

Performance measurement of magnetohydrodynamic code for space plasma on the various scalar-type supercomputer systems

Fukazawa, Keiichiro

Department of Earth and Planetary Sciences, Graduate School of Science, Kyushu University

Umeda, Takayuki

Solar-Terrestrial Environment Laboratory, Nagoya University

Miyoshi, Takahiro

Department of Physical Science, Graduate School of Science, Hiroshima University

Terada, Naoki

Department of Geophysics Science, Graduate School of Science, Tohoku University

他

<https://hdl.handle.net/2324/26035>

出版情報 : IEEE Transactions on Plasma Science. 38 (9 PART 1), pp.2254-2259, 2010-09. IEEE
バージョン :
権利関係 : (C) 2010 IEEE



Performance measurement of magneto-hydro-dynamic code for space plasma on the various scalar type supercomputer systems

Keiichiro Fukazawa, Takayuki Umeda, Takahiro Miyoshi, Naoki Terada, Yosuke Matsumoto, and Tatsuki Ogino

Abstract—The computational performance of magneto-hydro-dynamic (MHD) code is evaluated on several scalar type supercomputer systems. We have made performance tuning of a three-dimensional MHD code for space plasma simulations on the SR16000/L2, FX1 and HX600 supercomputer systems. For parallelization of the MHD code, we use four different methods, i.e., regular one-dimensional, two-dimensional, three-dimensional domain decomposition methods and a cache-hit type of three-dimensional domain decomposition method. We found that the regular three-dimensional decomposition of the MHD model is suitable for HX600 system, and cache hit type of three-dimensional decomposition is suitable for SR16000/L2 and FX1 systems. As results of these runs, we achieved a performance efficiency of almost 20% for MHD code on all systems.

Index Terms—Scalar massively computer system; Maganeto-Hydro-Dynamics simulation; Performance evaluation

I. INTRODUCTION

The numerical magneto-hydro-dynamic (MHD) code for space plasma has been optimized for vector-type supercomputers for a long time because most of supercomputers with vector processors have high performance in 1990's. These codes often have achieved a very high computational efficiency

(the ratio of the effective performance to the theoretical performance). However, almost 100% of the “top 500” supercomputer systems in the world recently adopt the scalar type processors and more than 85% of systems consist of the 64bit-x86 processor architecture [1]. The other scalar type computers are POWER and SPARC architectures.

The purpose of the present study is to make performance tuning of MHD code for space plasma simulations on scalar-type massively parallel supercomputer systems. In general, the computing efficiency of user applications on a scalar-type computer tends to be low (~5%) [2], although the computing efficiency of LINPACK sometimes exceeds 80%. In this paper, a performance measurement study of MHD codes is carried out on several scalar-type supercomputer systems at the Kyushu University (SR16000/L2) and Nagoya University (FX1 and HX600) in Japan.

The Hitachi SR16000/L2 supercomputer system at Kyushu University consists of 42 nodes and each node has 16 processor cores of IBM dual core POWER6 (4.7GHz, L2: 4MB/core, L3: 32MB/CPU) and 128GB memory which connects through the 2 InfiniBand DDR links with a bandwidth of 4 GB/sec per 1 link. The simultaneous multithreading (SMT) works on the dual core POWER6 processor so that four parallelized program can be executed on one processor.

The Fujitsu FX1 is often called as a pre-petascale computer which is the next-generation super computer in Japan. The FX1 supercomputer system at Nagoya University has 768 nodes. Each node consists of a quad core SPARC64 VII (2.5 GHz, L2: 6MB/CPU) and 32 GB memory which connects through an InfiniBand DDR links (2GB/sec).

The HX600 is a PC-Cluster type computer called as T2K open super computer in Japan [3]. HX600 supercomputer system at Nagoya University consists of 160 nodes and each node has 4 AMD quad core Opteron 8380 processors (2.5 GHz, L2: 512KB/core, L3: 6MB/CPU) and 64 GB DDR2 memory. As for the inter-node connections, each node has 4 InfiniBand links with a bandwidth of 2 GB/sec per 1 link.

This paper is structured as follows. In Section 2, numerical computation techniques and parallelization techniques are briefly described for the MHD code. In Section 3, the performance measurement results of the MHD code on SR16000,

Manuscript received December 1, 2009. This work was supported by Grant-in-Aid for JSPS Fellows No.09J01416352 (K.F.) from JSPS, Grant-in-Aid for Young Scientists (B) No.21740352 (T.U.) and in part Grant-in-Aid for Creative Scientific Research No.17GS0208 from MEXT of Japan. The computational resource was provided by Research Institute for Information Technology in the Kyushu University. This work was also conducted as a computational research project at Solar-Terrestrial Environment Laboratory in Nagoya University.

