スーパーコンピューターSX4の3次元グラフィック機能の利用と展望

大阪大学大学院 工学研究科
機械物理工学専攻
齋藤 賢一
saitou@mech.eng.osaka-u.ac.jp

1 本モニター活動の趣旨と目的

近年，計算機の演算スピードが飛躍的に向上し数値シミュレーションが量・質ともに確かなものになってきた．また，最近になりマルチメディアのかけ声とともに，特に計算機のグラフィック性能が急速に進歩してきている．このような状況から，シミュレーションする現象を人間が視覚的に把握するための方法，科学的可視化 (サイエンティフィックビジュアリゼーション) の方法が今後確立されて行くものと考えられる．本センターニュースでも計算化学の分野での統合的なシミュレーションシステムが紹介されている(1)．高速な数値演算性能を誇る大阪大学大型計算機センター (以下，大計センター) のスーパーコンピューターSX4(以下，単にSX4と呼ぶ) において，ポスト処理としてまたはリアルタイム処理の形態をとって，3次元 (3D) グラフィックによる可視化を行なうことが可能である．その実行の方法および環境についての調査が本モニター活動の目的である．SX4を用いて可視化をすることを思い立った背景を少し詳しく述べると，以下のようになる．

- SX4という高速な計算機を高速なグラフィック表示装置として利用できないか．可視化の並列化は重要な研究テーマになっている(2)．可視化サーバーとしてのSX4の秘められた能力について探りたい．
- 通常は，数値計算はSX4で行ない，出力された数値データをグラフィックワークステーション (GWS) で可視化することが勧められている．しかし，今後シミュレーション分野 (とくに数値流体力学，計算材料科学など) ではリアルタイムシミュレーション／アニメーションを目指す方向であり[1](3)，さらに進んだ方法として，シミュレーションの実行者がインタラクティブ (対話的) にシミュレーション対象を操作する場合[2]についても検討が始められている(4)．よって，今後のシミュレーション技術においては，数値計算の速度はさることながら，それとともにグラフィック出力能力も重要な鍵となると予想される．
- 可視化の動向は3Dおよびボリュームグラフィックス[3]に移行しつつあるため，画像処理についても高速な演算機能と大容量のメモリーが必要とされるようになる．
- 簡易的かつプラットフォーム非依存的な可視化方法の確立が必要である．これは，今後大規模なデータ処理を行なう必要があるため，複数の計算機 (中には異なったプラットフォームのもあるだろう) 間で並列的な処理を行なうことが普通になるからである．プラットフォームに依存した可視化ソフトは案外使いにくいものである．

SX4には，導入とともにリアルタイムアニメーションシステムSXview/IMGやビジュアルシミュレーションシステムSXview/GWS[4]が装備され，さらに2次元グラフィック・ライブラリ (GL) としてGKS，3次

[1] これは，数値計算が大規模になればなるほど，数値データを出力・保存することがより困難になるために，その場で研究者または視聴者が結果を即座に理解できるようなシステムが望ましいためである．

[2] 具体的には，例えば対象とする系の時間発展を調べたい場合，初期条件・境界条件を与えて計算をスタートさせた後，任意の時刻において境界条件などを設定しなおしたりできるものが考えられる．

[3] シーンを表現するためにボリュームバッファを用い，シーンを合成，操作，レンダリングするCG(コンピューター・グラフィックス) の1分野．ボリュームグラフィックスはラスターグラフィックスの3次元版として位置づけられる．

[4] SXview/IMGはSX4で画像生成を行なった結果を超高速画像表示装置 (UltraNet フレームバッファ) に出力することにより，リアルタイム・シミュレーションを可能にする．SXview/GWSはSX4とGWSとのネットワーク環境のもとで利用可能な分散処理指向の対話型システムである．

元GLとしてPhigsやOpenGLが装備され，利用可能になっている．その開発に際しては上に述べたような可視化システムとしての利用も十分に意識されている．

今回のSX4のモニター活動では，OpenGLライブラリを使用する．OpenGLは，もともと米シリコングラフィックス社(SGI)が開発したIRIS GLというGLをオープンにし発展させたもので，OpenGL ARBと呼ばれる企業連合が仕様を決めている(5)．OpenGLはグラフィックスハードウェアとアプリケーションプログラムとの橋渡しをするものであり，約120の独立なコマンドからなる一組のルーチン群である．また，OpenGLはX Window System，Windows NTなどの異なったウィンドウシステムに実装できるものである．

汎用的なアプリケーションを使うのと違い，GLを用いる場合には個人または研究グループでのソフトウェア開発が必要となる．汎用に耐え得るソフトウェア開発こそが有効であるという意見もあろう．しかし，そのようなシステマティックな開発を特にしなくともユーザーレベルで簡易な可視化が可能であれば，私たちのように表現方法にも何らかの興味を持ち，自由な発想で物理的・工学的な視点を加えたいシミュレーション実行者／研究者は嬉しい．今後，私たちと同じような理念でSX4を利用とするユーザーが出現した場合に，導入の手助けとなるような基本的な使い方の情報を得ることもとても大事なことであると考え，今回はその点に焦点を絞っている．

また，現時点で大計センターに設置されているGWSなど[5]でグラフィック出力を行なう際の能力・使い勝手と，研究室に居ながらODINS(大阪大学総合情報通信システム)ネットワークを用いてオンライン接続をして使用する場合の使い勝手などについて，必ずしも十分なものではないが，データが得られたので紹介し，今後の課題を探りたい．

2 大型計算機センター計算機のグラフィック表示機能の利用

2.1 一般的な利用形態について

私たちが通常大計センターの計算機のグラフィック表示機能を利用しようとするとき，その利用形態は表1に示すようになる．超高速ネットワークを用いたシステムはまだ試用の段階である．また，研究室に高性

