56

クラスライブラリによる並列化実時間可視化システムの構築

白山　晋（高度情報科学技術研究機構），太田高志（日本原子力研究所）

Parallelized Real-Time Visualization System by a Design of a Data Class

by

Susumu SHIRAYAMA (RIST) and Takashi OHTA(JAERI)

ABSTRACT

In order to develop a flow analysis system efficiently, it is important for a visualization system to keep independence from a flow solver. However, since a visual computing plays more important role to improve numerical methods, both flow and visualization systems have to exist together, especially on parallel computers. Moreover, real-time visualization system is helpful to analyze the flowfield. We propose one solution using a concept of a class library.

1. はじめに

計算流体力学(CFD)における可視化手法（技術）は成熟し，可視化システムとして普及の段階にある．このため，可視化に対する研究は停滞しているようである．可視化手法は，データの存在位置の表示，データの接続情報の表示（格子線，格子面表示），グラフ表示，等値線，等値領域，等値面，ベクトル表示，粒子追跡法の８種類に分類される．それぞれにいくつかの変化形があり，また，組み合わせによる可視化がなされているが，核となる方法が確立しているために，組み合わせ法やシステムの提示のみでは研究というよりも作業とみなされることが多くなっている．計算結果そのものの表示から，可視化をもとにした分野毎の解析（狭義の可視化翻訳）が主流となっていることも一般的な可視化手法，システムの研究の終焉が近いことを示しているのかもしれない．

ところが，近年，大規模計算から掃き出される巨大なデータの処理に対して，既存の可視化システムの不備が指摘されはじめた．どのような不備があるのかについて明確な報告はないのだが，計算機環境が分散並列計算機へパラダイムシフトしていくことから新たな可視化システムの構築が提案されているようである．そうした風潮の是非はともかく，これを機に既存の可視化手法，システムを見直してみる．

先に可視化システムは普及していると述べたが本当だろうか？　例えば，組織を単位とすればその観察は正しい．しかし，個人を対象とすれば，誰もが使えるようなものとはなっていないようである．よって，可視化専任者の需要が高い．結果を解析する力のないものの可視化はただの作業とみなされても仕方がない．好ましいことではないが，そうした状況を目にすることは多い．これを改善するにはソフトウエアとしての完成度を更に向上させることが必要である．ただし，この課題は教育による解析能力の向上，可視化技術の習得によっても解決できるはずである．

技術的な面でみた既存の可視化の問題点を探ってみる．

A 可視化手法に不具合はないのか?

B データベースの蓄積，検索技術等は完備しているのか?

C 計算環境の変化に追従しているのか?

可視化手法の不備に関しては，可視化の精度からのもの[1]と可視化操作に関連した不具合がある．数値解法と整合性のある可視化手法（可視化精度），粒子追跡法における粒子発生点の選択手法（可視化操作）などを考察する必要があるだろう．

データベースの蓄積は可視化に関わらず，CFD全体の問題であるが，特に画像データの収集と検索が大きな問題である．オブジェクトデータベース，データマイニング，数理的な考察からの特徴領域の自動抽出などが研究の対象となりそうである．

計算環境の変化というのは，一千万から一億自由度といった大規模計算が実現していること，並列計算環境，バーチャルリアリティ(VR)などの新しい可視化装置が普及しはじめていることから生じている．この変化に対応するためには

- 並列化可視化
- 実時間可視化
- ビジュアルコンピューティング
- 多様化する計算システムに柔軟に対応するパラダイム

を考察することが重要である．

本稿の目的は計算環境の変化に柔軟に対応する並列化実時間可視化システムの設計指針を示すことである．可視化システムの設計において重要な問題の一つにソルバーとの独立性をどのように保つかということがある．ソルバー自身が開発途中である場合，可視化ライブラリをソルバーに組み入れることは，開発環境を制限することになり好ましいことではない．一方で，ビジュアルコンピューティングによる支援環境を取り入れないと開発効率が上がらないことは事実である．この状況は並列環境であればなおのことである．ゆえに並列環境ではソルバーと可視化システムを独立に構築し，相互を補完しあうといった仕組みを考えることは重要である．

