

スカラー並列機を用いた高効率MHDコードの開発

萩 野 竜 樹 中 尾 真 季

近年、大規模数値計算に使用できるスーパーコンピュータが、ベクトル並列機からスカラー並列機へと急速に移りつつある。国内でも日立と富士通がクラスター型のスカラー並列機の市場化に移行し、ベクトル並列機の開発を継続しているのはNECのみとなった。こうした状況下、スカラー並列機で効率良く動くプログラムの開発が急務となってきたが、欧米でもその高効率の並列計算プログラムの開発が最近10年間の懸案問題で未だに成功したとは言えない状況である。私達は、そのスカラー並列機で高速計算できる一つの有効な方法を見出したので、その方法の要点と富士通のPRIMEPOWER HPC2500などのスカラー並列機を含むスーパーコンピュータで実施した現在までのテスト計算の結果を示す。

I. まえがき

コンピュータシミュレーションを実行するスーパーコンピュータをめぐる環境が急速に変わりつつある。それは、ベクトル並列機からスカラー並列機への移行である。欧米では、約10年前からスカラー並列機に代わり、ベクトル並列機はコスト高とみなされたために一時は市場から駆逐された。しかし、2003-2004年にベクトル並列機である日本の地球シミュレータが世界最高速のスーパーコンピュータの栄誉を授かるに至って、再評価の気運が起こり、米国政府の援助を受けてクレイ社が再びベクトル並列機 (CRAY X1E) の開発生産に着手した。一方国内では、まず、日立がベクトル並列機からスカラー並列機 (Hitachi SR8000, SR11000, 疑似ベクトル機ともいわれる) へ移行して富士通もそれに続き (PRIMEPOWER HPC2500)、2004年度にベクトル並列機の開発を継続しているのはNECのみ (NEC SX6, SX7, SX8) となった。

なぜ、ベクトル並列機からスカラー並列機への移行が急激に起こったのか、それはコストパフォーマンスが良いに違いないと多くの人が信じたためである。それでは、スカラー並列機がベクトル並列機より本当にコストパフォーマンスが良かったのか。これが利用者にとって最大の疑問点であったが、実際にスーパーコンピュータの高速計算を必要としている実用プログラムで実証される前に、スカラー並列機への移行が急速に進行したのが現実であった。大規模計算のプログラム実行で、ベクトル並列機では約40%の絶対効率が出ていたものが、スカラー並列機では10年以上の努力が世界中でなされてきたにもかかわらず、3-7%の絶対効率しか到達できないと言われてきた。この低い絶対効率の壁は超えられないものなのか。私達は、スカラー並列機で高速計算できる方法をいろいろ検討してきた結果、可能性のある一つの有効な解決方法を見出した。その方法と種々のスーパーコンピュータに対して実施した現在までのテスト結果を示すと

共に、2005年に更新された名古屋大学情報連携基盤センターのスカラー並列型のスーパーコンピュータ Fujitsu PRIMEPOWER HPC2500 に対する予測と期待、及び3次元MHDコードを用いたテスト計算結果について述べる。

II. MHDシミュレーションコード

私達は、3次元MHD（電磁流体力学的）シミュレーションコードを用いて、太陽風と地球磁気圏相互作用のグローバルな計算機シミュレーションを実施してきた（[1]-[7]）。ダイポール磁場の軸が傾いている時（傾き角度 30° で北半球が夏）の地球磁気圏の磁場構造の例を、IMF（惑星間磁場）が南向きと北向きの場合について図1に示す。さらに、IMFが南北朝夕面で回転している時で、北向きから南向きに変化した後に起こるプラズモイドの尾方向への放出時のVRML（Virtual Reality Modeling Language）による3次元可視化（[5], [7], [9]）の例を図2に示す。このようなグローバルなモデルでは、外側境界は影響を減らすためになるべく遠くに置き、かつ空間分解能を上げる必要から、計算方法の改良が行われる一方、最大級のスーパーコンピュータの利用、それも効率的な利用が必須である。MHDモデルでは、Modified Leap-Frog法を用いて、MHD方程式とマクスウェル方程式を初期値境界値問題として解いている（[8], [9]）。

