ハイパフォーマンス
コンピューティング 61－6
（1996．5．24）

並列マシン Cenju-3 上での流体解析のための実時間可視化システム

†村松 一弘，*武井 利文，**松本 秀樹，土肥 俊

NEC C&C 研究所，*NEC 汎用アプリケーション事業部，**NEC 情報システムズ

大規模流体解析と同時に解析結果を可視化する、実時間可視化システムを開発した。本システムは UNIX ベースのネットワーク分散環境に対応しており、計算サーバとしての分散メモリ型並列マシン Cenju-3 と、X および Motif が装備されているクライアントワークステーション (WS) で動作する。並列計算サーバ上で、流体解析とイメージデータ生成までの処理を行ない、解析結果の可視化表示と可視化のための諸パラメータの制御をクライアントで実行する。また JPEG のような画像圧縮技術を実装することにより、サーバからクライアントへの転送データの削減を図っている。これにより、バンド幅の狭いネットワークでもサーバからクライアントへのデータ転送がネックにならないと考えられる。

REAL-TIME VISUALIZATION SYSTEM FOR COMPUTATIONAL FLUID DYNAMICS ON CENJU-3 PARALLEL COMPUTER

†Kazuhiro Muramatsu, *Toshifumi Takei, **Hideki Matsumoto and Shun Doi

C&C Research Laboratories NEC Corporation
*Application Software Divison, NEC Corporation
**NEC Informatec Systems Ltd.

A real-time visualization software system has been developed for concurrent visualization of on-going large-scale flow simulation. This software runs in UNIX-base distributed environment composed of a Cenju-3 distributed-memory parallel computing server and a client workstation equipped with X and Motif. The flow simulation and generation of pixel images for display are performed on the parallel server, while graphical monitoring of the simulation results and steering of visualization parameters are performed on the client. Image compression techniques (such as JPEG) are also employed for the data transfer from the server to the client, so that the software can work on a standard low-cost network such as the Ethernet.

1 はじめに

計算機の高速化、ネットワーク分散環境の普及により、高速な計算サーバ上での計算結果をユーザのクライアント上でオンラインで可視化したいという要求が高まっている。従来このような要求を満たすシステムとしては、例えば MIT の Visual3 などがある [1]。そこでは、サーバ上でポリゴンなどのグラフィクス

†現所属；日本原子力研究所 計算科学技術推進センター

オブジェクトを生成し (マッピング処理)、それらをネットワーク上のクライアントに送る。クライアント上では、OpenGL などの汎用グラフィクスライブラリや IRIS などの専用ハードウェアを用いてレンダリング処理を行う。一方、レンダリング処理を並列コンピュータ上で行うライブラリが NASA Langley 研究所、MIT やカルフォルニア工科大学から報告されている [2]-[4]。

このような背景の下で、筆者らは解析計算からレンダリング処理までを計算サーバにて行い、生成されたイメージデータをユーザのクライアント上で実時間で可視化 (トラッキング) するシステム RVSLIB(Real-Time Visual Simulation Library) を開発してきた [5]-[7]。このシステムでは、クライアントの GUI からユーザが解析パラメータあるいは可視化パラメータを解析の途中で変更することも可能になっている (ステアリング)。さらにサーバーからクライアントへのイメージデータ転送の際に、画像圧縮技術 JPEG(Joint Photographic Expert Group) を用いることにより、ネットワークの負荷を軽減することが可能になっている。またこのシステムは X Window、Motif、Socket などの UNIX の標準的環境の上に構築されているため、高いポータビリティを持っている。

上記と同じ設計方針で筆者らは、RVSLIB とは別システムとして分散メモリ型並列コンピュータ上での実時間可視化システムも開発してきた [8]。分散メモリ型並列コンピュータへの指向は、従来の逐次計算機に比較し、演算プロセッサの増加による計算能力の高速化が可能であるという点で、プログラムが複雑になるという欠点を補ってあまりある可能性を有している、という理由からである。本報では、この並列コンピュータ上での実時間可視化システムを中心に解説を行なう。初めに、解析と可視化の分散方法の定量的比較を行ない、次に並列マシンに実装するための並列化の手法および本システムの構成について解説する。最後に、今後実現すべき課題について議論する。