K. F. is with the Department of Earth and Planetary Sciences, Graduate School of Science, Kyushu University, Fukuoka, 812-8581 Japan (Phone: +81-92-641-3131 (ext 8387), e-mail: fukazawa@geo.kyushu-u.ac.jp).

T. U. is with Solar-Terrestrial Environment Laboratory, Nagoya University, Nagoya, 464-8601 Japan. (e-mail: umeda@stelab.nagoya-u.ac.jp).

T. M. is with the Department of Physical Science Graduate School of Science, Hiroshima University, Higashi-Hiroshima, 739-8526 Japan (e-mail: miyoshi@sci.hiroshima-u.ac.jp).

N. T. is with the Department of Geophysics Science Graduate School of Science, Tohoku University, Sendai, 980-8578 Japan (e-mail: teradan@stpp.gp.tohoku.ac.jp).

Y. M. is with Solar-Terrestrial Environment Laboratory, Nagoya University, Nagoya, 464-8601 Japan. (e-mail: ymatumot@stelab.nagoya-u.ac.jp).

T. O. is with Solar-Terrestrial Environment Laboratory, Nagoya University, Nagoya, 464-8601 Japan (e-mail: ogino@stelab.nagoya-u.ac.jp).

FX1, and HX600 systems are presented, and summary of the present study is given in Section 4.

II. NUMERICAL METHOD

We use a three-dimensional MHD code developed by Ogino et al. [4], which has been used for studies on global structures and dynamics of planetary magnetospheres. The MHD code uses the “Modified Leapfrog” method, in which partial difference equations are solved by the two-step Lax-Wendroff method for one time step and then by the Leapfrog scheme for $(l - 1)$ time steps and the procedure is repeated. Thus the method has second-order accuracy in both space and time. The Modified Leapfrog method is a kind of combination technique which balances numerical stability of the two step Lax-Wendroff method and dissipationlessness of the Leapfrog method. That is, smaller l is desirable for numerical stability, while larger l is desirable for suppression of numerical dissipation. Based on the linear analysis and preliminary simulations, the value of $l = 8$ has been chosen. It should be noted that the Modified Leapfrog method achieved high performances on various supercomputers (more than 90% computing efficiency on NEC SX-6 supercomputer system and more than 20% computing efficiency on Fujitsu HPC2500 supercomputer system, etc.).

As a parallelization technique, the domain decomposition method for dividing three-dimensional space is adopted. Usually in parallel computing on a distributed-memory computer, the domain decomposition is an easy way to decompose three-dimensional Eulerian variables. In the case of a three dimensional model, the dimension of domain decomposition can be chosen as one dimension, two dimensions, or three dimensions as shown in Fig.1. The computation time (T_{S1} , T_{S2} , T_{S3}) and communication time (T_{C1} , T_{C2} , T_{C3}) in these cases can be roughly estimated as follows [5].

i) One-dimensional decomposition.

$$T_{S1} = k_1 n^3 / p, \quad T_{C1} = k_2 n^2 (p - 1) \quad (1)$$

ii) Two-dimensional decomposition.

$$T_{S2} = k_1 n^3 / p, \quad T_{C2} = 2k_2 n^2 (p^{\frac{1}{2}} - 1) \quad (2)$$

iii) Three-dimensional decomposition.

$$T_{S3} = k_1 n^3 / p, \quad T_{C3} = 3k_2 n^2 (p^{\frac{1}{3}} - 1) \quad (3)$$

Here k_1 and k_2 are constant coefficients for computation time and communication time, respectively, n is the size of a three-dimensional array in one direction, and p is the number of processor cores. Here a cubic three-dimensional array ($n = n_x = n_y = n_z$) is considered as shown in Fig.1. To simplify the analysis, we assume that each dimension is divided by the same number of processor cores. Fig.2 shows the computation time T_S , communication time T_C , and total computation time ($T_S + T_C$) obtained from Eqs. (1), (2) and (3). Although the computation times, T_{S1} , T_{S2} , and T_{S3} , become shorter in inverse proportion to the number of processor cores p from these equations, communication times, T_{C1} , T_{C2} , and T_{C3} become longer as p increases. From Fig.2, one can find that the communication time of the three-dimensional domain decomposition is the shortest. It is noted that the coefficients k_1 and k_2 assumed to be independent of the way of decomposition. In general, three-dimensional domain decomposition is most efficient for a scalar parallel computer, while two-dimensional domain decomposition is most efficient for a vector parallel computer. This is because vectorization of the most inner loop reduces the coefficient of computation time k_1 substantially. Effective use of cache memory is important for obtaining better performance on a scalar processor. Since the size and structure of cache memory depend on CPU architecture and model, it is not easy to obtain the highest performance on scalar-processor computers.

