C-05

# 非構造格子ボリュームデータの可視化における動的負荷分散

# Dynamic Load Balancing on Visualization of Unstructured Grid

高山 征大 *, 丸山 悠樹, 木下 純, 森 眞一朗, 津邑 公暁, 五島 正裕, 中島 康彦, 富田 眞治

## 1 はじめに

我々は，数値シミュレーションとその結果の可視化を同じシステムで行う，分散メモリ型並列計算機の構築に取り組んでいる．このとき，可視化すべきデータはシミュレーションによって各計算ノードに分散して置かれる．そのため，可視化側にデータの配置の自由が与えられていない．シミュレーションとその可視化を行う同様のシステムとして，GeoFEM 可視化サブシステム [1] があるが，これは共有メモリ型並列計算機であるため，上述のような問題はない．また，分散メモリ型並列計算機で並列可視化を効率よく行うアルゴリズム [2] も提案されているが，データを自由に分配できることが前提となっている．

本稿では，上述のようなデータの配置に自由がない場合での並列可視化についての考察を行う．また，対象とするボリュームデータは，単純な構造格子 (図 1-a) ではなく，有限要素法などが出力する非構造格子 (図 1-b) であるとする．

## 2 シミュレーションと連携した並列可視化

### 2.1 ボリュームデータの分散配置

(1) まず，可視化処理を行う前に，全てのボリュームデータについて位置情報などを計算し，それを基に，可視化に適したようにデータを静的に再分配する方法が考えられる．この方法だと，可視化処理に集中することができる，従来の並列可視化アルゴリズムを使うことができる，といった利点がある．その一方で，ボリュームデータは一般に非常に大きく，分散メモリ型並列計算機だとその転送にかかる時間が問題となる．我々は，対話的なシミュレーションを支援するための可視化システムを目標としているため，可視化の前処理がボトルネックになるのは望ましくない．

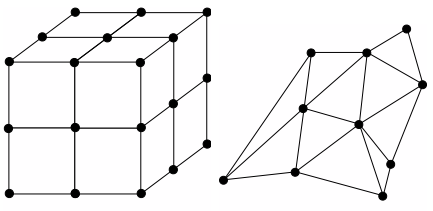


図 1: 各種格子

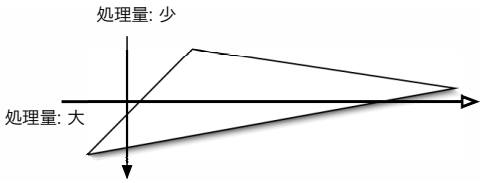


図 2: 非構造格子における視線方向での処理量の差

(2) 逆に，前処理によってデータの再分配をせず，各ノードは自分の持っているデータのみを処理し，持ちデータがなくなった時点で他のノードからデータを受け取って処理するといった動的な方法が考えられる．この方法だと，可視化処理中に通信割り込みが発生してしまい，処理の中断を招くが，大きなデータ転送という前処理を必要としない．

ボリュームデータは，cell と呼ばれる多面体の集合であり，個々の cell はその頂点にデータの値と座標値を持つ．構造格子は図 1-a に示すような立方体の cell から成っており，非構造格子ボリュームデータは，図 1-b に示すような多面体の cell から構成されている．従って，構造格子でも非構造格子でも，各 cell が持つデータ量は均質である．しかし，非構造格子の場合，cell の大きさは各 cell 毎に異なるため，可視化処理を行う計算量は cell 間において非均質である．また非構造格子においては，その cell を見ている方向によっても，可視化処理の計算量は著しく異なる (図 2).

我々の想定しているシミュレーションは巨大なデータを生成する一方，ノード間のネットワーク速度は低速であること，また非構造格子を対象としていることから，(2) が適していると考えられる．

### 2.2 ノード構成

1 つの制御ノードと N 台の計算ノードから成る構成とする．制御ノードは，各計算ノードの動作状況を busy 配列として持ち，最初に全ノードのフラグを立てる．各計算ノードは，自分の持っている範囲のデータを処理し終えたら制御ノードに idle 通知をする．それを受けた制御ノードは対応する計算ノードのフラグを落とす．次に，busy フラグが立っている計算ノードを一

* 京都大学

つ選び，idle 計算ノードに負荷を移送するように通知する．この idle ノードの選択はラウンドロビン方式で行えばよい．

計算ノードが処理中の負荷を移送しないよう，負荷は何らかの順で並べておく．処理は先頭から昇順に進め，移送する負荷は最後尾から降順でとっていくものとする．

制御ノードを設けずに，各計算ノードが idle になったときに idle 情報をブロードキャストする方法も考えられるが，そのためにはノード間で受け取りノードに関する適切な同期機構が必要となるため，この方法は採用しない．

