Vol.2016-HPC-156 No.3
2016/9/15

SX-ACE と FX100 における MHD シミュレーションコードの性能評価と最適化

深沢圭一郎[†1]　江川隆輔[†2]　磯部洋子[†2]　三吉郁夫[†3]

概要：SX-ACE は最新のベクトル型 CPU からなる計算機だが，スカラコアと ADB と呼ばれるキャッシュを持ち，過去のベクトル計算機とは構成が異なっている．FX100 は最新の SPACR64 アーキテクチャからなる計算機であるが，CPU のコア数が 32(+2 アシスタントコア)と，メニーコアに近い構成となっており，また，HMC と呼ばれる高帯域 3 次元積層メモリを採用している．このように大きな構成の変化がある計算機システムで実アプリケーションの性能評価を行うことは重要である．そこで流体シミュレーションコードの一つである MHD シミュレーションコードを用いて，性能評価，最適化を行った．SX-ACE では伝統的なベクトル最適化が効果的であるが，キャッシュ最適化も効果が高いことが分かった．また FX100 では今までの SPARC64V 系と異なり，キャッシュ最適化より，ベクトル最適化が効果的と明らかになった．

キーワード：性能評価，MHD シミュレーション，ベクトル機，SPARC64

Performance evaluation and optimization of MHD simulation code on SX-ACE and FX100

KEIICHIRO FUKAZAWA[†1]　RYUSUKE EGAWA[†2]　YOKO ISOBE[†2]
IKUO MIYOSHI[†3]

Abstract: SX-ACE is the latest vector-type supercomputer and has a scalar CPU core and CPU cache memory called ADB (Assignable Data Buffer), thus the architecture of SC-ACE is much different from the past vector computer. FX100 consists of the latest SPARC64V CPU which has 32 cores (+2 assistant cores) like a manycore CPU and high bandwidth 3D stacked memory HMC (Hybrid Memory Cube). It is important to evaluate the performance of these big architecture changing computer systems with a real scientific application. In this study we perform the evaluation and optimization of these computer systems using Magnetohydrodynamics (MHD) simulation code. Then we have found that the traditional optimization for vector computer is effective and the cache-hit tuning also effective on SX-ACE, and vectorization is more effective to FX100 than the cache-hit tuning, which is different from past SPARC64V architectures.

Keywords: Performance evaluation, MHD simulation, Vector computer, SPARC64

1. はじめに

1990 年代前半までベクトル型計算機がスーパーコンピュータの代名詞であったように，様々なベクトル型計算機が普及していたが, スカラ CPU の発展とコストの観点から, 2000 年代頃よりスカラ型計算機が普及し始めた．現在の Top500 ではほぼすべての計算機システムがスカラ型であり[1]，特に x86 系が占めている．このような中，最新の NEC SX シリーズである SX-ACE は唯一のベクトル計算機となっている．この SX-ACE は今までのベクトル CPU とは異なり，スカラ処理部を含んだ 4 つのベクトルコアで構成される CPU となっている. また, ADB（Assignable Data Buffer）と呼ばれるキャッシュを CPU に持っている．CPU とメモリの計算性能に対するメモリバンド幅も過去のベクトル型計算機と比べて，小さくなっておりいわゆる B/F 値は 1 となっている．このように計算機としての特徴が過去のベクトル型計算機と大きく変わっているため，これまでにベクトル型計算機に最適化を行ってきたアプリケーションが効率よく動作するか明かではない．更に，近年はスカラ型計算機が普及しているため，スカラ型計算機しか利用したことがないアプリケーション開発者やユーザが増えており，ベクトル型計算機でどの程度の性能が出て，どのような最適化を行うべきなのか理解されていない場合も多い.