2 解析と可視化の分散方法の定量的比較

解析から可視化までの一連の処理 (解析、マッピング処理、レンダリング処理) のサーバとクライアントへの分散方法には、次の 3 通りが考えられる。

アプローチ A 解析をサーバ、マッピング処理・レンダリング処理をクライアントでそれぞれ行う。サーバからクライアントに転送されるデータは解析結果である。

アプローチ B 解析とマッピング処理をサーバ、レンダリング処理をクライアントで行う。サーバからクライアントに転送されるデータは主にグラフィクスオブジェクトから成るグラフィクスコマンドである。

アプローチ C 解析・マッピング処理・レンダリング処理全てをサーバで行い、ピクセル表示のみをクライアントで行う。サーバからクライアントに転送されるデータはピクセルデータである。

ここでは、上記 3 つのアプローチのデータ転送量を簡単な例で定量的に比較する。簡単のために、計算格子は $n_{\rm gr} \times n_{\rm gr} \times n_{\rm gr}$の直交正方格子とする。

アプローチ A の場合 解析データとしては速度と圧力を取り上げ、各格子点上の速度および圧力のデータを、単精度実数としてクライアントに転送すると仮定する。各タイムステップで転送されるデータ量は、以下のようになる。

$$4 \times n_{\rm gr} \times n_{\rm gr} \times n_{\rm gr} \times 4 \text{ bytes} = 16 \times n_{\rm gr} \times n_{\rm gr} \times n_{\rm gr} \text{ bytes}$$

これは、格子のサイズ $n_{\rm gr} = 51$ とすると 2.0MByte、$n_{\rm gr} = 100$ とすると 15.25MByte となる。

アプローチ B の場合 等高線、パーティクルトレーサ、オブジェクトを重ね合わせるものとする。それぞれの図種において、以下の仮定をおく。

等高線

1. 256 本の等高線を生成する。
2. 最外側の等高線は断面上の $n \times n$ の格子の最外周を通る。一方、最内側の等高線は中央の格子内に入る。その他の等高線は、その間に一様に分布するものとする。
3. 等高線は、格子点と等高線の交点をつなぐポリラインとして定義される。
4. 等高線毎に色データが加わる。

1 等高線面当たりのデータ量

$$0.25 \times 4 \times n_{\mathrm{gr}} \times 256 \times 3 \times 4 \text{ bytes} = 3072 \times n_{\mathrm{gr}} \text{ bytes}$$

$$n_{\mathrm{gr}} = 1 \text{ 方向当たりの格子点数}$$

8 等高線面では

$$D_{\mathrm{Contour}} = 8 \times 3072 \times n_{\mathrm{gr}} \text{ bytes} = 24576 \times n_{\mathrm{gr}} \text{ bytes}$$

パーティクルトレーサ

1. 10000 個のパーティクルトレーサを描画するものとする。
2. 各トレーサ毎の転送データは、座標値と色データである。

10000 個のトレーサのデータ量

$$D_{\mathrm{Tracer}} = 10000 \times 4 \times 4 \text{ bytes} = 160000 \text{ bytes}$$

オブジェクト

1. 格子内の立方体 (複数) を描画する。
2. 一辺が $0.7n_{\mathrm{gr}}$の大きな 1 つの直方体と $0.1n_{\mathrm{gr}}$ の小 さな 2 つの立方体を想定する。
3. カラーマップ されたワイアーフレームモデルを考える。
4. オブジェクトの位置は変わらない。従って、各タイムステップのデータ量にはカウントしない。
5. 格子点上の色情報を転送するものとする。

