第１１回ビジュアリゼーションカンファレンス
2005 年 10 月 21 日、東京

大規模データからの高速動画作成システム

上原　均○、川原　慎太郎、大野　暢亮、古市　幹人、荒木　文明、陰山　聡

（海洋研究開発機構 地球シミュレータセンター）

A High Performance Movie-Making System for Large-Scale Simulation Data

H. Uehara, S. Kawahara, N. Ohno, M. Furuichi, F. Araki, and A. Kageyama

ABSTRACT

We have developed a high performance movie making system, MovieMaker, for the visualization of large scale simulation data generated by the Earth Simulator (ES). In many cases of the simulations on the ES, more than a Giga byte data is generated per one step for a single variable. However, available visualization tools today cannot handle such large scale data within acceptable time. MovieMaker is a parallel rendering system with dynamic load balancing, on the shared memory architecture. It can generate a sequence of visualization images within about 40 seconds per each image from a sequence of giga byte scale data. The implemented visualization methods in MovieMaker are volume rendering, isocontouring, and streamlines.

Keywords: Earth Simulator, Large Data, Volume Rendering, Parallel Rendering

１．はじめに

地球シミュレータ[1]は、全球超高解像度での大気大循環モデルなどの大規模シミュレーションの実行を目的として開発されたベクトル並列計算機である。現在、地球シミュレータでは、大気や海洋、固体地球などの多種多様な分野の数値シミュレーションが実行され、膨大なサイズのデータが出力されている。これらの出力データサイズは１ステップ・１変数当り数 GB 以上になることも珍しくなく、１ジョブの総出力で１TB オーダーに達する事も稀ではない。

標準的な可視化ソフトウェアでは、このような莫大な出力データを取り扱うことが出来ない。大規模データに対応したソフトウェアもあるが、その処理速度が充分ではないのが現状である。時系列データの解析には動画の作成が必須である。そのための連番画像を作成するのに数日以上かかるようでは実際上使い物にならない。

そこで我々は大規模シミュレーションデータから高速に動画用連番画像を作成するソフトウェア MovieMaker の開発に着手した[2,3]。MovieMaker の開発目標は「１TB の数値シミュレーションデータから一晩で一連の動画用連番画像を生成する」ことである。

本稿では MovieMaker の開発および性能評価、動画作成事例について述べる。

２．高速動画作成ソフトウェア MovieMaker

2.1　開発の方針

１TB のデータから一晩で動画を作るためには、1 ステップ当り１GB のデータが 1024 ステップ分、一晩を 12 時間と想定して、画像１枚当りの平均生成時間をおおよそ 42 秒以内にしなければならない。この目標を実現するためには、並列レンダリングが必須である。並列レンダリングシステムについては既に多数の研究事例[4,5]があるが、その多くは等値面表示などのポリゴンレンダリング、またはボリュームレンダリングのいずれかのみの並列処理に焦点を絞っている。MovieMaker ではポリゴンレンダリングとボリュームレンダリングの両方の並列処理を実現することを目指した。

並列レンダリングシステムの開発事例としては、近年、分散メモリ型マシンであるクラスタを用いた事例が主流のようである（例：[6]）。しかし、我々は可視化処理に優れた共有メモリ型計算機を複数台所有している（SGI Onyx3800, Onyx4 など）ので、MovieMaker を共有メモリ型アーキテクチャ向けソフトウェアとして開発することとした。

並列レンダリングシステムにおいて、ディスク I/O にかかる時間的コストの問題は深刻である。Yu ら[7]は並列ファイル I/O によってこの問題を解決した。一方、MovieMaker では単一プロセスによるファイル I/O をレンダリング処理とオーバーラップさせることで、この問

