HA8000 システムでの地球ダイナモシミュレーションと可視化

陰山 聡，大野 暢亮

海洋研究開発機構 地球シミュレータセンター

1. はじめに

HPC 特別プロジェクトの一つとして，今回我々は，地球シミュレータ向けに開発・最適化した地球ダイナモコードを HA8000 システムに移植した。このコードは，地球シミュレータ上の主に 512 ノード（4096 プロセッサ）を使った大規模な並列計算によるプロダクトランに用いていたものである。本研究の目的は，地球シミュレータに最適化された我々のコードが，そのまま HA8000 システムの多ノードで走るか？ HA8000 システムへの移植にはどの程度の作業が必要か？そのまま移植した場合，HA8000 システムの多ノードでどの程度の速度が出るか？限られた時間内で最小限の最適化によってどの程度の加速効果が見られるか，などの疑問に答えることであった。ここではその結果について報告する。なお，我々はベクトル計算機の利用とチューニングに関してはある程度経験をもっているが，スカラー型の，特に大規模な並列計算機を使うのは今回が初めてであり，スカラープロセッサへのチューニングについて知識も経験もほとんどない。従って，我々の技量不足からこのシステムの性能を十分に引き出していないことは間違いないので，この報告はその点を斟酌して読んでいただきたいと思う。また，今回のプロジェクトでは，シミュレーションデータをスーパーコンピュータ上で直接解析するために我々が開発している並列可視化ツールも移植し，最大 8192 コアまで使って様々な可視化も試したので，これに関してもあわせて報告する。

まず始めに，地球ダイナモシミュレーションについて簡単に紹介する。コンパスが北を指すのは，地球が双極子磁場に覆われているからである。その双極子磁場は，地球内部に流れる 10 億アンペア程度のリング状の電流が作りだしている。このリング状電流を生み出す発電のメカニズムが地球ダイナモである。この電流の直接のエネルギー源は，地球内部に存在する液体金属の流れである。地球の中心部，岩石でできたマントル層の内側には核と呼ばれる領域がある。核は鉄でできている。この核は二層に分かれており，外側は外核と呼ばれる液体領域，内側は内核と呼ばれる固体領域である。外核の液体鉄は対流運動しており，この対流運動が磁気流体力学（MHD）ダイナモと呼ばれる機構を通じて地球磁場を作り出している。磁場中を電気伝導性流体が流れると，誘導起電力により電流が生じ，その電流が元の磁場と同じ形の磁場を作っていればこれは正のフィードバックとなり磁場が強まる。これが MHD ダイナモの原理である。外核の対流の駆動源は未解明だが，我々のシミュレーションでは簡単のため熱対流のみを考える。基本方程式は MHD 方程式である。

シミュレーションモデルは以下の通りである：地球の外核を想定し，球殻状の容器に電気伝導性流体（MHD 流体）が入っているものとする。内側の球面は高温，外側の球面は低温に保たれている。球の中心方向に重力がはたらき，二つの球殻は同じ角速度 Ω で回転する。温度差が十分に大きければ内部の流体は熱対流運動し，MHDダイナモ機構によって，対流の運動エネルギーが磁場のエネルギーに変換され，磁場が生成される。比較的低い解像度で十分であればこの問題を数値的に解くのはそれほど難しいものではない。我々も含めて世界中のグループが

地球ダイナモシミュレーションに挑戦し，これまでに双極子磁場の自発的生成やその非周期的逆転現象など，地球磁場に特徴的な性質を計算機の中で少なくとも定性的には再現することに成功している。スーパーコンピュータの進歩に伴い，地球ダイナモシミュレーションも着実に進歩してきた。その進歩の指標の一つとしてよく使われるのは，この系を特徴づける無次元量の一つであるエクマン数 $Ek = \nu /2\Omega R^2$ である。ここで ν は動粘性率，Rは外核の半径である。地球外核のエクマン数は $O(10^{-15})$ 程度と見積もられる。エクマン数は粘性率に比例するので，その値が小さな系ほど，高い空間解像度を要求され，シミュレーションが難しくなる。現在のスーパーコンピュータでも$Ek = 10^{-15}$のシミュレーションは不可能である。1990 年代に我々が地球ダイナモシミュレーション研究を始めたとき，計算で用いたエクマン数は$O(10^{-4})$であった。スーパーコンピュータの進歩と共に，地球ダイナモシミュレーションで使われるエクマン数はゆっくりとではあるが着実に小さくなっており，我々は最近，$Ek = 10^{-7}$の計算に世界で初めて成功した[1]。

地球ダイナモシミュレーションで用いられる空間離散化手法としては球面調和関数を基底関数としたスペクトル法が長年主流であったが，この手法は大規模な計算には向かないので，近年では有限体積法や有限差分法，有限要素法などに基づく地球ダイナモシミュレーションコードが開発されている。 我々は長年，有限差分法に基づく地球ダイナモシミュレーションコードを使っている。基本計算格子として現在採用しているのは，図 1 に示したインヤン格子である。

