ハイパフォーマンス
コンピューティング 61－5
（１９９６．５．２４）

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

土肥俊，村松一弘，*武井利文，†松本秀樹，‡秋葉幸範
NEC C&C研究所，*NEC汎用ｱﾌﾟﾘｹｰｼｮﾝ事業部，†NEC情報ｼｽﾃﾑｽﾞ，‡NEC製造業ｼｽﾃﾑ開発本部

本論文では，高速な計算サーバ上でのシミュレーション結果を計算と同時に可視化する「リアルタイム可視化システム」が満たすべき要件について考察し，筆者らが開発したシステム RVSLIB(Real-Time Visual Simulation Library, Version 2.3)がこれら要件をどのように満たしているかを紹介する．また，関連する技術動向を踏まえ，システムが今後どのように発展すべきかを考察する．従来の可視化(ポストプロセッシング)技術とは異なる領域の技術が現在将来共に求められることが示される．

Real-Time Visualization for High-Performance Computing

Shun Doi, Kazuhiro Muramatsu, *Toshifumi Takei, †Hideki Matsumoto, ‡Yukinori Akiba
C&C Research Laboratories, NEC Corporation
*Application Software Division, NEC Corporation
†NEC Informatec Systems, Ltd.
‡Manufacturing Industries System Development Division, NEC Corporation

This paper discusses techniques for the Real-Time Visualization of on-going calculation using High-Performance Computers. After requirements analysis for Real-Time Visualization systems, we will show how these requirements are achieved in the RVSLIB (Real-Time Visual Simulation Library, Version 2.3), which is an experimental system developed by the authors. Discussions will also be made for the future extension of the present system. It will be shown that techniques from a wide range of fields are required to realize an efficient Real-Time Visualization system.

1. はじめに

計算機シミュレーション技術の高度化に伴って，計算結果を可視化する技術の重要性が増している．従来可視化というと，計算結果のファイルに対して可視化を行うポストプロセッシングを意味していた．計算機の高速化・低価格化に伴って，計算をしながら可視化を行うことがポピュラーになりつつある．ここでは，計算をしながら可視化を行うことを「リアルタイム可視化」と呼ぶことにする．

リアルタイム可視化は市販のポストプロセッサを用いても行うことができる．実際，世界的に最も広く使われている AVS(Application Visualization System)[1]も同様の機能を有する．しかし，ポストプロセッサに対する要請とリアルタイム可視化システムに対する要請は必ずしも同じではないので，本来ポストプロセッサとして開発されたシステムの有用性については別途検証を要するといえる．一方，リアルタイム可視化を指向したシステムとしてはVisual3[5]，pV3[6]，PGL[2]などがある．

本論文では，まず，リアルタイム可視化のシステムが備えるべき要件について考察する．特に，ネットワーク環境への親和性，マルチメディア環境（例えばディジタル画像に関する技術）への親和性などが重要と考える．次に，筆者らが開発したリアルタイム可視化システムRVSLIB(Real-Time Visual Simulation Library, Version 2.3)がこれら要件をどのように満足しているかを概説する．考察では，RVSLIB(V2.3)では未解決の要件について，それらを今後どのように実現していくかについて議論する．

2. リアルタイム可視化システムの要件

前提条件 一口にリアルタイム可視化といっても，その利用環境(ハードウェア特性，応用プログラム特性など)によって要件が変わってくる．さらに，業務の形態によっても変わってくるだろう．まず議論の前提条件を述べる(Hypotheses 1-3).

H1. 計算サーバは GFLOPS クラスの性能のマシンを想定する．具体的には，ベクトルスーパーコンピュータや並列マシンである．

H2. 応用プログラムは3次元非定常流体解析に代表される場の問題であり，また計算に伴って場が変化するものである．定常問題における「計算の収束過程」も可視化の観点では同一視できる．

H3. 更に，上記サーバ上で場の変化が利用者が感知できる速さで変化する程度のものであることを想定する．典型的には秒数コマの計算結果が出力され，それを可視化したい場合があてはまる．

以下に，筆者らが考える「リアルタイム可視化システムの要件」を述べる(Requirements 1-10).