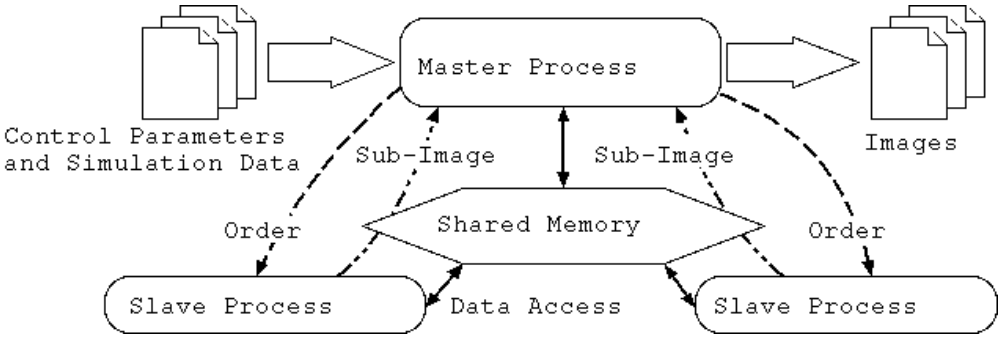

Fig.1 MovieMaker: Master/Slave Model for Parallel Processing on the Shared-Memory Architecture

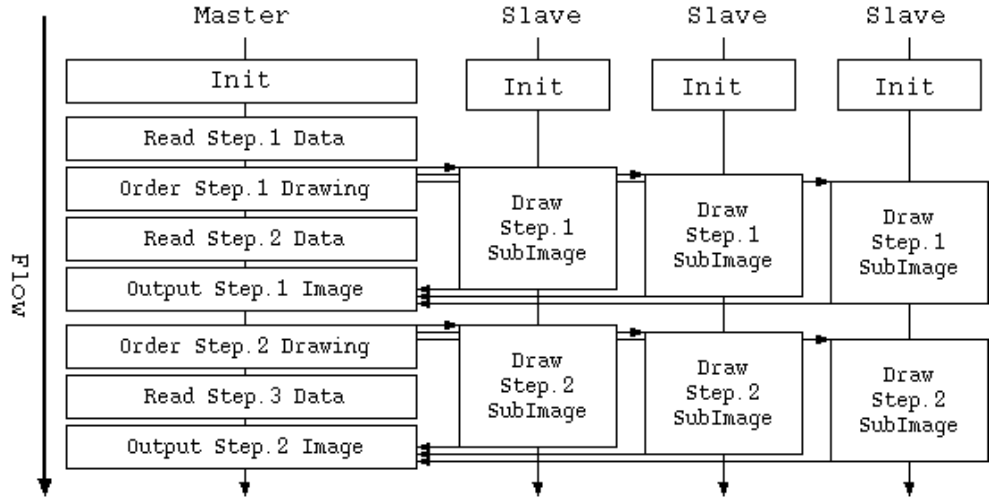

Fig.2 Flowchart of MovieMaker: Overlapping of Data Read Step and Rendering Step

題を解決する。

なお、地球シミュレータ向けに開発された可視化システムとして、固体地球シミュレーション向けの大規模有限要素法解析プラットフォーム GeoFEM の可視化サブシステム[8]がある。このサブシステムは GeoFEM 本体と強く結合しているのに対して、MovieMaker は数値シミュレーションプログラムから独立した後処理システムとして実現した。

2.2 設計

共有メモリ型アーキテクチャの特長を活かし、MovieMaker は共有メモリを用いたマスタ＝スレイブ方式の並列レンダリングシステムとして設計した。**Fig.1** に示したように一つのマスタプロセスと複数のスレイブプロセスによって共有するメモリ領域に、シミュレーションデータを格納する。マスタプロセスは、１）設定ファイルの読込、２）シミュレーションデータの読込と共有メモリへの格納、３）スレイブプロセス群の制御およびレンダリングタスクの割当て、の３つの処理を行う。スレイブプロセスは、マスタプロセスのタスク割当てに基づいてレンダリングを行い、その結果は共有メモリを介してマスタプロセスに返す。マスタプロセスでは、スレイブプロセス群から返ってきた部分画像群を合成し、最終的な画像を出力する。共有メモリアクセス以外のプロセス間通信は MPI[9]で実現する。