Fujitsu FX100 は最新の SPARC64 アーキテクチャ CPU から構成されているように, 京コンピュータの CPU アーキテクチャ的には後継機である．この CPU は 32 コア＋2 アシスタントコアを持ち, メニーコア的な CPU かつ補助コアがあるという新しい構成となっている．京コンピュータを利用するアプリケーションの多くは京コンピュータ向けに最適化を施されているが, FX100 の CPU は京コンピュータの CPU とは構造が明らかに異なっており，最適化も異なることが想像される． また，SPARC64 はスカラ CPU ではあるが RISC 系であり, x86 系に比べて計算機システムとして採

†1 京都大学・学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University
†2 東北大学・サイバーサイエンスセンター
Cyberscience Center, Tohoku University
†3 富士通株式会社
Fujitsu Ltd.

用されていない[1].

このような状況では，実際に科学アプリケーションがベクトル型計算機や SPARC64 アーキテクチャ計算機でどの程度の性能になるのか分からないことが多い．そこで，本研究では，実際に研究に利用されている MHD（Magnetohydrodynamics）シミュレーションコード[2]を用いて，SX-ACE と FX100 の性能評価と最適化を行う．詳細は後述するが，MHD シミュレーションコードは流体シミュレーションコードの一種であり，電磁場を扱う分，計算量が中性流体を扱う場合より増加するが，流体シミュレーションとの共通性は多い．

2. MHD シミュレーションコード

宇宙空間は真空と思われているが，その 99 %はプラズマで満たされている．プラズマとは電離した気体のことであり，帯電している電子とイオンが分かれて存在する状態である．宇宙空間，特に我々の暮らす太陽系においては太陽から太陽風と呼ばれるプラズマの風が常時吹き出しており，太陽系全体にそのプラズマが充満している．このようなプラズマの振る舞いを記述する方程式として Vlasov-Maxwell 方程式がある．これは，無衝突 Boltzmann 方程式と Maxwell 方程式から成る．Vlasov（無衝突 Boltzmann）方程式は以下の形をとる．

$$\frac{\partial f_s}{\partial t}+\vec{v}\frac{\partial f_s}{\partial \vec{r}}+\frac{q_s}{m_s}(\vec{E}+\vec{v}\times\vec{B})\frac{\partial f_s}{\partial \vec{v}}=0 \qquad (1)$$

ここで $\vec{E}$，$\vec{B}$，$\vec{r}$ と $\vec{v}$ はそれぞれ電場，磁場，距離，速度を表す．また，$f_s(\vec{r},\vec{v}_s,t)$ は位置－速度位相空間における分布関数であり，s はイオンや電子など種類を示す．q_s は電荷を m_s は質量を表す．

しかしながら，Vlasov 方程式は多くの成分からなる非線形方程式であり，計算機システムを用いても解くことが非常に難しい．そこで，Vlasov 方程式のモーメントをとることで求められる電磁流体力学（MHD）方程式が，グローバルなプラズマ構造を調べるときには使用されている．MHD 方程式は以下のようになる．

$$\begin{aligned}
&\frac{\partial \rho}{\partial t}=-\nabla\cdot(\mathbf{v}\rho)\\
&\frac{\partial \mathbf{v}}{\partial t}=-(\mathbf{v}\cdot\nabla)\mathbf{v}-\frac{1}{\rho}\nabla p+\frac{1}{\rho}\mathbf{J}\times\mathbf{B}\\
&\frac{\partial p}{\partial t}=-(\mathbf{v}\cdot\nabla)p-\gamma p\nabla\cdot\mathbf{v}\\
&\frac{\partial \mathbf{B}}{\partial t}=\nabla\times(\mathbf{v}\times\mathbf{B})
\end{aligned} \qquad (2)$$

上から，連続の式，運動方程式，圧力変化の式（エネルギーの式），最後が磁場の誘導方程式となる．簡単に言えば，電磁場を考慮した流体力学方程式と呼べる．詳しい導出方法は参考文献を参照されたい[3].