表 1: 大計センターのグラフィック機能の利用形態

大計センターからネットワーク経由で研究室で			
	センターSX4	⟶	研究室GWS
	センターSX4	⟶	研究室パソコン
○	センターGWS	⟶	研究室GWS
○	センターGWS	⟶	研究室パソコン
	センター画像処理端末	⟶	研究室パソコン

大計センターから超高速ネットワーク経由で			
▲	センターSX4	⟶	フレームバッファ

大計センターに出向いて			
○	センターGWS	→	センターGWS
○	センターSX4	→	センターGWS
	センターSX4	→	センター画像処理端末

(左:画像生成，右:画像表示．は良く使われていると思われる形態．▲はあまり無い形態．)

能のGWSを用意できる場合はかなり限られているのではないだろうか．しかし，実際には，現在のパソコン(PC)に高性能なグラフィックボードを詰めば，十分低価格のGWSに匹敵するグラフィック表示性能が出せると言われている．

2.2 私たちの主に使っている環境

私たちは研究室において，主に図1のような環境で大計センターのグラフィック環境を使う．主にX Window Systemを使用できるようにしたPCで，SX4またはGWSのグラフィック機能を使っている．また，より高速なグラフィック表示環境を求めて大計センターに出向いて作業することもある．

[5] 1998年度から導入されるSGI Onyx2, HP Visualizeなどの GWS，画像処理端末の利用が今後可能になる．現状よりさらに高パフォーマンスが期待できる．

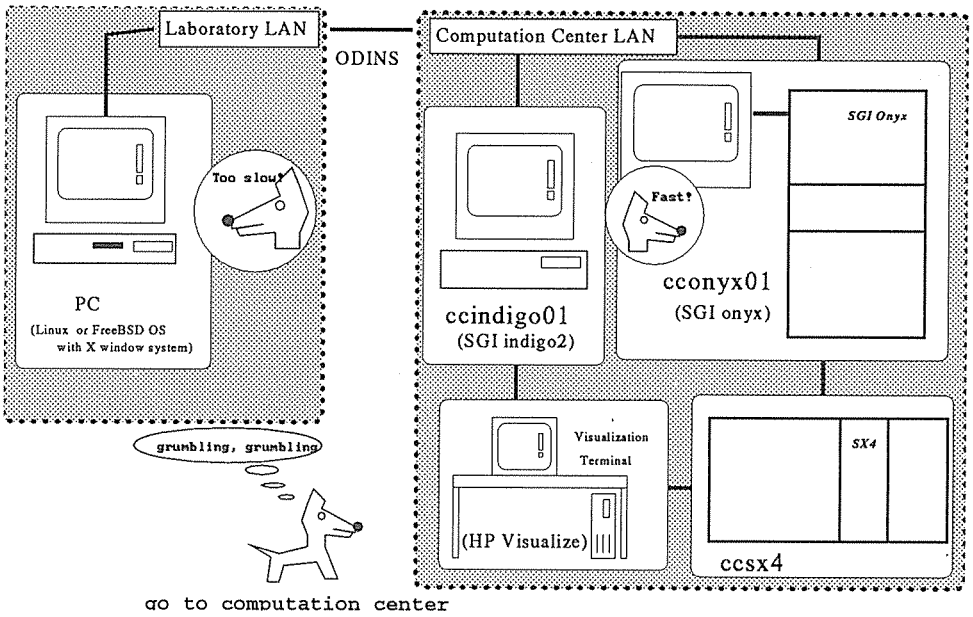


図 1: できるだけ速いグラフィック表示を求めて！

2.3 新計算機導入による環境の改善見込み

今年度，大計センターに画像処理端末が導入されるので，センターに出向いて可視化する場合の環境にはかなりの改善が予想される．本レポートは 1997 年度の利用可能なシステムにおける形態を基にしている点をここで断っておく．

3 SX4 の 3 次元グラフィック機能利用例

ここでは，具体的に研究活動に 3D グラフィック環境を利用した例を紹介する．

3.1 原子集合体挙動の分子動力学計算

近年の計算機能力の発展に伴い，物質の基本構成単位である分子／原子の運動を直接的に計算する，分子動力学 (MD) 法が盛んに利用されるようになってきている．数億個の原子を扱う超大規模な計算も行なわれている．しかしながら，分子／原子の位置と運動量の出力データは膨大なものになるので画像への変換 (可視化) が有効になる．その際に，MD 計算結果の離散的なデータから，生じている現象や物理量を積極的に捉え，人間が理解しやすいように表現する手法の確立が望まれている[6]．このサイエンティフィックビジュアリゼーションとして位置づけられている分野においては，とくに表示の難しいベクトル量・テンソル量を有効に可視化する方法が，今後十分に検討すべき大きなテーマとなっている．

2 階のテンソル量である原子応力は MD 計算の結果を評価するのに多く用いられてきており，原子レベルでの金属のアモルファス化を示す量として[7]，また結晶粒界のような不均質構造での微視的な強度評価の基準として[8] 注目されている．結晶粒界や表面などの界面では原子構造の不均質性により原子のエネルギーが集中し，原子や格子欠陥の運動が駆動され易いので，相変化，破壊や合体という非平衡的な現象の発端となり易い．今回紹介する例は，工学的応用が注目されている数十から数千の原子から成る微細な金属原子の集合体 (クラスター)[9] の合体過程の MD 計算についてであり，各原子に生じる原子応力の可視化を試みている[10]．