In the MHD model, the number of treated physical variables is eight: plasma density N , velocity (V_x , V_y , V_z), pressure P , and magnetic field (B_x , B_y , B_z). In the present MHD code, an array of these physical variables are defined as $f(n_x, n_y, n_z, n_m)$ (here referred as “Type A”) with $n_m = 1$ and $n_m = 5$ being N and P , $n_m = 2$ being V_x , $n_m = 3$ being V_y , $n_m = 4$ being V_z , etc. n_x , n_y and n_z are the array size in x , y , and z -direction, respectively. Since these physical variables on the same grid point are used repeatedly in computation, an array defined as $f(n_m, n_x, n_y, n_z)$ (here referred as “Type B”) is considered to be more effective for the cache hit. In the present study, we evaluate performance

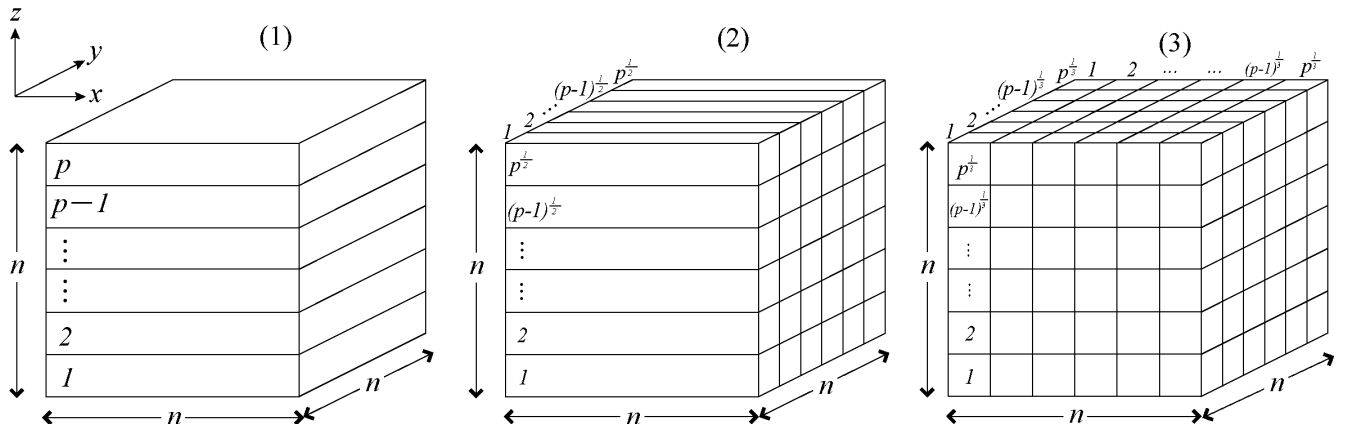


Fig. 1. Schematics of three kinds of the domain decomposition methods. (1) One-dimensional domain decomposition in z -direction. (2) Two-dimensional in y - and z -directions. (3) Three-dimensional domain decomposition.

of both Type A and B on scalar processors. If the Type B is suitable for the computer systems, then it is believed that the performance is followed by Eqs. (1) ~ (3) and Fig.2.

III. RESULTS

We use the array of 64 MB/core for the computational domain and additionally 192 MB/core for workspaces for computing the MHD equations with the Modified Leapfrog method.

To minimize the communication time, we use a buffer array which stores all the boundary data for inter-core and inter-node communications. In these communication processes, the “MPI_sendrecv” function is adopted because MPI_sendrecv is a bidirectional communication function which includes both sending data and receiving data in one process. The other communication functions, for example pairs of MPI_send/MPI_isend and/or MPI_irecv/MPI_recv, need two processes for each sending and receiving processes. It is also noted that the MPI_sendrecv function is usually optimized on each supercomputer system while blocking and non-blocking send/receive functions sometimes give buffer overflow when sending/receiving large amount of data.

In general, the one-dimensional decomposition is a classical technique for vector-type parallel computers (not for large number of CPUs) and two-dimensional decomposition is most suitable for vector massively-parallel computers (such as the Earth Simulator). Then the three-dimensional decomposition is most suitable for scalar massively parallel computers because the communication time is shortest as shown in Eq. (3).