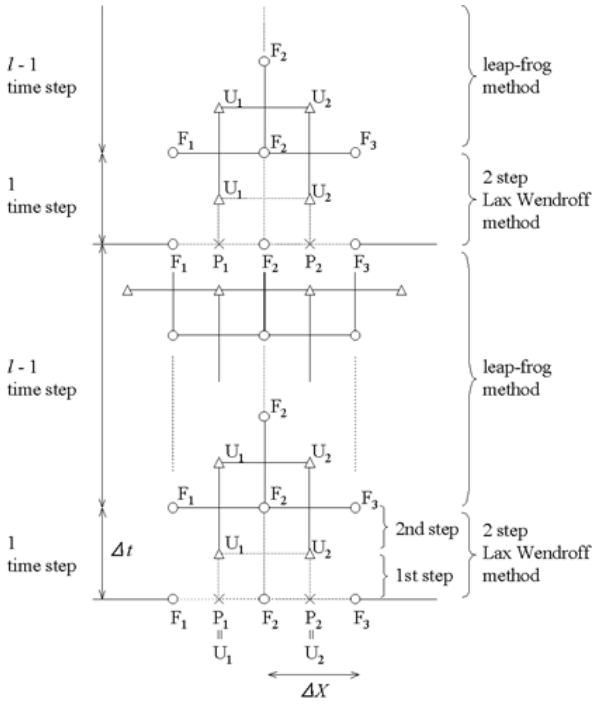

図 1　Modified Leap Frog 法の計算スキーム
Figure 1　Scheme of Modified Leap Frog method

MHD 方程式を解く数値計算法としては，Modified Leap Frog（MLF）法[2，4]という計算法を使用する．これは最初の 1 回を two step Lax-Wendroff 法で解き，続く （l － 1）回を Leap Frog 法で解き，その一連の手続きを繰り返す．図 1 に MLF 法の計算スキームを示す．l の値は数値的に安定の範囲で大きい方が望ましいので，本手法で採用する 2 次精度の中心空間差分では，数値精度の線形計算と予備的シミュレーションから l = 8 に選んでいる．

並列化には MPI と OpenMP を使用する．プロセス並列化手法としては 3 次元空間を分割する領域分割法を用いる．領域分割には，1 次元，2 次元，3 次元分割が考えられ，本性能評価ではこれらすべての評価を行う．領域分割の次元数により，計算ループのベクトル長が変わるため，それぞれの性能評価を行う．

一般的にスカラ CPU で性能を出すためにはキャッシュの有効活用が重要である．基本的な動作としてはメモリアクセス時に，その周辺数 KB のデータをキャッシュに格納する．キャッシュの量や，一度にキャッシュに格納するデータ量は CPU アーキテクチャ毎に変わるので，最高のパフォーマンスを出すにはそれぞれの調整が必要である．MHD シミュレーションにおいては，物理変数がプラズマ密度，速度 3 成分，圧力，磁場 3 成分の計 8 変数となる．そのため，配列を $f(i, j, k, m)$ （これを Type A とする）と定義し，m = 8 としている．数値計算時に同じ場所の物理変数を何度も使うことになるため，一般に $f(m, i, j, k)$ （これを Type

B とする）と定義した方がキャッシュヒット率は上がることがわかっている[5]．そのため，本性能評価においてもこの配列定義を使った性能評価も行う．

3. 性能評価

3.1 SX-ACE

東北大学の SX-ACE を利用して性能評価を行う．この計算機システムは 2,560 ノード（10,240 コア）からなり，理論性能 707 TFlops となっている．それぞれのノードは 1 つの CPU（4 ベクトルコアと 1 スカラコア， 理論性能 276 GFlops）64 GB のメモリ（バンド幅 256 GB/s）から構成されている．ノード間通信は 4 GB/sec で接続されている．SX-ACE での性能評価はすべて Flat MPI 並列で weak scaling で評価を行っている．計算サイズはプロセス当たり，512 MB（100^3 グリッド）を利用した．