3.2 原子応力の表示方法

粒子系の時間発展を記述する Liouville 方程式から導かれる，

$$\sigma_i^{\alpha\beta} = \frac{1}{V_i}\left\{-mv_i^{\alpha}v_i^{\beta} + \sum_{j\neq i}^{neighbor} U'_{eff,i}(r_{ij})\frac{r_{ij}^{\alpha}r_{ij}^{\beta}}{r_{ij}}\right\} \tag{1}$$

を原子応力テンソルの定義として用いる[11]．ここで，V_iは原子 i が占有すると考える体積であり，$U'_{eff,i}(r)$ は今回の MD 計算で原子間相互作用として用いる銅に対する有効媒質理論に基づくポテンシャル関数[12] から導かれる有効ポテンシャルである．α,βはデカルト座標成分 (x,y,z) を表わす．原子応力は非局所的な性質をもつものであり，適切な平均化によって連続体力学における応力の概念と結び付くものであるが，ここでは瞬間の各原子の力学状態を表わす量が要求されるため，式 (1) そのものを評価することにする．

銅の原子クラスターは 0K の安定構造において内部の原子がほぼ f.c.c.(面心立方格子) 構造もしくはそれから僅かに歪んだ構造を保ちつつほぼ球状であり，原子応力が球対称に配置されると予測されるので，3 次元的に主応力とその方向を把握することが望ましい．原子応力は対称テンソルであり 3 つの主量が決定できる．これらをここでは主原子応力$\sigma_{i,k}(k=1,2,3;i=1\sim N;N$原子数) と呼び，それらは，

$$\det(\sigma_i^{\alpha\beta} - \delta^{\alpha\beta}\sigma_{i,k}) = 0 \tag{2}$$

の解となるものである．また，各々に対応する直交する 3 主方向の方向ベクトルも決定できる．それらを局所的な座標軸 (X,Y,Z) として用いた場合に，

$$(X/\sigma_{i,1})^2 + (Y/\sigma_{i,2})^2 + (Z/\sigma_{i,3})^2 = 1 \tag{3}$$

で定義される応力楕円体[13]を原子の可視化に用いる．この図形 (楕円体面) に主応力の大きさと向きの情報が含まれ，各原子の 3 次元での力学状態が表わされることになる．

3.3 原子応力の CG による可視化テクニック

以上の議論から，3 次元的に原子 (質点) を中心とする原子応力テンソルを表わす楕円体を描画したいということになるわけだが，汎用的なソフトウェアを用いて原子集合体の配置を可視化するときは原子自体をある半径の球として描くことが多い．原子の配置のみであれば，これまで計算化学の一分野として発展してきた分子グラフィックスのソフトウェアを利用することができるが，原子間力や原子応力を表示するようにはできていない．また，AVS [6]や IRIS Explorer [7]などを利用して等値面などを描画することで実現することもできる．しかし，直接的でないことと，基本的にソフトウェアのプラットフォームに依存する環境の制限を避けたかったこと，ならびに，基本的なグラフィック生成／表現法についての私たちの研究グループの興味[14] から，グラフィックライブラリ OpenGL を用いて専用のソフトウェアを作成することにした[8]．

ライブラリまで手を加えることは現実的ではないので避けたいが，表示方法にはある程度自由度を残しておいて欲しいと言う向きは多いと思う．内容が理解できている可視化プログラムがあれば，既開発の計算プログラムにサブルーチンとして連結して利用することもできる．

3.4 可視化の具体的手順と表示例

ここでは，可視化の具体的方法と実行例について述べる．

[6] Advanced Visual System 社のデータフロー型ビジュアリゼーションソフトウェア．大計センターでも利用可能である．
[7] AVS と同様のコンセプトで作られた可視化ソフトウェア．大計センターでも利用可能である．
[8] 具体的なプログラミングに際しては，入門者向けの書籍[15] を参考にした．

3.4.1 ソースファイルの作成とコンパイルの方法

プログラミング言語にはCを用いている．作成したプログラムの例として，stress.c というものを以下で用いる．このプログラムのヘッダー部分は下のようになっている．

```
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
/************************************************************************/
/*   Presentation of Atomic Principal Stress      (stress.c)           */
/*                        Author: Masao KOMAYA, Osaka University       */
/*                                Ken-ichi SAITOH, Osaka University    */
/*          Copy-right: 1998, Inaba Laboratory, Osaka University       */
/************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <math.h>
#include "aux.h"
#include "material.h"
#define  PI    (float)3.14159265357989323844
#define NMAX 5000
static GLfloat  spin_x    = 0.0;
static GLfloat  spin_y    = 0.0;
static GLfloat  scale     = 1.0;
static GLint    Drag      = 0;
static GLint    Scale     = 0;
                          :
                          :
```

まず，グラフィック表示に必要なライブラリの場所をSX4のファイルシステムの中で探さなければならない．ここで必要なものはccsx4の /usr/lib/ 以下 (実際は /usr/lib0/) に存在する libGLU.a,libGL.a 等であった．また，SX4用のOpenGLのサンプルプログラムが/usr/lib/GL/samples に存在した．

今回はとくにプログラムをモジュール化していないので，次のようなコンパイル用のスクリプトを用意した．Xウィンドウシステムを利用して表示するので，それに必要な各ライブラリもリンクする．aux.c はウィンドウ処理に関する関数群を有するAUXライブラリと言われるもののソースファイルであり，tk.c はXツールキット (TK) のソースファイルであり，ともにSGIの著作権の元に配布されている物を利用する．

```
#!/bin/sh
cc -o a.out.sx4 stress.c aux.c tk.c -lGLU -lGL -lXext -lX11 -lm
```