並列レンダリング時の負荷分散[4]は、マスタプロセスが行うスレイブプロセスへのタスク割当てで実現する。負荷分散の方式は一般に静的、動的、適応的の三つに大別できるが、MovieMaker では動的負荷分散を行う。静的方式では視点の変化やデータの変化に伴う負荷変動に追従できず、適応的方式でも適応化処理のコストが無視できないと判断したためである。MovieMaker では、あるステップの画像を作成する上でのレンダリング処理をスレイブ総数などに基づいて適度に小さく分割し、その分割したタスクをスレイブプロセスに動的に割り当てる事で負荷を均等化する。

MovieMaker が対象とするような大規模データでは、シミュレーションデータをファイルからメモリに読み込む時間が無視できないほど大きい[7]。そこで MovieMaker では **Fig.2** に示すように、スレイブプロセス群での並列レンダリング中にマスタプロセスで次のステップ分のデータ読込をさせるというオーバーラップを行うことで、データ読込時間を隠蔽する。ただし、マスタプロセスがシミュレーションデータを読み込んでいる最中にもスレイブプロセスでのタスクが終了する可能性がある。そこで、マスタプロセスでのデータの読込では、並列レンダリングとオーバーラップしない最初のステッ

プ分のデータ読込を除き、1) 次のステップデータの読込と、2) スレイブプロセスでのタスクの終了チェックおよび未処理タスクの割当て、の二種類の処理をタイムシェアリング方式で実施する。このデータ読込時間の隠蔽により、１TB 級のデータを可視化する際でもデータ読込時間をレンダリング時間に隠蔽できる。

2.3 入力データと実行方法

現在、MovieMaker が入力として受け取るシミュレーションデータは等間隔あるいは不等間隔のカーテシアン座標系でのスカラーデータおよびベクトルデータである。データ型は 4 バイト実数とした。

MovieMaker の実行は、可視化処理内容の詳細をテキストで記述した設定ファイルに基づいてバッチ処理で行う。設定ファイルには、各データファイルのパスやデータのメッシュサイズといった必須事項の他に、視点位置や角度、投影関係やライティング関係のパラメータ、後述する各種可視化手法の詳細なパラメータ等を指定できる。視点位置や角度はステップ毎の記述が可能であり、フライスルーのような動画作成も容易である。

2.4 実装

MovieMaker は C++言語で記述し、ポリゴンレンダリングは OpenGL のオフスクリーンレンダリングで実装した。現在の MovieMaker で実装済みの可視化機能は、ボリュームレンダリング、等値面生成、流線追跡の三種である。これらの機能は単独だけでなく、複数を同時に使用することも可能である。各機能について以下で簡単に説明する。

1) ボリュームレンダリング機能

近年では、ハードウェア処理による３D テクスチャマッピングを用いた高速なボリュームレンダリングの実装[10]が可能になったが、ハードウェアへの依存性と画質の問題を解決できなかったため、我々はレイキャスト法に基づくソフトウェアレンダリングを採用した。パラメータとして色調以外にも、光線追跡精度やアーリーレイターミネーションの閾値も指定できる。レンダリング時間の短縮と I/O サイズの縮小に有効なデータ圧縮とエンコーディングなどの手法[11]は今のところ採用していない。

2) 等値面生成機能

等値面生成はよく知られた Marching Cubes 法[12]に基づいて実装した。設定ファイルで指定した複数の閾値に対して、それぞれ色調の異なる等値面生成が可能である。

3) 流線追跡機能

流線（力線）追跡は 6 次精度の Runge-kutta 法に基づいて実装した。複数のベクトルデータに対し、任意の本数の流線について始点と色調を指定できる。追跡精度や追跡ステップ数も指定できる。