まず，3 種類の領域分割と配列の要素を入れ替えた場合の性能を測定した．図 2 に 256 ノードまで利用した場合の測定結果を示す．これまでのベクトル型 CPU と同様にベクトル長が長くなる 1 次元，2 次元領域分割の性能が高くなっている．最大で 256 ノード利用時に 1 次元領域分割で 26 TFlops（実行効率 36.7 %）となっている．図 3 に図 2 の計測時におけるベクトル化率を示す．ここにあるように，性能が高い場合（1 次元，2 次元領域分割）はベクトル化率が高くなっていることが分かる．

また，図 2 では，これまで性能が半分以下に落ちていた配列の要素を入れ替えた場合（3 次元領域分割 Type B）で，それほど性能劣化が見えていない．図 4 にキャッシュの働きをする ADB の利用率を示すが，Type B だけが ADB 利用率が高い．このことから，配列要素入れ替えにより想定通りキャッシュヒット率が高くなることが分かり，この高利用率のために，Type B の場合でもそれほど性能劣化が起きていないと考えられる．この結果からスカラ型 CPU に最適化されたコードでもある程度の性能は出ることが予想される．

実際にシミュレーションを行う場合，並列数が大きい場合，1 次元や 2 次元領域分割は扱いづらい．そのため，3 次元領域分割において，MHD シミュレーションコードの SX-ACE に対する最適化を行い，現実的な性能がどの程度になるか調べる．1 次元領域分割で性能が高いという結果から，配列 *f*(*nx, ny, nz*)において，*nx* を長くし，*ny, nz* を変更し，それに伴い，*ny, nz* の領域分割数（*npy, npz*）を変更した．表 1 にその結果を載せる．ここでは通信要素数と実際に通信にかかった時間も計測している．まず，*ny, nz* ともに極端に小さい場合は余り性能が出ておらず，100 程度が最適なサイズと考えられる．また，この場合が最も通信要素が少なく，実際に通信にかかった時間も短い．3 次元領域分割での並列化性能の向上の面からもこの *ny* = *nz* の設定が最適だった．