A. SR16000/L2

On SR16000/L2 system, the Hitachi Optimized Fortran Compiler, the IBM XL Fortran Compiler and the IBM Parallel Environment for AIX 5L including the MPI-1.2 communication library are installed. As compiler options, we adopt

-Oss -parallel=0 -divopt -pvfunc=0 -nolimit

for the Hitachi Compiler and

-qfree=f90 -O3 -qarch=prw6 -qtune=pwr6

for the IBM Compiler. The meanings of these options are

followings [6, 7]. -Oss: highest optimization option,

-parallel=0: no automatic parallelization option, -divopt:

conversion of division to multiplication, -pvfunc=0: no use of built-in vectorized mathematical functions, -nolimit: no time and memory limits to compile.

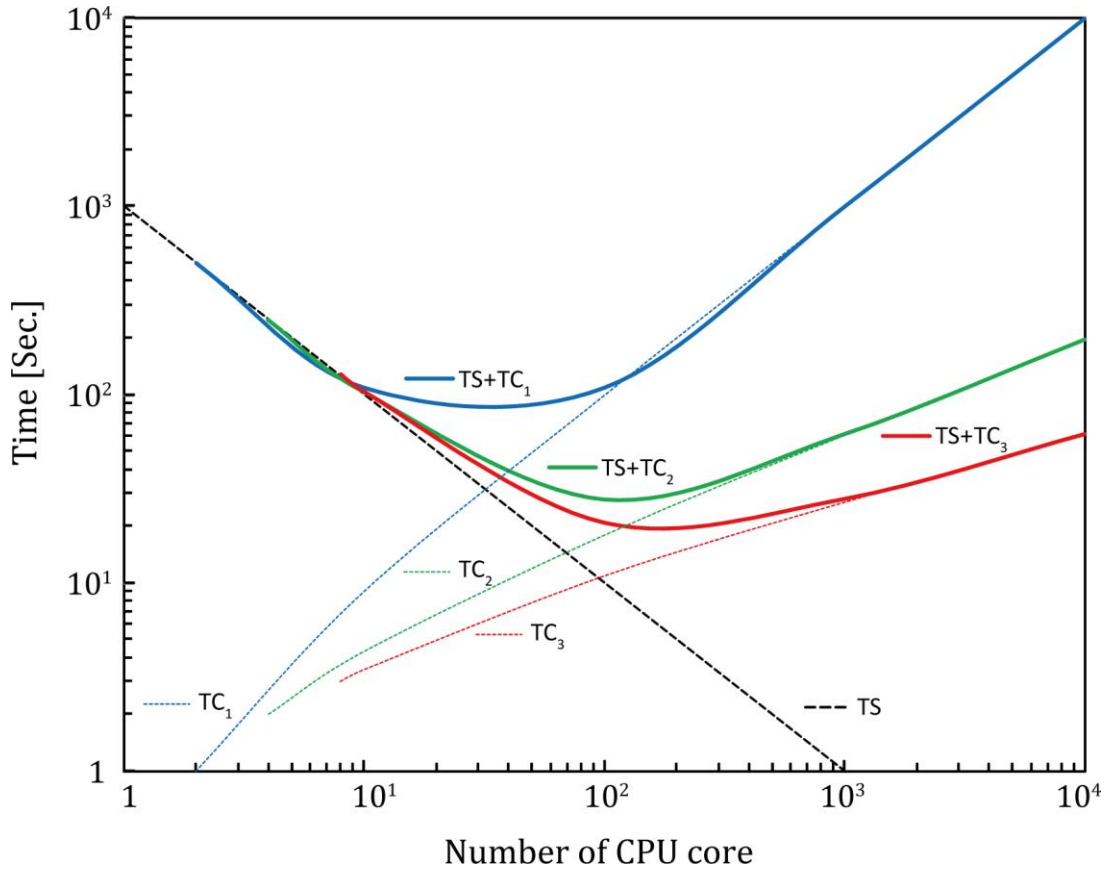


Fig. 2. Computation time (T_S), communication time (T_C), and total parallel computation time ($T_S + T_C$) as a function of the number of processor core obtained from Eqs. (1)-(3). The dashed line is the computation time (T_S), the dotted lines indicate the communication times of one-dimensional, two-dimensional and three-dimensional decomposition (TC_1 , TC_2 , TC_3). The total computation times are represented as solid lines. Here k_1 and k_2 are set to be 1 and 0.01, respectively, to simplify this figure.

