電磁流体コードによる大規模惑星磁気圏シミュレーション

深 沢 圭 一 郎

九州大学大学院理学研究院地球惑星科学部門

梅 田 隆 行、荻 野 瀧 樹

名古屋大学太陽地球環境研究所

１．はじめに

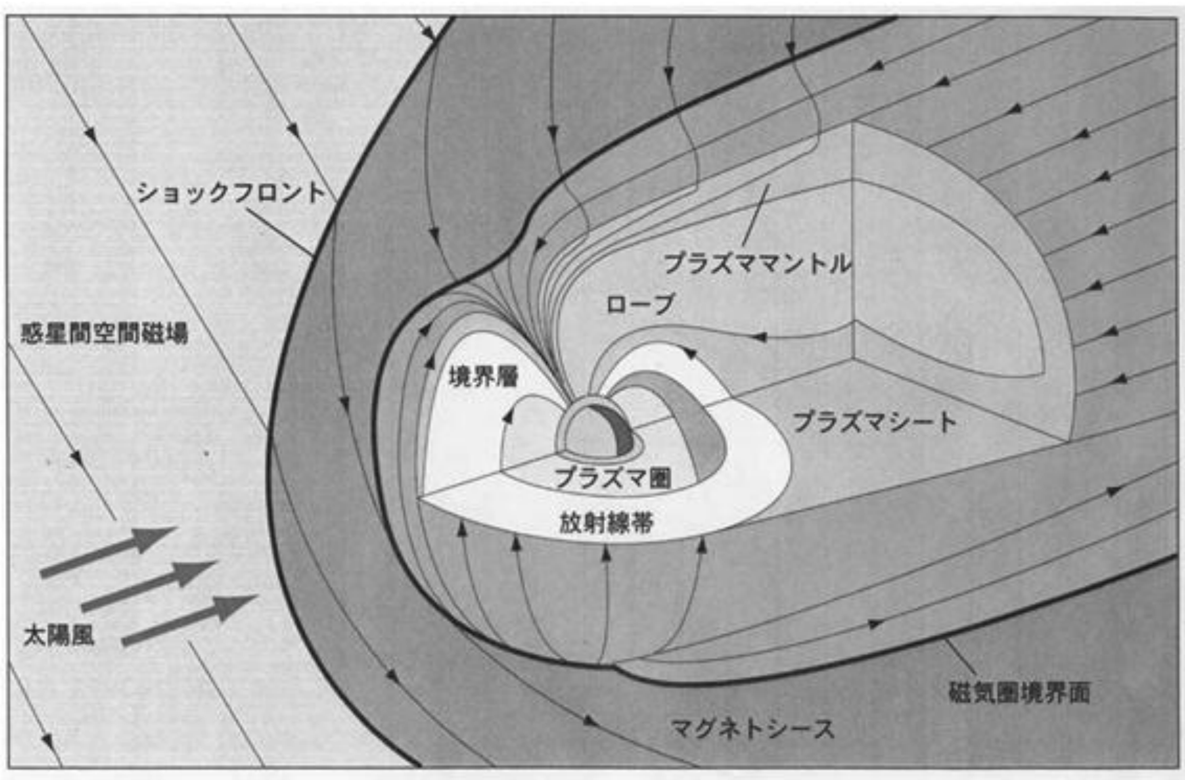
１．１　太陽地球惑星系科学分野の紹介

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。しばしば物質の第４の状態とも呼ばれている。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。宇宙プラズマは導電率が高いため、プラズマは磁力線に沿って動きやすく、また磁力線を横切る動きを取りにくい特徴がある。そのため、太陽風プラズマは太陽の磁場を伴って超音速で吹き出しており、地球のような磁化惑星に衝突すると、その磁場を伴ったプラズマの風が惑星の固有磁場と相互作用する。その結果、惑星磁場が変形し、磁気圏という第１図に示すような形をとる。惑星磁気圏の太陽側は太陽風の圧力により圧縮された形をしており、反太陽側は太陽風によって引き延ばされた形をしている。図の左側から太陽風が流れ込み、磁気圏の前面には、弓形の冠衝撃波(bow shock)が形成され、その内側にはマグネトシースが存在する。磁気圏は磁場構造により、内部磁気圏(中低緯度に根ざす閉じた磁力線からなる領域)と外部磁気圏(高緯度側に根ざす開いた磁力線からなる領域)の２つに分けられる。その内部磁気圏と外部磁気圏の昼側境界にあたるのが、カスプ領域である。ローブ領域は外側磁気圏で開いた磁力線領域であり、希薄なプラズマが存在している。ローブに挟まれた閉じた領域がプラズマシートと呼ばれる部分で地球の極域電離圏と磁力線を通してつながっている。その境界領域はエネルギーが高くなっている。特に、このプラズマシートは、南と北のローブ領域の磁場で囲まれているため、エネルギーが高く、また、地球に向かう高速の流れが存在し、かつ密度粒子が高く、地球の極域電離層で起こる様々な現象の源となっている。磁気圏境界と呼ばれる部分が、地球磁気圏の殻である。その内側において、尾部の子午面では、カスプからの延長のプラズママントルが、赤道面では、低緯度境界層が存在し、これらの領域では、100‐150km/s の粒子の流れが観測されている。この磁気圏は基本的に太陽風プラズマに対するシールドとして働いているが、いくつかの条件下では太陽風プラズマが磁気圏内に侵入することがある。その結果、例えば身近な現象としては極域地方でのオーロラ発光という形で表れる。より詳細な紹介については、参考文献[1]などを参考にしていただきたい。

宇宙プラズマ研究において、我々は主にこのような太陽から吹いてくる磁場を伴ったプラズマの風(太陽風)と地球の磁場が相互作用して起こる様々な現象を研究ターゲットにしている。これらは宇宙空間で起きる現象であるため探査機を打ち上げて観測を行うが、基本的に“その場”の観測しか行えない（立体空間情報を得ることができない)。そのため、3 次元空間構造、

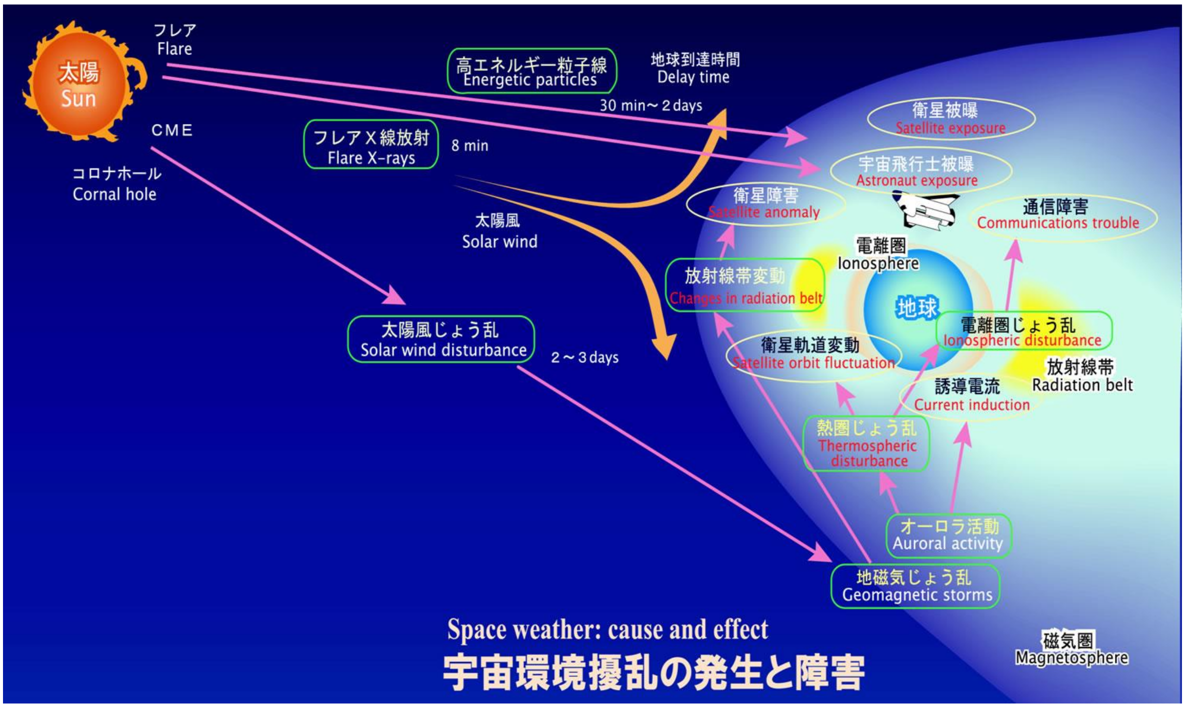
さらにその時間発展などを調べることのできる宇宙プラズマ計算機シミュレーションがこの分野の理論の発展、また観測結果の理解の促進に非常に重要な役割を果たしてきている。

地球周辺の宇宙空間において、前述のような宇宙プラズマに起因する様々な現象を気象にならい、宇宙天気と呼んでいる。その宇宙天気と呼ばれる現象を第2図にまとめた。基本的にこれらの現象は太陽の活動により生じる。例えば、太陽表面でのフレア爆発、コロナから高速にガスの固まりが放出される現象（コロナ質量放出：CME）等がある。それら太陽の活動の結果、身近なものとしては、衛星に障害が生じ、衛星放送に不具合が生じることや、国際宇宙ステーションなどで活動している宇宙飛行士が被爆することなどがある。また磁気圏より地球に近いところに存在する電離圏という領域では太陽活動により、電子密度が変動し、地上でGPS衛星からの電波をうまく受信できないことも起こる。このような現象を引き起こす太陽活動だが、実はその活動度は11年周期で変動している。つまり11年毎に高い活動度（極大期）、低い活動度（極小期）を繰り返している。現在（2009年）は極小期が数年続く珍しい時期に入っており、宇宙天気現象を耳にする日も少ない。しかしながら、今後太陽活動度が上昇していくことが見込まれるので、宇宙天気という言葉を目にする機会も増えるであろう。