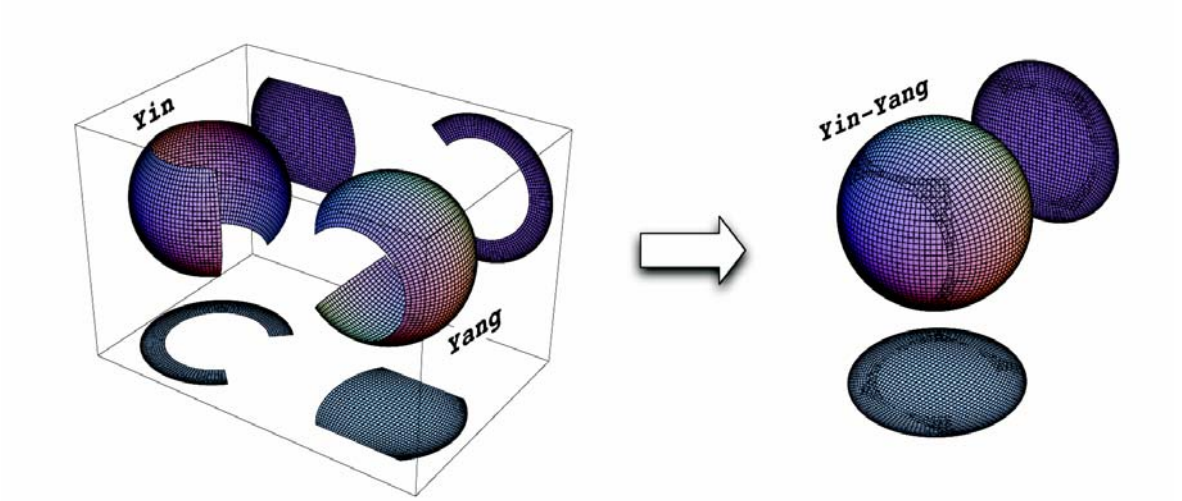

図 1. インヤン格子

二つの合同な要素格子を組み合わせて球面を覆うキメラ格子の一種。球座標の低緯度部分を切り出して一つの要素格子（イン格子）を作る。それを回転させると一つの要素格子（ヤン格子）ができる。

通常の球座標に基づく格子（緯度経度格子）は，高緯度部分，極の近くで格子間隔が不必要に集中するために計算格子として望ましくない。しかしながら低緯度部分の領域をだけを見れば，これは数値計算向きの「タチの良い」格子系である。直交格子であり，格子間隔はほぼ一定で，しかもメトリックは単純でよく知られているからである。通常の球座標上，緯度方向に赤道を挟んで南北 45 度ずつの領域を切り取り，経度方向には（360 度ではなく）270 度切り取った領域を考える。これは球座標の計算空間では長方形の領域である。これを一つの要素格子とし，まったく同じものをもう一つ用意して，実空間で回転させて組み合わせれば球面全体を覆う。これがインヤン格子である。我々のコードではこのインヤン格子上で MHD 方程式を解

いている。空間離散化は 2 次の中心差分，時間発展には 4 次精度のルンゲ＝クッタ法を用いている。

地球シミュレータ上の計算ではベクトル化は球殻の半径方向にかけていた。並列化は，イン格子とヤン格子のそれぞれで水平方向（緯度・経度方向）に 2 次元領域分割し，MPI を用いる。それぞれの要素格子で独立の MPI コミュニケータを構成することで MPI 通信のためのコーディングを簡略化させている。イン格子，ヤン格子それぞれの水平方向の境界上でのデータは，キメラ格子手法に基づき相互補間で設定する。

2. HA8000 システムへの移植

移植作業は順調であり，コードの変更は以下に述べるわずかな点のみで済んだ。

変更点 1 ：我々のコードでは，全エネルギーを計算する場合などに MPI_ALLREDUCE 関数の SUM 機能を使っている。この関数は（MPI では）第一引数と第二引数に与える送信バッファと受信バッファを個別に用意しなければいけなかったが，MPI2 では第一引数に MPI_IN_PLACE を与えることで余分なバッファが不要になるので便利である。HA8000 システムではデフォルトの MPI がこの MPI_IN_PLACE の機能に対応していなかったので，利用する MPI を MPI2 に切り替えようとしたが，それがなぜかうまくいかなかった。そこで今回は，我々のコード中の MPI_ALLREDUCE で MPI_IN_PLACE を使わないように変更した。なお，我々のコードでは，MPI_ALLREDUCE も含めてほとんどの MPI 関数は，mpiut.f90 という自作の wrapper モジュールで wrap している。これは Fortran90 に用意されている関数の多重定義機能を利用して，ソースコード中の MPI 関数の呼び出しを簡潔でわかりやすくするためである。従ってソースコード中の MPI_ALLREDUCE の変更はわずかで済んだ。