### 2.3 可視化手法

ボリュームデータはシミュレーション側によって既に各ノードに分散配置されているが，同じノードに隣接する cell があるとは限らない．ボリュームレンダリングで一般的なアルゴリズムである Ray Casting 法といった，スクリーンからオブジェクトに処理を進める方法は，処理の単位がスクリーン上の 1 ピクセルの値を求めることであり，隣接する cell に頻繁にアクセスするため，我々の場合には適さない (図 4-a)．従って我々は，cell を一つずつ処理し，その結果をスクリーンに投影する projection 型のアルゴリズムを採用する (図 4-b)．projection 型のアルゴリズムには，Cell Projection [3]，Sweep Plane, Vertex Projection などがあるが，データの配置における自由度の高さから，Cell Projection を採用する．

### 2.4 並列処理の単位

Cell Projection のアルゴリズムを説明する．(図 5)

(1) 最初に，ボリュームデータを視線方向 (z 方向) に並べておき，各 cell をスクリーンに投影する．(2) 次に，頂点の関数値に基づいてスクリーン上で走査線変換を行い，その cell がスクリーンに寄与する領域の関数値と z 値を計算する．(3) そして，各 cell がスクリーンに投影した平面を z 値に基づいて合成し，最終画像を得る．

次に，移送する並列処理の単位について考察する．

Cell Projection では，cell を処理の単位として可視化が行われていくので，cell を並列処理の単位とすることがまず考えられる．この場合，2.2 で述べた負荷の順は，cell の視線方向の順番

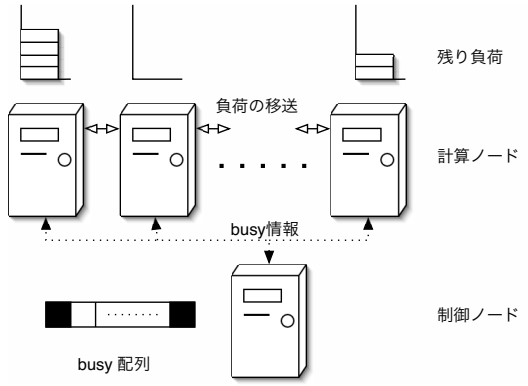


図 3: ノード構成

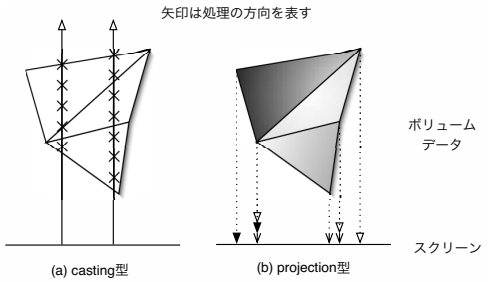


図 4: ボリュームレンダリング処理の方向

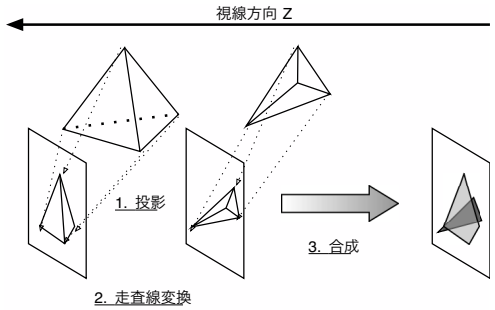


図 5: Cell Projection

とすればよい．つまり，可視化処理は手前の cell から行い，他の計算ノードに渡す cell は奥から行う．このとき，どれだけの cell を一度に渡せばよいかは検討の必要がある．

また，各 cell による中間画像を合成する処理を並列処理の単位とすることも考えられる．この場合，合成を中間画像の走査線単位で行うか，平面単位で行うか，といった選択が考えられる．

## 3 おわりに

今後は，ここで述べた並列可視化システムを実装し，並列処理の単位，移送する負荷の量などの比較検討を行う．

## 参考文献

[1] L. Chen, I.Fujishiro, and K. Nakajima. Parallel Performance Optimization of Large-Scale Unstructured Data Visualization for the Earth Simulator. Fourth Eurographics Workshop on Parallel Graphics and Visualization 2002.

[2] K.L. Ma and T.W. Crockett. A scalable parallel cell-projection volume rendering algorithm for three-dimensional unstructured data. , IEEE Parallel Rendering Symposium, pages 95–104. IEEE, November 1997.

[3] Nelson Max, Pat Hanrahan, and Roger Crawfis. Area and volume coherence for efficient visualization of 3d scalar functions. Computer Graphics, 24(5), December 1990. San Diego Volume Visualization Conference Proceedings.