum for vectorization, since vectorization efficiency becomes good in the one where the vector length is longer, and since the amount of communications was simultaneously reduced using two-dimensional domain decomposition, it is thought that the highest calculation speed was obtained. This is seen more notably than the case where enlarged the array and the number of CPUs is increased with 1024 as shown in Table 2. Moreover, the difference in the vectorization efficiency which vector length depends is a difference of 3Da and 3Db, that is, 3Da has brought a result with quick calculation speed rather than 3Db. If the number of CPUs becomes large 512, although the calculation speed of 3Da becomes slow and the calculation speed of 3Db is approached, it thinks because the do loop length inside the maximum became short with 512/8=64. In Earth Simulator, the difference of 3Da and 3Db is small rather. Furthermore, it seems that it should be surprised that the difference of 2D and 3Db is not large, either. Although it expected that a difference generally came out more, since the data transfer between CPUs is very excellent in Earth Simulator at high speed, three-dimensional domain decomposition (3Da and 3D b) can also be understood that parallel computing speed seldom falls. Moreover, although calculation speed has deteriorated in one-dimensional domain decomposition of 256CPU use of Figure 8, since the variable of the domain decomposition direction is set to nz+2=256 including a boundary, and the number of parallelization nodes has the necessity which should be set below to the half, it is thought that it is because automatic parallel was used only in that case. The further investigation is required for this point. It was checked that very good vector-parallel computing efficiency can be acquired from these results by using the two-dimensional domain dividing method in a vector-parallel machine.

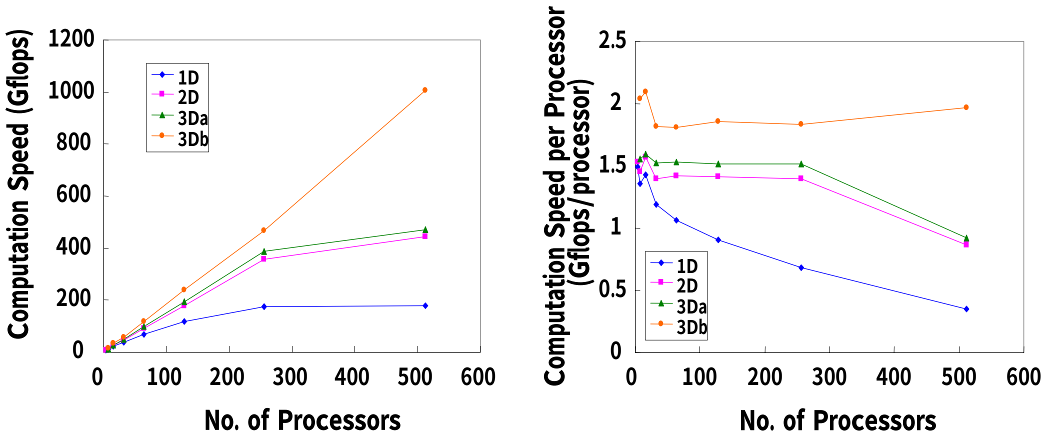
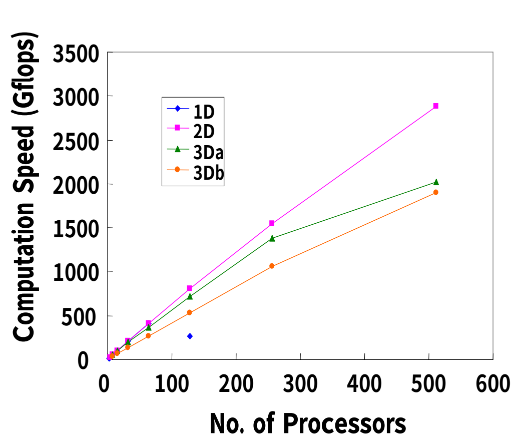