コンパイルの実行は対話方式でもバッチ方式でもどちらでも可能である．

3.4.2 GWS,OpenGL 非サポートのマシンにおける実行

OpenGLの関数群はプラットフォームによらずほぼ同一であるので，SX4と同じ手順でSGIのGWSでも利用できる．また，GL拡張されていない X Window System でも，OpenGLとほぼ同様の内容のMesaGL [9] というグラフィックライブラリを用いることで同様な実行が可能である．

3.4.3 ベクトル化および並列化の可能性について

cc コマンドによるコンパイルによって，自動的に部分的なベクトル化が行なわれる．コンパイルオプションとして-h vector を加えると，各繰り返しループ (for ループおよび while ループ) のベクトル化情報が出力される．さらに，C-ANALYZER/SX という性能向上支援ツールを用いるとさらに詳細なプログラムの情報が得られる (コマンド ca)．SX4でのバッチ処理スクリプトにおいて，

```
ca -tm -A0 loop -AL stdout lfunc inst msec \
-af afaf.tm -cf sca-filelist -lGLU -lGL -lXext -lX11 -lm
```

と記述して実行してみた．ca のプログラムの解析結果は，例えば，次のように出力される．

[9] MesaGL は，マシンが OpenGL をサポートしていない場合，OpenGL API(Application Programming Interface) をサポートしていない場合に用いる．Mesa はウィスコンシン大学の Brian Paul によって記述された．公式には OpenGL とは言えないという指摘(16)もある．

```
*-------------------------*
   Summary List
*-------------------------*
C-ANALYZER/SX Revision :  Rev.063
Analyzed Date          :  Thu Mar 19 11:10:54 1998
ca Options             :  -tm -AL stdout msec inst
                             : lfunc all programs
                          -af afaf.tm -AO
                             : loop all programs

Execution Time         :  CPU Time       =   0 : 00 ' 08 " 803418 (  8803.418 msec)
                          Elapsed Time   =   0 : 03 ' 44 " 934741 (224934.741 msec)

Instruction Information:  Instruction Count                =      144294090
                          Floating Point Element Count =       10889172
                          Vector Instruction Count     =            396
                          Vector Element Count         =           5940

Performance            :      16.391 MOPS      1.237 MFLOPS (By CPU Time)

Memory Information     :  Bank Conflict :  None

Vector Information     :  Vector Operation Ratio =   0.00 %
                          Average Vector Length  =  15.0

         Execution Terminated But Not in main Program

          Stopped  at auxQuit
          Called from KeyDown
          Called from DoNextEvent
          Called from tkExec
          Called from auxMainLoop
          Called from main

*-------------------------------------------------------------*
   Function Summary List
*-------------------------------------------------------------*

Function    Atr.  Code  Frequency  Inclusive          Exclusive          MOPS  MFLOPS  V.Op.  Aver.  Bank
                                   CPU Time(   % )    CPU Time(   % )                  Ratio  V.Len  Conf

tkExec      GLOB              1  6270.162( 71.2)  5170.153( 58.7)   9.9     0.0   0.00    0.0
sphere      LOCL            588  2386.760( 27.1)  2279.261( 25.9)  34.8     4.3   0.00    0.0
ModelSpin   GLOB         246673  1019.325( 11.6)   934.736( 10.6)   2.6     0.0   0.00    0.0
calcnormal  GLOB          65856   107.499(  1.2)   107.499(  1.2)  26.3     3.7   0.00    0.0
main        GLOB              1  8803.418(100.0)    95.495(  1.1)  48.6     0.8   0.00    0.0
tkGetMouseLoc
            GLOB           1100    56.171(  0.6)    56.171(  0.6)  10.1     0.0   0.00    0.0
cardanoi    GLOB            294    34.676(  0.4)    34.676(  0.4)  52.2    10.6   0.00    0.0
ModelDraw   GLOB            297    53.584(  0.6)    34.606(  0.4)  11.3     0.0   0.00    0.0
DoNextEvent
            LOCL           1151    71.268(  0.8)    23.745(  0.3)  17.6     0.0   0.00    0.0
          :                   :     :                  :
          :                   :     :                  :
```