変更点 2：我々の元々の Fortran90 コードでは MPI ライブラリを使う各モジュールで MPI モジュールを use する（use mpi）ことで MPI ライブラリ使うようにしていたが，デフォルトではこの方法は使えないようであった。この問題も MPI2 を使えば解決できるらしいことがわかったが，上述したとおりなぜか MPI2 への切り替えに失敗したので，今回は ‘include mpi.f’ とすることで回避した。

以上，述べたとおり，地球シミュレータから HA8000 システムへの移植は簡単で，短時間のうちに終了することができた。

3. Numacntl の仕方

移植した我々コードを flat MPI で計算した場合，numactl を以下の指定で行うときに最も大きな加速効果（約 20%）が見られた。

```
#!/bin/bash
MYRANK=$MXMPI_ID
MYVAL=$(expr $MYRANK / 4)
NODE=$(expr $MYVAL % 4)
numactl --cpunodebind=$NODE --membind=$NODE $@
```

以下に述べる計算は全てこの numactl で計算したものである。また，

```
numactl --cpunodebind=$NODE --interleave=$NODE $@
```

としたときも（わずかながら遅かったが）ほぼ同様の効果が見られた。

4. 高速化に関するいくつかの試み

ここでは計算時間短縮のために行ったいくつかの試みと，その結果について述べる。今回は時間が限られていたので，網羅的で系統的なテストにはなっていない。なお，今回試した並列化手法は，flat MPI だけである。

コンパイラによる速度の違い：日立のコンパイラ（オプション –Oss –noparallel）と比較して，インテルのコンパイラ（オプション –O4）の方が約 10%ほど高速であった。

配列サイズ依存性：我々のコードでは，物理量は（半径，緯度，経度方向に対応する）3 次元配列で表されている。ある物理量 p を p(N1,N2,N3)と書いたとき，第 1 次元のサイズ N1 は，地球シミュレータの場合，ベクトルレジスタの数から決めていた。一方，第 2 次元と第 3 次元方向には領域分割をかけるので，利用するノード数（MPI プロセス数）によってそれぞれの大きさは変動する。分割する前の第 2 次元と第 3 次元方向のサイズをそれぞれ（我々のダイナモシミュレーションのプロダクトランで典型的な値である）514 と 1538 に固定し，第 1 次元のサイズ N1 を以下の表のように変化させて速度を測定した。これは 128 ノード，2048 コアでの計算である。

N1	507	508	509	510	511	512	513
GFLOPS	502	535	531	531	528	543	482

これから分かるとおり，N1=512 のときに最も早く，それから 1 グリッド増やしただけで（N1=513）速度は約 11%落ちた。この結果から，今回は N1=512 の計算を主に行った。

ホットスポット：128 ノードでの計算結果を見ると，コード中で，MHD 方程式のソルバに約 80%の計算時間費やされ, ルンゲ＝クッタ法による時間積分ルーチンに 10ー15%の時間が費やされていた。これは地球シミュレータにおける割合とほぼ同じである。そこで基本アルゴリズムは変えずに，MHD ソルバのソースコードを変更して加速する試みをいくつか行った。その中で，意外にも効果がなかったものと，逆に意外にも効果があったものを紹介する。

効果のなかったもの： MHD 方程式には基本物理量（3 次元配列）が 8 個ある。これらをひとまとめにして，第 1 次元のサイズが 8 の 4 次元配列を定義すれば，キャッシュを有効利用できると期待される。この方法は，スカラー型スーパーコンピュータでの MHD シミュレーションでは効果的な加速手法であることが知られている [2]。今回，我々のコードに対してもこの方法を試してみたが，残念ながら効果はなく，むしろ遅くなってしまった。

効果があった方法： MHD ソルバの基本部分は 3 次元配列に対する 3 重 do loop である。ある物理量を p(i,j,k) とすると, 3 重 do loop の再内側ループのインデックス i が走る範囲は 2≦i≦N1-1 である。この第 1 次元方向に一種の領域分割をかけ，配列を無理矢理 4 次元化してみた。つまり，第 1 次元方向を NDIV 個に分割し，4 次元の新しい配列 pp(ii,j,k,n)を作る。ここで 2≦ii≦N1D-1, 1≦n≦NDIV, i=(n-1)*(N1D-2)+ii である。そして，MHD ソルバにこの変数を渡すには，分割したそれぞれの領域毎に以下のように渡す。

```
do d = 1 , NDIV
```

```
        call mhd_solver(pp(:,:,:,d))
      end do
```

サブルーチン mhd_solver は，3 次元配列を受け取る MHD ソルバであり，基本的には元々の 3 次元 MHD ソルバと同じものである。この方法により，20%ほどの高速化がみられた。ただし，以下に述べる 512 ノードの計算では，この加速手法は使っていない。