第1図：地球磁気圏の構造。

図の左側から超音速の太陽風が惑星間空間磁場を伴って吹いており、それが地球に達すると、地球の前面には衝撃波面が形成される。その衝撃波面より地球側に磁気圏の境界を表す磁気圏境界面が存在し、その中が磁気圏になる。磁気圏は惑星固有磁場の勢力範囲であり、太陽風から地球をシールドする働きを持つ。磁気圏内は、いろいろな領域に分けられており、それぞれ図中にあるような名前が付いている。



第２図：様々な宇宙天気現象（情報通信研究機構提供）。

太陽で起きる、フレア、CME（コロナ質量放出）や太陽風により、様々な現象が地球周辺で起こる。例えば、高エネルギー粒子線であれば、宇宙空間にいる宇宙飛行士に被爆をもたらし、太陽風の擾乱により、磁気圏で擾乱が起こり、それに伴い地球周辺の様々な領域で擾乱が起こり、オーロラ活動が活発になったり、衛星の軌道に影響を与えたりする場合がある。日本の宇宙天気研究、予測は歴史的に独立行政法人情報通信研究機構で行われている。

このような宇宙天気現象は地球だけでなく、磁場を持つ木星や土星でも起こる。第１表に太陽系内の代表的な磁化惑星の特徴を示すが、惑星に固有磁場が存在すると、前述のように太陽風プラズマとの相互作用が起こり、磁気圏が形成される。そのため、木星、土星でもオーロラが観測されている（第３図を参照）。また第１表にあるように木星、土星は高速自転しており、地球とは異なった磁気圏構造をしていると考えられる。木星は巨大な磁場を持ち、更に磁気圏内に大量のプラズマを保持したまま高速自転しているために、磁気圏が遠心力によりディスク状に伸びていると考えられている（第４図参照）。また固有磁場が非常に強力なためその勢力範囲である磁気圏も非常に大きく、土星磁気圏が木星磁気圏の尾部（反太陽側）に入っている観測結果もある。土星はその特徴からよく地球と木星の中間の惑星と言われる。それは、地球程度の磁場しか無く、一方でガス惑星であり、木星と同様に高速自転し、大きさも太陽系では木星の次の規模であることからきている。しかしながら、最近の研究では二つの惑星の中間という特徴ではなく、土星固有の現象、特徴も見つかってきている。

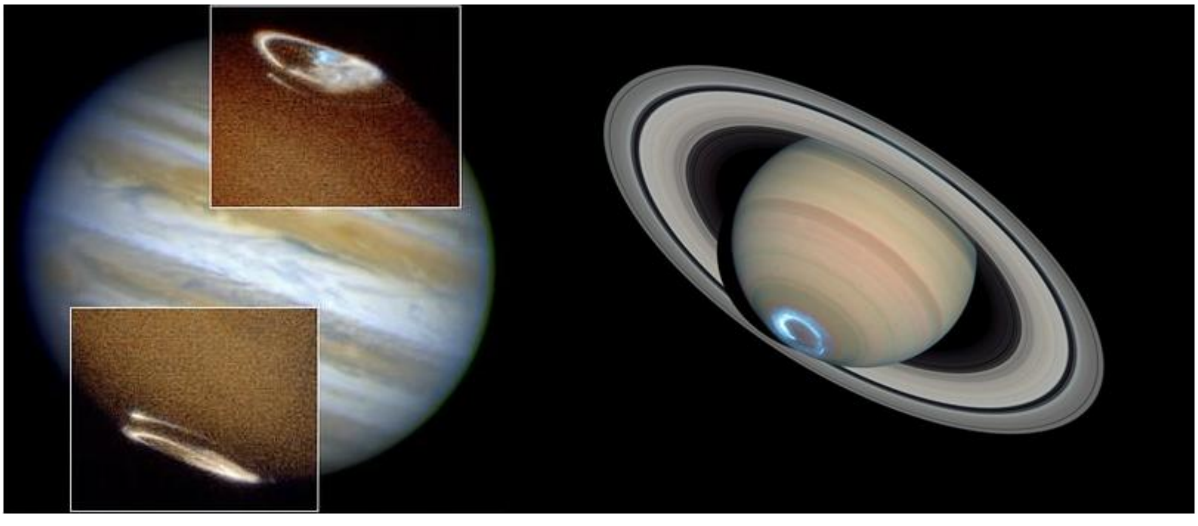
地球においては今までに、また現在もたくさんの衛星が磁気圏を観測し、宇宙天気現象について理解が進んできているが、その他の惑星ではそれほど進んでいない。それでも木星においては、今まで８つの探査機（Voyager 1、2、 Pioneer 10、11、 Ulysses、 Galileo、 Cassini、New Horizon）が観測を行い、その中でもGalileo探査機は1997年から2002年まで木星を周回観測し、さまざまな情報を与えてくれた。我々のグループにおいてもこのGalileo探査機の結果に対するシミュレーションを行い、その構造、現象の理解につなげている[2][3][4]。一方で

土星においては、今までに4機の探査機（Voyager 1、 Pioneer 10、11、Cassini）が観測を行っており、2002年から今現在もCassini探査機が土星を周回観測している。Cassini探査機はGalileo 探査機以上に様々な現象を観測しており、現在我々もその理解のため、シミュレーションを行っている[5][6]。第5図に我々の土星磁気圏シミュレーション結果を載せる。シミュレーション自体は3次元で行っているが、ここでは土星磁気圏赤道面におけるプラズマ対流の磁力線方向に対して垂直（上図)、平行（下図）の渦度を表している。左から右に時間発展を示しており、磁気圏境界面で激しく乱れるプラズマ対流が見て取れる。これは地球、木星磁気圏では見られない構造であり、土星磁気圏の特徴とも考えられている。近年Cassini探査機がこの対流構造に似た観測を行い、シミュレーションの正当性が強くなってきている。

第1表: 地球、木星、土星の特徴。

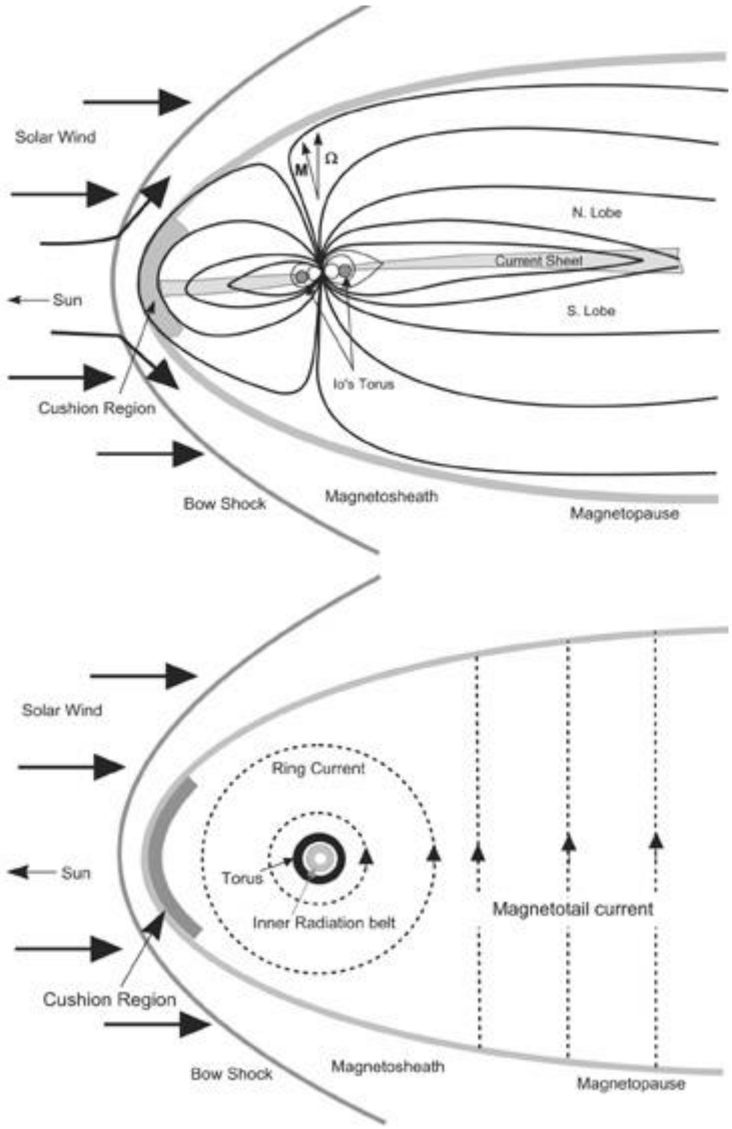
固有磁場は木星が飛び抜けて大きく、地球と土星ではそれほど変わらない。自転速度、赤道半径は木星、土星で同程度であるが、地球に比べると遙かに大きい。プラズマ源は各惑星磁気圏内に広がるプラズマの発生源を示している。イオ、エンケラダスは衛星であり、地球で言うところの月に当たるが、大気を持っている等、その特徴は異なる。木星はおよそ太陽から、地球と比べて5倍以上離れており、土星はさらに9.5倍離れている。

	固有磁場 [nT]	磁極	自転周期 [hr]	プラズマ源	赤道半径 [km]	太陽からの距離 [A.U.]
地球	31,000	N極が南	24	電離圏	6,378	1
木星	420,000	N極が北	10	イオ、電離圏	71,492	5.2
土星	21,000	N極が北	10.65	エンケラダス、電離圏	60,268	9.55