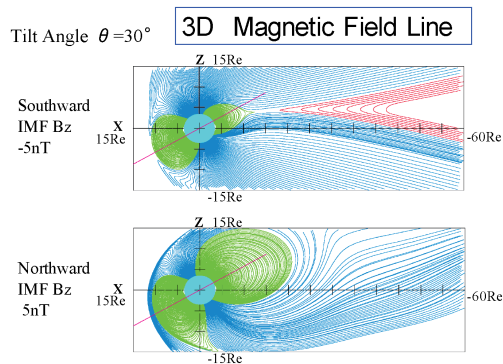


図1 惑星間磁場（IMF）が南向きと北向きの場合で、傾いたダイポール磁場に対する地球磁気圏の磁場構造

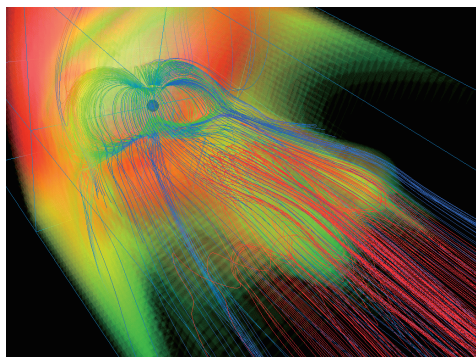


図2 IMFが北向きから南向きに変化した後に起こる尾部磁気リコネクションの発生とプラズモイドの尾方向への放出時の地球磁気圏構造

〈基礎方程式〉

MHDモデルの基礎となる規格されたMHD方程式とMaxwell方程式

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

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

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

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

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

大規模シミュレーションに必要な並列計算を可能にするMPI (Message Passing Interface) ([10]-[16]) を用いた並列化Fortran MHDコードの構造を図3に示す ([8], [9])。Modified Leap-Frog法は、Two step Lax-Wendroff法と同じように2ステップの計算になる。MPIを用いたMHDコードにおける並列計算の特徴は、それぞれのステップの計算の直前に、分散メモリの並列計算に必要な通信をまとめていることである。この一括通信を有効に利用する方法は、効率的な並列プログラムを作成するためには極めて重要な点である。

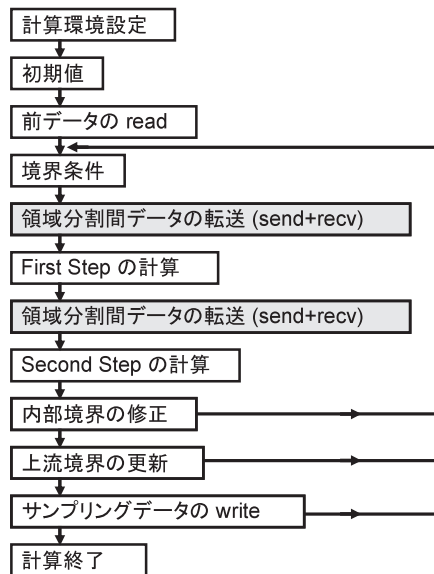


図3 MPIを用いたMHDコードの構造

III. 領域分割法による並列計算

分散メモリ型の並列計算機を用いた並列計算では、3次元配列に対して領域分割を用いるのが通常である ([8], [9])。3次元モデルの場合、領域分割の次元を1次元、2次元、3次元に選ぶことができる。その場合の計算時間と通信時間は大まかにつぎのように見積もることができる。

	計算時間	通信時間
1次元領域分割	$T_s = k_1 N^3 / P$	$T_c = k_2 N^2 (P-1)$
2次元領域分割	$T_s = k_1 N^3 / P$	$T_c = 2k_2 N^2 (P^{1/2} - 1)$
3次元領域分割	$T_s = k_1 N^3 / P$	$T_c = 3k_2 N^2 (P^{1/3} - 1)$

ここに、 k_1 と k_2 は一定の係数、 N は3次元配列における1方向の変数量、 P は並列CPUの数である。計算時間と通信時間の和が並列計算に要する時間になるが、その領域分割による並列計算効率を図4に示す。計算時間 T_s はCPU数 P に反比例して短くなるが、通信時間 T_c は P の増加に伴って長くなる。しかし、その通信時間の長くなる様子は、1, 2, 3次元領域分割によって大きく異なる。すなわち、3次元領域分割が最も通信時間を短くできることと、1次元と2次元領域分割の間でもその差は大きいことが理解できる。ただし、この比較では通信時間を決める係数 k_2 が同じであると仮定したが、その条件も通信部分のプログラムの工夫により実現できる。こうして、スカラー並列機では3次元領域分割が、一方ベクトル並列機では、1つの次元方向はベクトル化に利用する必要があるために2次元領域分割が最も効率的であろうと予想できる。