Figure 7. Calculation speed (Gflops) to four kinds of MHD codes (1D : One-dimensional domain decomposition, 2D : Two-dimensional domain decomposition, 3Da : Three-dimensional domain decomposition and f(nx2,ny2,nz2,nb), 3Db:Three-dimensional domain decomposition and f(nb,nx2,ny2,nz2)) at the time of increasing the number of CPUs by scalar-parallel machine PRIMEPOWER HPC2500, and calculation speed per 1CPU (Gflops/CPU)



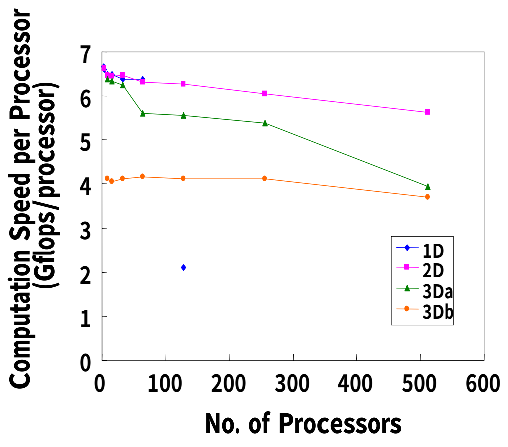


Figure 8. Calculation speed (Gflops) to four kinds of MHD codes (1D : One-dimensional domain decomposition, 2D : Two-dimensional domain decomposition, 3Da : Three-dimensional domain decomposition and f(nx2,ny2,nz2,nb), 3Db:Three-dimensional domain decomposition and f(nb,nx2,ny2,nz2)) at the time of increasing the number of CPUs by Vector-parallel machine (Earth Simulator), and calculation speed per 1CPU (Gflops/CPU)

As a left-behind problem, the period until 128 parallel is possible in a program in one-dimensional domain decomposition, when using 256CPU, the combination of 128 process parallel and 2 thread parallel needed to be used, when using 512CPU, the combination of 128 process parallel and 4 thread parallel needed to be used. When the hard transfer function of data is turned OFF (-ldt=0) in these cases also in the case of the grid of (nx, ny, nz) = (520,260,260), calculation speed falls considerably. Moreover, the result which falls extremely was obtained in 512CPU. However, if the data transfer unit was turned ON (- ldt=1), the fall tendency of the calculation efficiency will have improved greatly. Moreover, because it is restricted to until a maximum of 512 process parallel in PRIMEPOWER HPC2500 of Nagoya University, in 1024CPU use, the combination of 512 process parallel and 2 thread parallel is used, and in 1536CPU use, the combination of 512 process parallel and 3 thread parallel will be used. However, in the combination of the process parallel and thread parallel which uses these many CPU, in the test of hard transfer function OFF (- ldt=0) of data, the efficiency of parallelization did not become good but was getting very worse. However, if the function was turned ON (- ldt=1), calculation efficiency will have improved remarkably so that the result of f (8, 1024, 1024, 1024) and f (8, 2046, 2046, 2046) at the time of using the three-dimensional domain dividing method of Tables 1 and 2 may show. Thus, when using the hard-data transfer function and using the three-dimensional domain dividing method, it has checked that scalability was extended very well to 1536CPU use. We notices one more point, when using many CPU, the method of reducing a process parallel number and increasing a thread parallel number is causing degradation of calculation efficiency. This tendency has appeared notably in the result of f (8,522,262,262) , f (8, 1024, 1024, 1024), and f (8, 2046, 2046, 2046) at the time of using the three-dimensional domain dividing method of Table 1. The result by these present is considered also as the method (for example, flat MPI) of using only user parallel compared with the method of combining the respectively optimal user parallel and automatic parallel is pointing to the view that efficient parallel computing is realizable. The numerical value in the parenthesis of VPP5000 shows for comparison what calculation speed of the peak price had come out by 32-64PE until now.