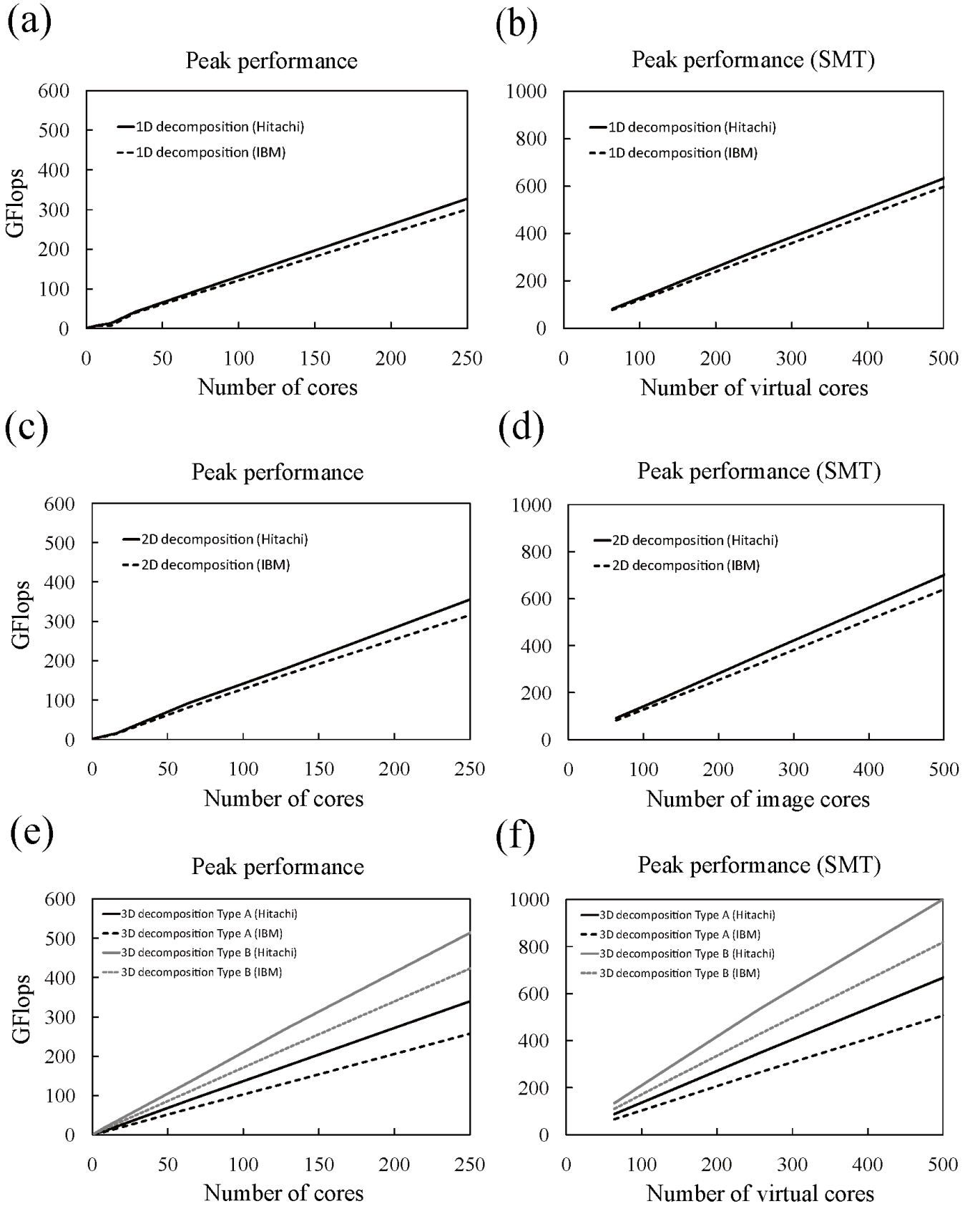


Fig. 3. Computational peak performance with the one (a and b), two (c and d) and three (e and f) decomposition. Left panels show the results not using SMT and right is results with SMT.

We evaluate the 8 nodes (256 cores) at maximum. Fig.3 shows the computational peak performance of the one (top panel), two (middle panel) and three (bottom panel) dimensional decompositions with the Hitachi and IBM compilers. The solid line shows the result with the Hitachi compiler, and the dashed line shows the result with the IBM compiler. Left panel shows the no SMT case and right is the results of SMT working. In the all dimensional decompositions, the computational peak performance scales linearly as the number of cores increases with both Hitachi and IBM compilers. As the peak performance with 256 cores, the one-dimensional decomposition gives 335.5 GFlops with Hitachi compiler and 308.4 GFlops with the IBM compiler. With SMT, we can use the 512 virtual cores on 256 real cores and then obtain 646.5 GFlops (Hitachi) and 611.5 GFlops (IBM) in one-dimensional decomposition. In these cases the performance efficiency exceeds 7% with 256 cores and 13.7% with 512 virtual cores (using SMT). Note SMT makes the number of core double virtually without change in the peak performance.