一つの解決策としてクラスライブラリによる並列化実時間可視化システムを提案する．具体的にはソルバーと同一のデータクラスを利用することが考えられる．特にそのデータクラスがメモリのレベルで共有できれば，並列化及び実時間可視化を簡単に行うことができるだろう．そうでない場合でも可視化システムをクラスライブラリにすることによって従来型の可視化ライブラリの埋めこみによる方法などを含めて，様々なシステムに柔軟に対応する可視化システムを構築することができる．同一の構想で格子生成システムを作成すれば，プリポストとソルバーを同一の思想で設計することが可能となり，プリポストを意識しないシームレスな実行環境を構築できる．

2. 理想的な可視化システム

前節の内容と多少重複するが，理想的な可視化システムについてまとめておこう．

- 分散並列計算環境を含めてどんな計算機環境にも柔軟に対応できる
- 本計算部と独立である
- ビジュアルコンピューティングを容易に実現できる
- 実時間可視化が実現できる
- ステアリングが実現できる
- 動画作成を含めた操作が容易である
- 機能追加が容易である
- データベースとのアクセスが自由である
- VRなどの新しい可視化装置に対応できる

これに特徴領域の自動抽出機能が加われば，しばらくの間，可視化システムを議論することはなくなるだろう．そのような可視化システムを構築するためには，ソフトウエア面で解析システム全体の変革が必要になる．オブジェクト指向やエージェント指向が一つの考え方である．しかしながら，システム全体を変革することは資産（プログラム資産，プログラミング資産など）の利用という面で難しいことは容易に予想できる．そこで，どういった概念が必要かを以下の節で示していく．

3. データクラスの利用

3.1 データクラスの考え方

オブジェクト指向ではボトムアップ的な抽象化を行う．これを流体解析における場に適用したものが図1である． もちろん，こうした分類には任意性があるし，実装まで考察の対象にすれば，データと流れ場の計算をカプセル化する方法も有効である[2]．ここでは，本計算部と独立した可視化システムの構築がデータの分離により実現できることを強調した．独立に作られた本計算部と可視化部をカプセル化することは後で考えればよい．

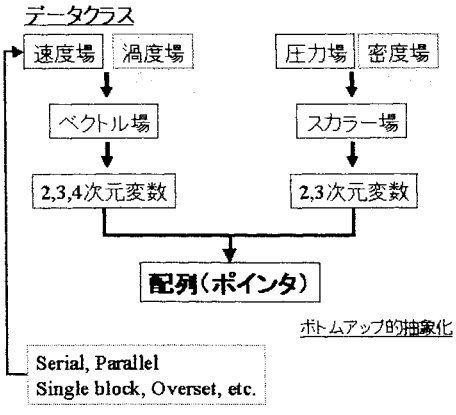

図1. オブジェクト指向によるフィールドデータの抽象化

図1で重要なのはボトムアップ的な抽象化によって，流れ場を示すデータクラスをポインタとして示すことができる点である．ポインタの先にあるデータ構造は実体と接続情報を引き出すことで確定する．並列化のためのデータであるとか，多重連結によるものである等はデータクラスが吸収し，ほとんどの可視化手法の実装は生のデータから幾何データを抽出することに集中することで実現できる．また，データクラスのポインタに直接辿り着くことができるので，本計算部と同一のデータクラスであれば実時間可視化が容易に実現できる．

この考えを根底におけば，図2に示す統合的な解析システムの構築も可能となるはずである．この場合，ほぼ三つのデータクラスを軸にシステムを構築すればよいことがわかる．それらは，