5. HA8000 システム 512 ノードでの性能

この章では，地球シミュレータ用に最適化したコードに，第 2 章で述べた軽微な変更のみを加えたコードをそのまま 512 ノードで走らせた結果についてまとめる。これほど大きな規模の並列計算の実行では，何らかのトラブルが起きるものと覚悟していたが，ジョブ投入から実行まで，全て順調に進んだ。

ノード当たり 16 コア，計 8192 コアで約 40 分間実行したところ，プログラムの初期化処理のルーチンに全体の計算時間の 14%もかかった。これは異常に大きい。原因は不明である。この初期化処理では，メモリアロケーションや計算格子のメトリック設定，計算初期条件の設定など，シミュレーション開始時に一度だけ行う比較的複雑な処理を集めている。初期化処理の部分を除いて評価した計算速度は 1.8 TFLOPS であった。1 コアあたりの性能を 9.2 GFLOPS として計算すれば，8192 コアで 1.8 TFLOPS という性能は理論ピーク性能の約 2.4%に相当する。

次に同じ 512 ノードで，1 ノードあたり 8 個のコアだけを使った flat MPI を試してみた。その結果，初期化処理にかかる時間は全体の 4.7%に減少した。初期化処理の部分を除いて評価した計算速度は 1.5 TFLOPS であった。これは理論ピーク性能の約 4.0%である。

次に 512 ノードで，1 ノードあたり 4 コアを使った計算をしてみた。今度は初期化処理にかかる時間が 1.5%にまで下がった。地球シミュレータでの経験から言えば，この程度が通常の数値である。全体の計算速度は 0.87 TFLOPS であった。これは理論性能の 4.6%に相当する。

6. データ可視化について：インヤン 座標用可視化プログラム Armada

地球ダイナモシミュレーションの結果を可視化する上で，常に問題になるのは：(1) インヤン座標という特殊な座標系，(2) 出力されるデータサイズの大きさ，の 2 点である。

インヤン座標上のデータを直接可視化することが可能なソフトウェアは，我々の知る限り存在しない。ただし，インヤン座標は球座標の一部分の単純な組み合わせであるので，工夫すれば AVS/Express 等で可視化は可能であるし，球座標上にデータをマッピングすれば，多くの可視化ソフトウェアで可視化が可能となる。(2)については，我々のシミュレーションで出力するデータは単精度で出力しており，グリッドサイズはイン座標とヤン座標それぞれ 1 タイムステップ・1 変数ごとに約 3GB である。この規模のデータとなると，たとえインヤン座標上のデータを可視化できるソフトウェアが存在したとしても，インタラクティブに可視化するのは困難である。また，データ転送に多大な時間がかかる。

我々は，インヤン座標上の時系列データを直接可視化することが可能な，グラフィックス関係のハードウェアを一切必要としない並列可視化ソフトウェア“Armada”を開発している[3]。このソフトウェアは，C++で記述されていて，MPI および OpenMP で並列化されている。またグラフィックス関係のハードウェアを用いずに画像を描画するので，スーパーコンピュータ上で

も動作する。今回我々は，Armada を HA8000 上で動作させることにより，計算された結果をネットワーク経由で他のグラフィックスワークステーションなどの可視化用計算機に転送することなく,また球座標にデータをマッピングすることなく直接可視化を行う方針を立てた。今後，スーパーコンピュータの規模が拡大して計算結果がさらに大規模になると，これに類する可視化手法が一般的になると思われる。

Armada は地球ダイナモシミュレーション専用というわけではなく，汎用の CFD データ可視化ツールである。また，すべての可視化機能をソフトウェアレンダリングで実現している（グラフィックス関係のハードウェアを必要としない)。使用できる可視化機能は，スカラーデータでは，等値面・ボリュームレンダリング・カラースライス，ベクトルデータでは，流線・矢印表示である。スカラーデータの可視化については，すべての可視化手法にレイ・キャスティング法を用いている。レイ・キャスティング法でボリュームレンダリング以外の等値面やスライスを生成するのは，容易である。等値面生成は，2 つの連続するサンプリングポイント上の値を比較することで，レイが等値面を通過しかた否か判定できる。スライスについては，幾何学的にスライス面をレイが通過したか否か判定できる(使用するスカラーデータの可視化手法がスライスのみ場合は，幾何学的にレイとスライス面の交点を計算して描画する)。

スカラーデータとベクトルデータの可視化手法は，合計 5 種類存在するが，単独でも使用可能であるし，組み合わせて使用することも可能である。データは，複数読み込むことが可能で，例えば，温度の等値面と渦度の z 成分のスライス面，速度場の流線を同時に描くことも可能である。

Armada には，1 度の実行で，多数の可視化画像を生成する機能が備わっている。具体的には：(a) 各タイムステップ毎に複数，多数の視点を設置する機能，(b)各タイムステップ毎に複数の可視化パターンを適用させる機能，の二つを実装している。