3次元グローバルMHDコードにおいて3次元領域分割の具体的導入方法の要点をつぎに示

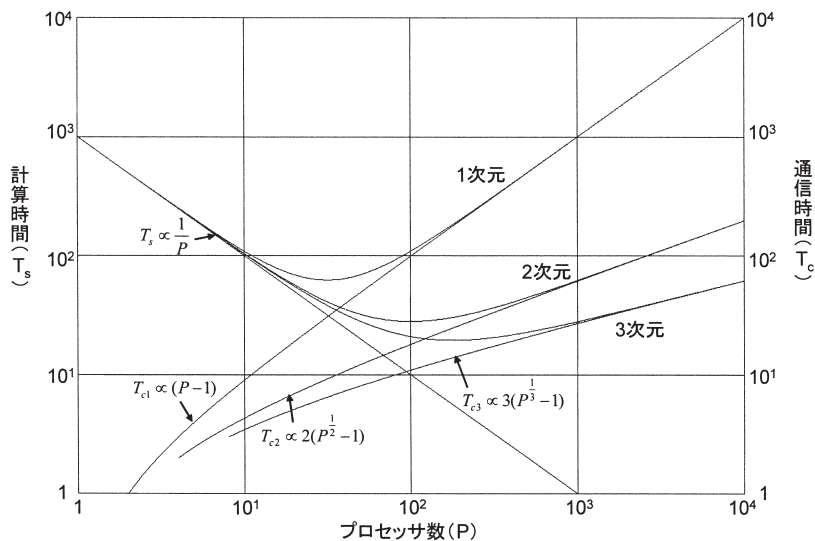


図4 1, 2, 3次元領域分割法による並列計算効率, 並列計算時間は計算時間 (T_s) と通信時間 (T_c) の和となる

す。MPI ([10], [14], [15]) を用いて 3 次元空間 (x, y, z) で領域分割を行うような以下の例の場合, x, y, z 方向にそれぞれ 2 分割 (npex=2, npey=2, npez=2) で, 全体で npe=npex*npey*npez=8 分割となり, 8 個の CPU を使用する。それぞれの CPU に入る配列変数は分割されて, f (nb, 0:nxx+1, 0:nyy+1, 0:nzz+1) の配列が割り当てられ, nb=8 は MHD 方程式の成分の数である。すなわち, 配列変数は元々は領域分割された隣の CPU に存在する, 3 次元方向にそれぞれ両側のデータ 1 面が加わる。itable (-1:npex, -1:npey, -1:npez) は wild card を含んだ CPU 間のデータ転送のための参照表である。また, ftemp1x, ftemp2x などは, 一括データ転送のために用意した 2 次元境界面のバッファ配列である。このように, 参照表とバッファ配列を準備することにより, 3 次元領域分割のために生じる分散型メモリ間のデータ転送を簡単にかつ効率的に実行できる。

《3 次元領域分割の方法》

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

分散型メモリ間のデータ転送のプログラムはつぎに示すように極めて簡単になる。まず, wild card を変換表 itable で使用することにより, プログラム中の x, y, z 方向の端での特殊性がなくなる。これによって if 文などの条件文を使う必要性がなくなる。つぎに, ftemp1x=f (:, is, :, :) で f (nb, 0:nxx+1, 0:nyy+1, 0:nzz+1) の x 方向の左端の yz 面のデータをバッファ配列 ftemp1x に一時保存して, mpi_sendrecv で ileftx の CPU の ftemp1x から irightx の CPU の ftemp2x に一括転送する。続いて, f (:, ie+1, :, :) =ftemp2x を用いて f (nb, 0:nxx+1, 0:nyy+1, 0:nzz+1) の x 方向の右端に追加された yz 面のデータに移すことにより, データ転送を完結する。このようにして, 3 次元領域分割された各方向の両面のデータを分散型メモリを持つ

CPU間で容易に一括転送できる。プログラムであきらかなように、この方法ではifなどの条件文とdo loopが一切不要である。1次元領域分割との比較で問題となる3次元領域分割によるオーバーヘッドが、ほとんど生じないであろうことはそのことから推測できる。結果として、図4に示すように通信時間 T_c の比例係数を1, 2, 3次元領域分割でほとんど同じにすることができる。

《領域分割されたCPU間のデータ転送》

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

1, 2, 3次元領域分割の実際の3次元MHDコードは後で示すホームページで見ることができる ([9], [17])。

IV. 領域分割法を用いたMHDコードの計算効率の比較

情報連携基盤センターのスカラ並列型スーパーコンピュータ富士通PRIMEPOWER HPC2500でMPIを用いて書かれた3次元MHDコードをコンパイル, 実行した時のシェルを(A)と(B)に示す ([10], [11], [13])。 (A) は128 CPUのプロセス並列のみの使用で, (B) は128 CPUを用いて32プロセス並列と4スレッド並列 (自動並列) ([12], [13]) を併用した場合である。 また, コンパイルのオプションは最大の計算効率を得られた場合のものを示しているが, これらのオプションは今では特に指定する必要はない。 今回は使用しなかったが, スレッド間バリア機能を有効にする `-Khardbarrier` というコンパイルオプション (デフォルトではOFF) を加えるのもスレッド並列での高速化には有効であるとのコメントを富士通SEからもらっている。