- 格子や流れ場などのフィールドデータのクラス
- グラフィックスデータのクラス
- ビジュアルデータのクラス

ビジュアルデータのクラスとしては，可視化結果そのもの，生のデータ，外からのデータ（実験，観測，他の計算）を想定している．

これから先の議論はデータクラスが存在しているものとして行う．流れ場の解析システムにおけるデータクラスの詳細な考え方は文献3,4を参照してほしい．

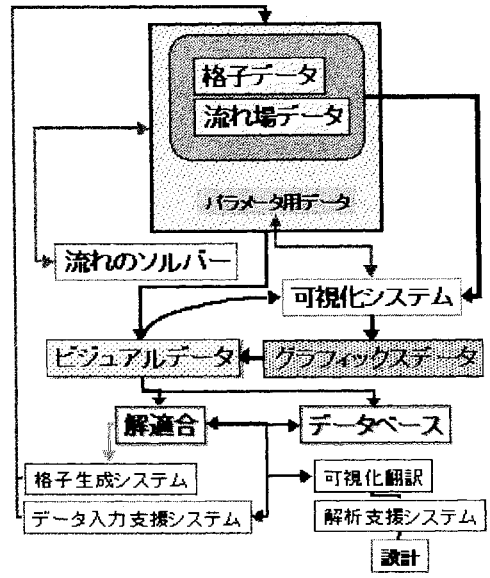

図2. データクラスを中心とした統合的開発環境

3.2 データクラスと計算機環境

計算機環境としては，PCやWSを含めた単一CPUのもの，共有メモリを用いた並列機，分散メモリによる並列機の三つを考える．図3にソルバーからポストへのデータのながれを示した．図に示す様々な形態のうちいくつかのものは具現化されている．

ソルバーの実行環境と可視化処理の環境は，同一である場合，独立である場合の二通りを考えてある．また，並列機においては，可視化プログラムがソルバーとCPUを共有する場合，分散する場合の二通りを考慮する．データはメモリまたはディスクに存在するものとし，並列化I/Oによるデータ転送も考察の対象とする．図中の前処理は主としてデータ圧縮操作を示す．

並列化に着目していくつかの計算機環境を挙げておく．本計算部に可視化プログラムを搭載するものとしては，

- 計算と可視化を同一ノードで行うもの
- データを一ヶ所に集めて可視化処理を行うもの

- 数箇所の可視化専用ノードに対してデータを再分散して可視化処理を行うもの

が考えられる．これらの形態は本計算において可視化対象となるデータの大きさによって使い分けられる．実際には計算と可視化の負荷分散という立場で可視化サブシステムを用いる場合が多い．後述するように可視化処理の計算時間は本計算に比べて小さいことから，可視化サブシステムは本計算部の1/100程度の計算能力のものを利用することが多い．本計算部と可視化サブシステムにはメモリを共有できる仕組みがあれば理想的だが，通常はファイルを介在することでデータのやり取りが行われる．可視化のために並列サブシステムが利用されるのは，大規模計算によって得られたデータが一つの計算ノードの記憶容量を越える場合を想定してのことである．