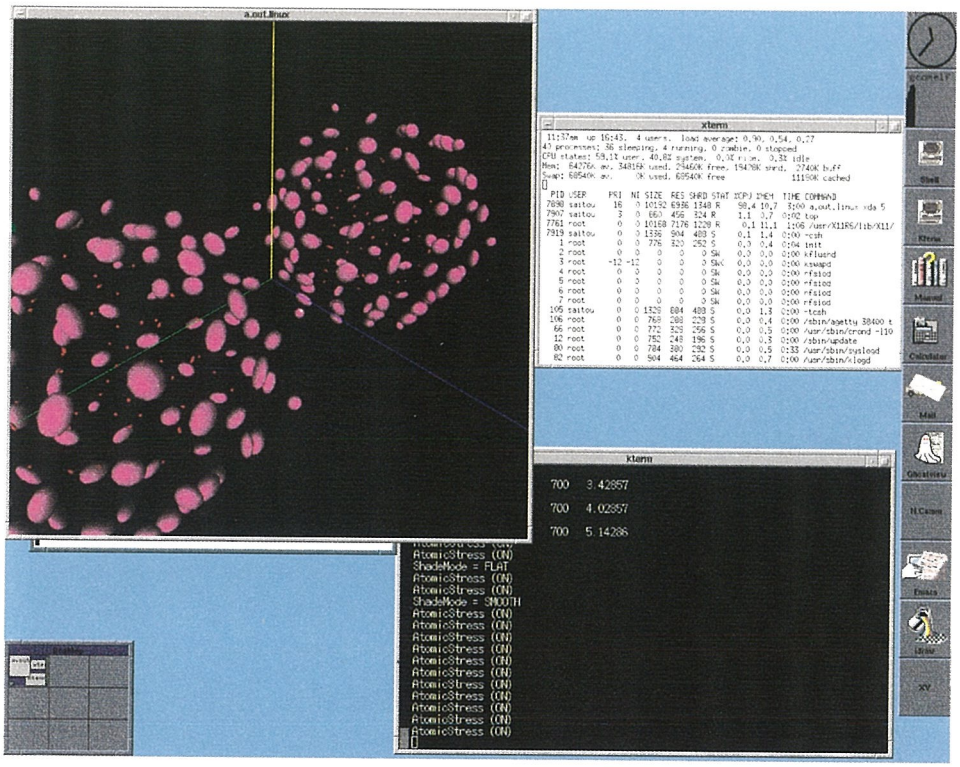
実行に時間がかかっている関数は，関数別リストの上のほうに示されている，tkExec, sphere, ModelSpin, calcnormal, main (それぞれ，TK による画像生成，原子の球の描画，回転操作，面の法線の計算，メイン (入力ルーチンを含む)) などである．この例は，研究室の PC で実行したものである．実際の作業／表示時間は数分に及ぶにも関わらず，CPU 時間はあまり消費されていない．主に最初の主応力の計算・画像生成に CPU 負荷がかかるようである．一旦画像が生成されると，マウスでの回転・拡大／縮小，シェードモードの切替え，などは対話的に行なえる．

また，残念なことにベクトル化率はほとんど 0 であり，OpenGL の関数はことごとくベクトル化不可能であった．画像生成に CPU 時間がかかることを考えると，ベクトル化の恩恵を受けるのはその部分であろう．今後，ベクトル化可能なようにプログラミングして，さらに多くの原子／分子を表示する場合に用い，ベクトル化率の向上が可視化のパフォーマンスへどの程度寄与するかを検討することは興味深い．並列化については現在検討中である．

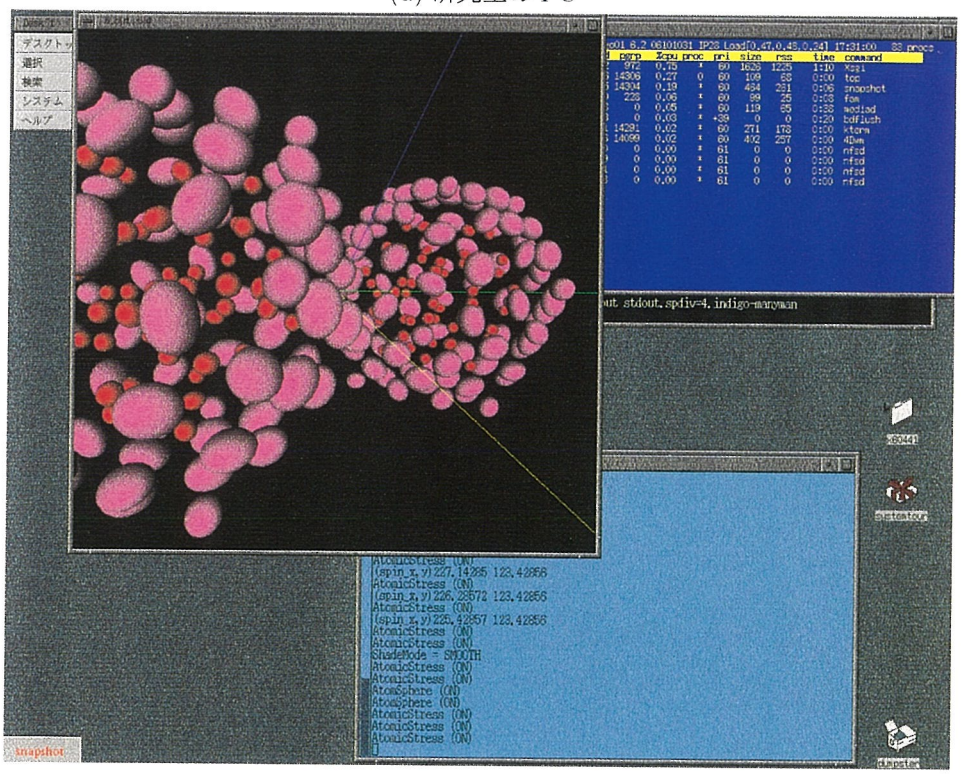
3.4.4 可視化結果

可視化の実行状況を示す例として，研究室の PC のディスプレイ上で実行している様子および，大計センターの ccindigo01 で実行している様子を図 2に示す．

図 3は非常に微細な銅の原子集合体同士が合体する計算結果として，原子応力の楕円体を実際に可視化した例である．原子を表わす楕円体が長く引き延ばされれば，その方向に引張りが加わっていることが示