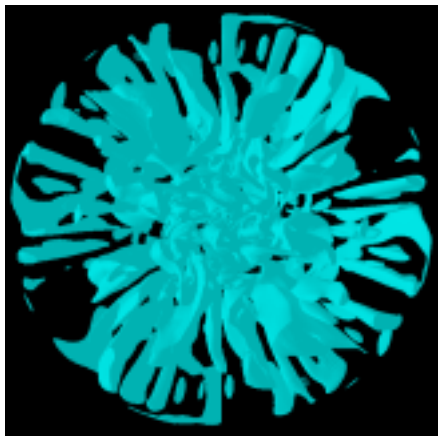

Fig.3 Sample Image Using Isocontouring

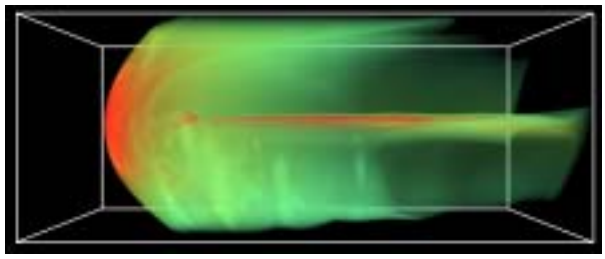

Fig.4 Sample Image Using Volume Rendering (the Earth's Magnetosphere)

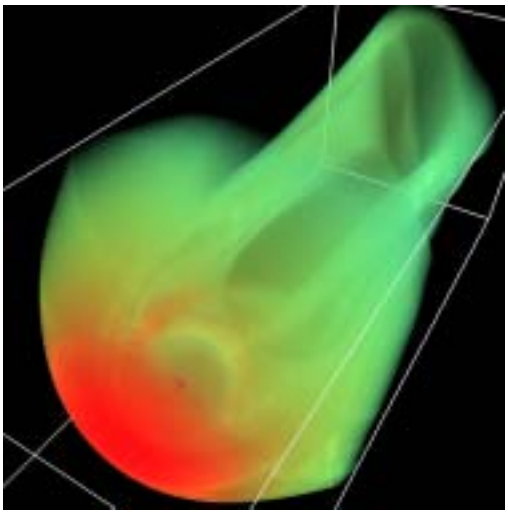

Fig.5 Same as **Fig.4** Viewed from Different Angle

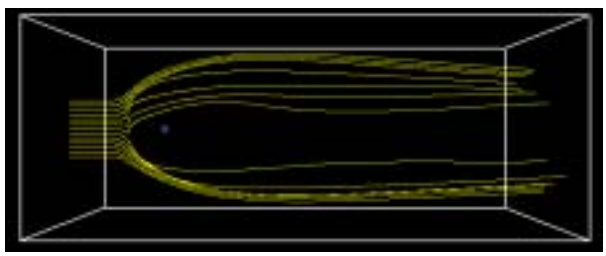

Fig.6 Sample Image Using Streamlines

３．性能評価

SGI Onyx 3800 (12CPU, 主記憶 24GB) 上で MovieMaker の性能測定を行った。今回の評価では 11 プロセス（マスタプロセス１、スレイブプロセス 10）での実行とし、テストデータには地球ダイナモシミュレーションの出力データを用いた。テストデータはメッシュサイズ 640×640×640、1 ステップ当たり約１GB で 10 ステップ分のデータである。各可視化機能を用いたテストの結果、画像１枚当たりの平均生成時間はボリュームレンダリング機能を用いた場合で約 40 秒、等値面生成機能を用いた場合で約 33 秒、流線追跡機能を用いた場合で約 10 秒であった。この時に出力された画像のサンプルを **Fig.3** に示す。この画像は渦度を等値面生成機能で可視化したものである。各可視化機能の設定条件により、処理時間の増減は予想されるが、今回の性能評価実験では **2.1** 節で述べた「画像１枚当りの平均生成時間を 42 秒以内とする」とした当初の目標は達成できた。ただし、複数機能を同時に使用した場合の処理時間は、単独で使用した場合の処理時間の和となる。処理時間はアルゴリズムの改善などにより更に短縮可能と考えている。