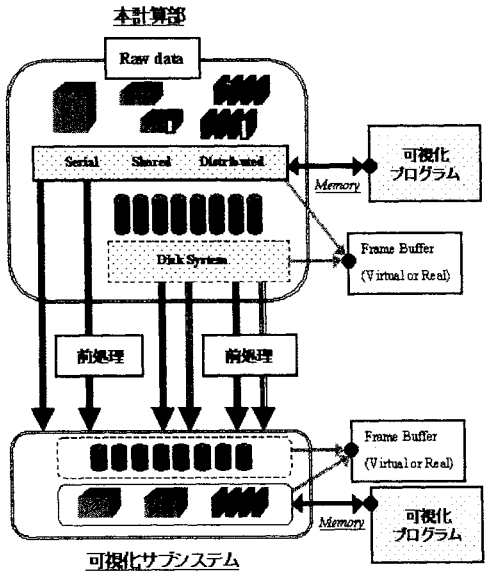

図3. データのながれ

可視化対象データに着目しながらシステムをまとめると主として三つのものとなる．

- 生のデータ（本計算からのデータ）をファイル化したシステム
- ソルバーの中に可視化関数を加えるもの
- 可視化計算環境にソルバーを埋めこむもの

現在最も広く用いられているシステムは，対象データを一旦ファイル化してしまうものである．このシステムを用いれば計算プログラムと可視化プログラムを分離することができる．しかしながら，実時間可視化に適さない，分散型並列機におけるデータ収集の問題など，実時間，分散並列という点を強調する場合，可視化システムとして十分な機能があるとはいえない．また，近年の大規模計算が分散並列環境に移行しつつあることから巨大ファイルの操作という問題が生じる．一方，ファイルを介在することなくビジュアルコンピューティングを実現したシステムとしては，ソルバーの中に可視化関数を加えるタイプのものと可視化計算環境（AVSなど）にソルバーを埋めこむものがある．ソルバーに可視化関数を加えるものはソルバー自身の修正が必要になる．可視化計算環境を利用するものでも，その環境に合わせたソルバーの修正を要求される．

ファイルを介在するシステムの考察から，ソルバーと可視化プログラムに共通するデータクラスを定義すれば，データクラスの部分を除いてソルバーと可視化プログラムの独立性を保つことができることがわかる．並列化はデータクラスで吸収する．データクラスへのアクセスに関しては様々な方法が考えられるだろう．その点は実装の段階で効率的な方法を採用すればよい．ファイルを介在する場合はオブジェクトそのものをファイルに書き出せばよい（ビジュアルデータクラス）．そうすることでオブジェクトデータベースの概念も利用できる．理想的にはメモリを共有できるような仕組みが実現できればよい．ファイルを利用してしまうと現在のシステムと変らないと指摘されるだろうが，データクラスという概念を用いることで，様々な計算環境に柔軟に対応できるという点を強調しておきたい．例えば，あるシステムではファイルを介在する必要があるが，別のシステムではメモリを共有できるとした場合，最小限のプログラムの書き換えで双方に対応する可視化システムを構築できる．

計算機環境としては画像表示装置まで考える必要がある．表示系（レンダリングと表示そのもの）まで含めると，生のデータ，幾何データ，表示用グラフィックスデータ（ラスター型，ベクトル型）の三種類のデータを考える必要がある．特に二番目と三番目のデータの配置がデータ圧縮や並列環境を考えた場合に重要となる．表示装置と幾何データの抽出やレンダリングのための計算機は同一である場合と異なる場合の双方を考慮する必要がある．幾何データ，表示用グラフィックスデータをグラフィックスデータのクラスとして定義することで，可視化における幾何データの計算と表示を分離することができる．

4. システムの構築

可視化システムの設計に際して押さえておくべきことは，

1. 可視化によって生じるデータ量と通信量
   - どのようなデータがどの程度生じるか
   - 大規模計算に対応できるか
2. 可視化の計算時間
   - もっとも普及している計算機でどの程度の可視化が可能なのか
   - 大規模計算に対応できるか
3. 実時間可視化の可能性
4. 可視化技術は十分なのか
   - 比較，取り込み技術
   - 並列化技術

である．

4.1 データ量と通信量

実時間可視化，並列化可視化を含めたシステムを設計する場合，計算データ（生のデータ）から，幾何データ，表示用グラフィックスデータがどのプロセスで生じるか，それぞれのデータの配置，表示用グラフィックスデータの種類，可視化処理の種類を把握しておく必要がある．図4に可視化処理のながれを示す．図中の仮のボクセル空間（データ）は，計算データを直交等間隔格子上で再定義したものと考える．このボクセルデータの生成に可視化処理のアルゴリズムを利用する場合もある[5]．幾何データは領域設定等に再利用されることもある．このため接続情報の作成が必要な場合が生じる．このような幾何データは，二次的な処理を経て別種の幾何データを生成する．このため再帰的な定義のためのタグや接続情報もデータ量として見積もる必要がある．