Actually, in three-dimensional MHD code which was usually being moved by 16CPU of VPP5000, the speed of 110-120 GFLOPS had come out. If the same code is moved by PRIMEPOWER HPC2500, 80-90GFLOPS comes out by 64CPU, 90-140GFLOPS has come out by 128 CPU, and conventional calculation can be continued as it is. Of course, in three-dimensional MHD code which had realized high efficiency by VPP5000, if a CPU number increases further, the tendency for calculation speed to be saturated will be seen. If process parallel and thread parallel are used together and change of calculation speed is investigated, when using thread parallel together, 6-7% calculation speed may improve. It mentioned above when a data transfer unit is not used, the tendency restricted when it does not have so much CPU that calculation efficiency improves by thread parallel combined use (less than 128-256 CPU) is accepted. However, if the hard transfer function of data is turned ON, the maximum of the CPU number will go up considerably. In this way, combined use of process parallel and thread parallel can also expect achievement of quite high parallel computing efficiency.

5. Outline of Renewal Supercomputer and Anticipation and Survey to High-Speed Calculation

The outline of the supercomputer of the Information Technology Center of Nagoya University is shown below. It was updated in March, 2005 from conventional vector-parallel machine Fujitsu VPP5000/64 to scalar-parallel machine Fujitsu PRIMEPOWER HPC2500 ([10], [11]).

＜The supercomputer＞

Fujitsu PRIMEPOWER HPC2500 23 node
(64CPU／memory 512GB × 22 node、128CPU／memory512GB × 1 node)

The total performance	**：12.48Tflops**
All memory capacities	**：11.5TB**
The disk capacity	**：50TB**

By the renewal, how to become high-performance, is shown at catalog performance ratio of VPP5000/64 and PRIMEPOWER HPC2500 about supercomputing 4 function of calculation speed, the main memory, magnetic-disk capacity, and network speed in Figure 9. By updating to scalar-parallel machine PRIMEPOWER HPC2500 from vector-parallel machine VPP5000/64, theoretical performance becomes good sharply. However, because there is relative inefficiency of a scalar-parallel machine, it is dangerous to expect the improvement in performance easily. It was expected as follows whether high-speed calculation would be how far expectable by PRIMEPOWER HPC2500 after updating based on the conventional test. Then, test calculation was carried out, and the survey result was also added to Table 3.

If the anxiety at the time of updating is written collectively ([8], [9]), especially degradation in MHD code of the one-dimensional domain decomposition which had realized high efficiency with the vector-parallel machine may be miserable, and calculation efficiency may fall to 1/10 of VPP5000/64. If it becomes such a situation, the conventional three-dimensional MHD code will not become useful at all. Two and three-dimensional domain decomposition are likely to be adopted well, and comparable calculation efficiency expects at last. When the MHD code of the three-dimensional domain decomposition which rearranged the array variable is used, 3.4TFLOPS may be able to expect by the most optimistic anticipation. However, it is important to actually perform whether absolute efficiency becomes how much good in three-dimensional MHD code.

As mentioned above, updating to scalar-parallel machine PRIMEPOWER HPC2500 had much source of anxiety. The results which actually carried out the operation start and which were energetically tested from March, 2005 are Tables 1 and 2. It is Table 3 which was summarized as survey. It became clear that the three-dimensional MHD code which demonstrated high efficiency with the vector-parallel machine conventionally almost also has no change, and appropriate improvement in the speed is realizable with these results. Actually, even as for the conventional three-dimensional MHD code using the one-dimensional domain dividing method, calculation speed of 116GF is realized by 128CPU and calculation speed of 175GF is realized by 256CPU. In this way, the large-scale three-dimensional MHD simulation of space plasma was smoothly maintainable satisfactorily also about the shift to scalar-parallel machine PRIMEPOWER HPC2500 from vector-parallel machine VPP5000 of the Information Technology Center. Moreover, in the case of the three-dimensional domain dividing method using the grid of (nx2, ny2, nz2) = (2048, 2048, 2048), the following maximum high-speed values were able to be acquired: 929GF by 512CPU, 1700GF by 1024CPU, 2303GF by 1536CPU. If the compile option which enables the above-mentioned barrier function between threads is used, the further improvement in the speed expects.