(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
```

まず、富士通のPRIMEPOWER HPC2500を用いて、プロセス並列のみとプロセス並列とスレッド並列を併用した場合の3次元MHDコード（1次元領域分割手法）の計算効率を比較した結果を表1に示す。表では、使用した総CPU数、プロセス並列数、スレッド並列数に対して、1回の時間ステップに要する計算時間、計算速度 (GFLOPS) 及び1 CPU当たりの計算速度 (GFLOPS/CPU) を示した。MPIのFORTRANプログラムなので、プロセス並列数は2以上に設定する必要がある。また、その制約からスレッド並列数も自動的に64以下に選ぶことになる。128 CPUの場合、プロセス並列のみの場合が最も計算効率が良いが、プロセス並列とスレッド並列の併用の場合でもそれほど遜色のない計算効率がでている。ただし、スレッド並列数が32、64と大きくなるにつれて計算効率が劣化する傾向がみられる。

一般に、大きな共有メモリが必要であるとか、プロセス並列を大きくは取れない制限がある場合はスレッド並列を使用しなければならない ([12], [13])。例えば、表1のz方向の1

表1 PRIMEPOWER HPC2500でスレッド並列を用いた場合のMHDコードの計算効率

総CPU数	プロセス並列数	スレッド並列数	計算時間 (sec)	計算速度 (GFLOPS)	計算速度/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	9.080	2015.17	1.312
1536	256	6	16.462	1111.54	0.724

次元領域分割の3次元MHDコードの場合、外部境界条件設定の制限から、z方向の配列変数 ($nz2=nz+2=262$) の半分以下にプロセス並列数を取る必要がある。したがって、CPU数が256以上になる場合、必然的にスレッド並列を使うことになる。このように、プロセス並列とスレッド並列を併用する必要が生じた場合、3次元MHDコードではスレッド並列数を4～16程度に取れば効率的な計算ができることが分かる。もちろん、どのくらいのスレッド数を用いれば高効率を得られるかは、プログラムに強く依存すると思われるが、まずは、16以下のあまり多くないスレッド数を試すことから始めればよさそうである。表1は、最大の並列計算速度が得られたデータのみを示しているが、スレッド数を多くする場合計算速度のばらつきがかなりあらわれてくる。これは、データ通信との兼ね合いとも考えられるが、実際の計算ではもう少し計算速度が遅くなることも起こりそうである。また、256CPUと512CPUで64スレッドの場合のデータがないが、これはワーク領域不足で実行できなかったためである。

3次元領域分割の3次元MHDコードで配列の大きさが $f(8, 522, 262, 262)$ の場合、512CPUに固定してプロセス並列とスレッド並列数を変えた場合は面白いデータが得られた。最大計算速度は512プロセス並列のみの場合で、1007 GFLOPSを得た。これに対して、スレッド並列数を増やしていくと、2スレッド並列で少し計算速度が低下し、4, 8スレッド並列でさらに大きく計算速度が低下した。特に、512プロセス並列のみと512CPU使用で64プロセス並列と8スレッド並列の場合の計算速度の差は重要である。後者ではx, y, z方向とも同じ4分割なので、後者も前者とほとんど同程度の計算速度が得ると期待していたが実際は違っていた。この結果は、3次元領域分割法を使って通信量を最小にしようとする方法と8スレッド並列内の自動並列化の方法がうまく整合しないことを意味していると考えられる。すなわち、8スレッド並列内の配列変数 $f(nb, nx2, ny2, nz2)$ の順序付けがどのようになされているのかが、問題となりそうである。さらに大きな配列の場合、 $f(8, 1024, 1024, 1024)$ と $f(8, 2046, 2046, 2046)$ でも似た傾向がみられる。これらは、自動並列の最適化とユーザ並列の最適化が必ずしも両立しないことを示唆しているようにも思われる。特に、 $f(8, 2046, 2046, 2046)$ の配列に対して、1024CPUと1536CPU使用でそれぞれ4スレッド並列と6スレッド並列を指定した場合、並列計算効率は上がらずにむしろ低下している。このスレッド並列数を増やした場合の計算効率劣化に対して、スレッド間バリア機能を有効にする-Khardbarrierのコンパイルオプションが有効であろうとのコメントを富士通からもらっているが確認はまだである。