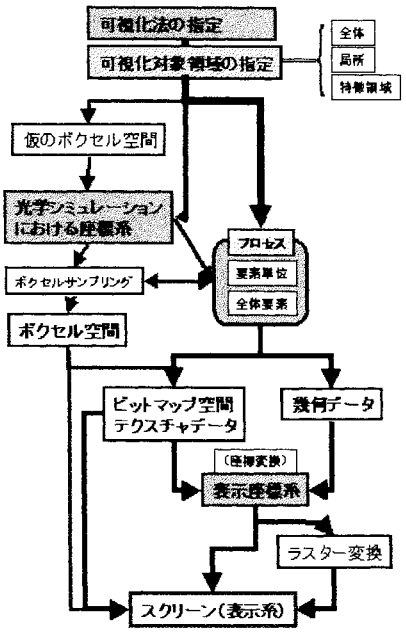

図4. 可視化処理のながれ

さて可視化によるデータ量を見積もってみよう．三方向にほぼ同数の点である三次元データを扱う．総格子点数を n とすると次のようにデータ量を見積もることができる．

・ベクトル型

(a) 点や線による可視化結果から

・格子図，ベクトル図，等値線図

$(n^{\frac{2}{3}}) * O(1)$

・仮想粒子による可視化

発生粒子数*積分ステップ数

(b) 多角形による可視化結果から

・格子面図

$(n^{\frac{2}{3}}) * O(1)$

・等値面図

$(n^{\frac{2}{3}}) * 4*$枚数$*O(1)$

・ラスタ型（イメージデータ）

画素数*3bytes　（例）640*480*3：約1MB

座標 (x, y, z)，流れ場の変数 (u, v, w, p) の7変数が1千万格子点の計算で得られたとすれば，一時刻あたり単精度で280MBのデータとなる．このデータから1枚の格子面と3枚の等値面を表示したとすれば，幾何データは約2.6MBとなる．幾何データはもっとも単純な圧縮（4bytesから2bytesデータへの変換）によって半分に，さらにポリゴンリダクションなどの圧縮法を用いれば，数十分の一になる．仮に2.6MBのデータを低速のLAN(10Mbps～100Mbps)で転送させるとすれば1秒程度である．また，比較的データが多くなる粒子追跡では，1000個の粒子を1000ステップ追跡した結果のデータが4MBであるから，この場合も2秒程度で転送ができる．一般的にはデータ量が固定されているラスタ型のものがデータの転送をともなう可視化システムでは好まれる．640x480の解像度であれば，常に1秒以下でデータが転送される．この形式はデータの通信量を考えれば有利であるが，視点変換が重要になる解析では幾何データを扱う必要が生じる．例えば，大規模計算における流れの可視化を想定する場合は，全体的な現象と局所的な可視化は関連するものの，データとしても見ると空間のコヒーレンスが良いとはいえず[6]，常に拡大縮小といった操作が必要になる．また，小規模の粗い計算と大規模計算での可視化は手法として同じものを用いることになるので可視化翻訳の部分にまで踏み込まなければ視点を固定した可視化を行うことは少ない．つまり，視点変更に対応するためのシステムの柔軟性は重要な要素となる．数秒で幾何データが転送できる範囲では，表示系に三次元機能を持たせた方が効率的であろう．

通信量の削減は可視化手法を問わず重要な問題である．削減のためには，

- 一般的なデータ圧縮
  1. 生データの圧縮
  2. 画像データの圧縮（プリミティブ型）
  3. 画像データの圧縮（ラスター型）
- CG技法の応用
  1. ポリゴンリダクションの利用
  2. テクスチャマッピングの利用
  3. カリング処理など

等を考察する必要がある．ラスター型画像データの圧縮[7]，テクスチャマッピング[8]を取り入れた可視化システムは存在する．

4.2 可視化の計算時間

データ量と通信量に加えて，そのデータを生成するための可視化処理の計算時間を見積もってみる．もっとも普及している計算機でどの程度の可視化が可能なのかがわかれば図4に示したプロセスをどういった計算機で行えばよいのかがわかる．基準となる計算機として，PentiumII 400MHzのパソコンを用いた．