R1. **ネットワーク分散環境における親和性**

利用者は通常のLAN環境下において，自席のUNIXワークステーションないしパソコン(クライアント)から高速計算サーバ上の計算をリアルタイム可視化する．ここで「通常の」とはEthernetクラスのネットワークをさす．この程度のネットワーク上で，所望のレスポンス・画質の可視化ができなければならない．

R2. **ディジタル動画(マルチメディア)環境における親和性**

従来可視化結果の保存にはもっぱら(アナログ)ビデオのコマ撮りを行っていた．これには高価なビデオ機器とコマ撮りのための長い時間が必要であった．一方，近年画像圧縮などのディジタル動画処理技術の進展が著しく，実際パソコンでも容易にディジタル動画を扱えるようになってきた．今後の可視化システムではディジタル動画環境への親和性が重要である．

R3. **応用プログラムへの組み込み容易性**

当然のことながら，利用者はそれぞれ自プログラムでの計算結果を可視化したい．任意の応用プログラムから容易に利用できる必要がある．

R4. **グラフィカルな操作性**

市販のポストプロセッサ同様，移動・回転・拡大などの可視化対象の操作や表示図種の指定・変更などがGUIを用いてグラフィカルに行える必要がある．

R5. **実行性能**

リアルタイム可視化にふさわしい処理速度などの性能要件を満たす必要がある．具体的には，応用プログラムの処理時間に比べて可視化のための処理時間が十分に小さいことが要求される．

R6. ソルバステアリング

R4, R5 の要件が満たされると，次にユーザは応用プログラム自体のパラメータも計算実行中に操作(ステアリング)したくなる．このステアリングは，応用プログラム内の単なるパラメータの場合から計算格子自体をも操作する場合まで考えられる．当然後者の要請は高度である．

R7. クライアントモジュールの可搬性

従来の可視化には専用の高性能グラフィックスハードやグラフィックスライブラリが必要であり，クライアントは一般に高価なものであった．LAN 環境の日常化に伴い，安価な UNIX ワークステーションや更にパソコンをクライアントとして使いたいという状況が通常となってくる．即ち，UNIX 上の標準的環境(X, Motif)や Windows 上でもリアルタイム可視化できることが望まれる．ここには二つの課題がある．一つは，比較的処理性能の低いクライアントでも対応できること．一つは，UNIX, Windows などのマルチ OS への対応である．

R8. マルチサーバアーキテクチャへの対応

上記 R3 はクライアント側の広がりに関する要請であるが，同様にサーバ側に関しても広がりが必要である．「ハイパフォーマンコンピュータ」は単一ベクトルマシンであるかも知れないし，並列ベクトルマシン(共有メモリ/分散メモリ)や分散メモリ高並列マシンのこともある．即ち，サーバ側の可視化ソフトはこれら種々のアーキテクチャに容易に適合できることが望まれる．

R9. 対応する格子の汎用性

利用者の応用プログラムで使用している格子は(単一の)BFC(Boundary Fitted Coordinate)格子，(BFC 格子ベースの)マルチブロック格子，FEM/FVM などの非構造格子などがある．これらに対して可視化できる必要がある．一般に非構造格子においては格子間の接続関係に規則性がないため，内挿計算などの処理のコストが他の格子系に比べて高い．従って，全ての格子を非構造格子と捉えて可視化する(例えば[2],[5])のではなく，それぞれの格子に応じて内挿プログラムが用意されていることが望ましい．

R10. WAN 及びグループウェアへの対応

特に共同研究開発においては，互いに離れた(組織的にも異なる)利用者が同時に同一のシミュレーション実行経過を可視化したいという要求が現れる(例えば[4], [7])．具体的には，インターネットにおける通信速度やセキュリティの問題への考慮が必要である．また，複数のクライアントへの可視化結果の送出やサーバの制御権に対するクライアント間の排他制御などグループウェアシステムとしての機能も考慮されなければならない．

3. 開発事例 - RVSLIB [3],[11]

この節では筆者らが開発したリアルタイム可視化システム RVSLIB(Real-Time Visual Simulation Library, Version 2.3)を例に取り上げ，前節の各要件がどのように満たされているかを説明する．

3.1 イメージベースのクライアント/サーバシステム (R1, R2 に対応)

可視化における一連の処理は，一般に，マッピング，レンダリング，イメージ描画に分割される．マッピングは計算格子上で与えられる計算結果からポリゴン・ポリラインなどのグラフィックスオブジェクトを生成する過程である．レンダリングはこれらを実際の視点情報やライティング条件などに

応じてピクセルを生成する過程である．このとき，計算サーバとクライアントへのこれらの処理の分割は表１の３方式(A, B, C)が考えられる．各方式の長所・短所は表１に示した通りである．筆者らが注目している点は，方式Cにおいて解析規模の大型化に伴って転送データ量が増大しないということである．方式CではJPEG(Joint Photographic Expert Group)などの画像圧縮技術を用いることによって更にデータ量を減らすことができる．定量的解析については別報[9]を参照されたい．