1, 2, 3次元領域分割法を用いた場合の3次元MHDコードの計算効率の比較を表2に示す。数値は、1個当たりのCPUに対する計算速度 (GF/PE) を示した。VPP5000, NEC SX6, NEC ES (Earth Simulator) はベクトル並列機であり、PRIMEPOWER HPC2500とHitachi SR8000, SR11000はクラスター型のスカラー並列機である。3次元領域分割法では2種類のコードがあり、 $f(nx2, ny2, nz2, nb)$ は配列の順序がx, y, z方向が先でベクトル成分(8個)が後であり、 $f(nb, nx2, ny2, nz2)$ は計算に関係する変数のメモリを近くにするためにベクトル成分(8個)を配列の最初に移動している。この配列順序の入れ替えは、スカラー並列機ではキャッシュのヒット率が高くなり、計算効率の改善が期待されるが、PRIMEPOWER HPC2500ではその効果が顕

表2 3次元MHDコードによる計算効率の比較

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)	1.312 (2.015 TF)				
f (8, 2048, 2048, 2048)	1536		(MPI, 6 thread)	0.724 (1.112 TF)				

著にあらわれている。また、512 CPUを用いたベクトル並列機のNEC ESの結果をみるとx方向でベクトル化を活かした2次元領域分割が最も効率的な結果を得ている。

スカラー並列機で高効率を実現するためには、キャッシュのヒット率を上げることと3次元領域分割を行って通信量を減らすことが重要であると提案してきたが、表2のPRIMEPOWER HPC2500の512 cpuまでの結果からそのことの重要性を明瞭にみることができる。PRIMEPOWER HPC2500での2、3次元領域分割の計算結果は512cpuまですべてプロセス並列のみを利用したものである。1次元領域分割の場合、HPC2500で非常に悪い結果が出ているが、

これは3次元MHDコードで境界を含めたグリッド点数 $(nx, ny, nz) = (510, 254, 254)$ を2のべき乗 $(512=510+2, 256=254+2)$ に取ったため、グリッド点数を少し変える $(nx, ny, nz) = (520, 260, 260)$ と括弧で示す数値のように効率が大きく改善された。すなわち、領域分割方向の配列の大きさを2のべき乗にとり、CPU数も2のべき乗にとった場合、メモリアクセスの競合が生じて計算速度が極端に劣化する現象が発生したと考えられる ([12], [13])。したがって、これまでのほとんどのMHDコードは、最初に高効率を得られない場合でもCPU数や配列の大きさを少し変えるなどの変更さえすれば大きな問題なく高速に計算できることが分かった。

残された問題点として、1次元領域分割では128並列までがプログラム上可能で、256cpuを使う場合128プロセス並列と2スレッド並列の組み合わせ、512cpuを使う場合128プロセス並列と4スレッド並列の組み合わせを使う必要があった。これらの場合 $(nx, ny, nz) = (520, 260, 260)$ のグリッドの場合でも、データのハード転送機能をオフ ($-ldt=0$) にした場合は計算速度はかなり低下して、さらに512cpuでは極端に低下する結果が得られた。しかし、そのデータトランスファユニットをオン ($-ldt=1$) にすれば、その計算効率の低下傾向は大きく改善された。また、名古屋大学のPRIMEPOWER HPC2500では最大512プロセス並列までに制限されているので、1024cpu使用では512プロセス並列と2スレッド並列の組み合わせ、1536cpu使用では512プロセス並列と3スレッド並列の組み合わせを用いることになる。しかし、これらの多くのcpuを使用しているプロセス並列とスレッド並列の組み合わせでは、データのハード転送機能オフ ($-ldt=0$) のテストでは並列化の効率は良くならないばかりか、逆に非常に悪くなっていた。しかし、その機能をオン ($-ldt=1$) にすれば、表1, 2の3次元領域分割法を用いた場合の $f(8, 1024, 1024, 1024)$ と $f(8, 2046, 2046, 2046)$ の結果から分かるように、計算効率は著しく改善された。このように、ハードデータ転送機能を使い、かつ3次元領域分割法を用いればスケーラビリティが1536cpu使用まで非常によく伸びることが確認できた。もう一点、気が付くことは、多数のCPUを使用する場合、プロセス並列数を減らしてスレッド並列数を増やす方法は計算効率の劣化を招くことである。この傾向は表1の3次元領域分割法を用いた場合の $f(8, 522, 262, 262)$, $f(8, 1024, 1024, 1024)$, $f(8, 2046, 2046, 2046)$ の結果に顕著にあらわれている。これら現在までの結果は、それぞれ最適のユーザ並列と自動並列を組み合わせる方法より、ユーザ並列のみを利用する方法 (例えばフラットMPI) の方が、高効率の並列計算を実現できるという考え方を指示しているようにも思われる。VPP5000の括弧内の数値は、32-64PEでこれまでどのくらいの計算速度の最高値が出ていたかを比較のために示したものである。