可視化手法の中で計算時間を必要とする等値面処理とボリュームレンダリング法を扱う．粒子法の結果を可視化対象とする．ある物理量に対してそれぞれが局所分布をもっている10^4個の粒子を考える．個々の粒子の分布を直接表示することも可能だが，多くの場合は局所分布から場が形成されているので，はじめにその場を補助空間を利用して算出する．補助空間を101x101x101の百万点の格子として場を計算し，三枚の等値面を描いたのが図5である．等値面はポリゴンではなく三角形の枠を線として表示した．等値面探索に要した時間は2.73秒であった．

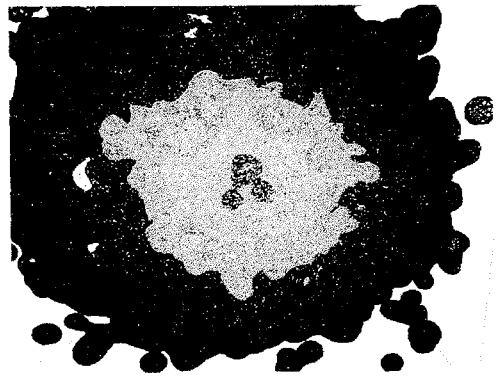

図5. 10^4個の粒子に対する場の分布

次にボリュームレンダリングによって場を可視化してみる．ボリュームレンダリングとしてはスライス法[5]を用いる．ス

ライスの数は150枚，ピクセル数は640x480である．計算に要した時間は，約200秒であった．

図6. ボリュームレンダリングによる場の分布の可視化

4.3 実時間可視化

本稿では理想的な実時間可視化を以下のように定義しておく．

「計算しているプロセスがあるとすると，可視化のプロセスがメモリに直接アクセスして，現在進行中の計算プロセスにインターラクティブに関与できるような可視化を理想的な実時間可視化とする」

実時間可視化はデータクラスの利用によって実現が容易であることは先述した通りである．つまり，同じインスタンスを利用するだけである．計算パラメータのクラスを用意し，そのインスタンスを共有できるようにすれば，ステアリング処理も簡単である．

しかしながら，すべての計算システムでメモリへの直接的なアクセス権が提供されているわけではない．可視化システムを構築する場合は，この点を前もって調べておく必要がある．もし直接アクセスできない場合は，リアルタイム性は犠牲になるが，ファイルを介在させるとか，可視化ライブラリを埋め込むなどの方法に切り替えればよい．ここでも可視化システムの多様化が生じるが，オブジェクト指向の考えを用いればその多様性を最小限のプログラム変更で吸収できる．

プロセス間でメモリ共有を許す計算環境（2CPUのパソコン）で実時間可視化のための数値実験を行った．本計算のプロセスと可視化プロセスを同時実行し，フィールドデータとその属性（時間ステップ）の共有指定以外は本計算と可視化は完全に独立している． 図7に結果を示す．

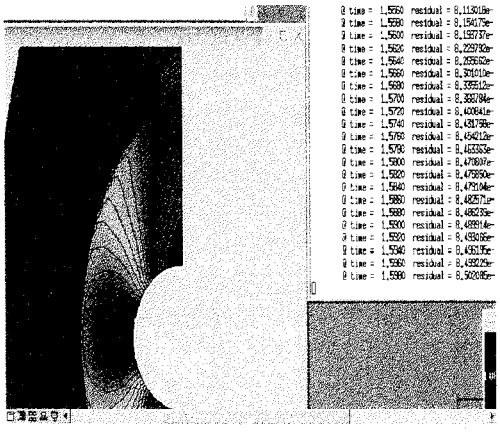

図7. 実時間可視化の例

図の左に示す可視化の結果は可視化システムからの出力である．右は本計算からの数値解の収束履歴などの出力である．

この例では，可視化の方で，常に時間ステップを監視し，時間ステップが変化した時点で描画作業が開始されるといった仕組みになっている．ただし，本計算に対するデバック作業などでは同期をとる必要がないので，フィールドデータの格納されているメモリの状態をリアルタイムで表示することも可能である．