The computational peak performance of the two-dimensional decomposition method with 256 cores is 363.9 GFlops (Hitachi) and 323.9 GFlops (IBM). With SMT this method obtains 717.5 GFlops (Hitachi) and 655.7 GFlops (IBM) on 512 virtual cores. These performances are higher than the results of one-dimensional decomposition. The performance efficiency exceeds 15.0% (Hitachi) and 13.6% (IBM) with 512 virtual cores.

In the three-dimensional decomposition, we examined the two types of array (Type A and Type B) as described in Section 2. The format of (e) and (f) in Fig.3 is the same as (a)~(d) in Fig.3 while the gray solid line and the gray dotted line are added for results of Type B with the Hitachi compiler and the IBM compiler, respectively. The peak performance with virtual 512 cores of Type B with the Hitachi compiler (1023.7 GFlops) is the best in the all cases, and the peak performance of Type A (683.8 GFlops) is much worse than that of Type B. With the IBM compiler, the peak performance of Type B (838.5 GFlops) is better than that of Type A (518.2 GFlops).

Fig.4 shows the comparison among the three decompositions. Left panel shows the results with the Hitachi compiler and right shows those with the IBM compiler. The three-dimensional decomposition Type B gives the best performance with both Hitachi and IBM compilers. The results of other decompositions are not so different but the three-dimensional decomposition Type A is worst in both Hitachi and IBM. In general, the three-dimensional decomposition gives a better performance than the two-dimensional decomposition on scalar parallel computers. This indicates the importance of cache hit tuning for POWER6 architecture. If its tuning is not considered in codes, vector like code (one and two-dimensional decomposition) obtain the good performance as compared with the scalar type code such as three-dimensional decomposition Type A. POWER6 has L2 4MB./core and L3 32MB/CPU. Thus using

cache effectively is much more important.

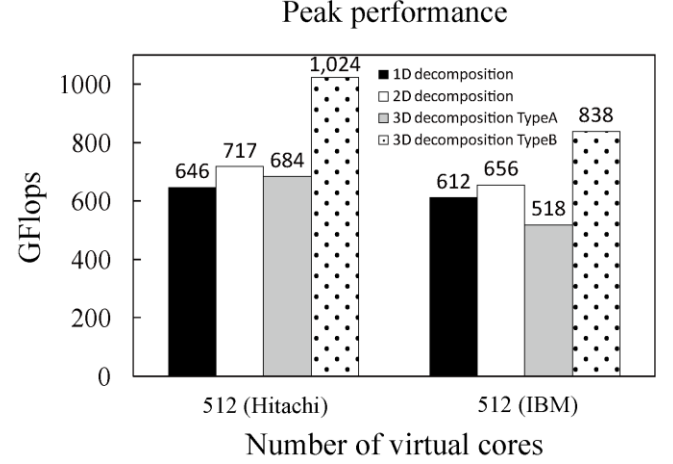


Fig. 4. Comparison of the three type of decompositions with the Hitachi (left) and IBM compilers (right).

B. FX1 and HX600

On FX1 system, the Fujitsu Fortran Compiler and the Parallelnavi Language Package 3.2 including the MPI-1.2 communication library are installed. As compiler options, we adopt

-Am -Kfast, V9, prefetch_model=FX1, prefetch_indirect, pre fetch_strong, mfunc=2
for the Fujitsu Compiler.

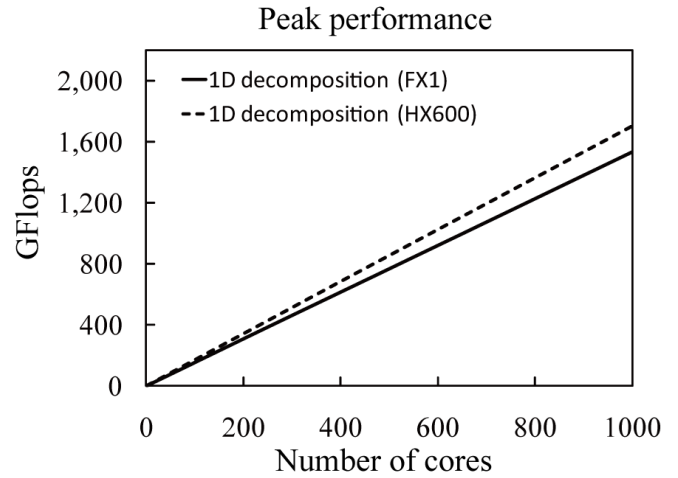
On HX600 system, the Fujitsu Fortran Compiler, Parallelnavi Language Package for Linux V3.0L30 including the MPI-1.2 communication library are installed. As compiler options, we adopt