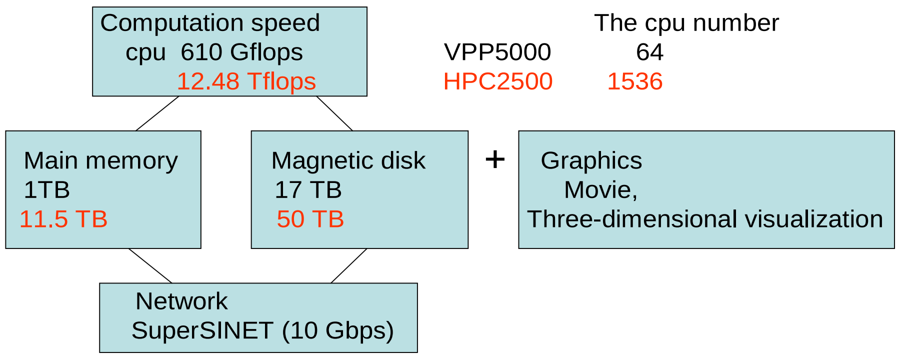


Figure 9. The comparison of PRIMEPOWER HPC2500/1536 and VPP5000/64 (The Information

Table 3. How can the high speed computation be expected at PRIMEPOWER HPC2500 ?

VPP5000/64			
The theory performance：9.6 GF x 64 =	614 GF		
The observation	410 GF (66.8%)		
PRIMEPOWER HPC2500/1536			
The theory performance：8.125 GF x 1536 =	12,480 GF		
Expectation value	3,400 GF (27.2%)		
	GF/CPU	CPU number	computation efficiency speed
The three-dimensional domain decomposition	1.416 x (8.125/5.2) x	1536 =	3,400 GF (27.2%)
The two-dimensional domain decomposition	0.202 x (8.125/5.2) x	1536 =	485 GF (3.9%)
The one-dimensional domain decomposition	0.028 x (8.125/5.2) x	1536 =	67 GF (0.5%)
The observation			
	GF/CPU	CPU number	computation efficiency speed
The three-dimensional domain decomposition	1.831 x	256 =	469 GF (22.5%)
	1.968 x	512 =	1008 GF (24.2%)
	1.660 x	1024 =	1700 GF (20.4%)
	1.499 x	1536 =	2303 GF (18.5%)
The two-dimensional domain decomposition	1.396 x	256 =	357 GF (17.1%)
The one-dimensional domain decomposition	0.907 x	128 =	116 GF (11.1%)

6. Summary

The supercomputer of the Information Technology Center of Nagoya University, which has been used at computer application cooperative research of Solar-Terrestrial Environment Laboratory, was also renewed from vector-parallel machine Fujitsu VPP5000/64 to scalar-parallel machine Fujitsu PRIMEPOWER HPC2500 in March, 2005. The variable (for example, x direction) of the do loop of the most internal has been taken as long as possible, because it was important to improve the vectorization efficiency first of all in the vector computer. In this way, it has succeeded in making simplicity comparatively high-efficient parallel program even in either methods such as 1, 2, 3-dimensional domain decomposition and MPI and HPF (High Performance Fortran) ([6],[8]). However, it is the inadequate to take the variable of the do loop of the most internal in the long variable like the space one-direction, because rate of hit of the cache is the most important, in the scalar-parallel machine. For example, it is desirable that 8 vector components of MHD equation to which be directly concerned in the calculation are taken. In this way, when especially CPU increased, it was feared whether many programs which had realized high efficiency with the vector-parallel machine would face the problem which cannot raise high efficiency with a scalar-parallel machine at all. However, it has proved clearly that appropriate improvement in the speed can be realized without hardly changing the three-dimensional MHD code which demonstrated high efficiency with the vector-parallel machine conventionally by PRIMEPOWER

HPC2500, either. Moreover, in three-dimensional MHD code (3D Domain Decomposition by f (nb, nx2, ny2, nz2)) using improvement in a hit ratio of cash and three-dimensional domain decomposition, the maximum high-speed value (they are 929GF at 512CPU) is acquired only by the process parallel by MPI, and it has checked that scalability was also very good. However, in order to acquire such high efficiency, it is indispensable to use the hard transfer function of data.