4.4 並列可視化

分散並列環境下での大きなメモリを必要とする大規模計算を想定する．可視化作業で必要な変数の数と本計算に用いる変数の数は異なる．物体適合座標系を用いた三次元の差分計算では，座標値と一つのスカラー値を可視化作業でのものとすれば本計算の1/4程度である．もし最大限のメモリを用いた4つ以上のノードによる計算を行った場合，可視化のためのデータを一つの計算ノードに集めることができない．もちろん，大規模計算というのは最大限のメモリを用いたものばかりではないし，座標値と一つのスカラー値の数十倍の変数を使用した計算であれば，一つのノードにデータを集めることはできる場合もある．先述したように可視化処理時間の問題は小さいと考えられるので，生のデータを集める方法の方が効率の良い場合も多い（可視化の並列化が不要ということ）．可視化に対して並列化が要求されるのは，可視化対象データを分散して存在させなければならない場合である．

可視化手法の並列化と一般のコンピュータグラフィックスにおける並列化の違いは，計算データが空間を占めていることに起因する．このため独自の並列化手法が必要になっている．並列化に対応した可視化システムは，計算データと表示用グラフィックスデータの配置に留意しながら設計される．

4.1節で示したように視点変更に対するシステムの柔軟性は重要な要素であるために，分散記憶型の並列機に対する効率的な可視化を考えると，領域の指定と可視化対象となる領域の局在化のためにロードバランスが崩れることが問題となる（図8）．但し，4.2節で示したように一般的な可視化手法に関しては可視化に要する計算負荷はそれほど大きなものではないことからロードバランシングがそれほど重要な問題とならない場合も多いことには留意すべきことである．

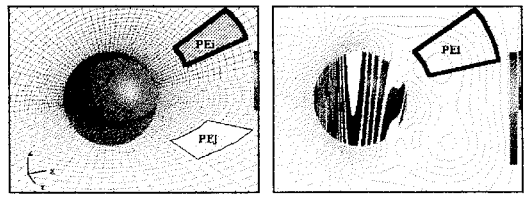

図8.1 ロードバランシングの問題1

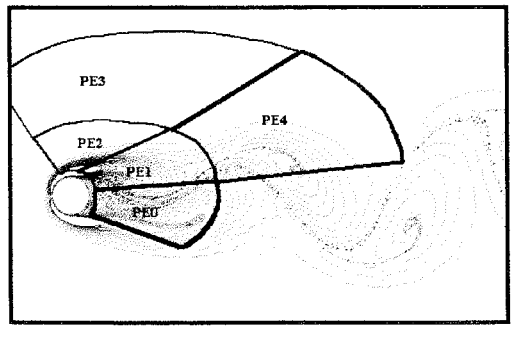

図8.2 ロードバランシングの問題2

可視化手法の多くは要素単位での処理が可能である．この場合，各ノードでの幾何データ生成までは問題なく並列化ができる．並列化に関する問題は，得られたデータから表示用のグラフィックスデータを生成するまでのレンダリング操作における隠線・隠面処理の段階で生じる．各ノードでzバッファをもつピクセルデータを生成し合成する方法，表示用のグラフィックスデータをベクトル型としてそれらを表示装置に送ることでノード毎の隠線・隠面処理を回避する方法が考えられている．

一部の粒子追跡法（図9）やボリュームレンダリングのような計算空間全体を対象とするようなものに対しては効率的な並列化手法を考えなければならない．隠面処理の効率的な並列化手法が見つかれば，ボリュームレンダリングの並列化の問題はおこらない．結局，並列化にともなう大部分の問題は，粒子追跡法と隠線・隠面の効率的な並列化処理法が提案できれば解決できるはずである．著者の知る限りでは，現時点においてそのような処理法はない（提案はあるが効率面で問題がある）．