RVSLIBは方式Cを用い，更にネットワークの負荷状況に応じてJPEG圧縮を用いる．この画像圧縮はまた，動画の保存(その場合，例えばMotion JPEG)にも用いられる(図１参照)．

表１：処理の分散方式と転送データの内容

分散方式	A	B	C
サーバの処理	CFD計算 一部のマッピング処理 (ﾄﾚｰｻ,ﾎﾞﾘｭｰﾑﾚﾝﾀﾞﾘﾝｸﾞ)	CFD計算 マッピング処理	CFD計算 マッピング処理 レンダリング処理
クライアントの処理	マッピング処理 レンダリング処理	レンダリング処理	(ピクセル描画)
S/C間転送データ	計算結果(格子データ)	グラフィックスデータ	イメージデータ
長所	サーバの負荷小(CFD計算に専念できる)	グラフィックスハードの有効活用	可搬性，ｸﾗｲｱﾝﾄの負荷小，転送データ量が一定，画像圧縮が適用できる
短所	解析規模の増大に対して，転送データ増大	同左	サーバの負荷大

3.2 メモリリファレンスモデル/ライブラリインターフェース (R3, R5, R6に対応)

従来の可視化システムの多くでは，計算結果はファイルにより受け取っていた．例えば，100^3の計算格子を用いて計算を行い，その計算結果(数十MB)を各時刻ステップ毎にファイルに吐き出すことは，それだけで相当の処理時間を消費する(秒オーダー)．従って，計算結果をメモリ上で受け渡すことが必要になってくる．更に，このような大きな配列をメモリ上でコピーすることは，メモリ容量的に得策ではない．RVSLIBでは，(数学ライブラリでよく行っているように)応用プログラムから可視化システムへはアドレスのみを渡している(図２参照)．この方式のもう一つのメリットは，サブルーチンコールにより制御が可視化システムに一旦渡るため，そこで応用プログラムのパラメータをステアリングすることが可能になることである．

3.3 Visualization Based Input/Action Window (R4, R7に対応)

市販のポストプロセッサでは表示対象(オブジェクト)の移動・拡大・回転その他諸々のグラフィックス処理をマウス操作で行うことが一般的である．リアルタイム可視化システムにおいても同様の操作性が望まれる．

分散方式Cの場合，クライアントは３次元情報を持っていないので，このような処理が単純には実現できない．RVSLIBでは，オブジェクトの輪郭線，計算格子の輪郭線，コンター表示断面の輪郭線などの概要情報をクライアントに送り，それらに対してマウスによる諸処理を行う．この機能を

Visualization Based Input/Action Window と呼んでいる[3]. RVSLIB ではこれを X, Motif で実現している. これにより, かなり処理性能の低いクライアントでも, マウス操作に対する対話性・レスポンス性を保つことが可能になっている.

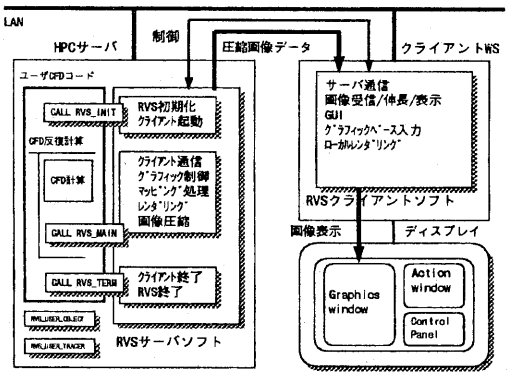

図 1：RVSLIB の構成

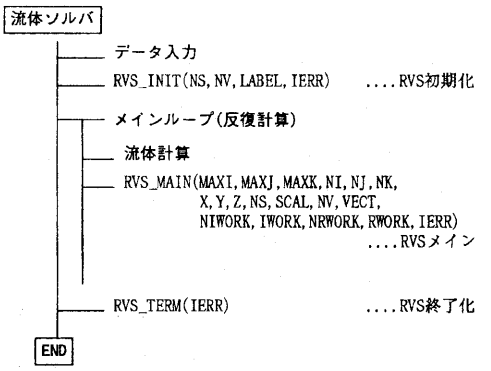

図 2：RVSLIB の呼び出し

3.4 高速可視化技法(ダイレクトイメージ生成) (R5 に対応)

方式 C の最大の欠点がサーバの CPU をマッピング・レンダリング処理に使うことである. RVSLIB では, ダイレクトイメージ生成[11]という手法を導入することによって, サーバのグラフィックス処理コストを低減している. 通常は格子点上の計算結果からポリゴンやポリラインを生成するが, このダイレクトイメージ生成は, 表示する計算格子の格子番号やその格子内での内挿係数を表示画面の全ピクセルについて予め計算しテーブルに保存しておき, 以降の処理においてはこれらの係数を用いて計算結果からダイレクトにピクセルデータを生成する手法である(性能については[11]を参照のこと). ここで特筆すべきは, テーブル生成後の処理時間が計算格子のサイズに依存しないということである.