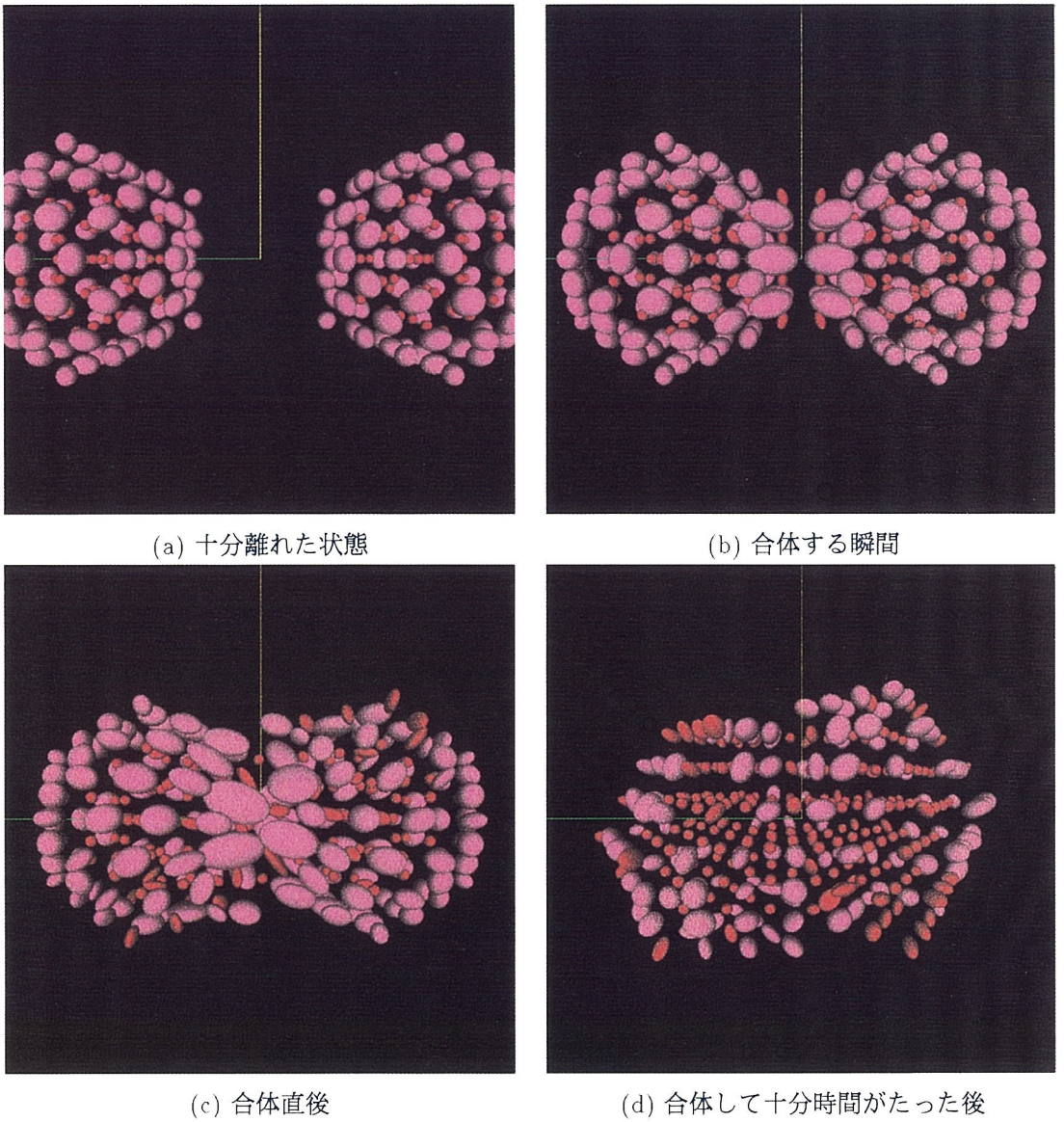
(a) 研究室の PC



(b) 大型計算機センターの GWS(ccindigo01)

図 2: 研究室の PC および大型計算機センター GWS 上での実行風景

されている．合体寸前に表面原子間で大きな引張り合いが生じている．ピンク色は3主応力が全て引張であることを示しており，一つでも圧縮であれば赤色で描画している．



(a) 十分離れた状態　(b) 合体する瞬間

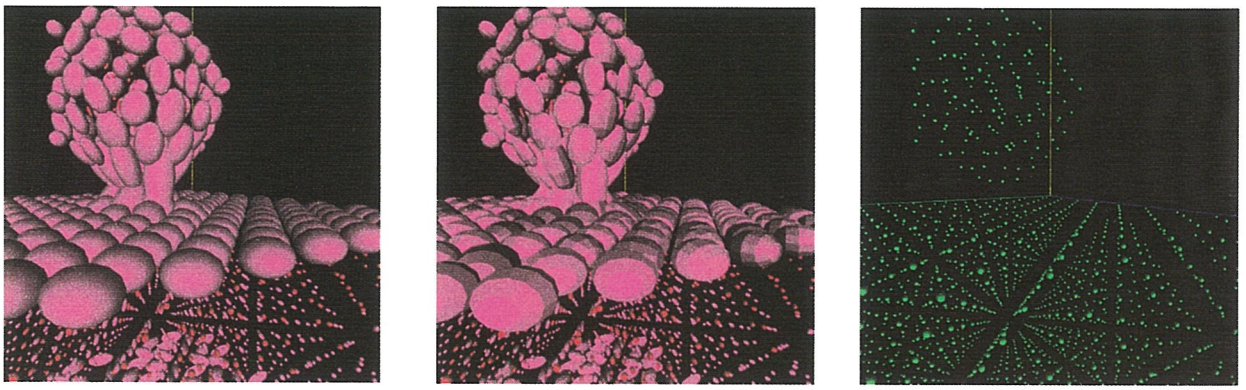
(c) 合体直後　(d) 合体して十分時間がたった後

図 3: 可視化例・その 1(原子集合体合体過程における原子応力の３次元表示)

3次元処理のお蔭で，ディスプレー上では回転／ズームアップなどが自在に行なえる．また，OpenGLでの影の扱い (シェーディングモード) の切替えをすることで，図 4(a)(b) のような表面の表現を変化させることもできる．また，原子配置のみの情報を得るために，応力楕円体を描画せずに一定の半径の球のみで表示すると図 4(c) のようになる．このように見え方が随分と変わり，確かに物理量の情報を入れたほうが現象を理解し易い場合があることが分かった．ちなみに，図 4では小さな原子集合体が周期的に広がった表面に接近している状態を可視化していて，ともに表面応力 (張力) の存在が表現されている．