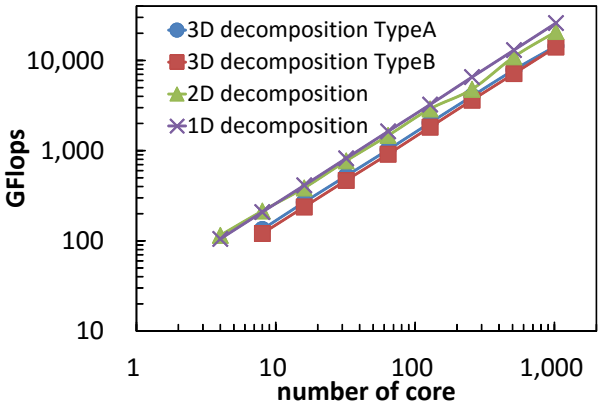

図 2　SX-ACE での MHD コードの性能
Figure 2　Performance of MHD code on SX-ACE

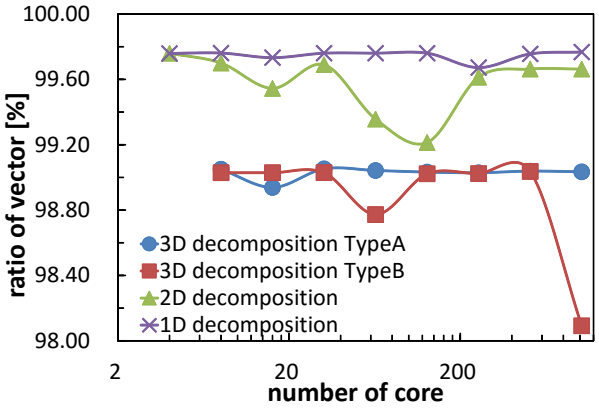

図 3　MHD コードのベクトル化率
Figure 3　Vectorized ratio of MHD code

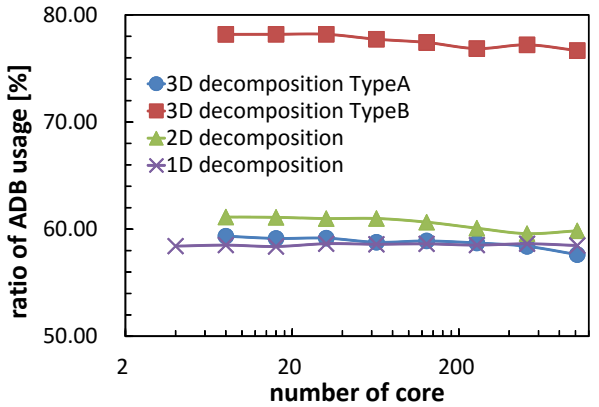

図 4　MHD コードの ADB 利用率
Figure 4　ADB usage ratio of MHD code

表 1　*ny*，*nz* を変更した場合の通信性能と実行性能

Table 1　Performances of communication and execution in changing the size of *ny* and *nz*

nx	*ny*	*nz*	*npx*	*npy*	*npz*	通信要素数				通信処理		GFLOPS/proc (効率:%)
						nx×*ny*	*ny*×*nz*	*nz*×*nx*	Total	[sec]	[%]	
600	25	400	2	8	8	15,000	10,000	240,000	265,000	5.54	25.8	24.01(37.5%)
600	25	400	2	4	16					5.42	25.3	24.09(37.6%)
600	50	200	2	8	8	30,000	10,000	120,000	160,000	4.93	23.6	24.86(38.8%)
600	50	200	2	4	16					4.88	23.5	24.89(38.9%)
600	100	100	2	8	8	60,000	10,000	60,000	130,000	4.05	19.2	24.71(38.6%)
600	100	100	2	4	16					4.14	19.7	24.78(38.7%)
600	200	50	2	8	8	120,000	10,000	30,000	160,000	5.12	23.7	23.95(37.4%)
600	200	50	2	4	16					4.95	23.0	24.14(37.7%)
600	400	25	2	8	8	240,000	10,000	15,000	265,000	7.26	30.4	21.46(33.5%)
600	400	25	2	4	16					7.17	30.0	21.61(33.8%)