まず(a)について詳しく述べる。Armada は時系列のデータを扱えるが，タイムステップ毎に 1 枚の画像だけでなくユーザーがあらかじめ設定した視点から，複数の画像を生成する機能がある(図 2 参照)。例えば，視点（位置，視線方向，視線上方の組）をあらかじめ 100 個用意した場合，データが 100 タイムステップあると 100x100 で合計 1 万枚の画像が生成される。

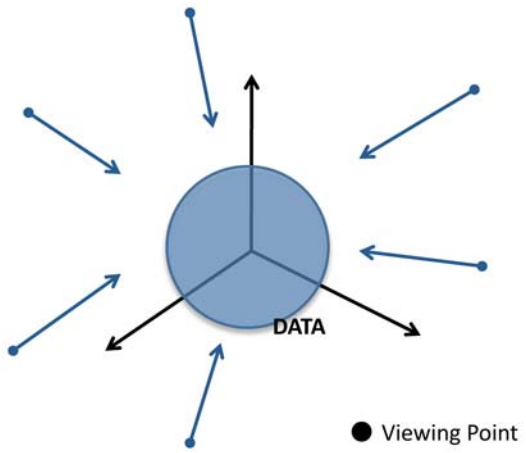

図 2. 複数視点からのデータ可視化の概念図

Armada ではあらかじめ設定した複数の視点からの可視化画像を作成することができる。

次に(b)について説明する。例えば,『可視化１：等値面（値 10.5)+スライス(z=10)，可視化 2：

等値面(値 6.8)，可視化 3：スライス（z=0)+流線 5 本』等の可視化パターンをユーザーが用意すると，各タイムステップで 3 種類ずつの可視化画像を生成する。視点が 1 つで 100 ステップのデータがあるとすると，合計 300 枚の画像を生成する。1. と 2. を組み合わせた場合，例えば，視点 100，可視化パターン 10，タイムステップ 100 あるとすると，合計 100x10x100=10 万枚の画像を生成し出力する。我々は，この大量画像生成機能を用いたリアルタイム可視化実験を将来おこなう予定である。

Armada の並列化については，ノード間並列を MPI，ノード内の並列化を OpenMP でおこなうハイブリッド並列化を採用している。

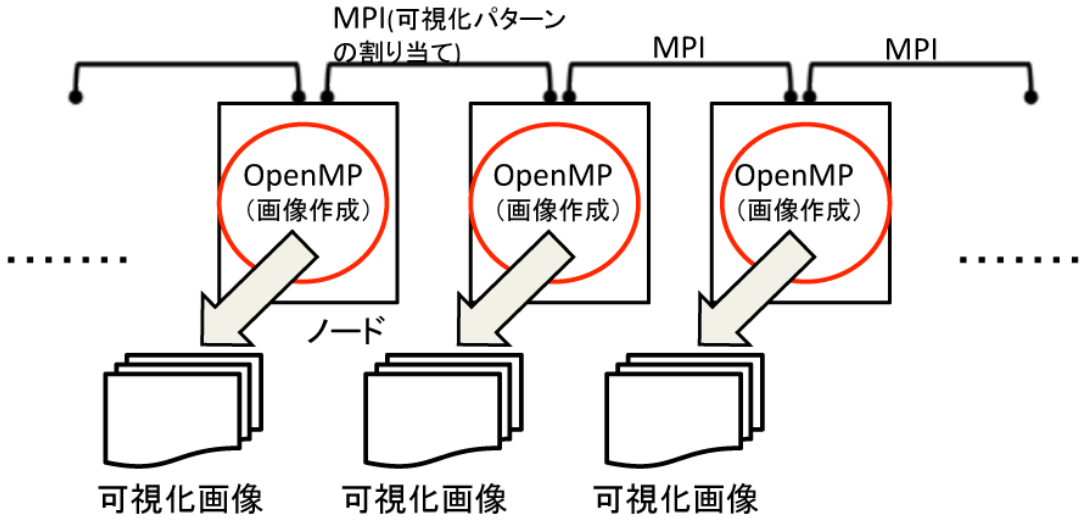

図 3. Armada の並列化

各ノードが独立して可視化をおこない画像を出力する。

Armada を実行すると，各ノード（正確には各 MPI プロセス）は基本的に独立して動作する。マスタープロセスから MPI 通信で割り当てられた“可視化”(可視化パターンとタイムステップが割り当てられる)を実行する（図 2 参照)。ノード内の並列化は，OpenMP を用いておこなわれており，可視化するデータは 1 ノード内に載ることが必要である。つまり 1 ノード内で走る可視化ソフトウェアが，複数のノードで同時に実行され，多数の可視化画像を出力するというイメージである。多数のノードに（領域）分割されたデータを読み込みこんで，１つの（１タイムステップの）巨大データを可視化する方式とは異なる。このソフトウェアの目的は，複数の可視化パターンの画像を多数の視点から可視化して大量の画像を保存することであること，スーパーコンピュータの 1 ノード内の共有メモリのサイズが増加傾向にあること，などの理由によりこのように並列化した。HA8000 の 1 ノード内の CPU コア数は 16，共有メモリは 32GB 搭載されているので，我々の 1 ステップ・1 変数あたり約 3GB のデータでも十分に可視化することが可能である。ただし，今後 1 ノード内に可視化すべきデータが載らなくなるような場合には，並列化を再検討する。