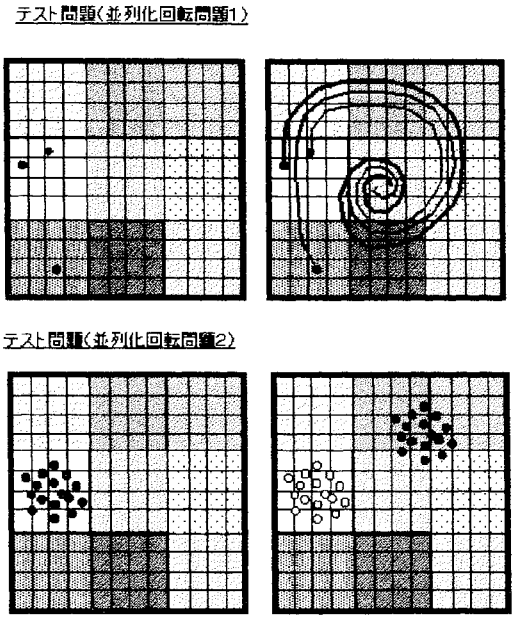

図9. 粒子追跡法の並列化

5. データクラスの役割

フィールドデータのクラス，グラフィックスデータのクラスが決定すれば（フィールドデータのクラスは文献3,4のものを用いる），システム構築において解決すべき問題は，データクラスに対する何種類かの可視化処理のクラスを定義することに帰着できる．可視化処理のクラスの構造は，フィールドデータのオブジェクトに働きかけてグラフィックスデータのオブジェクトを生成するといった単純なものでかまわない．データクラスを用いることで，可視化処理のクラスにおいて全体のデータ構造を意識する必要がなくなる．また，フィールドデータのクラスで並列化は隠蔽されているので[3,4]，要素単位での可視化処理は並列化を意識する必要がない．このため，メンバ関数として従来の手続き型の処理法を定義すればよい．

可視化処理から画像表示系への移行に対してはグラフィックスデータのクラスを用いる．このクラスの導入により並列効率の問題はあるが，データ圧縮，隠線・隠面処理を可視化処理とは独立に扱うことができる．また，表示装置に対する依存性の少ない可視化システムを構築することができる．

おわりに

データクラスの導入により，様々な計算機環境に柔軟に対応できる可視化システムの設計が可能となる．

並列化は，要素単位のものはフィールドデータのクラス，グラフィックスデータのクラスが吸収するので新たに考える必要はない．問題は粒子追跡法の一部，ボリュームレンダリングなどの全計算領域を対象とした可視化手法における効率の良い並列化手法の開発に集約できる．

実時間可視化という意味ではメモリを覗くことができるシステムが望ましい．この仕組みはシステム運用の問題さえ解決できれば実現可能である．

グラフィックスデータのクラスを用いることで，データ通信量の削減，表示装置に対する依存性を可視化処理と独立に扱うことができる．

並列環境で大規模計算を扱う場合は，不要なアルゴリズムの開発を避けるために，1PEの可視化処理時間を考慮した上でのシステム設計及び可視化システムの実効性能の見積もりを行うべきである．

参考文献

1. Shirayama, S., "Several sources of errors in numerical flow visualization techniques", AIAA-95-1714-CP, Proceedings of the AIAA 12th Computational Fluid Dynamics Conference, SanDiego, CA, June, 1995.
2. 城之内忠正，千葉　賢，"ながれ計算Javaコンポーネント:flow beans"，日本流体力学会年会'98講演論文集, July, 1998, pp.419-422
3. Ohta,T.,"Design of a Data Class for Parallel Scientific Computing", Scientific Computing in Object-Oriented Parallel Environments, Springer, 1997, pp.211-217
4. Ohta,T.,"An Object-Oriented Programming Paradigm for Parallel Computational Fluid Dynamics on Memory Distributed Parallel Computers", Parallel CFD '97
5. 白山　晋, 桑原邦郎, "仮想粒子密度とボリューム・レンダリングによる流れの可視化"，ながれマルチメディア論文集 Vol.**1**, July, 1998
6. 白山　晋，"大規模計算とボリューム・ビジュアライゼーション"，航空宇宙技術研究所特別資料SP-27,December,1994, pp.351-356
7. Doi,S., et al, NEC Research and Development, Vol.**37**, No.1, 1996, pp.114-123.
8. 小山田耕二，"VRMLを用いたCFD結果の可視化技術"，日本流体力学会年会'98講演論文集, July, 1998, pp.425-426