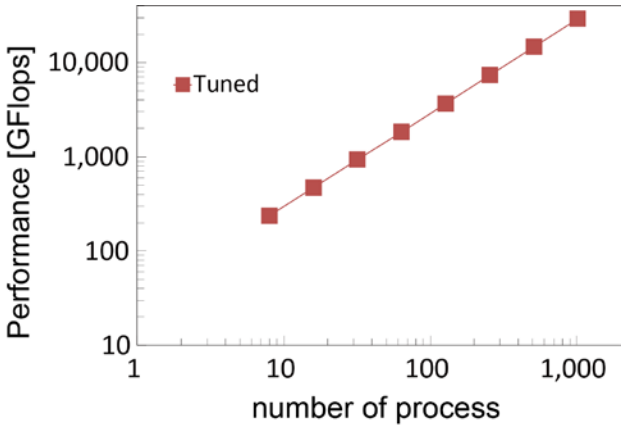

図 5　SX-ACE へ最適化後の MHD コードの性能

Figure 5　Performance of optimized MHD code to SX-ACE

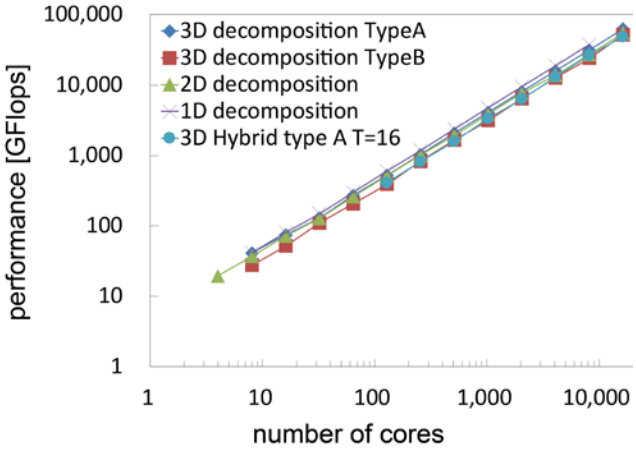

図 6　FX100 での MHD コードの性能

Figure 6　Performance of MHD code on FX100

次に *x* 方向のサイズ変化に伴う性能を調べた結果，*x* 方向のサイズが大きいほど性能が出る傾向があった．これは，1 次元領域分割の性能が高い結果と同じ傾向であり，ベクトル型 CPU の特徴と言える．ただし，そのサイズが 2 のべき乗の場合，バンク競合により極端に性能が劣化することが見えている．競合を避けるため 2 のべき乗より −1 をした，*nx* = 1023 場合に性能が最も出ている．ここでは最大で 44.7 %の実行効率が得られた．また，*ny, nz* を 100 から 200 にすることで性能が少し上がることも確認された（実行効率 45.2 %）

この条件で並列化性能を再度計測してみると，図 5 のようになった．ここでは，プロセス当たりの計算サイズを (1023, 200, 200)として，1024 プロセスまで測定している．今まで並列化性能が下がっていたが，並列化効率 99.997 %を達成することができた．

更なる最適化のため，MHD シミュレーションコードのメインループ解析を行った．コード上では，B/F 値が 3.59 となっているが，ADB を考慮した実際の実行時の B/F 値は 0.53 となっており，実効性能は演算ネックとなっていた（SX-ACE の B/F 値は 1）．MHD コードのメインループ内では，差分計算に伴う加算と乗算か多数出てくるが，それらの命令数の差が大きく，演算バランスが良くない．これにより理論性能がうまく出ていないことがわかり，冗長演算を共通化することで性能が向上することが確認できた．しかしながら，4 %程度の性能向上で有り，この最適化によりコードの保守性が著しく損なわれるため，一般的には薦められない結果となった．

3.2　FX100

名古屋大学の FX100 を利用し，性能評価を行う．この FX100 は 2,880 ノードからなり，ノード当たり 1 つの SPARC64XIfx（32 コアと 2 アシスタントコア，理論性能 1.1 TFlops）と 32 GB の HMC （Hybrid Memory Cube）メモリ（バンド幅 480 GB/s）から構成される．ノード間通信は

表 2　各条件におけるキャッシュミス率とメモリスループット
Table 2　Cache miss rate and memory throughput on four conditions

	L1D ミス率 [%]	L2D ミス率 [%]	メモリスループット [GB/s]
ハイブリッド並列	1.76	0.77	140.4
Flat MPI 並列	0.89	0.54	142.7
Flat MPI 並列 ＋ 配列分割	0.85	0.52	143.7
Flat MPI 並列 ＋ タイリング	0.81	0.44	33.9

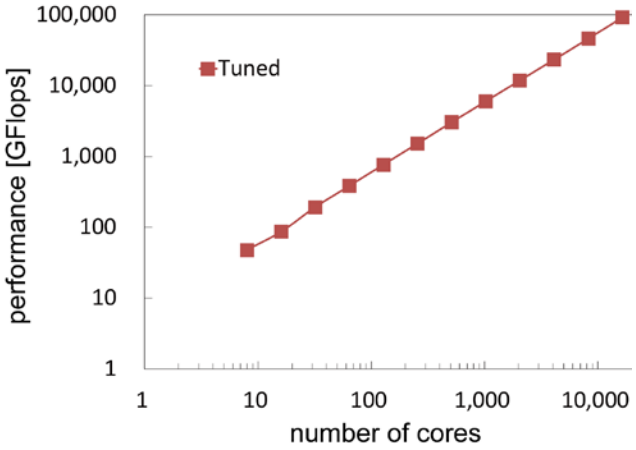

図 7　FX100 へ最適化後の MHD コードの性能
Figure 7　Performance of optimized MHD code to FX100

Tofu2（バンド幅 12.5 GB/s）で接続されている．システム全体での理論性能は，3.2 PFlops を達成している．FX100 での性能評価は Flat MPI 並列とハイブリッド並列を利用し，weak scaling で評価を行っている．計算サイズはプロセス当たり，512 MB（100^3 グリッド）を利用した．