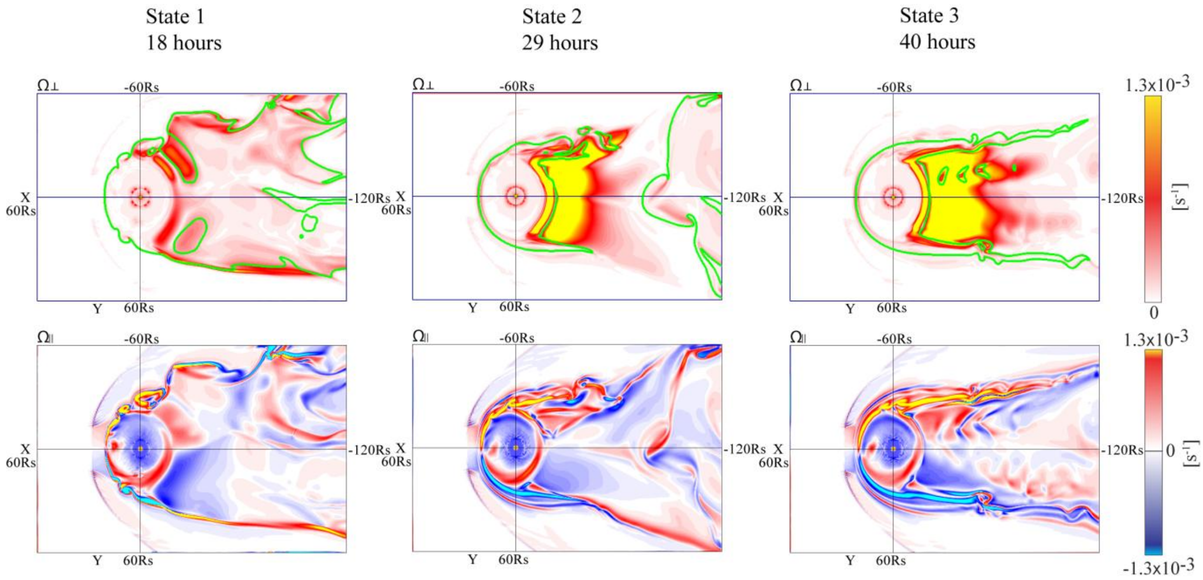
第3図: 木星と土星におけるオーロラ発光[NASA提供]。

左側がハッブル宇宙望遠鏡で撮像された木星におけるオーロラ発光（紫外線領域）の写真であり、右図が土星におけるオーロラ（紫外線領域）の写真である。青白く光って見えるのは紫外線領域での撮像のためである。



第 4 図：木星磁気圏の概略図[7]。

上図が子午面（縦に切った面）における木星磁気圏であり、遠心力によって磁力線が左右に伸びているのがわかる。この構造を磁気ディスクと呼ぶ。下図は赤道面における磁気圏構造を示している。木星の衛星イオから吹き出したプラズマが作るイオトーラスが木星の囲むように輪を描いている。これらは地球にはない木星の特徴である。これらによって木星磁気圏独特の性質ができあがっている。



第 5 図: 土星磁気圏シミュレーション結果[6]。

赤道面における磁場に垂直なプラズマ対流の渦度（上図）、平行な渦度。各図の左側から太陽風が吹いてきている。特に下の図で Y 軸の負側でプラズマ対流が乱れているのが見える。上図内の緑色の線は磁力線が閉じているか開いているかの境界を表す線である。つまり磁気圏境界面を書いている。

１．２　プロジェクトの目標

一般に磁気圏の大規模構造をシミュレーションする場合、プラズマを電磁流体と近似した、電磁流体力学（MagnetoHydroDynamics：MHD）シミュレーションを行う。このシミュレーションでは数値計算的な意味においてスケールフリーであり、解像度は計算機の資源によって決まる。現在我々がシミュレーションでは 600×400×400（×MHD 変数:8）の直行格子を使い、解像度は 0.3R_S（1R_S は土星半径を表す：60,000km）であるが、近年の土星磁気圏シミュレーション結果に表れるプラズマ対流の構造（第 4 図参照）を表すには解像度が足りないことが明らかになっている。そこで、本共同研究プロジェクトでは高精細の土星磁気圏グローバルシミュレーションを行うことを目的としている。最終的な目標としては、0.1R_S の解像度（現在の 1/3）で土星磁気圏を解くことである。前述のように土星磁気圏は地球磁気圏と違い自転速度が早く、惑星付近では磁気圏プラズマが自転とともに回転しており、太陽風との相互作用の結果が準定常状態に達するまでに非常に時間がかかり、意味のある結果を得るためには、シミュレーションを長時間走らすことが必要となる。そのため、計算機の実行効率が非常に重要になってくる。

平成 20 年度共同研究では、シミュレーションを始めるに当たって、いわゆる T2K 型、PC クラスタ型の計算機に対する我々の二つのシミュレーションコードの性能評価を行う。一つは今述べた土星磁気圏グローバル MHD コードであり、もう一つは次（々）世代惑星磁気圏シミュレーションコードと言われる Vlasov コードである。共同研究で使用できる T2K オープンスパコンの資源は 64node、1024core までであるので、その core 数までの性能評価を行う。

MHD 方程式は Vlasov 方程式において速度空間を落とすような近似、更には単一流体近似により導かれる（詳細は次のセクションを参照）。この近似により扱えなくなった現象についてはモデル化されたパラメータを使い、現象論として組み込んでいる。現在磁気圏の大規模構造を調べるには、計算資源から考えても MHD シミュレーションを使うしかなく、またその結果も観測、

理論とよく合う。しかしながら、マクロスケールからミクロスケールに、またはその逆を含んだ現象までも調べるためには、MHD シミュレーションでは不可能であり、将来の計算資源の向上を見込んで、Vlasov シミュレーションによる新しい磁気圏コードの開発が行われている。

これら両コードともにベクトル機である地球シミュレータ、NEC SX シリーズ、またスカラ機である富士通 HPC2500 でも最適化により、良い性能を示しており[8]、本稿では、その性能評価について詳しく述べる。

２．磁気圏シミュレーションモデルの概要と特徴

２．１　電磁流体方程式

宇宙プラズマの密度はとても低いために、その平均自由行程が非常に長くなる。例えば、太陽プラズマの平均自由行程は１天文単位（太陽と地球の距離）にも達する。そのため宇宙プラズマは基本的に衝突が無いと見なされる。その無衝突プラズマの振る舞いは以下の Vlasov（無衝突 Boltzmann）方程式によって記述される。

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

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。電磁場の時空間発展は以下のように Maxwell 方程式によって記述される。

$$\begin{aligned}
&\nabla\times\vec{B} = \mu_0\vec{J} + \frac{1}{c^2}\frac{\partial \vec{E}}{\partial t}\\
&\nabla\times\vec{E} = -\frac{\partial \vec{B}}{\partial t}\\
&\nabla\cdot\vec{E} = \frac{\rho}{\varepsilon_0}\\
&\nabla\cdot\vec{B} = 0
\end{aligned} \tag{2}$$

ここで $\vec{J}$ は電流密度、ρ は電荷密度、μ_0 は真空中の透磁率、ε_0 は真空中の誘電率、c は光速を示す。電流密度 $\vec{J}$ は帯電した粒子の動きによって以下のように記述できる。

$$\vec{J} = \sum_s q_s \int f_s\vec{v}\,d\vec{v} \tag{3}$$

また、電流密度 $\vec{J}$ は以下の電荷保存則を満たしている。

$$\frac{\partial \rho}{\partial t} + \nabla\cdot\vec{J} = 0 \tag{4}$$

ここまでにあげた式が無衝突プラズマの第一原理の方程式である。

次に電磁流体力学（MHD）方程式を求めていく。MHD 方程式は Vlasov 方程式(1)の 0 次、1 次、2 次のモーメントをとり、運動論的効果を無視することで得られる。まず Vlasov 方程式の 0 次のモーメントをとる（Vlasov 方程式(1)を速度空間について積分する）と連続の式が求まる。

$$\frac{\partial n_s}{\partial t} + \nabla \cdot (n_s \vec{u}_s) = 0 \qquad (5)$$

$\vec{u}_s$ は各プラズマの平均速度を表す。次に Vlasov 方程式(1)に $m\vec{v}$ を掛けてから速度空間で積分する（1 次のモーメントをとる）と、運動方程式が求まる。

$$\frac{\partial}{\partial t}(m_s n_s \vec{u}_s) + \nabla \cdot (m_s n_s \vec{u}_s \vec{u}_s + \vec{P}_s) - \rho_s \vec{E} - \vec{J}_s \times \vec{B} = 0 \qquad (6)$$

$\vec{P}_s$ は圧力テンソルを表す。更に、粒子の運動エネルギー $\frac{1}{2}m\left|\vec{v}\right|^2$ を Vlasov 方程式(1)に掛けて速度空間で積分する（2 次のモーメントをとる）と、エネルギー方程式が求まる。

$$\frac{\partial}{\partial t}\left(\frac{1}{2} m_s n_s \left|\vec{u}\right|_s^2 + \frac{3}{2} p_s\right) + \nabla \cdot \left(\frac{1}{2} m_s n_s \left|\vec{u}\right|_s^2 \vec{u}_s + \frac{3}{2} p_s \vec{u}_s + \vec{P}_s \cdot \vec{u}_s + \vec{h}_s\right) - \vec{E} \cdot \vec{J}_s = 0 \qquad (7)$$

p_s は $p_s \equiv \frac{1}{3}\sum_{i=1\sim 3} P_{i,i,s}$ で定義される圧力である。$\vec{h}_s$ は熱流束密度である。ここで以下の操作を行って、単一流体近似をこれらの流体方程式に適応すると、

$$\sum_s m_s n_s \equiv \xi、\ \frac{\sum_s m_s \vec{u}_s}{\sum_s m_s} \equiv \vec{U}、\ \sum_s \rho_s \equiv 0、\ \sum_s \vec{J}_s \equiv \vec{J} \qquad (8)$$