４．MovieMaker による動画作成事例

地球磁気圏の MHD シミュレーションの出力データから、**3** 章の評価実験と同じプロセス数にて作成した連番画像の例を **Fig.4** から **6** に示す。使用した出力データは、メッシュサイズが 502×202×202、72 ステップ分のデータである。**Fig.4** と **5** は温度分布をボリュームレンダリング機能で可視化したものである。透明度の調整により、高温部のみが表示されている。**Fig.6** は太陽風の速度場を流線追跡機能で可視化したものである。地球磁気圏前面（太陽側）に生じたバウショックと呼ばれる衝撃波面が見てとれる。ここで用いたデータのサイズは我々が最終目標とする１ステップ当り１GB というサイズよりも小さいため、この動画の 1 フレーム当りの生成時間は約 10 秒前後であり、かなり高速である。

５．まとめ

地球シミュレータで生成されるような大規模シミュレーションデータから、動画用連番画像を高速作成するシステムとして MovieMaker を開発した。我々は 1 ステップ当り 1GB のデータに対しては、まだ最大 30 ステップ分までしか MovieMaker による動画生成を行っていないが、これまでのテスト結果から「１TB の数値シミュレーションデータから一晩で一連の動画用連番画像を生成する」という開発目標は充分達成できると考えている。今後は、表現機能の追加や球座標系への対応を検討している。

謝辞

地球磁気圏の MHD シミュレーション結果を御提供いただきました名古屋大学太陽地球環境研究所の荻野竜樹教授に感謝いたします。

参 考 文 献

1) Web-site http://www.es.jamstec.go.jp/ .
2) 上原均、川原慎太郎、大野暢亮、古市幹人、荒木文明、陰山聡: MovieMaker: 大規模・高解像度データの高速動画作成システム、地球惑星科学関連学会 2005 年合同学会 (2005) CDROM-Proceedings J031-005.
3) Uehara, H.,Kawahara, S.,Ohno, N.,Furuichi, M.,Araki, F. and Kageyama, A.: MovieMaker: A High Performance Movie-Making System for Large Scale Simulations, In Proc. of 19th International Conference on Numerical Simulation of Plasma and 7th Asia Pacific Plasma Theory Conference (2005) pp.42-43.
4) Crockett, T. W.: Parallel Rendering, NASA CR-195080, ICASE Report No.95-31 NASA Lagley Research Center Hampton, VA 23601-0001 (1995).
5) Wittebrink, C. M.: Survey of Parallel Volume Rendering Algorithms, In Proc. of International Conference on Parallel distributed Processing Techniques and Applications, Las Vegas, Nevada (1998) pp.1329-1336.
6) Takayama, M.,Shinomoto, Y.,Goshima, M.,Mori, S., Nakashima, Y. and Tomita, S.: Implementation of Cell-Projection Parallel Volume Rendering with Dynamic Load Balancing, Int'l Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA2004) (2004) pp.373-379.
7) Yu, H.,Ma, K. L. and Welling, J.: I/O Strategies for Parallel Rendering of Large Time-Varying Volume Data, In Proc. of Parallel Graphics and Visualization (2004).
8) Fujishiro, I.,Chen, L.,Takeshima, Y.,Nakamura,Y. and Suzuki, Y.: Parallel Visualization of Gigabyte Datasets in GeoFEM, J. of Concurrency and Computation: Practice and Experience, vol.14, No.6-7 (2002) pp.521-530.
9) Web-site http://www.mpi-forum.org/ .
10) Cabral, B.,Cam, N. and Foram,N.: Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware, In Proc. of Symposium on Volume Visualization (1995) pp.91-98.
11) Ma, K. L. and Lum, E.: Strategies for Visualizing Time-Varying Data, In Proc. of CAD/Graphics 2001 (2001).
12) Lorensen, W. E. and Cline, H. E.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm, In Computer Graphics (SIGGRAPH '87 Proceedings), Vol.27 (1987) pp.163-169.