SX-ACE と同様に，3 種類の領域分割と配列の要素を入れ替えた場合の性能評価を行った．SPARC64 系では今までスカラ型 CPU の最適化に効果的な Type B の性能が良かったが，FX100 ではベクトル型と同様の場合に性能が高くなっており，512 ノード（16,384 コア）利用時に 3 次元領域分割 Type A の場合に 62 TFlops となっている．Type B では効率的にキャッシュが利用できているが，SIMD の効率が他の場合よりも低いために，性能が悪くなっていることが FX100 で利用できるプロファイラから分かっている．

このことから FX100 ではベクトル向けの最適化にさらにキャッシュの有効利用を加えるような最適化が必要と考えられるため， 3 次元領域分割 Type A に対していくつかの最適化を行った．メインの配列がキャッシュに載りやすくするために，物理成分毎に配列を分割，計算ループ中での配列要素の同時参照率を上げるためにタイリングを行った．表 2 に最適化前後の詳細 PA からの情報を載せる．最適化前では，Flat MPI 並列の方がハイブリッド並列よりも，キャッシュミス率が低く，メモリスループットも高い．これは図 6 の性能結果と同じだが，本 MHD シミュレーションコードはスレッド間で共有するデータが無いため，キャッシュの共有が無く，Flat MPI よりも効率が悪くなったものと考えられる．配列分割は想定通り効果が見えるが，タイリングを行った場合はキャッシュへの効果はあったが，メモリスループットが大きく下がり，計算性能が悪くなった．タイルサイズをいくつか変更した結果，タイル数が少ないほど性能劣化が少なく，現状の計算サイズでは逆にタイリングが非効率になることが分かった．更に SIMD 向け（ベクトル化）として，*nx* の計算サイズを SX-ACE と同様にし，それに伴うプロセス分割数の調整，計算ループの分割などを行った性能が，図 7 になり，最大で 512 ノード利用時に 91.49 TFlops となった．

4. 他計算機システムとの比較

最適化後の SX-ACE と FX100 が他の計算機システムと比べてどのような性能になっているかを比較するために MHD シミュレーションコードの他計算機システムでの性能を表 3 に示す[5, 6]．CPU（GPU，コプロセッサー）自体の性能を比較しやすいように，CPU 当たりの性能を表に計算している．現在稼働中の計算機システムでは，CPU 当たりの性能が京コンピュータで 27.9 GFlops，FX10 が 48.9 GFlops となっており，3 次元領域分割 Type B が最適となっている．Xeon 系では CX400（Sandy Bridge）が CPU 当たりの性能で京を越えており，HA8000（Ivy Bridge）と XC30（Haswell）は，FX10 より少し性能が落ちている．Xeon 系ではベクトル化が効く 2 次元領域分割や 3 次元領域分割 Type A が最適となっている．これらに比べて SX-ACE は CPU 当たりの性能が 114.0 GFlops で，FX100 は 178.7 GFlops となっており，大きく性能が高いことが分かる．今後は，新しい Xeon Phi（Knights Landing）の理論性能が現行 Xeon Phi の 3 倍であり，実行効率が下がらないのであれば，FX100 の性能を越える可能性がある．また GPU は Kepler 世代でも高い性能を示しており，現行の Pascal 世代の Tesla であればこちらも 3 倍程度の性能が見込まれる．