今回我々は Armada を HA8000 システムの 512 ノード，8192 コアでも動作することを確認した。以下に HA8000 で作成した地球ダイナモシミュレーションデータの可視化画像を紹介する。図 4 は比較的粗い解像度でのシミュレーションデータの可視化例である。図 5，6，7 は，１変

数約 3GB のデータを高解像度データを可視化した例である。

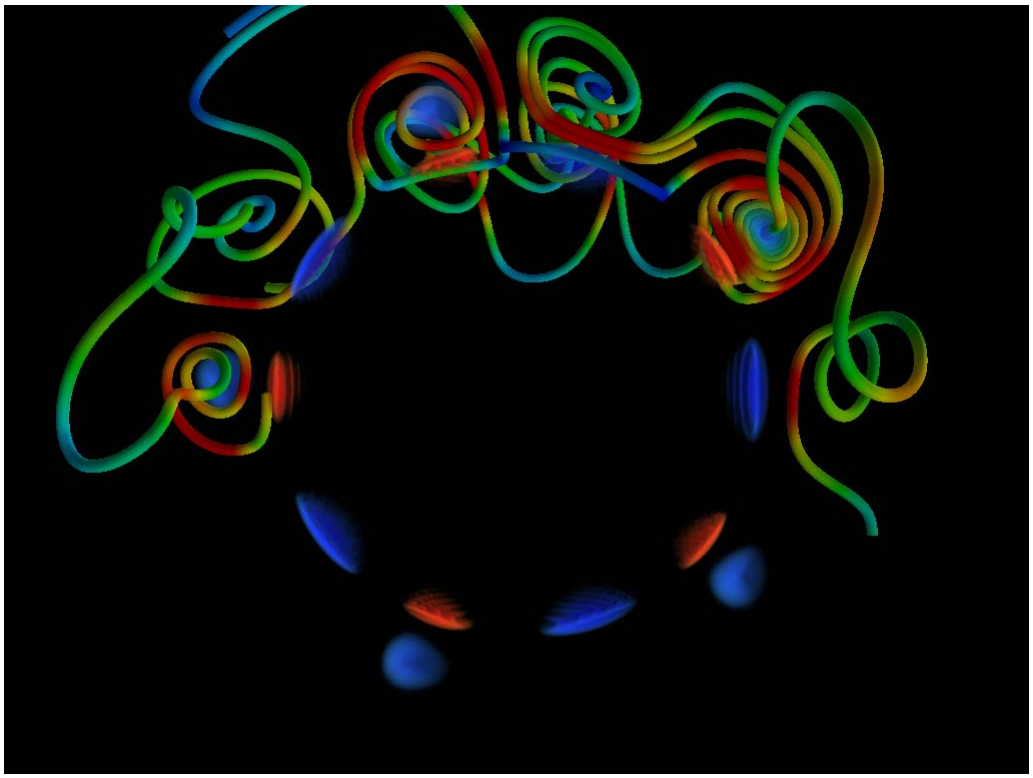

図 4. 速度場と渦度の z 成分

速度場を流線で可視化し，渦度の z 成分をボリュームレンダリングで可視化している。流線の色は，ベクトルの大きさを表している。

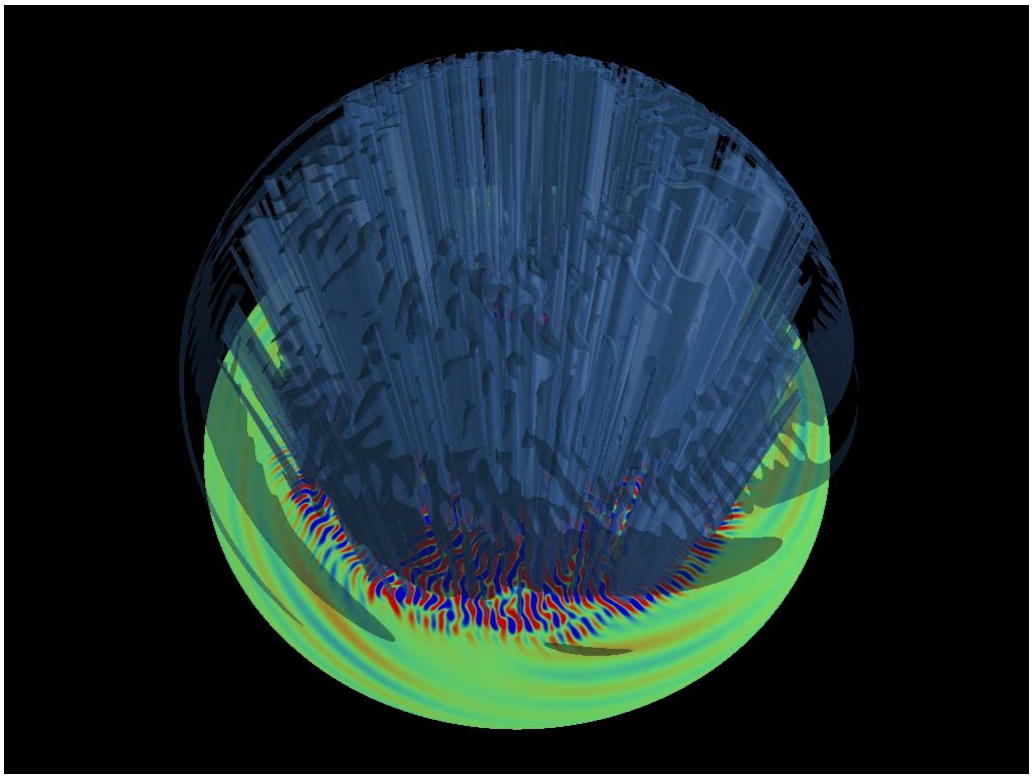

図 5. 渦度の z 成分の可視化

渦度の z 成分を赤道面のスライス，および半透明な等値面で可視化している。

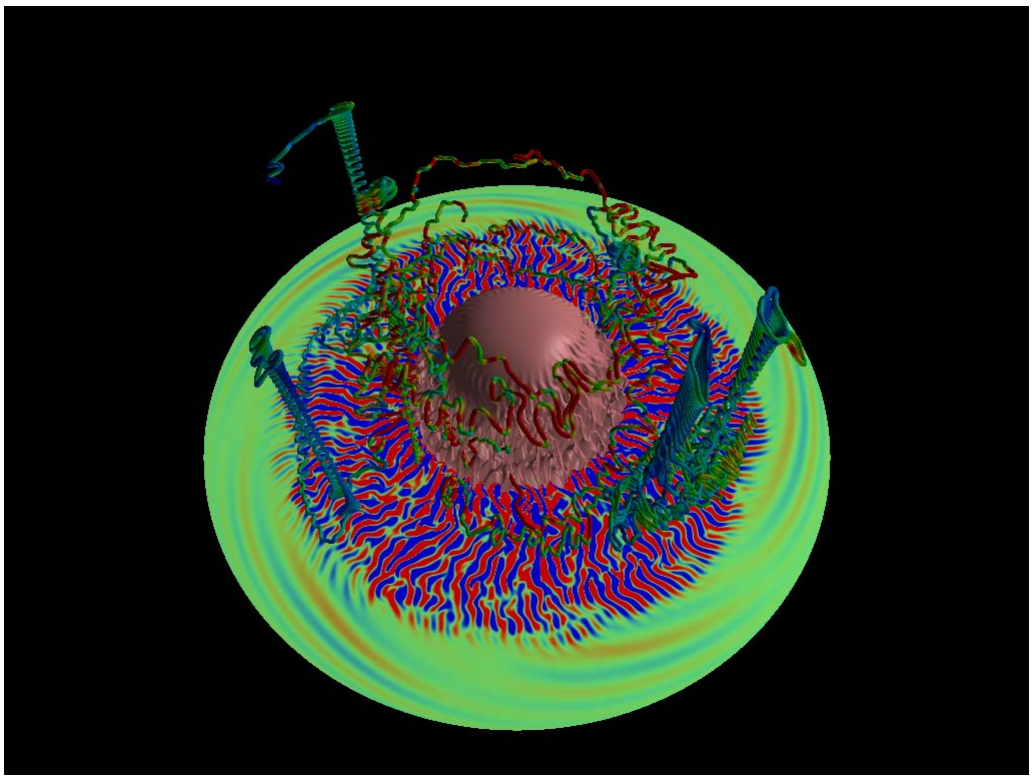

図 6. 温度，渦度の z 成分および速度場の可視化

温度を等値面，渦度の z 成分をスライス，速度場を流線で可視化している。

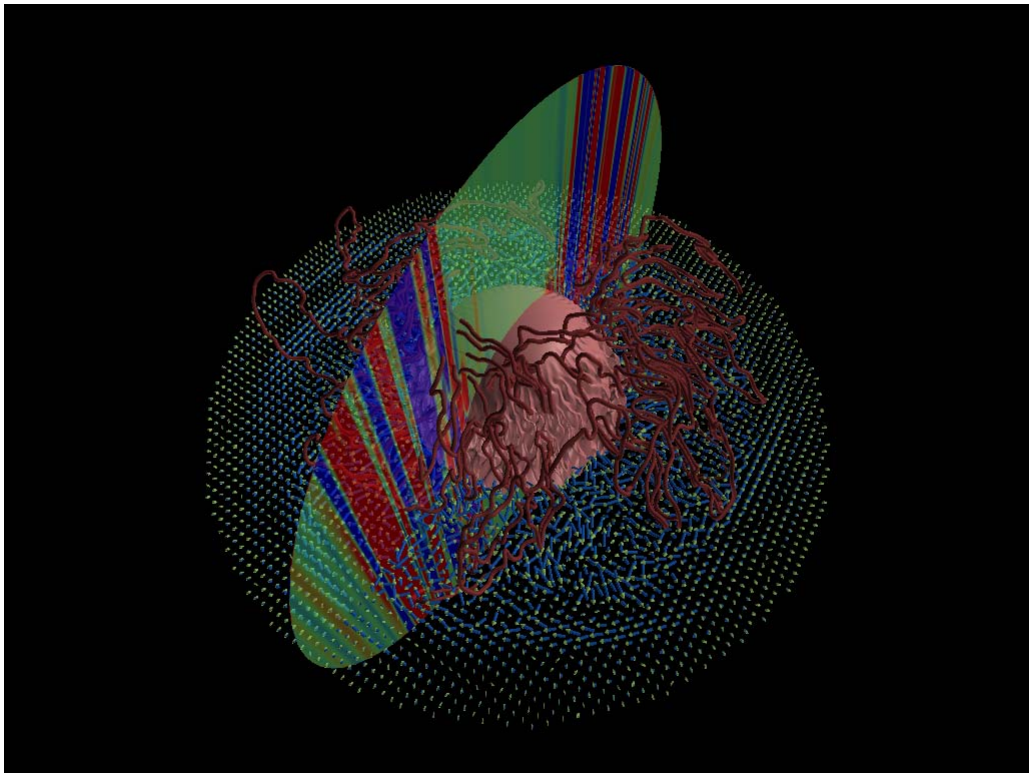

図 7. 温度，渦度の z 成分，速度場，磁場の可視化

渦度の z 成分に半透明なスライスを用いている。速度場を矢印，磁場を流線で可視化している。1 変数が約 3GB であるので，合計約 24GB のデータを可視化している。