さらに、いくつかのテンソル計算を行って、以下の式を得る。

$$\frac{\partial \xi}{\partial t} + \nabla \cdot (\xi \vec{U}) = 0 \qquad (9)$$

$$\xi \frac{\partial \vec{U}}{\partial t} + \xi (\vec{U} \cdot \nabla) \vec{U} + \nabla p - \vec{J} \times \vec{B} = 0 \qquad (10)$$

$$\frac{\partial p}{\partial t} + (\vec{U} \cdot \nabla) p + \gamma p \nabla \cdot \vec{U} = 0 \qquad (11)$$

ここで $\gamma = \frac{5}{3}$ は比熱比であり、等方対角テンソル $\sum_s \vec{P}_s \equiv pI$ （I は単位テンソル）と仮定したため、熱流束密度は無視できる。更に、磁場凍結条件により、

$$\vec{E} + \vec{U} \times \vec{B} = 0 \qquad (12)$$

が求まり、Darwin 近似を用いて、

$$\nabla \times \vec{B} = \mu_0 \vec{J} \qquad (13)$$

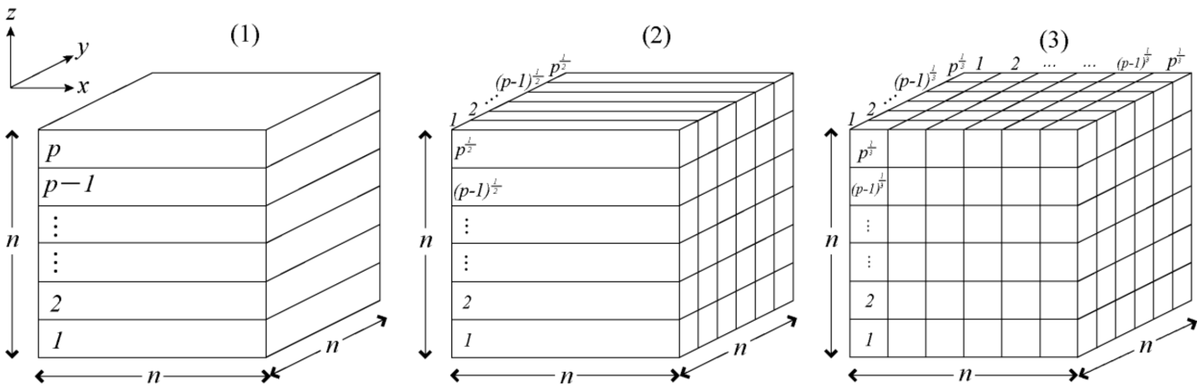
が求まる。これら(9)～(13)式が MHD 方程式となる。より詳細な導出は参考文献[9]を参考にされたい。

２．２　シミュレーションモデル

２．２．１　MHD モデル

MHD 方程式を解く数値計算法としては、Ogino et al. [10]によって開発された Modified Leap Frog 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。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、２次精度の中心空間差分を採用するとき、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。Modified leap-frog 法は、two step Lax-Wendroff 法の数値的安定化効果を一部取り入れて、leap-frog 法の数値的減衰と分散の小さい効果をより多く取り入れた、数値的減衰と分散にバランスの良くとれた一種の組み合わせ計算方法となっている。また、パラメータを変化させることによって、性質の良く分かった２つの計算方法に一致させることができるので、結果に与える数値誤差の影響も理解し易い利点を持っている。更にこの手法を用いた計算で、今まで様々な計算機で性能評価を行ってきたこともあり、同様の手法をもちいることで、過去の結果と比較できる利点もある。



第 6 図: 3 種類の領域分割法。

左から 1 次元領域分割、2 次元領域分割、3 次元領域分割の概要図を示す。n^3 の配列を並列 p で分割している。全並列数を p としているため、2 次元領域分割では各次元で $p^{1/2}$ 並列、3 次元領域分割の場合、x, y, z 方向に $p^{1/3}$ 並列を適用している[11]。

並列化には MPI を使用する。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、第 6 図に示すように、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。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。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(T_{S1}, T_{S2}, T_{S3})と通信時間(T_{C1}, T_{C2}, T_{C3})は大まかに次の様に見積もることができる。

i)1 次元領域分割

$$T_{S1} = \frac{k_1 n^3}{p} \text{、} T_{C1} = k_2 n^2 (p-1) \tag{14}$$

ii)2 次元領域分割

$$T_{S2} = \frac{k_1 n^3}{p} \text{、} T_{C2} = 2k_2 n^2 (p^{\frac{1}{2}} - 1) \tag{15}$$

iii)3 次元領域分割

$$T_{S3} = \frac{k_1 n^3}{p} \text{、} T_{C3} = 3k_2 n^2 (p^{\frac{1}{3}} - 1) \tag{16}$$

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 は総並列数である。ここでは簡単のため、領域分割は第 5 図に示すように x, y, z 方向に同じ数(n)の配列を使用し、

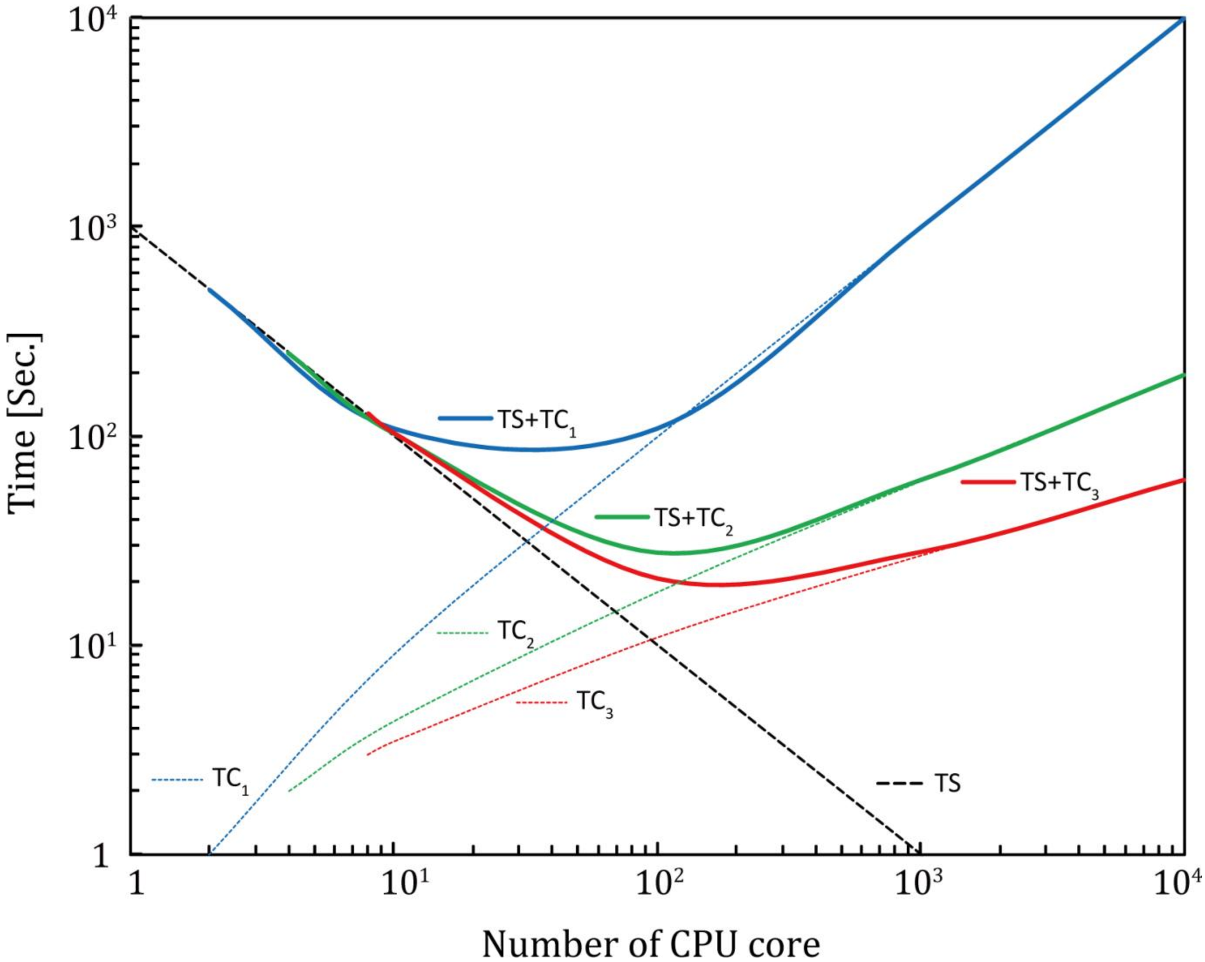
各次元を並列化する場合は等しい並列化数を設定している。計算時間と通信時間の和が並列計算に要する時間と考えられる。これらの式を第７図にグラフで示した。式、図より明らかに計算時間 T_{S1} T_{S2} T_{S3} は並列数 p に反比例して短くなるが、通信時間 T_{C1}, T_{C2}, T_{C3} は p の増加に伴って長くなる。 しかし、その通信時間の長くなる様子は、T_{C1}, T_{C2}, T_{C3} によって大きく異なる（第７図参照)。即ち、３次元領域分割が最も通信時間を短くでき、また、１次元と２次元領域分割の間でも通信時間の差は大きくなることが理解できる。ただし、この比較では簡単のため、通信時間を決める係数 k_2 が同じであると仮定した。これは通信部分のプログラムの工夫によりある程度小さくすることが可能である。こうして、スカラ並列機では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。