5. まとめ

SX-ACE と FX100 に対して，宇宙プラズマを解く MHD

表 3　様々な計算機システムにおける性能の傾向[5，6]

Table 3　Performance trend of various computer systems [5，6]

	Core/CPU	Rmax [TFlops]	Rpeak [TFlops]	Rpeak /CPU [GFlops]	Efficiency [%]	Suitable domain decomposition	CPU architecture
SX-ACE	1024/256	65.50	29.20	114.0	45	3D_A	Vector
K	262144/32768	4194.30	914.12	27.9	22	3D_B	SPARC64 VIIIfx
FX10	76800/4800	1135.41	234.59	48.9	21	3D_B	SPARC64 IXfx
FX100	16384/512	576.72	91.49	178.7	17	3D_A	SPARC64 XIfx
CX400	23616/2952	510.11	104.23	35.3	20	3D_A	Xeon (SB)
HA8000	23160/1930	500.26	83.42	43.2	17	2D	Xeon (IB)
XC30	448/32	16.49	1.37	42.8	8	2D	Xeon (HSW)
SR16000/L2	1344/672	25.27	5.38	8.0	21	3D_B	POWER6
Xeon Phi 5120	60/1	1.00	0.08	84.0	8	3D_A	Knights Corner
Tesla K20X	896/1	1.31	0.15	153.3	12	3D_A	Kepler

シミュレーションコードの性能測定を行った．SX-ACE では過去のベクトル機と同様にベクトル長が長い場合に性能が高いが，ADB と呼ばれるキャッシュの効果により，キャッシュ最適化をしたスカラ型 CPU 向けコードでも性能の劣化が少ない．ベクトル長の調整や通信の最適化などを行った結果，SX-ACE において 3 次元領域分割においても高い性能が出ることが分かった．

FX100 では，FX10 や京で最も性能の高かった 3 次元領域分割 type B の性能が最も性能が低く，ベクトル型 CPU 向けのコードの性能が高かった．これはスカラ型 CPU ではあるが，SIMD 最適化が重要で，キャッシュヒットを考慮するよりも，ベクトル化を考慮すると性能が高くなることを示している．最適化として，3 次元領域分割 type A にループや文の分割，プロセス分割の調整，配列分割を行うことでキャッシュミスが減少し，最終的には 16.7%の実行効率となった．

謝辞　本研究の一部は，学際大規模情報基盤共同利用・共同研究拠点，および，革新的ハイパフォーマンス・コンピューティング・インフラの支援による．また，計算機資源を名古屋大学宇宙地球環境研究所「計算機利用共同研究」，および，名古屋大学情報基盤センターの支援による．

参考文献

[1] Top500 Supercomputing Sites. (http://www.top500.org/)
[2] Ogino. T, R. J. Walker, M. Ashour-Abdalla, "A global magnetohydrodynamic simula-tion of the magnetopause when the interplanetary magnetic field is northward", IEEE Trans. Plasma Sci. vol. 20, 1992, 817-828.
[3] F. F. Chen, 1974. Introduction to Plasma Physics. Plenum Press, NY.
[4] Fukazawa, K., T. Ogino, and R. J. Walker (2012), "A Magnetohydrodynamic Simula-tion Study of Kronian Field-Aligned Currents and Aurora", J. Geophys. Res., 117, A02214, doi:10.1029/2011JA016945.
[5] Fukazawa, K., T. Nanri and T. Umeda, "Performance Measurements of MHD Simulation for Planetary Magnetosphere on Peta-Scale Computer FX10", Parallel Computing: Accelerating Computational Science and Engineering (CSE), Advances in Parallel Computing 25, pp.387-394, IOS Press, 2014. (DOI: 10.3233/978-1-61499-381-0-387)
[6] Fukazawa, K., T. Umeda, and T. Nanri, “Performance Evaluation of MHD Simulation Code with X86 CPUs and Manycore Systems”, JSST2016, accepted.