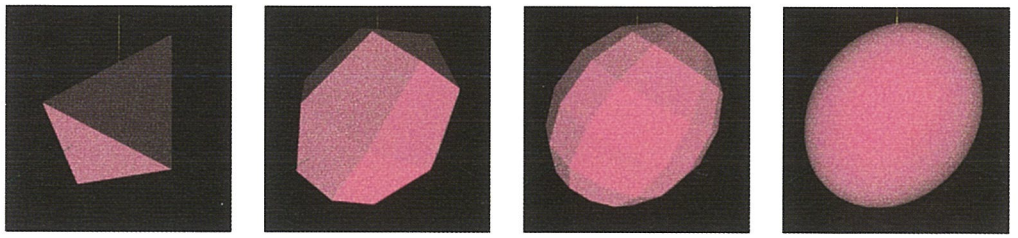
楕円体のような曲面を描画する場合，表面を周方向とその直交する方向にそれぞれ分割して多角形を張り付けるようにする．また，図 3, 図 4(a) のような滑らかな曲面の描画には，多面体の各点での法線を計算することが必要であり，多大な計算機パワーを必要とする．そして，分割の数がグラフィック出力のスピードを左右するようである．図 5に原子一つについていろいろな分割数を用いて，2種類のシェーディン

グモードで描画した結果を示している．少ない分割でもスムースシェーディングモードを用いれば十分に滑らかな描画が可能である．



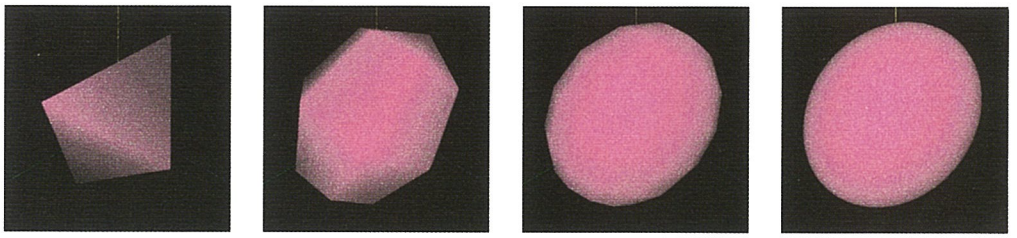
(a) スムースシェーディングモード　(b) フラットシェーディングモード　(c) 球による原子配置のみの描画

図 4: 可視化例・その 2(可視化方法による違い:原子数 2253 個 分割数 6)



2 分割　4 分割　8 分割　32 分割

(a) フラットシェーディングモード



2 分割　4 分割　8 分割　32 分割

(b) スムースシェーディングモード

図 5: 可視化例・その 3(多面体への分割数による違い)

3.5　グラフィック機能のパフォーマンスのチェック

表 1のような利用形態があるわけだが，どの方法がどれだけのパフォーマンスを出せるかは是非知りたいところである．そこで，利用形態に依存したベンチマークを行なった．ただし，共同利用していたり，データ通信の形態や状態などが必ずしも明白でないため[10] に全く厳密な結果では無いことを断っておく．表 2 がその結果である．この原稿を書いている時点で，大計センターの画像処理端末が利用可能になったので，それも含めて検討した．

[10]私自身がネットワーク，計算機内のデータ通信の方法などまだまだ不勉強のせいもあることを付け加えておく．

OpenGL によってグラフィックの計算を行なうサーバー，すなわちグラフィッククライアント (実際にプログラムを実行させるほう) に SX4,cconyx01,ccindigo01, 研究室の PC とし，グラフィックサーバー (実際に画像を描画するほう) を cconyx01,ccindigo01, 研究室の PC, 画像処理端末とした．研究室の PC での画像生成には OpenGL の代わりに MesaGL を使用している．

表 2: OpenGL プログラムの実行結果

		グラフィックサーバー (OpenGL プログラムの実際の画像表示)			
		cconyx01 (SGI onyx)	ccindigo01 (SGI indigo^2)	研究室 PC	画像処理端末 (HP Visualize)
グラフィック クライアント (OpenGL プログラムの実行)	ccsx4	— —	速 (即) 遅 (25 秒)	速 (即) Core	速 (0.5 秒) 遅 (30 秒)
	cconyx01	速 (即) やや遅 (7 秒)	速 (即) 遅 (5 秒)	速 (即) —	— —
	ccindigo01	— —	速 (即) やや遅 (5 秒)	速 (即) —	速 (即) 遅 (10 秒)
	研究室 PC	遅 (3 秒) 遅 (15 秒)	— —	— —	遅 (2 秒) 反応無 (15 秒)
	画像処理端末	— —	— —	— —	— —

速／遅などは回転操作・拡大操作の反応の速さについてである．() 内は表示が始まるまでの時間．
上段: 26 原子を描画，下段: 2253 原子を描画，ともに 5 分割×5 分割の多面体表示．
Core はコアを吐いてしまったケース．cconyx01 は使用中，不明なエラーを発して実行できないことが多くあった．
研究室の PC には DEC 社の Venturis FX5200s(Linux+Accelerated X) を用いた．