In the present system of FUJITSU PRIMEPOWER HPC2500 because the maximum of process parallel is 512, when using CPU beyond it, process parallel and thread parallel need to be used together ([10]-[13]). Improvement in the speed was unrealizable in the combination of the process parallel exceeding 512CPU, and thread parallel with the test result of hard transfer function OFF of the early data. However, if it was made hard transfer function ON of data, the parallel computing efficiency exceeding the 512CPU will have improved remarkably. In this way, in the three-dimensional MHD code using three-dimensional domain decomposition of f (8, 2046, 2046, 2046) which enlarged the scale of calculation, the maximum high speed of 1.7TFLOPS was able to be obtained by combined use of the process parallel (512) using 1024CPU, and thread parallel (2). Furthermore, the maximum high speed of 2.3 TFLOPS was able to be obtained by 1536CPU. Thus, when using process parallel and thread parallel together well, it proved that remarkable parallel computing efficiency is actually acquired. However, when using CPU of such a large number, and the number of threads is increased, there is a tendency for parallel computing efficiency to fall conversely. This is directly related also to the question of the following most fundamental parallel computer use, when using many CPUs by a parallel computer. Can performing all by user parallel (flat parallel what is called by MPI etc.) acquire high efficiency? or Can performing in user parallel and the optimal automatic parallel combination acquire high efficiency? That is, if the optimized automatic parallel in a node is used and a user realizes optimal parallel between nodes, it is necessary to ask the credibility of the myth that the greatest parallelization efficiency should be acquired currently believed by many people although it is unidentified. In user parallel and an automatic parallel combination said to be the optimal, the thing that there is no example in which the greatest parallelization efficiency was acquired is a counterargument to it. These propositions are considered that a dispute continues in the future, only in user parallel, efficient parallel computing is realized at this parallel computing. Because FUJITSU has said that it is scheduled to raise the maximum of a process parallel number to over 1024 CPUs, this is a talk very delightful for a user.

In order to do the high-efficient computation by scalar-parallel machine which will become the mainstream in future more and more, it is required much more that it conducts by arranging that it changes the order of the arrangement in order to raise rate of hit of the cache, that it introduces the three-dimensional domain decomposition method and the communication method by the consolidation system transmission and reception. By the introduction of new three-dimensional MHD code which satisfies such condition, the way seems to open it for the first time in the possibility of the effective utilization of the large scale scalar-parallel machine. Of course, the necessity for a high-speed hard transfer function is a thing needless to say. Though the high-efficient computation of absolute efficiency of 40-70% of the vector-parallel machine realized in three-dimensional global MHD code which solves interaction between the solar wind and earth's magnetosphere is impossible, it is also certainly likely to be expectable that it exceeds 20% with a scalar-parallel machine from the test calculation using this FUJITSU PRIMEPOWER HPC2500, or calculation efficiency near 20% can be realized. It is just going to expect the excellent technical capabilities which a Japanese supercomputer maker has, and the further development power very much.

Acknowledgments

The computer simulation of this paper is made using the supercomputer of Information Technology Center of Nagoya University, Fujitsu VPP5000/64, and PRIMEPOWER HPC2500, NEC SX6 of JAXA/ISAS, Earth Simulator (ES) of the Earth Simulator center, and Hitachi SR8000 and SR11000/j1 of National Institute of Polar Research. Moreover, in the development and the test of the three-dimensional MHD code of a MPI version, I am thankful to Tomoko Tsuda assistant of Information Technology Center of Nagoya University and FUJITSU, LTD. who obtained many advices, and the Masaki Okada of a National Institute of Polar Research who cooperated in supercomputer use of Hitachi.