実際、VPP5000の16cpuで通常動かしていた3次元MHDコードは110-120 GFLOPSの速度が出ていたが、同じコードをPRIMEPOWER HPC2500で動かすと64cpuで80-90 GFLOPS、128cpuで90-140 GFLOPSが出ていて、これまでの計算はそのまま継続できる。もちろん、VPP5000で高効率を実現していた3次元MHDコードはcpu数がさらに増えると計算速度が飽和する傾向が見られる。プロセス並列とスレッド並列を併用して、計算速度の変化を調べると、スレッド並列を併用する場合6-7%計算速度が向上する場合もあるが、前述したようにデータ

トランスファーユニットを利用しない場合はスレッド並列併用で計算効率が向上するのはcpuがあまり多くない場合（128～256cpu以下）に限られる傾向が認められる。しかし、データのハード転送機能をオンにしておけば、そのcpu数の上限はかなり上昇する。こうして、プロセス並列とスレッド並列の併用でもかなり高い並列計算効率の達成を期待することができる。

V. 2005年更新のスーパーコンピュータ PRIMEPOWER HPC2500の概要と高速計算への予想と実測

平成16年度に更新された情報連携基盤センターの計算機の概要をつぎに示す。従来のベクトル並列機Fujitsu VPP5000/64からスカラー並列機Fujitsu PRIMEPOWER HPC2500に2005年3月に更新された（[10], [11]）。

〈スーパーコンピュータ〉

Fujitsu PRIMEPOWER HPC2500 23ノード

（64 CPU / メモリ 512 GB × 22ノード, 128 CPU / メモリ 512 GB × 1ノード）

総合性能 : 12.48 TFLOPS

総メモリ容量 : 11.5 TB

ディスク容量 : 50 TB

更新によって、どれだけ高性能になるかを計算速度、主メモリ、磁気ディスク容量及びネットワーク速度のスーパーコンピューティングの4機能について、VPP5000/64とPRIMEPOWER HPC2500のカタログ性能比として図5に示す。ベクトル並列機VPP5000/64からスカラー並列機PRIMEPOWER HPC2500への更新によって、理論上の性能は大幅によくなるが、スカラー並列機の相対的な非効率性があるので性能向上を安易に期待するのは危険である。更新後のPRIMEPOWER HPC2500でどこまで高速計算が期待できるかを従来のテストを基に予想するとつぎのようになった。その後、テスト計算を実施したので、その実測結果も表3に加えた。

更新当時の危惧をまとめて書くと（[8], [9]）、特に、ベクトル並列機で高効率を実現していた1次元領域分割のMHDコードでの劣化は悲惨であるかも知れないし、計算効率がVPP5000/64の10分の1に落ちる可能性があるかも知れない。もし、このような状況になるとすれば、従来の3次元MHDコードは全く使いものにならないことになる。2, 3次元領域分割をうまく採用してやっと同程度の計算効率が期待できそうである。配列変数の並び替えを行った3次元領域分割のMHDコードを用いた場合、最も楽観的な予想で3.4 TFLOPSが期待できる可能性がある。しかし、絶対効率でどれだけよくなるかは、実際に3次元MHDコードで実行することが肝要である。

上記のように、スカラー並列機PRIMEPOWER HPC2500への更新に多くの懸念材料があったが、2005年3月から実際に稼働開始したので、精力的にテストした結果が表1, 2であり、実測としてまとめたものが表3である。これらの結果から、従来ベクトル並列機で高効率を発揮していた3次元MHDコードも、ほとんど変更なしでそれなりの高速化を実現できることがあきらかになった。実際1次元領域分割法を用いた従来の3次元MHDコードでも、128cpuで116GF、256cpuで175GFの計算速度を実現している。こうして、スペースプラズマの大規模3次元MHD

シミュレーションは、情報連携基盤センターのベクトル並列機VPP5000からスカラー並列機PRIMEPOWER HPC2500への移行に際しても問題なくスムーズに維持することができた。さらに、 $(nx2, ny2, nz2) = (2048, 2048, 2048)$ のグリッドを用いた3次元領域分割法の場合、512cpuで929GF, 1024cpuで1700GF, さらに1536cpuで2015GFの最高速値を得ることができた。また、前述のスレッド間バリア機能を有効にするコンパイルオプションを使用すれば、さらなる高速化が期待できそうである。

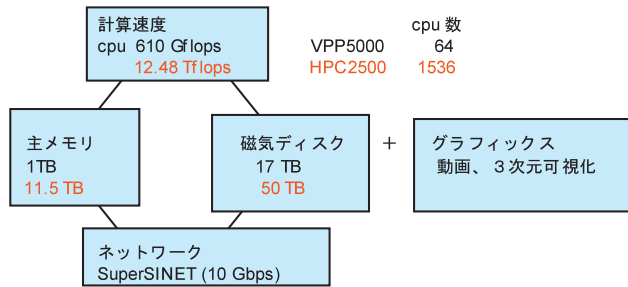


図5 PRIMEPOWER HPC2500/1536とVPP5000/64の比較（名古屋大学情報連携基盤センター）とスーパーコンピューティングの4つの重要な機能

表3 PRIMEPOWER HPC2500でどこまで高速計算が期待できるかとその実測値

VPP5000/64					
理論性能	: 9.6 GF x 64 =				614 GF
実測					410 GF (66.8%)
PRIMEPOWER HPC2500/1536					