スカラ機で性能を出す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データアクセス時に、その前後含めて数 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 u(i,j,k,m) と定義し（Type A)、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ので、一般に u(m,i,j,k,) と定義した方がキャッシュヒット率は上がると考えられる (Type B)。そのため、本性能評価においてもこの配列定義を使った評価も行う。

２．２．２　Vlasov モデル

今回の性能評価では Umeda et al. によって開発された 5 次元 Vlasov コード[12]を使う。5 次元のうち２次元は位置空間 (x, y) であり、３次元は速度空間 (v_x, v_y, v_z) である。この Vlasov コードは 4 次精度の無振動、正値性のある、保存型スキームであり、多次元保存型 semi-Lagrangian 法の１種である。

Vlasov モデルでは７つの物理変数を取り扱う。分布関数 F、電場（Ex, Ey, Ez）と磁場（Bx, By, Bz）である。電荷密度 ρ と電流密度（Jx, Jy, Jz）も使用するが、これらは分布関数のモーメントを取ることで得られる。この分布関数の計算上の量は電場、磁場の比べて非常に大きい。この Vlasov コードでは電磁場の配列定義には Type B の定義法を使用し（u(m, nx, ny))、分布関数ではモーメントを計算するとき（$\sum v_x \sum v_y \sum v_z F$）にキャッシュヒット率が最大になるような定義を用いた（F(nvx, nvy, nvz, nx, ny))。このため２次元領域分割を今回は適用した。一方で速度空間はモーメント計算時間の短縮のため、分割は行わなかった。



第７図:並列数に対する各領域分割における計算時間の変化[11]。

式(14)～(16)によって描かれたグラフ。横軸は並列数、縦軸は時間を示す。点線はそれぞれ計算時間、通信時間を示し、実線が各領域分割にかかる時間（T_S+T_C）を示している。ここでは簡単のため、k_1=1、k_2=0.01 としている。k_n の値に依るが、ある並列数から、計算時間の上昇が見られる。しかしながら、3 次元領域分割の場合、その上昇を最小限に抑えられている。

３．性能評価結果

今回 T2K オープンスパコンでは 3 つの Fortran Compiler が利用できたので、その中から日立最適化 Fortran Compiler と Intel Fortran Compiler Ver. 11.0 を使用した。PGI 製コンパイラではうまく最適化する時間が足りなかった。コンパイラオプションとしては、日立コンパイラには、

```
-Oss -noparallel -autoinline=2
```

を使い、Intel コンパイラには、

```
-O3 -msse3 -xSSE3 -ipo
```

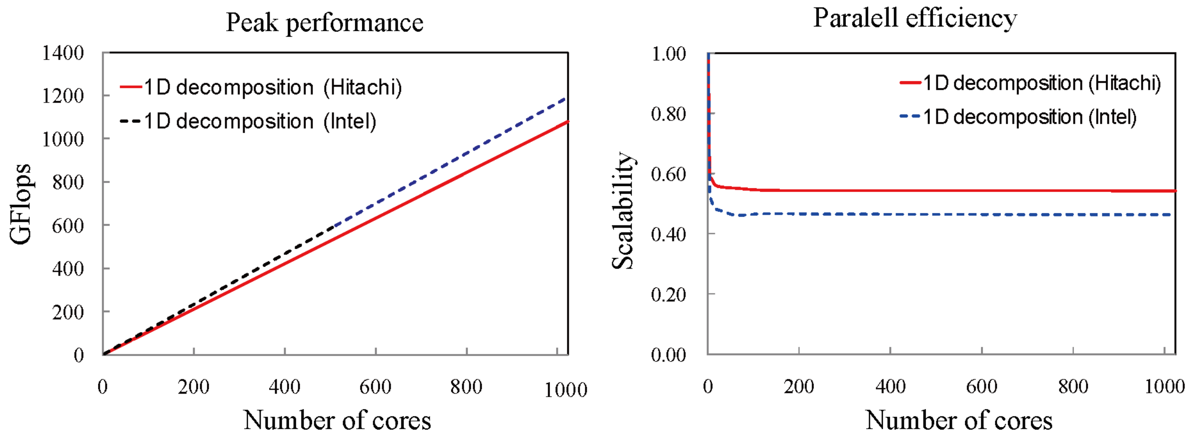
を使用した。これらからわかるように並列化はすべて MPI だけで行った。ここで Intel コンパイラは SIMD Extensions (SSE)をサポートしているが、日立コンパイラはサポートしていない。

並列化時の通信において、通信時間を最小限にするために、すべての境界値を入れるためのバッファー配列を用いた。また、実際のMPIで通信する際には“MPI_sendrecv”を用いた。これは送受信を一括で行う関数になり、送受信に伴うプロセスが一つで済む。例えば、MPI_send/MPI_isend”と”MPI_recv/MPI_irecv”などを使用するとそれぞれに1プロセスが必要となる。さらに、blockingとnon-blockingのsend/receiveがよく大容量通信時にバッファオーバーフローするのに対して、MPI_sendrecvはほとんどのスーパーコンピュータシステムにおいて最適化されており、安定した大容量通信が行える。

３．１　MHDシミュレーション

MHDシミュレーションでは64MB/core（1GB/node）サイズの配列を計算するが、MHD方程式をModified LeapFrog法で解くためのワーク配列として192MB/core（3GB/node）を追加で使用する。ここでは、1次元、2次元、3次元領域分割の結果を順に述べる。まず、1次元領域分割の結果について、述べる。基本的に1次元領域分割は、小並列のベクトル計算機に最適な並列化手法と言われている。ベクトル化のためにdoループを長くとる取る必要があったためと、並列化数が少ない分、通信コストも低いので、次に述べる2次元、3次元領域分割の場合のように通信用のバッファーにデータを集める動作の方が、コストがかかるからである。この手法はFlat MPIを使う上では超並列計算には対応していない（並列化される次元の配列数を並列化数が越えてしまう場合）が、自動並列やOpenMPを併用することで、超並列計算に対応することが可能である。ただし、性能が出るかどうかは現状では不明瞭である（確実なデータが無い）。そのため1次元領域分割は昔のベクトル計算機（富士通VPP5000、NEC SX-6など）によく使われていた手法で、今回1次元領域分割の評価に使用した我々のコードベクトル計算機向けに最適化されたコードでもある。

第8図に1次元領域分割の結果を載せる。左側の図が並列化数（使用core数）に対する実効性能を表しており、右図が並列化効率を示している。赤色の実線が日立製コンパイラを使用した結果を示し、青い破線がIntel製コンパイラを使用した結果を示している。実効性能は図を見ると明らかなように、グラフが直線を描いており、並列化数に対する理想的な性能向上を示している。最終的な実効性能としては、1024core（64node）使用時に日立製コンパイラで1,077.8GFlopsを達成し、Intel製コンパイラでは1,192.8GFlopsを達成した。このときの理論性能に対する実効性能の割合を表す実行効率は日立製コンパイラで11%、Intel製コンパイラで13%であった。このようにそれぞれ最高で実効性能1TFlops（1,000GFlops）、実行効率10％以上の結果になった。コンパイラでは常にIntel製コンパイラの方が良い実効性能を出していた。一方、右図の並列化効率を見てみると、日立製コンパイラの方がIntel製コンパイラよりも高い効率を出している。両コンパイラの結果ともに4並列で一気に効率が下がり、16並列以上では効率は変わらず、図においても横一直線になっている。効率自体は1024core使用時に日立製コンパイラで約56％、Intel製コンパイラで約46％となっている。

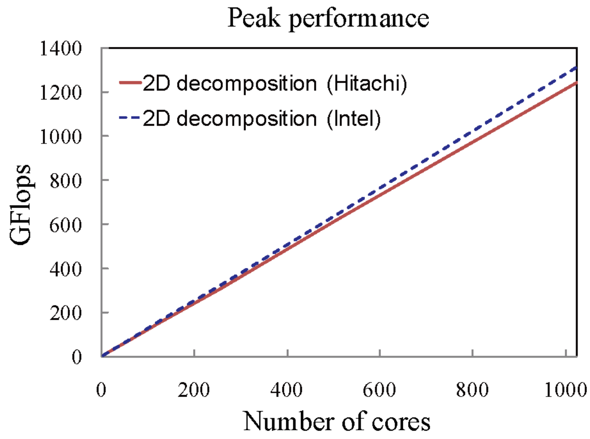


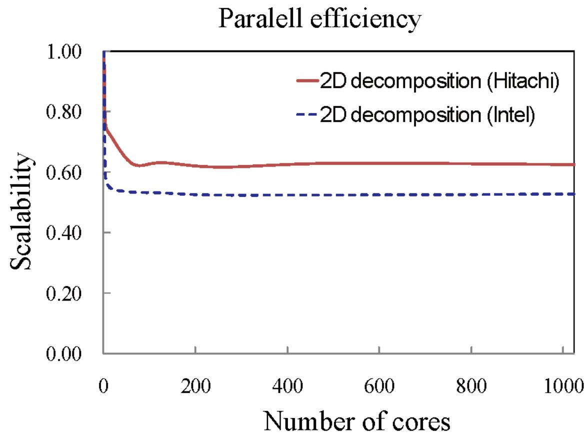
第 8 図:並列数に対する 1 次元領域分割の実効性能と並列化効率[11]。

左図が実効性能を表し、横軸が並列化数（使用 core 数)、縦が GFlops 値を表す。右図は並列化効率を表し、横軸が並列化数（使用 core 数）で縦軸がスケーラビリティを示す。赤色の実線が日立コンパイラの結果を示し、青色の破線が Intel コンパイラの結果を表す。実効性能としては Intel コンパイラの性能が良いが、並列化効率では日立コンパイラに劣る。