SX4 に関して言えば，予想されたように SGI の GWS で表示するのが最適なようである．また，画像処理端末も良好な表示能力を示した．ただし，原子数が数千個になると回転操作などで表示を更新するのに一秒くらいかかってしまう．さらに，GWS で画像生成する方が表示までが幾分速いことがわかった．画像生成・表示を試みたものの，反応が返って来ないケースやコアを吐いてしまうケースも多く，ここで用いたプログラムでさらに多く (例えば数十万個くらい) の原子を描画するには，楕円体面の分割数をかなり少なく押える必要がある．

3.6 SX4 のグラフィック機能の評価／GWS より優る点

表 2から明らかなようにたくさんの原子を速く表示しようと思うと表示は GWS および画像処理端末に限られ，画像生成に関しては現状では GWS，SX4 はほぼ同様のパフォーマンスのようである．しかし，GWS にはプログラムの使用するメモリーの制限が避けられない．原子数が 281625 個の場合の表示を試みたところ，ccindigo01 ではコアが吐かれたが，SX4 では結局は表示しなかったものの，しばらくは表示をしようとしていた．予測であるが，大規模数値計算を行ないつつ画像生成するなら SX4 が適しているはずである．

4 結論

大阪大学大型計算機センターの SX4 モニター活動として以下の点を検討した．OpenGL グラフィックライブラリを用いた計算結果の可視化を行なう際の手順を確立すること，そのパフォーマンスの利用形態への依存性を調べることを行なった．今回は SX4 に備わっている機能であるベクトル化や並列化が行なえなかったが，可能性は残っている．

分子動力学法によるシミュレーション結果の可視化などのように，多数の多角形の描画が必要となるとき，すなわち多量のメモリーを使用するようなときには，PC ではもちろんのこと，GWS でも難しい可視化が可能になりそうな印象を受けた．ただし，現環境では画像を表示する方のコンピュータのグラフィック表示能力／環境が支配的になる．現段階では大計センターに出向いて SGI の GWS および新しく導入さ

れた画像処理端末を画像表示に利用するのが一番適していると考えられる．一方，研究室で可視化を行なう場合に問題となる，通信に要する時間などについては全く検討していないので今後の課題である．今回の結果は可視化対象や利用環境に大きく依存したものであるために一般的な評価とはならないが，可視化実行時の使用形態の選択などには幾分かの参考になるのではないだろうか．

おわりに

大計センターのグラフィック機能をネットワーク経由で使おうと思うと，大学外のユーザーはもちろん，大学内でもネットワークの速さの制限に縛られる．よって，これまでは多くの場合大計センターに出向いて，僅かなGWSを使用することが主であったように思われる．しかしながら，画像処理端末の導入によって，同時にたくさんのユーザーがSX4その他を通じて大計センターが提供するグラフィック機能を(大計センターで)使用できる環境が整えられつつあることは嬉しい限りである．一方で，大計センターへ通う手間の点で各研究室における，シンプル(簡単)かつチープ(安価)な可視化の必要性も感じる．今後の可視化にはその2つの方向性があることを念頭に置いた上で，さらに題目に関しての利用を進め，できるだけ有効な方法を模索していきたい．

最後に，可視化プログラムを共同で作ってくれた大阪大学工学部 機械工学科 稲葉研究室 駒谷 政男 君(現 機械物理工学専攻 大学院)，筆者が分子動力学プログラムの開発に携わらさせて頂いた大阪大学大学院工学研究科 知能・機能創成工学専攻 北川研究室，計算機の詳しい使い方などを示唆して下さった大阪大学大型計算機センターの皆様にお礼申し上げます．

参考文献

[1] 近江・三浦 他, 大阪大学大型計算機センターニュース, **27**-3(1997-11),11-18.

[2] Chalmers,A. and Jansen,F.W., (Guest Editorial) Parallel Graphics and Visualization, *Parallel Computing*, **23**(1997), 817-818.

[3] 可視化情報学会 編, 流れのコンピュータグラフィックス, (1996), 154, 朝倉書店.

[4] Rapaport,D.C., Interactive Molecular Dynamics, *Physica,A*, **240**(1997), 246-254.

[5] 川西, OpenGL入門[第1回]3次元グラフィックス機能のインターフェース OpenGL, 日経 *Computer Graphics*, **1995年1月号** (1995), 203-209.

[6] Rosenblum,L. et al., Scientific Visualization, Advances and Challenges (chap.7), (1994), 103, Academic Press.

[7] Kulp,D.T. and Egami,T. et al., *J.Alloys and Compounds*, **194**(1993), 417-427.

[8] 齋藤・北川 他, 分子動力学法による対応粒界の強度評価, 材料,**46**-3(1997), 238-243.

[9] 茅・西, クラスター,(1994),32, 産業図書.

[10] 齋藤・稲葉 他, 界面近傍の原子移動と原子応力の表示方法, 日本機械学会 第75期通常総会講演会,(1998-4), 発表予定.

[11] Irving,J.H. and Kirwood,J.G., The Statistical Mechanical Theory of Transport Processes. IV.The Equations of Hydrodynamics, *J.Chem.Phys.*, **18**-6(1950), 817-829.

[12] Jacobsen,K.W., Bonding in Metallic Systems: An Effective-Medium Approach, *Comments Cond.Mat.Phys.*, **14**-3 (1988), 129-161.

[13] Fung,Y.C. , 連続体の力学入門, (1974), 99, 培風館.

[14] 駒谷・齋藤 他, CGを用いたMD結果のプレゼンテーション手法の研究, 日本機械学会 関西学生会卒業研究発表会予稿集,(1998-3),250.

[15] 相川, OpenGLプログラミング・ガイドブック, (1996), 1, 技術評論社.

[16] Kilgard,M.J., OpenGL Programming for the X Window System, (1997),462, アジソン・ウェスレイ・パブリッシャーズ・ジャパン.