4. 考察と課題

この節では, 前節で紹介した RVSLIB(V2.3)において満たされていない要件について考察する (Subjects 1-5).

S1. **トータルシステムのステアリング** (R6)

物理実験との対比において, これはシミュレーションの一究極の要請といえる. 可視化以前に, 計算格子の動的変更・再生成など, プリプロセッシングの技術課題のほうが大きいと思われる.

S2. **クライアントのマルチ OS 化** (R7, R10)

特にパソコンをクライアントとするケースが増えてくる. 一つの方策は, Netscape Navigator などの WWW(World Wide Web)ブラウザを用いることであるが, このようなブラウザでどこまで対話性のよい GUI が実現できるかは未知である.

S3. **サーバのマルチアーキテクチャ化** (R8)

別報[9]で分散メモリマシン上でのリアルタイム可視化システムについて報告している. この並列対応システムと RVSLIB(V2.3)とは別システムとして実現されている. 共有メモリ型並列ベクト

ル計算機，分散メモリ型並列計算機などのバリエーションに対して，統一的に対応することが重要になってくる．データ構造のアーキテクチャ依存性，プログラミングパラダイムの問題がある．ただし，この問題はソフトウェア一般に共通する問題であろう(ソースの一元化)．

S4. 対応する格子の汎用化 (R9)

例えば，Visual3[5]やPGL[2]はシステム全体が非構造格子にのみ対応している．BFC格子などの構造格子も非構造格子の特殊な場合として扱う．これらのシステムの欠点は，非構造格子がゆえの処理速度の遅さにある．RVSLIBの現バージョンは単一BFC格子にのみ対応しているが，(BFC格子ベースの)マルチブロック格子は現機能の拡張によって比較的容易に対応できると考える．非構造格子については別途専用の内挿プログラムが必要になる．その場合の速度低下は避け難い．

S5. グループウェア (R10)

リアルタイム可視化がポピュラーになってくると，多数のクライアントで同時に同一のシミュレーション結果を可視化し，またその上で議論をするといった利用形態，即ちグループウェア的利用形態が所望される．このような研究開発としてCOVISE[7]やTempus Fugit[4]がある．あるいは，既存のグループウェアシステム(例えばMERMAID[10])との通信プロトコル共通化も一方策である．

5. むすび

本論文では，高速な計算サーバ上でのシミュレーション結果を計算と同時に可視化する「リアルタイム可視化システム」の要件について考察し，筆者らの開発したシステムRVSLIB(V2.3)がこれら要件をどのようにして満たしているかを紹介した．また，未解決の要件についてはそれらをどのように満たしていくべきかについて若干の考察を加えた．　一言でいうと，WANを含めたネットワーク分散，ベクトル/スカラプロセッサ・共有/分散メモリなどの組み合わせによる種々のサーバアーキテクチャ，ディジタル動画を中心としたマルチメディア環境，グループウェア環境など，広範かつ進展著しい諸技術分野を見据えたシステムの発展が求められているといえる．

参考文献

[1] AVS Developer's Guide, Release 4.0, AVS Inc., 1992.

[2] T. W. Crockett, NASA Contractor Report 194935, 1994.

[3] S. Doi, et al., NEC Research & Development, Vol. 37, No. 1, pp. 114-123, 1996.

[4] M. Gerald-Yamasaki, RNR Technical Report RNR-92-007, 1992.

[5] M. Giles and R. Haimes, Comput. Systems in Engineering, Vol. 1, No. 1, pp. 51-62, 1990.

[6] R. Haimes, AIAA paper 94-0321, 1994.

[7] U. Lang, et al., Future Generation Computer Systems (Elsevier), Vol. 11, pp. 419-430, 1995.

[8] 村松，他, NEC技報, Vol. 48, No. 12, pp. 142-148, 1995.

[9] 村松，他，情報処理学会HPC研究会報告, 96-HPC-61-5, 1996.

[10] T. Ohmori, et al., Proc. 12th Int. Conf. on Distributed Computing Systems, IEEE, pp. 538-546, 1992.

[11] 武井，他，第7回数値流体力学シンポジウム講演論文集，pp. 701-704, 1993.