次に 2 次元領域分割の結果を述べる。2 次元領域分割は 1 次元領域分割と同様にベクトル計算機向けの手法と考えられているが、特に並列化数の多いベクトル計算機向けである。理由は 1 次元領域分割の場合とベクトル化向けにある次元向けの do ループを長くとることができることと、並列化数が多くなりすぎて、通信コストが無視できなくなっているからである。この場合には、並列化プロセスで通信データをまとめて、各プロセス間で一気に通信する方がコストを低くできると考えられる。この手法は地球シミュレータで特に有効で、2 次元領域分割の評価に使うコードは地球シミュレータで最適化されたコードでもある。

2 次元領域分割の評価結果を第 9 図に載せる。基本的に図の書式は第 7 図と同じであり、左図が実効性能を表し、右図が並列化効率を表している。まず、実効性能だが、ここでも 1 次元領域分割の結果と同様にきれいな右肩上がりの直線を示しており、並列化数の上昇に対する良い性能上昇を表している。1024core における実効性能としては、日立製コンパイラで 1,241.6GFlops を達成し、Intel 製コンパイラで 1,313.0GFlops を達成した。このときの実行効率は日立製コンパイラで 13%、Intel 製コンパイラで 14%となった。これらの結果は日立製コンパイラ、Intel 製コンパイラのなかでそれぞれ最高の結果である。また、1024core 時の日立製コンパイラ、Intel 製コンパイラの性能差が、1 次元領域分割の場合に比べて、少なくなっている。次に並列化効率だが、これも 1 次元領域分割と同様に日立製コンパイラの方が Intel 製コンパイラよりも良い効率を出した。基本的に並列化効率の下がり方などの変化は 1 次元領域分割と似ており、4 並列ですぐに性能が落ち、その後はフラットな直線を描いている。ただし、並列化効率自体は各コンパイラで 1 次元領域分割よりも 10%程度上昇しており、1024core 使用時に日立製コンパイラで 62%、Intel 製コンパイラで 53%の効率が出た。これらも今回の評価のなかで最高の結果である。





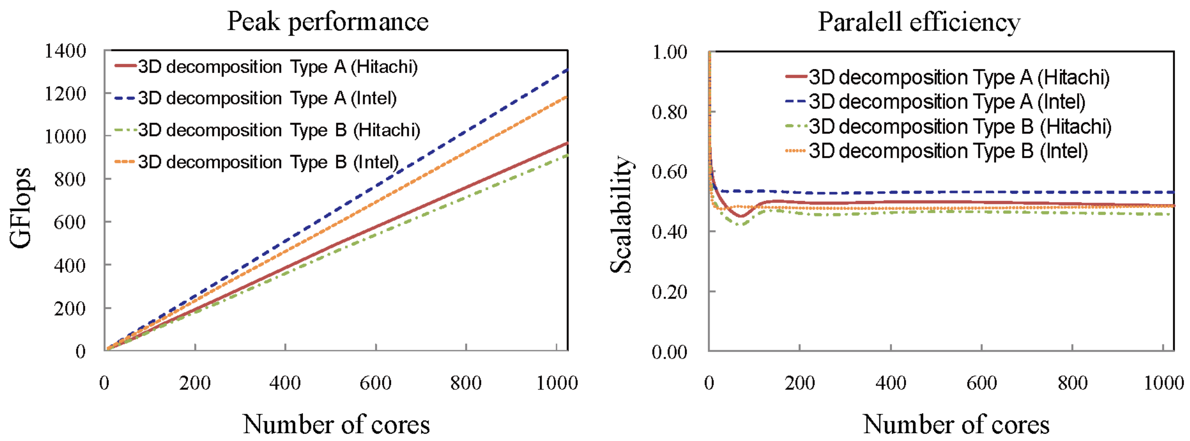
第 9 図:並列数に対する 2 次元領域分割の実効性能と並列化効率[11]。

左図が実効性能を表し、横軸が並列化数（使用 core 数)、縦が GFlops 値を表す。右図は並列化効率を表し、横軸が並列化数（使用 core 数）で縦軸がスケーラビリティを示す。赤色の実線が日立コンパイラの結果を示し、青色の破線が Intel コンパイラの結果を表す。両コンパイラの結果でも並列化効率が他の領域分割よりも良い結果になっている。

次に 3 次元領域分割の性能評価について述べる。3 次元領域分割は、1 次元、2 次元領域分割と比べて、並列化数の上昇に対する通信コストが低いために、超並列計算機向けの手法と言われる。3 次元領域をすべて並列化するために、基本的にはスカラ計算機向けの手法である。われわれは、一昔前にベクトル計算機からスカラ計算機（富士通 HPC2500）へのシフトを経験し、そのときに 3 次元領域分割による最適化を行い良い性能を得た。また、最近ではプレペタスケールコンピュータとも言われる富士通 FX1、日立 SR11000、SR16000 においても 3 次元領域分割で良い性能評価結果を得ている。これらはスカラ計算機ではあるが、T2K オープンスパコンのような一般的な x86 系のプロセッサーを使った PC クラスタタイプの計算機ではないので、同じ傾向が出るとは限らない。

第 10 図に 3 次元領域分割の結果を載せる。前述のように 3 次元領域分割では二つの配列定義を用いたため、図中に 4 つの結果が並んでいる。Type A の配列定義は f(i, j, k, m)の順番であり、Type B の配列定義は f(m, i, j, k)と MHD 変数を示す m を配列の初めに持ってきている。図の書式自体は第 7 図、第 8 図と同様であり、左図が実効性能、右図が並列化効率を示す。まず、実効性能を見ると、Intel 製コンパイラの Type A、Type B の方が、日立製コンパイラの結果より、明らかに性能が出ている。この性能差は前述の 1 次元、2 次元領域分割よりも大きい。実効性能自体は、1024core において、日立製コンパイラでは 1,000GFlops に達せず、Type A で 965.2GFlops（実行効率 10%)、Type B で 908.4GFlops（同 9.6%）であった。一方 Intel 製コンパイラでは、Type A で 1,306.6GFlops（同 14%)、Type B で 1,183.5GFlops（12.6%）を達成した。Intel 製コンパイラでの Type A は 2 次元領域分割の結果と同程度であり、最高の性能を出している。また、配列定義の結果に注目してみると、日立、Intel 製の両コンパイラで Type A の結果が Type B を上回っている。この結果は今までのスカラ計算機(HPC2500、FX1、SR11000、SR16000）とは異なる結果であった。この手法でのキャッシュチューニングは T2K オープンスパコン HA8000 には適さないようだ。並列化効率は並列化数増加に対する変化の様子は、1 次元領域分割、2 次元領域分割と同様であった（すぐに効率が下がり、その後はフラット)。Intel 製

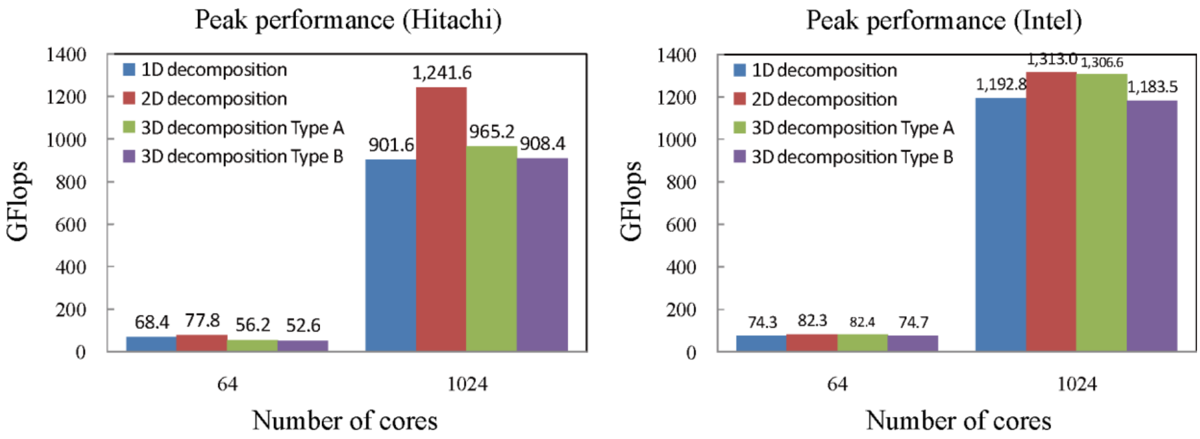
コンパイラで Type A の場合以外は、並列化効率が他の領域分割と比べて良くない。



第 10 図:並列数に対する 3 次元領域分割の実効性能と並列化効率[11]。

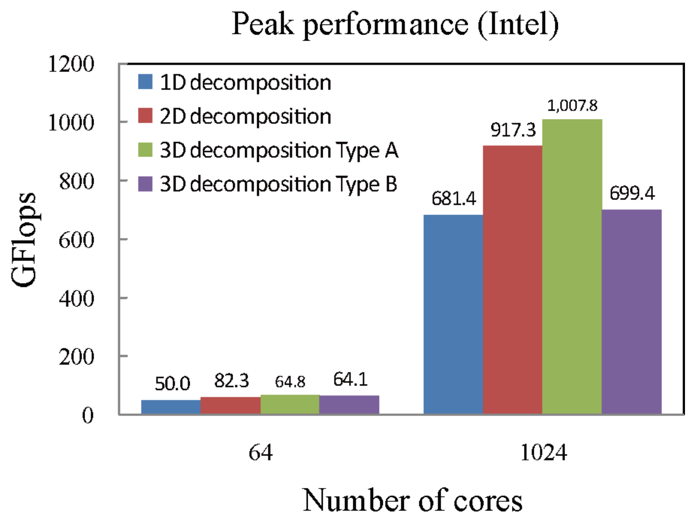
左図が実効性能を表し、横軸が並列化数（使用 core 数）、縦が GFlops 値を表す。右図は並列化効率を表し、横軸が並列化数（使用 core 数）で縦軸がスケーラビリティを示す。ここではキャッシュヒットを考えて、2 種類の配列定義をテストした。Type A が 1 次元、2 次元領域分割と同じ配列定義（f(i,j,k,m)）であり、Type B は MHD 変数を配列の 1 次元目に移動させた定義（f(m,i,j,k,)）になる。他のスカラー計算機と比べて、この性能評価では Type B の結果があまり良くない。