理論性能	: 8.125 GF x 1536 =	12,480 GF			
期待値		3,400 GF (27.2%)			
		GF/cpu	cpu数	計算速度	効率
3次元領域分割	1.416 x (8.125/5.2) x		1536 =	3,400 GF (27.2%)
2次元領域分割	0.202 x (8.125/5.2) x		1536 =	485 GF (3.9%)
1次元領域分割	0.028 x (8.125/5.2) x		1536 =	67 GF (0.5%)
実測					
		GF/cpu	cpu数	計算速度	効率
3次元領域分割		1.831 x	256 =	469 GF (22.5%)
		1.968 x	512 =	1008 GF (24.2%)
		1.660 x	1024 =	1700 GF (20.4%)
		1.312 x	1536 =	2015 GF (16.1%)
2次元領域分割		1.396 x	256 =	357 GF (17.1%)
1次元領域分割		0.907 x	128 =	116 GF (11.1%)

VI. まとめ

太陽地球環境研究所の計算機利用共同研究で使用している情報連携基盤センターのスーパーコンピュータも2005年3月にベクトル並列機Fujitsu VPP5000/64からスカラー並列機Fujitsu

PRIMEPOWER HPC2500へ更新された。ベクトル計算機はまずベクトル化効率を上げることが重要だったので最内のdoループの変数(例えばx方向)をできるだけ長く取ってきた。こうして、1, 2, 3次元領域分割やMPIやHPF (High Performance Fortran) ([6], [8]) などいずれの方法でも、比較的簡単に高効率の並列プログラムを作成することに成功してきた。しかし、スカラー並列機では、キャッシュのヒット率が最も重要なので、最内のdoループの変数は空間一方向などの長い変数に取るのは不適切で、例えば計算に直接関与するMHD方程式の8個のベクトル成分を取るのが望ましい。こうして、特にCPUが増えた場合、ベクトル並列機で高効率を実現していた多くのプログラムは、スカラー並列機で全く高効率を上げることができない問題に直面するのではないかと懸念していた。しかし、従来ベクトル並列機で高効率を発揮していた3次元MHDコードも、PRIMEPOWER HPC2500でほとんど変更なしでそれなりの高速化を実現できることが明確に分かってきた。さらに、キャッシュのヒット率の向上と3次元領域分割を用いた3次元MHDコード (3D Domain Decomposition by f (nb, nx2, ny2, nz2)) では、MPIによるプロセス並列のみで最高速値 (512cpuで929GF) が得られていて、かつスケラビリティも非常に良いことが確認できた。ただし、これらの高効率を得るためにはデータのハード転送機能を利用することが不可欠である。

富士通PRIMEPOWER HPC2500の現在のシステムではプロセス並列は512が上限なので、それ以上のcpuを使用する場合、プロセス並列とスレッド並列の併用が必要である ([10]-[13])。その初期のデータのハード転送機能オフのテスト結果では512cpuを越えてのプロセス並列とスレッド並列の組み合わせで高速化を実現することはできなかった。しかし、データのハード転送機能をオンにすれば、その512cpuを越えての並列計算効率は著しく改善された。こうして、計算のスケールを大きくした。f (8, 2046, 2046, 2046) の3次元領域分割を用いた3次元MHDコードでは、1024cpuを用いたプロセス並列 (512) とスレッド並列 (2) の併用で1.7 TFLOPS、さらに1536cpuで2TFLOPSの最高速を得ることができた。このようにプロセス並列とスレッド並列をうまく併用すれば、かなりの並列計算効率が実際に得られることが分かった。ただし、このような多数のCPUを使用する場合、スレッド数を増やすと並列計算効率が逆に低下する傾向がある。これは、並列計算機で多数のCPUを使う場合、すべてユーザ並列 (いわゆるMPIでのフラット並列など) で実行するのが高効率を得られるか、ユーザ並列と自動並列の最適の組み合わせで実行するのが高効率を得られるかという、最も基本的な並列計算機利用の問いにも直接関係している。すなわち、最適化されたノード内自動並列を利用し、利用者が最適のノード間並列を実現すれば、最大の並列化効率を得られるはずである、という未確認だが多くの人に信じられている神話の信憑性を問う必要がある。それに対する反論は、ユーザ並列と自動並列の最適と言われる組み合わせで、最大の並列化効率を得られたという実例がないというものである。これらの命題は今後も論争が続くと思われるが、今回の並列計算ではユーザ並列のみの方が高効率の並列計算を実現している。富士通はプロセス並列数の上限を1024CPU以上に上げる計画であると言っている。これは利用者にとって大変嬉しい話である。

今後ますます主流となってきたようなスカラー並列機での高効率な計算を実現する場合、キャッ