-Am -Kfast, fsimple, fuse, prefetch_indirect, prefetch_sequential, SSE4, loop, nomfunc, noparallel, tiling
for the Fujitsu Compiler. The meanings of these options are followings [8]. -Am: use of module in programs, -Kfast: optimization option for the code running fast, -KV9: creation of an object program that uses SPARC V9 instructions, -Kprefetch_model=FX1: prefetch optimization for FX1 system, -Kprefetch_indirect: creation an object with prefetch instruction for the array data accessed indirectly(list access), -Kprefetch_strong: creation of strong prefetch instructions, -Kmfunc=2: use of multi-operation functions in place of mathematical functions, -Kfsimple: simplification of floating-point operation, -Kfuse: fusion of neighboring loops, -Kprefetch_sequential: creation an object with prefetch instruction for the array data accessed sequentially, -KSSE4: use of the SSE4 instructions, -Kloop: optimization for multiple loop, -Knomfunc: no use of multi-operation functions, -Knoparallel: no automatic parallelization, -Ktiling: optimization of loop tiling.

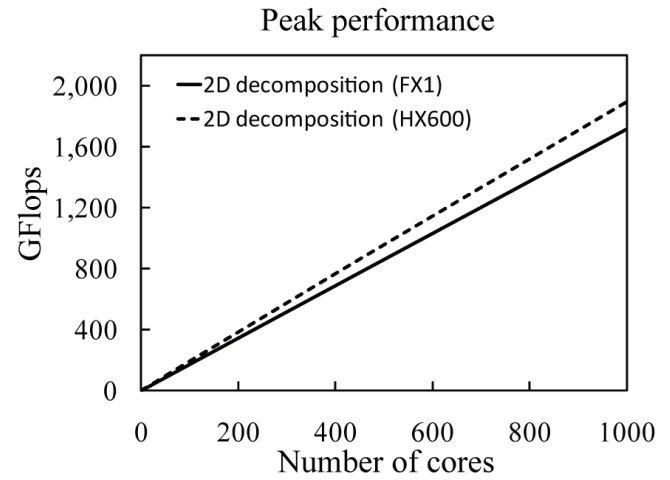
We evaluate the 256 nodes (1024 cores) at maximum on both FX1 and HX600 systems. The CPU architecture is different between FX1 and HX600 systems but the ideal performance of FX1 system (quad core SPARC64VII 2.5 GHz) is the same as HX600 system (quad core AMD Opteron 2.5 GHz). Fig.5

shows the computational peak performance of the one (top panel: a), two (middle: b) and three (bottom: c) dimensional decomposition on the FX1 and HX600 systems. The solid line

(a)



(c)



(e)

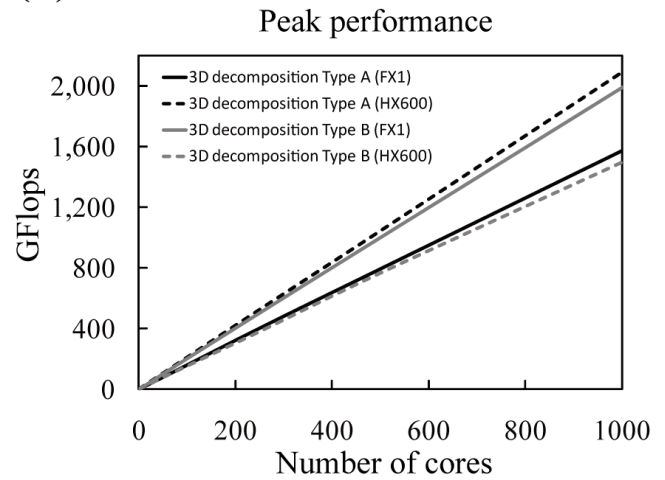


Fig. 5 Computational peak performance with one, two and three dimensional decomposition on FX1 and HX600.

shows the result on the FX1, and the dashed line shows the result on the HX600 system. In all decomposition on FX1 and

HX600 systems, the computational peak performance scales linearly as the number of cores increases. As the peak performance with 1024 cores, the one-dimensional decomposition gives 1.5708 TFlops on FX1 system and 1.7406 TFlops on HX600 system. In this case the performance efficiency is 15.8% and 17.0%.

The computational peak performance of the two-dimensional decomposition method with 1024 cores is 1.7542 TFlops and the performance efficiency exceeds 17% on FX1 system. On HX600 system this method obtains the peak performance 1.9395 TFlops and performance efficiency 18.9%.

In the three-dimensional decomposition, we also examined the two types of array (Type A and Type B). The format of (c) in Fig.5 is the same as (a) and (b) in Fig. 5 while the gray solid line and the gray dotted line are added for results of Type B with the FX1 and HX600 systems, respectively. The peak performance of Type B on FX1 system (2.0318 TFlops) is the best in the all decompositions, and the peak performance of Type A (1.6071 TFlops) is slightly worse than that of Type B. On HX600 system, on the other hand, the peak performance of Type A is 2.1391 TFlops and that of Type B is 1.5311 TFlops. These results indicate that the three-dimensional decomposition with the array of Type B is not effective on HX600 system.