7. まとめ

地球シミュレータ向けに開発・最適化した地球ダイナモシミュレーションコードを HA8000 システムに移植した。このコードは磁気流体力学（MHD）方程式を球ジオメトリ（インヤン格子）の下で解く流体系のコードである。ソースコードのわずかな変更で移植でき，2 次元領域分割による flat MPI 並列化の下，そのままのコードで最大 512 ノード，8192 コアまでの計算を問題なく行うことができた。

512 ノードでの計算性能は，1 ノード当たり 4 コアを使った場合，理論性能の 4.6%，1 ノードあたり 8 コアを使った場合は 4.0%であった。ベクトル計算機向けにチューニングされたコードを何もしないでそのままスカラー計算機で走らせたことを考えれば，512 ノード x 8 コア = 4096 コアの計算で 4%台のスピードが出たことは我々には驚くべきことのように感じた。しかしながら 512 ノード x 16 コアの計算では残念ながら 2.4%の性能しかでなかった。128 ノード x 16 コアの計算でも性能は 2.9%であったので，これは総コア数の問題ではなく，我々のコードがノード内の全てのコアを十分生かし切れていないことを意味する。いずれにせよ，この数字は悪すぎるように感じるので，何らかのチューニングが必要なのは明らかだが，我々の知識と技量では，今回の限られた時間内にこの性能を引き上げることはできなかった。

なお，今回移植した地球ダイナモコードのアルゴリズムと，その地球シミュレータ上での性能評価については，文献[4]に記述してある。

8. 謝辞

情報基盤センターの中島研吾教授には，HA8000システムの利用方法に関して様々な技術的アドバイスを頂きました。深く感謝します。本研究は科研費（17540404）と三菱財団助成金の補助を受けました。

参 考 文 献

1. A. Kageyama, T. Miyagoshi, and T. Sato, Formation of current coils in geodynamo simulations, Nature, vol. 454, pp. 1106-1109, 2008
2. 荻野竜樹，中尾真季，スカラー並列機を用いた高効率ＭＨＤコードの開発，名古屋大学情報連携基盤センターニュース，Vol.4，No.4，pp.284-299，2005
3. 陰山聡，大野暢亮，″固体地球シミュレーションの可視化″，可視化情報，Vol.28，No.110，pp.180-185，2008
4. A. Kageyama et al., A 15.2 TFlops Simulation of Geodynamo on the Earth Simulator, Proc. ACM/IEEE Conference SC2004 (Super Computing 2004), pp. 35-43, 2004