ここで、64core と 1024core の結果を使い、各領域分割の性能を第 11 図に載せ、比べてみる。左図では日立製コンパイラ、右図では Intel 製コンパイラの結果を載せている。まず、全体を見て明らかなように、2 次元領域分割が日立製コンパイラでも Intel 製コンパイラでも良い性能を出しており、HA8000 には 2 次元領域分割が適しているとわかる。特に日立製コンパイラの結果では、2 次元領域分割以外は明らかな性能下降が見られる。一方 Intel 製コンパイラでは、全体的な差は少なく、その中では 2 次元領域分割、3 次元領域分割の Type A が良い性能を出していることが分かる。Intel 製コンパイラの結果が全体的に良い理由は SSE が有効に使えていることが考えられる。事実、SSE を有効にしない場合の結果は、全体的に日立製コンパイラより 1%程度実行効率が悪く、3 次元領域分割の Type A でやっと 1TFLops を越える程度であった（第 12 図参照）。また、Type B の性能が良くない理由はキャッシュサイズに問題があると考えられる。今までに Type B により良い結果を得た計算機は基本的に 1core に対して大きな 2 次キャッシュ（2MB 以上）を有している。一方で HA8000 の CPU である AMD Opteron 8356 は L2 が 512 KB/core、L3 が 2 MB/cpu つまり、512KB/core である。このため、計算上使用する MHD 変数分の配列がうまくキャッシュに収まらないと考えられる。



第 11 図:各領域分割の実効性能。

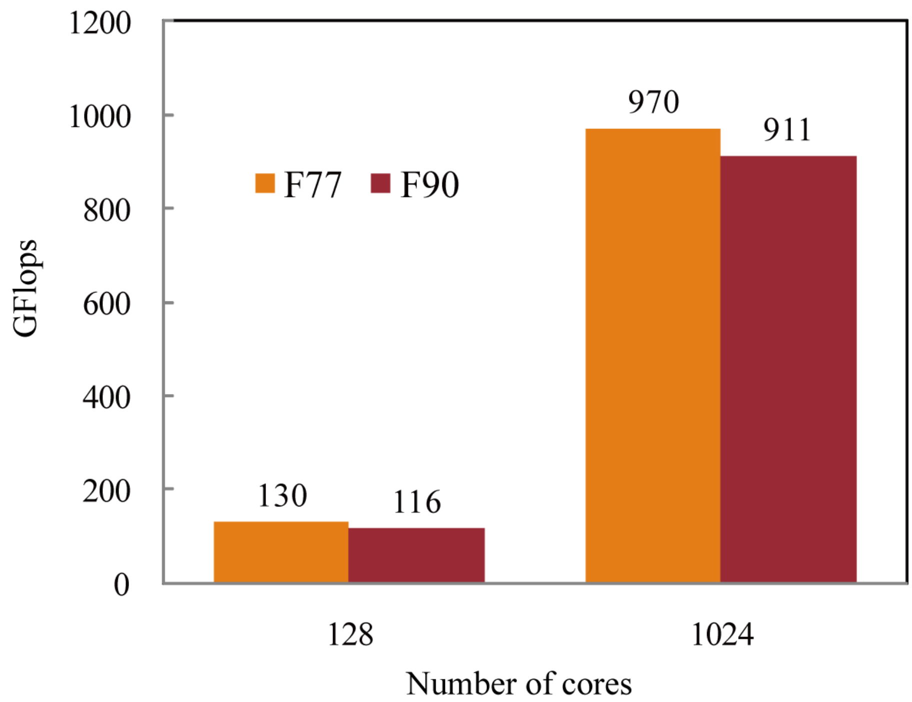
左図が日立コンパイラの実効性能を表し、横軸が並列化数（使用 core 数）、縦が GFlops 値を表す。右図は Intel コンパイラの結果を表し、書式は左図と同じである。Intel コンパイラは全体的に性能が出ているが、日立コンパイラでは 2 次元領域分割以外はあまり性能が出ていない。



第 12 図:SSE を無効にした各領域分割の実効性能。

横軸が並列化数（使用 core 数）、縦が GFlops 値を表す。コンパイラオプションにより、SSE を有効にしない場合、Intel 製コンパイラの性能は著しく劣化する。

ここからは、MHD コードの性能評価時に調べたいくつかの結果を紹介する。まず、Fortran のバージョンで性能が変わるかを調べた。よく Fortran77 の方が、Fortran90 以降に比べて、シンプルな分コンパイラが最適化しやすく、性能が出るといわれる。最近ではプログラムのメンテナンスなどの理由により、Fortran90 を使われる人が多いが、われわれのコードで違いが出るかを調べた。第 13 図にその結果を載せる。これは日立製コンパイラを使った、1 次元領域分割の結果であるが、1024core 使用時は明らかな（誤差範囲ではない）差が見えている。おおよそ 6%ほど Fortran77 の方が良い性能を出している。今回の性能評価全体では Fortan90 を使ったが、最高の性能を出すには Fortran77 を使用する方が良いと考えられる。

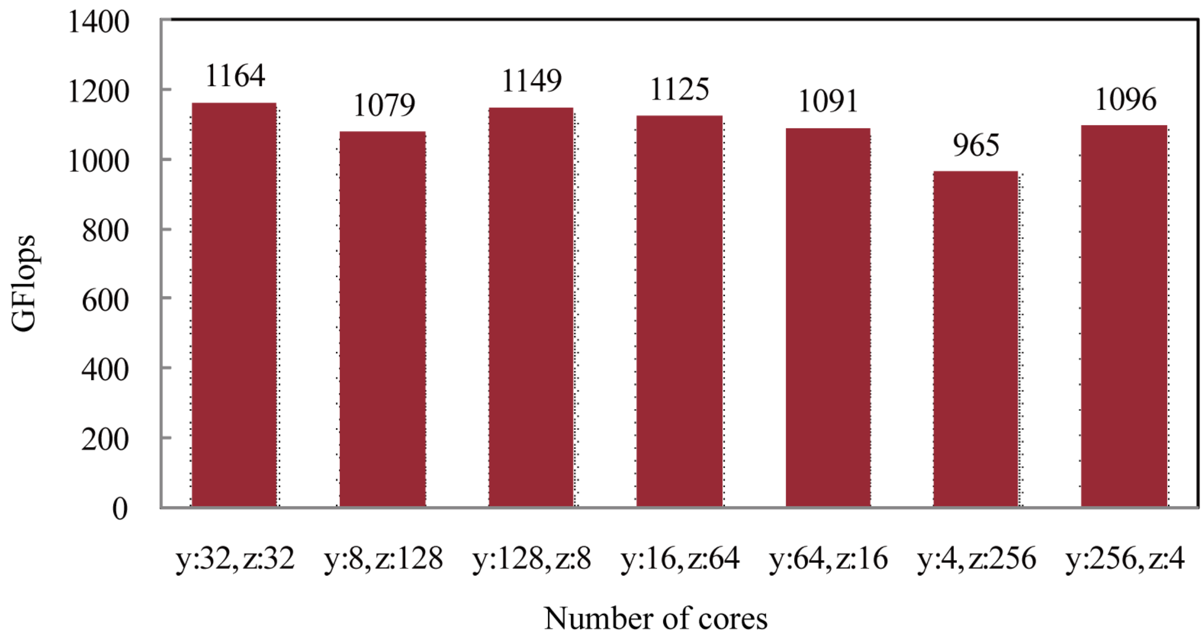


第 13 図:Fortran77 と Fortran90 の性能比較。

オレンジが Fortran77 の結果であり、赤が Fortran90 の結果である。左図が実効性能を表し、横軸が並列化数（使用 core 数)、縦が GFlops 値を表す。これは日立コンパイラの結果である。少しではあるが、Fortran77 の結果が良い性能を示した。

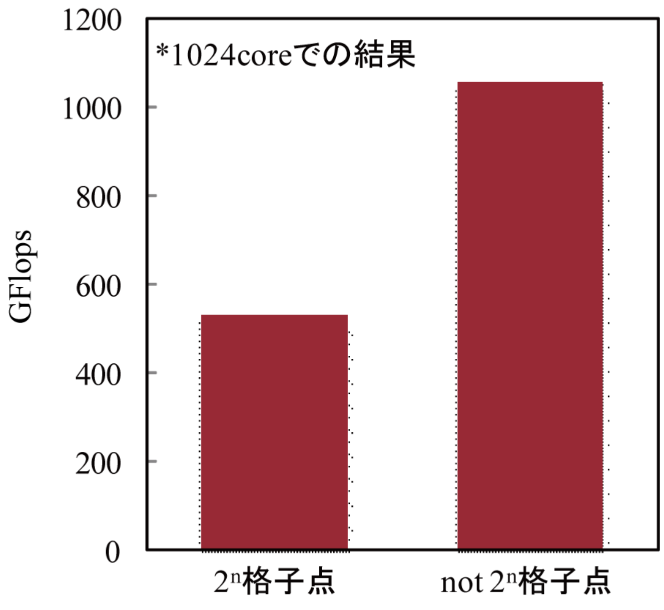
次に 2 次元領域分割における各次元の変列数を変化させた場合、どのように性能が変わるのかを調べた。第 14 図に 1024core 使用時の y 方向と z 方向の並列化数を変化させた場合の結果を示す。これらはすべて日立製コンパイラの結果になる。図を見て気付くのは、z 方向に 256 並列を行った場合の性能劣化である。Y 方向に 256 並列させた場合に比べて 100GFlops 程度性能が下がっている。Z 方向に 128 並列の場合もそれほど良い性能ではなく、極端に z 方向に並列化数を増やすと性能が落ちる傾向が見られた。おそらくメモリアクセスの最適化に問題が出るためと考えられる。この結果から 2 次元領域分割では 32 並列×32 並列が最も良い性能が出ることが分かったので、前述の 2 次元領域分割では 32×32 並列を使用した。