オブジェクトのデータ量

$$D_{\mathrm{Object}} = 8 \times 0.7 \times 0.7 \times n_{\mathrm{gr}} \times n_{\mathrm{gr}} \times 4 + 2 \times 8 \times 0.1 \times 0.1 \times n_{\mathrm{gr}} \times n_{\mathrm{gr}} \times 4 \text{ bytes} = 16.32 \times n_{\mathrm{gr}} \times n_{\mathrm{gr}} \text{ bytes}$$

以上の仮定のもとで、転送するデータ量 D_{Total}を計算すると、以下のようになる。

$$\begin{aligned} D_{\mathrm{Total}} &= D_{\mathrm{Contour}} + D_{\mathrm{Tracer}} + D_{\mathrm{Object}} \\ &= 24576 \times n_{\mathrm{gr}} + 160000 + 16.32 \times n_{\mathrm{gr}} \times n_{\mathrm{gr}} \text{ bytes} \end{aligned}$$

これは、格子のサイズ $n_{\mathrm{gr}} = 51$ とすると 1.4MByte、$n_{\mathrm{gr}} = 100$ とすると 2.65MByte となる。

アプローチ C の場合　ここでは以下の仮定をおく。

1. RGBデータは 4 バイト整数としてストアする。
2. ビューポートのサイズは 512×512 とする。

表 1: アプローチ B と C におけるデータ転送量の比較

アプローチ	データ量 (MByte)	
	$n_{\mathrm{gr}} = 51$	$n_{\mathrm{gr}} = 100$
A	2.0	15.25
B	1.4	2.65
C	1	

このとき、1 画面当りのデータ量は 1MByte となる。以上を $n_{\mathrm{gr}} = 51$ と$_{\mathrm{gr}} = 100$ の場合についてまとめる

と、表1のようになる。アプローチCではデータ量は計算格子サイズに依存せず一定である。この例に関する限り、アプローチCの方が転送データ量は少ない。ただし、その差は高々1〜3倍程度の差である。

アプローチCの場合、JPEG・MPEGといった画像圧縮技術を用いることによって、データ量を数十分の一から百分の一のオーダまで圧縮することが可能になる。例えば筆者らの評価では、512*512ドットのフルカラーの流体の可視化データは、JPEGにより1/15〜1/20程度に圧縮される。一方、アプローチBの場合、数値データのため、圧縮率は高々数分の一程度と思われる。従って、データ転送に圧縮技術を用いることを考えるとアプローチCは更に有利になってくる。アプローチCは昨今のマルチメディアパソコンでの動画利用の普及を考えると、今後よりポピュラーになってくると考えられる。即ち、ユーザの端末としてパソコンを用いることによって、より安価かつ手軽に計算サーバ上の計算の実時間可視化を実現できる。

3 Owner Computation Ruleに基づく可視化処理の並列化

次に、分散メモリ型のサーバ上での処理の分割・割当の問題を考える。一般に、前節のアプローチAでは、サーバ上ではCFD計算のみが実行されるので、CFD計算におけるデータ(配列)のみを分割すればよい。アプローチBの場合、さらにポリゴンの生成があるが、その処理は基本的に格子に付随した処理と考えられるので、その分割は計算格子の分割に準じればよい。この分割法を以後、Owner Computation Ruleと呼ぶことにする。

さてアプローチCの場合、状況は少し異なり、ピクセルイメージ化された画像処理の並列化においては、画面分割(ラスタ分割など)を行なうのが普通である。さらに、画面内の可視化対象が極存している場合、動的画面分割を行なうことによるPE(Processing Element)間の負荷の均等化が図られる(例えば[4])。具体的には、並列コンピュータのPEをCFD計算用と可視化計算用に分別して、それぞれに適当な分割(例えば、計算分割には領域分割、可視化分割にはラスタ分割)を行なう方法が思い浮かぶ。しかし、これは並列コンピュータ内で上記アプローチBを行なっていることになり、並列コンピュータ内でのデータ転送がネックになる可能性がある。