シュのヒット率を上げるために配列の順序を入れ替えること、3次元領域分割法を導入すること、及びその通信方法をまとめて一括方式で送受信することが一層必要となる。そのような条件を満たす新しい3次元MHDコードの開発導入により、大規模スカラ並列機の効率的利用の可能性に初めて道が開けてくると思われる。もちろん、高速なハード転送機能の必要性は言うまでもないことである。太陽風と地球磁気圏相互作用を解く3次元グローバルMHDコードで実現していた、ベクトル並列機の場合のような絶対効率で40—70%の高効率計算は無理としても、今回の富士通PRIMEPOWER HPC2500を用いたテスト計算からスカラ並列機で20%を超える、あるいは20%に近い計算効率が実現できることも確実に期待できそうである。日本のスーパーコンピュータメーカーの優れた技術力とさらなる開発力に大いに期待するところである。

謝 辞

本稿のコンピュータシミュレーションは名古屋大学情報連携基盤センターのスーパーコンピュータ、Fujitsu VPP5000/64とPRIMEPOWER HPC2500、JAXA/ISASのNEC SX6、地球シミュレータセンターのEarth Simulator (ES)、及び国立極地研究所の日立SR8000とSR11000/j1を利用してなされたものです。また、MPIバージョンの3次元MHDコードの開発とテストでは多くの助言を頂いた名古屋大学情報連携基盤センターの津田知子助手と富士通株式会社の方、さらに、日立のスーパーコンピュータ利用で協力して頂いた国立極地研究所の岡田雅樹助手に感謝いたします。

参考文献

- [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 Scientific Publishing 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] 荻野竜樹, 「太陽風と磁気圏相互作用の電磁流体力学的シミュレーション」
プラズマ・核融合学会誌, CD-ROM 特別企画 (解説論文), Vol.75, No.5, CD-ROM 20-30, 1999
<http://gedas.stelab.nagoya-u.ac.jp/simulation/mhd3d01/mhd3d.html>
- [6] 荻野竜樹, 「太陽風磁気圏相互作用の計算機シミュレーション」

名古屋大学大型計算機センターニュース, Vol.28, No.4, 280-291, 1997

- [7] 萩野竜樹, 「コンピュータシミュレーションと可視化」
愛媛大学総合情報処理センター広報, Vol.6, 4-15, 1999
<http://gedas.stelab.nagoya-u.ac.jp/simulation/simua/ehime985.html>
- [8] Ogino, T., 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] 萩野竜樹, 「並列3次元MHDコードと3次元可視化」
天体とスペースプラズマのシミュレーションサマーセミナー講義テキスト, 107-136,
2003
<http://center.stelab.nagoya-u.ac.jp/kaken/mpi/mpi-j.html>
- [10] 津田知子, 「MPIによる並列化」, 名古屋大学情報連携基盤センター MPI講習会資料,
2002年2月
- [11] 永井亨, 津田知子, 「新スーパーコンピュータシステムの概要と利用方法について」
名古屋大学情報連携基盤センターニュース, Vol.4, No.1, 6-14, 2005
- [12] 青木正樹「ベクトル (VPP5000) からスカラSMP (PRIMEPOWER HPC2500) へ」
名古屋大学情報連携基盤センターニュース, Vol.4, No.1, 20-27, 2005
- [13] 津田知子, 「新スーパーコンピュータ (HPC2500) 利用のしおり」
名古屋大学情報連携基盤センターニュース, Vol.4, No.2, 104-113, 2005
- [14] 富士通株式会社, 「MPIプログラミング ~ Fortran, C ~」第1.3版, 2001年4月, 第1.4
版, 2002年12月
- [15] 富士通株式会社, 「MPI使用手引き書 V20用」, UXP/V 初版 1999年9月
- [16] 青山幸也 (日本アイ・ビーエム株式会社), 「並列プログラミング虎の巻 MPI版」
虎の巻シリーズ2, 2001年1月
- [17] 本稿で使用した1, 2, 3領域分割の3次元MHDコード参照のホームページ
Homepage of New Domain Decomposition 3D MPI MHD Programs,
<http://center.stelab.nagoya-u.ac.jp/kaken/mpi/mpiex012.html>

(おぎの たつき: 名古屋大学太陽地球環境研究所附属ジオスペース研究センター)

(なかお まき: 名古屋大学太陽地球環境研究所附属ジオスペース研究センター)