最後に使用配列が 2 の n 乗であるとき、キャッシュミスによる性能劣化があるかどうかを調べた。これは配列のサイズにより、メモリアクセス時に毎回キャッシュミスを起こしてしまうような状況を想定している。第 15 図にその結果を載せる。左側が配列のサイズを 2 の n 乗に定義した場合、右側が 2 の n 乗ではない場合の結果である。明らかに 2 の n 乗に定義した場合で性能の劣化が見られる。これは基本的にどの CPU アーキテクチャでも起こる問題ではあるが、コンパイラオプションにより回避可能なことも多い。



第 14 図:2 次元領域分割における各次元の並列数の変化に対する実効性能。

ここでは、合計 1024core を用いて、y 方向の分割数、z 方向の分割数を変化させた。横軸が y, z 方向における並列化数、縦が GFlops 値を表す。y、z 共に等しい数の並列数が良い性能を出しているが、y の分割数が多い場合より、z の分割数が多い方が性能が落ちやすい結果になっている。これは日立コンパイラによる結果である。



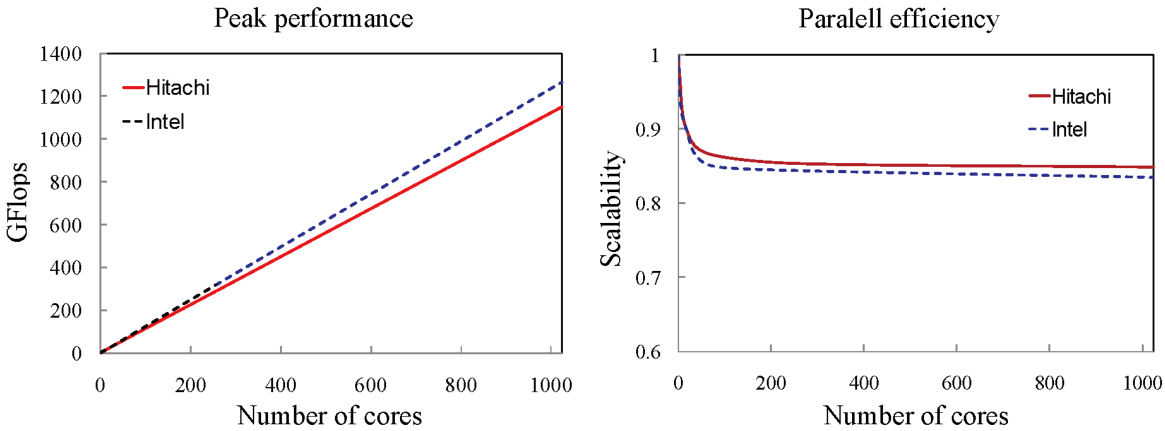
第 15 図:２のべき乗個の格子点を使った場合の性能劣化

左が 2^n 個の格子点数を使った場合、右側が 2^n 個ではない格子点を使った場合の結果。縦軸は GFlops 値を表す。明らかな性能劣化が確認できる。この結果は日立コンパイラによるものである。

３．２　Vlasov シミュレーション

Vlasov シミュレーションでは、1GB/core（16GB/node）の配列を使用した。ここで使用した無振動保存型スキームでは境界値が前後で 6 グリッド分必要のため、並列化時の通信が MHD コ

ードよりも多くなる。また前述のように 2 次元領域分割を位置空間に適用する。これは MHD コードの結果を受けて、2 次元領域分割が最適だと考えられるからである。第 16 図に Vlasov シミュレーションの性能評価を載せる。図の書式は第 8 図、第 9 図と同じで、左図が実効性能、右図が並列化効率を示す。赤色の実線が日立製コンパイラを使用した結果を示し、青い破線が Intel 製コンパイラを使用した結果を示している。まず実効性能をみると、リニアに性能が上がっており、MHD コードの場合と同様に良い性能上昇を示した。また、やはり Intel 製コンパイラが日立製コンパイラよりも良い性能を出しており、1024core において、日立製コンパイラで 1,149.3GFlops (実行効率 12%)、Intel 製コンパイラで 1,265,7GFlops (同 13%) を達成した。これは MHD コードの結果とほとんど変わらない性能である。また、並列化効率をみると、これも MHD コードの結果と同様に日立製コンパイラの方が Intel 製コンパイラよりも良い効率を出す。さらに、Vlasov コードの場合、並列化効率の下降が MHD 比べて著しく少ない。日立製コンパイラであれば、1024core 時に 85%の並列化効率を出しており、Intel 製コンパイラでも 83%の並列化効率を出している。これは、MHD コードの結果と比べて、20%程度良い結果である。つまり、MHD コードは非並列（単体）での性能が高いが、並列化により性能が落ちている。一方で Vlasov コードは非並列時の性能は MHD より低いが並列化効率が良く、性能の劣化が少ないため、最終的に MHD コードと同じ程度の実効性能を達成していると言える。



第 16 図:並列数に対する Vlasov コードの実効性能と並列化効率[11]。

左図が実効性能を表し、横軸が並列化数（使用 core 数）、縦が GFlops 値を表す。右図は並列化効率を表し、横軸が並列化数（使用 core 数）で縦軸がスケーラビリティを示す。赤色の実線が日立コンパイラの結果を示し、青色の破線が Intel コンパイラの結果を表す。Vlasov コードは MHD コードに比べて良い並列化効率を持っている。

４．まとめと今後の展望

H20 年度では、HA8000 を使い土星磁気圏シミュレーションを始めるに当たって、我々のコードの性能評価を行った。性能評価は MHD コードと Vlasov コードの二つを用いて行った。MHD コード、Vlasov コードをあわせて、実効性能では Intel 製コンパイラの方が性能が出るが、並列化効率は日立製コンパイラの方が良いという結果になった。MHD コードでは、2 次元領域分割を用いた場合が最も性能が良く、約 1.3TFlops（実行効率 14%）を達成し、Vlasov コードでは 1.27TFlops（同 13%）の性能を達成した。両コードとも 16core 以上は並列化することによる性能劣化は見えなかったため、このまま 1024 並列を越えてもリニアに性能が出ると考えられる。

別のシステムでは 10,000 並列でのベンチマークをとり、性能が出ることが実証されている。ただし、T2K オープンスパコンのような汎用 PC クラスタ型ではない、スパコンシステムの結果であり、実際に HA8000 などで大規模並列を調べる必要性は十分ある。

MHD コードの問題点として、並列化を行うと性能が 6 割近くに下がってしまうことが今回わかり、そこを改善することで実行効率 20%近く、最低でも 15%を越える性能を目指す。それらの最適化が終わり次第、高精細の土星磁気圏シミュレーションを始める。目標としては $0.1R_S$ の格子幅を用いたシミュレーションを行い、現在問題になっている土星磁気圏境界面上の乱れた対流構造をうまく取り扱えているかを調べる。また、次（々）世代磁気圏コードの Vlasov コードでは、単 core での性能上昇を考える。並列化効率は良いので、単体での性能を上げる事で、並列化後の性能を上げることが可能である。こちらでも 15%以上の実行効率を目指していく。

参 考 文 献

[1] Margaret G. Kivelson, Christopher T. Russell 編『Introduction to space physics』, Cambridge University Press, 1995.

[2] Ogino, T., R. J. Walker, and M. G. Kivelson, A global magnetohydrodynamic simulation of the Jovian magnetosphere, *J. Geophys. Res., 103*, 225, 1998.

[3] Fukazawa, K., T. Ogino, and R.J. Walker, "Dynamics of the Jovian magnetosphere for northward interplanetary magnetic field (IMF)", Geophys. Res. Lett., 32, doi:10.1029/2004GL021392, 2005.

[4] Fukazawa, K., T. Ogino, and R.J. Walker, "The Configuration and Dynamics of the Jovian Magnetosphere", J. Geophys. Res., 111, A10207, 2006.

[5] Fukazawa, K., T. Ogino, and R.J. Walker, "Magnetospheric Convection at Saturn as a Function of IMF B_Z", Geophys. Res. Lett., 34, L01105, 2007a.

[6] Fukazawa, K., T. Ogino, and R.J. Walker, "Vortex-associated reconnection for northward IMF in the Kronian magnetosphere", Geophys. Res. Lett., 34, L23201, 2007b.

[7] Khurana, K. K., M. G. Kivelson, V. M. Vasyliunas, N. Krupp, J. Woch, A. Lagg, B. H. Mauk and W. S. Kurth, The Configuration of Jupiter's Magnetosphere, In:Bagenal, F., T. Dowling and W. McKinnon (Ed.), Jupiter, Cambridge University Press, New York, 2004.

[8] 様々なスパコンにおける MHD コードの実効性能.
http://center.stelab.nagoya-u.ac.jp/web1/simulation/hpfja/comput04.html

[9] R. O. Dendy,『Plasma Dynamics』, Oxford University Press, 1990.

[10] 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.20, 817.828, 1992.

[11] Fukazawa, K., T. Umeda and, T. Ogino, Performance measurement of electromagnetic fluid codes for space plasma on the T2K open supercomputer system, submitted to

parallel computing.
[12] T. Umeda, K. Togano, T. Ogino, Two-dimensional full-electromagnetic Vlasov code with conservative scheme and its application to magnetic reconnection 180, 365 -374, 2009.