以上の議論は計算分割と可視化分割の間の"協調化"の問題が、アプローチCにおいては新たに生ずることを意味している。筆者らは、これまでのRVSLIBでの経験から、可視化処理の負荷は計算処理に比べて比較的少ないと考え、アプローチBで述べたOwner Computation Ruleをこのアプローチにも適用することにした。すなわちピクセルイメージ上での各ピクセルの処理は、対応する計算格子が割り当てられているPEに割り当てるというものである。現在実現されているシステムでは、3次元BFC(Boundary Fitted Coordinate)系のCFDソルバにのみ対応している。この場合について、上述の方法を説明する。

簡単化のために、2次元の障害物回りの流れを考える(図1)。図1の網掛け部分がユーザが見たい領域(クライアント上の表示画面)だとする。この時各PEは、図1に示されるようにそれぞれの計算領域の画面に対応する部分についてのみ可視化処理を行なう。図1から明らかなように、可視化領域を計算領域のある特定部分に限定(ズームアップ)した場合、その領域に対応するPEに可視化処理が集中することになり、PE内の負荷の不均等が生ずることになる。しかし前述のように、解析計算に比較して可視化処理の負荷は少

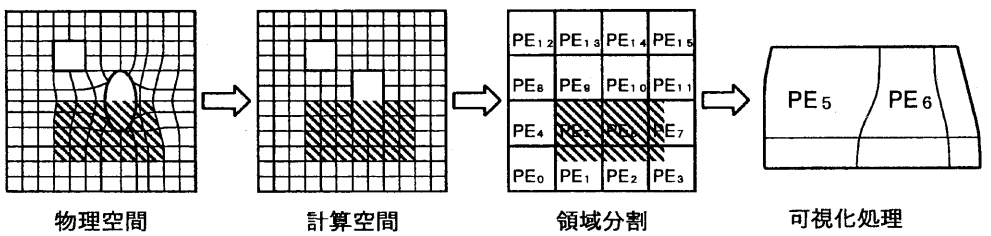

図1: 可視化処理の並列化手法

ないと考えられる。それゆえ、解析計算と可視化処理を合わせた処理全体としては、その不均等は小さいと予想される。

4 システム構成

以上のクライアント/サーバ分割および分散メモリ型のサーバ上の分割・割当の下で、本システムの構成は図2のようになる。本システムでは並列コンピュータの特徴を生かして、PEを解析・画像生成を行うPE(以下、解析PEと略す)群と合成・圧縮を行うPE(以下、合成PEと略す)とに分けた。これにより、解析・画像生成と合成・圧縮とがパイプライン的に処理され、実行速度の向上が図られている。