References

[1] T. Ogino, A three-dimensional MHD simulation of the interaction of the solar wind with the earth's magnetosphere: The generation of field-aligned currents,
J. Geophys. Res., 91, 6791-6806 (1986).

[2] T. Ogino, R.J. Walker and M. Ashour-Abdalla, A global magnetohydrodynamic simulation of the magnetosheath and magnetopause when the interplanetary magnetic field is northward,
IEEE Transactions on Plasma Science, Vol.20, No.6, 817-828 (1992).

[3] T. Ogino, Two-Dimensional MHD Code, (in Computer Space Plasma Physics), Ed. by H. Matsumoto and Y. Omura, Terra ScientifiCPUblishing Company,
161-215, 411-467 (1993).

[4] T. Ogino, R.J. Walker and M. Ashour-Abdalla, A global magnetohydrodynamic simulation of the response of the magnetosphere to a northward turning of the interplanetary magnetic field,
J. Geophys. Res., Vol.99, No.A6, 11,027-11,042 (1994).

[5] T. Ogino,　The electromagnetism hydrodynamic simulation of a Solar wind and a Magnetosphere　interaction,
Plasma/Nuclear fusion academic journal, Specially plans CD-ROM (description dissertation), Vol.75, No.5, CD-ROM 20-30, 1999.
http://gedas.stelab.nagoya-u.ac.jp/simulation/mhd3d01/mhd3d.html

[6] T. Ogino, The computer simulation of a Solar wind Magnetosphere interaction

Nagoya University mainframe center news, Vol.28, No.4, 280-291, 1997.

[7] T. Ogino, “Computer simulation and visualization”
Ehime University synthesis information processing center public relations, Vol.6, 4-15, 1999.
http://gedas.stelab.nagoya-u.ac.jp/simulation/simua/ehime985.html

[8] T. Ogino, Global MHD Simulation Code for the Earth's Magnetosphere Using HPF/JA,
Special Issues of Concurrency: Practice and Experience, 14, 631-646, 2002.
http://gedas.stelab.nagoya-u.ac.jp/simulation/hpfja/mhd00.html

[9] T. Ogino, The parallel three-dimensional MHD code and three-dimensional visualization
A heavenly body and the simulation summer seminar lecture text of space plasma, 107-136、2003.
http://center.stelab.nagoya-u.ac.jp/kaken/mpi/mpi-j.html

[10] Tomoko Tsuda, Parallelization by MPI, Information Technology Center of Nagoya University
MPI school data, February, 2002

[11] Tooru Nagai, Tomoko Tsuda, About the outline and the usage of a new supercomputer system
Nagoya University mainframe center news, Vol.4, No.1, 6-14、2005.

[12] Masaki Aoki, From a vector (VPP5000) to Scala SMP (PRIMEPOWER HPC2500)
Nagoya University mainframe center news, Vol.4, No.1, 20-27、2005.

[13] Tomoko Tsuda, The bookmark of new supercomputer (HPC2500) use
Nagoya University mainframe center news, Vol.4, No.2, 104-113、2005.

[14] FUJITSU, LTD., MPI programming -Fortran, C- The 1.3rd edition, April, 2001,
The 1.4th edition, December, 2002.

[15] FUJITSU, LTD., MPI use guidance document For V20, UXP/V First edition, September, 1999

[16] Yukiya Aoyama (IBM Japan Corp.), Parallel programming key, The MPI version, Key series 2
January, 2001

[17] The homepage of refer to the three-dimensional MHD code of 1, 2, and 3 domain decomposition used in this paper
Homepage of New Domain Decomposition 3D MPI MHD Programs,
http://center.stelab.nagoya-u.ac.jp/kaken/mpi/mpiex012.html