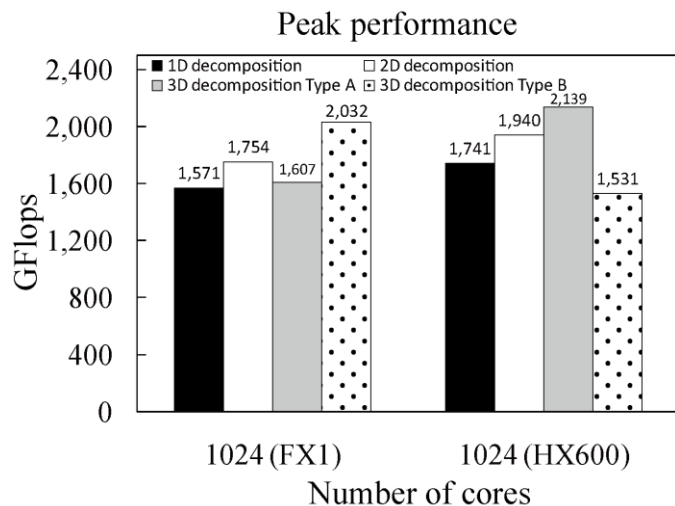


Fig. 6. Comparison of the three type of decompositions on the FX1 (left) and HX600 (right).

Fig.6 shows the comparison between the three decompositions with 1024 cores on FX1 and HX600 systems. The three-dimensional decomposition gives the best performance with both FX1 (Type B) and HX600 (Type A) systems. On FX1 system there is no difference of peak performance among one, two and three (Type A) decompositions. On the other hand, on HX600 system two-dimensional decomposition obtains a better performance than the FX1 system case, however three-dimensional decomposition Type B is worst performance in all cases. The behavior of FX1 system performance is similar to that of SR16000/L2 system while the SPARC64VII on FX1

system does not have much cache (L2 6MB/CPU) as compared to the POWER6 (L2 4MB/core, L3 32MB/CPU). As results of three-dimensional decomposition Type A and B, using cache effectively is important for FX1 system. On HX600 system three-dimensional decomposition Type A and two-dimensional decomposition give good performance. This results show the cache tuning is not effective to HX600 system (quad core AMD Opteron).

IV. SUMMARY

In this study we have made performance measurements and tuning of MHD code for three types of scalar supercomputers, which are the SR1600/L2 supercomputer system at the Kyushu University, FX1 and HX600 supercomputer systems at the Nagoya University. These supercomputers consist of POWER6, SPARC64VII and Opteron processors, respectively. For the MHD code, we evaluated four types of decomposition methods: regular one-dimensional, two-dimensional, three-dimensional domain decompositions and a cache hit considered type of three-dimensional decomposition. We found that the three-dimensional decomposition method is the most suitable for these three supercomputers. Although the cache hit considered type of three-dimensional decomposition is suitable for SR16000/L2 and FX1 systems, not cache hit considered type of three-dimensional decomposition is suitable for HX600 system. As results, we obtained the best performance of 1.0236 TFlops (performance efficiency 21.3%) with 512 virtual cores on SR16000/L2, 2.0318 TFlops (19.8%) with 1024 cores on FX1 system and 2.1391 TFlops (20.9%) with 1024 cores on HX600 system.

REFERENCES

- [1] Top500 Supercomputing Sites. (<http://www.top500.org/>)
- [2] Oliker, L., Canning, A., Carter, J., Shalf, J., Ethier, S.: Scientific computations on modern parallel vector systems. In: Proc. SC2004: High Performance Computing, Networking, and Storage Conference, 2004
- [3] T2K Open Supercomputer Alliance. (<http://www.open-supercomputer.org/>)
- [4] T. Ogino, R. J. Walker, M. Ashour-Abdalla, "A global magnetohydrodynamic simulation of the magnetopause when the interplanetary magnetic field is northward", *IEEE Trans. Plasma Sci.* vol. 20, pp. 817-828, 1992
- [5] K. Fukazawa, T. Umeda, T. Ogino, "Performance measurement of electromagnetic fluid codes for space plasma on the T2K open supercomputer system", *Parallel Computing*, submitted
- [6] Optimization FORTRAN90 User's Guide, Hitachi Ltd
- [7] User's Guide of XLF, Hitachi, Ltd
- [8] Fortran User's Guide, Fujitsu Inc.