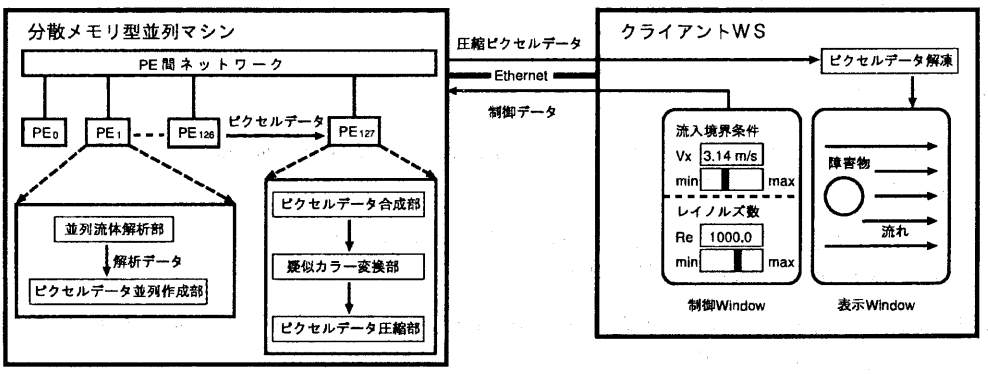

図2: システム構成

トラッキングの処理手順は、以下のようになる。

1. 各解析PEで並列に解析を行う。
2. 各解析PEで解析データに基づいて並列にピクセルデータを生成する。図種としては、オブジェクト表示・等高線図・ベクトル図・パーティクルトレースおよびそれらの重ね合わせ(等高線図・ベクトル図は複数断面)が可能である。
3. ピクセルデータを合成PEに送信する。
4. 合成PEでピクセルデータを合成し、1フレームの画像を生成する。
5. クライアントWSの同時表示可能色数が256色の場合、合成データのフルカラーから256色への疑似カラー変換を行う。
6. さらにJPEGによる圧縮を行う。
7. 圧縮ピクセルデータをクライアントWSに送信する。送信にはSocketを利用しているため、任意のUnix WS上で利用可能である。
8. クライアントWSにて、ピクセルデータを解凍する。
9. 解凍されたピクセルデータを表示する。

ステアリングの処理手順は、以下のようになる。

1. 制御Windowで決定された制御パラメータをクライアントWSから計算サーバに送信する。制御パラメータとしては、解析部では流入境界条件・レイノルズ数・時間刻幅など、可視化部では視点・カラーテーブル・光源の位置・光源のカラーテーブルマッピングするデータの範囲などを計算の途中で変更することができる。
2. 計算サーバにおいて、各解析PEに制御パラメータをブロードキャストする。
3. 各解析PEでは、新しいパラメータに基づいて計算およびピクセルデータ生成を行う。

5 今後の課題

前述のように、可視化処理は解析計算に比べてその負荷は比較的小さいと考え、Owner Computation Rule に基づいて並列化を行なった。しかし高細度の画像表示、すなわちピクセル数が大きい場合や、計算格子数が小さい場合には状況が異なり、可視化処理と解析計算との負荷が同程度になる。この時、視点や可視化領域に依存して、負荷のアンバランスが生ずる。このアンバランスの解消が課題の1つである。その方策として、視点や可視化領域に依存した動的負荷分散が考えられるが、解析計算との関係などで問題点も多い。

また現在の構成では、合成PEは1PEであるが、ピクセル数によっては合成PEの負荷が大きくなり、解析PEとの間でアンバランスが生ずる可能性もある。この場合には、合成部をラスタ分割によって並列化し、アンバランスを解消すればよい。各合成PEは独立にクライアントWSにピクセルデータを転送し、表示を行なう際に、同期を取ればよいと考えられる。

参考文献

[1] Haimes et al., *VISUAL3: Interactive Unsteady Unstructured 3D Visualization*, AIAA Paper 91-0794, 1991.

[2] Crockett, *Design Consideration for Parallel Graphics Libraries*, NASA Constractor Report 194935, 1994.

[3] Haimes et al., *pV3: A Distributed System for Large-Scale Unsteady CFD Visualization*, AIAA Paper 94-0321, 1994.

[4] Palmer et al., *Rotation Invariant Partitioning for Concurrent Scientific Visualization*, Proc. Parallel CFD'94, pp.409～416, 1995.

[5] 武井ほか,「流体解析のためのリアルタイムビジュアリゼーションシステム」, 第7回数値流体力学シンポジウム講演論文集, pp.701～704, 1993.

[6] Doi et al., *A Real-time Visualization System for Computational Fluid Dynamics*, NEC Research & Development, Vol.37,No.1,pp.114～123, 1996.

[7] 土肥ほか,「ハイパフォーマンスコンピュータのためのリアルタイム可視化」, 情報処理学会 HPC 研究会, 96-HPC-61-4, 1996.

[8] 村松ほか,「Cenju-3 上での流体解析のための実時間可視化システム」, NEC 技報, Vol.48,No.12, pp.142～148, 1995.