ハイパフォーマンス
コンピューティング 48－8
（１９９３．８．２０）

大規模実対称行列の状態密度の計算とその並列化

日向寺祥子、長嶋雲兵、細矢治夫、関口智嗣[1]、佐藤三久[1]
お茶の水女子大学理学部化学科、情報科学科
[1]電子技術総合研究所、計算方式研究室

行列の状態密度は、従来、行列を対角化することにより求められていたが、力学モデルを用いた対角化を行なわない手法が開発された。その方法によりこれまで不可能であった大規模行列の状態密度を求めることができるようになった。またこのアルゴリズムは離散的であるために並列化に適しているので、その並列化を行なった。

ＳＵＮ４　ｉｐｃ２８台によるワークステーションクラスタを用いた実験では、１台による実行時間から期待される性能より高い性能が並列化により得られた。この結果は並列計算は単純に性能向上が期待されるばかりでなく、メモリアーキテクチャ上の不都合も解消できることを示唆している。

Parallel Calculation of Density of States for Large Scale Real Symmetric Matrix

Sachiko HYUGAJI , Umpei NAGASHIMA , Haruo HOSOYA ,
Satoshi SEKIGUCHI[1] and Mitsuhisa SATOH[1]
Department of Chemistry and Information Sciences, Faculty of Science,
Ochanomizu University
[1]*Electrotechnical Laboratory*

The suitability of parallel processing for carring out efficient calculations of density of states (DOS) for large scale real symmetric matrix has been investigated using workstation cluster consisted of 28 SUN4 IPC's with 22 Mbyte memory.

Observed efficiency of parallel calculation of DOS is higher than the expected linear speed up. (super-linear speed up.) In this case, because the size of problem on each processor becomes small by parallel processing, the cause for the super-linear speed up is considered to avoid the neck of memory architecture such as cache miss and etc. This suggests that parallel processing realizes not only powerful CPU performance but also effective use of large size high speed memory system.

1．はじめに

化学の分野では物質の電子構造についての研究がいろいろとなされている。量子論的議論ではエネルギーは離散的な値をとる。しかしながら、固体や高分子など自由度の極めて大きな系に対しては、それらのエネルギー準位はほぼ連続と見なすことができ、バンドを形成する。そのため、それらの電子構造を議論するときには個々のエネルギー準位の代わりにバンド構造、つまり単位エネルギーあたりの状態密度（density of states , ＤＯＳ）が問題になる。

従来、状態密度の計算は行列の対角化を用いて個々の固有値を求め、ある区間での固有値の数を数え上げることで求められていた。例えば、巨大分子の電子状態を簡単な近似を用いて計算するヒュッケル分子軌道法と呼ばれる計算方法を用いると、まず大きな有限のネットワークについてヒュッケルの行列をつくり、それを対角化して離散的なエネルギー準位を計算する。それに経験的に知られているバンド幅をかけることによってその状態密度とするものである。しかし、大次元行列の精度のよい一律の対角化は数千次元が限度であるため、この方法では巨大次元の行列についての状態密度の計算はできない。さらに疎行列に対してもＮの２乗に比例する主記憶、Ｎの３乗に比例するＣＰＵ時間が要求されるため、巨大な主記憶を持つ高度なスーパーコンピュータを用いても、バンド構造が問題となる大規模行列の状態密度の計算は容易ではない。

WilliamsとMaris[1]と矢久保、中山[2]らは、少ない主記憶およびＣＰＵ時間で、巨大次元のダイナミック行列の状態密度を求める方法を新たに開発した。これは高分子の巨大なハミルトニアン行列をバネのつながった力学モデルに見立てて計算を行なう方法であり、固有振動状態を持つ系が、周期的な外場との共鳴条件を満足することを利用している。バネモデルに周期的な外力を加えバネが共鳴する外力を固有値（エネルギー準位）とするものである。この方法によって大規模実対称行列の状態密度の計算が可能となった。精度そのものはあまり良いものではないが、大きな分子の状態密度を見る上では大変有効である。

ポリアセンというベンゼンが一列に並んだ形をしたポリマーでは、炭素原子が数千くらいでは原子数に対して状態密度が不規則に変化するため、今までの方法では収束しなかった。この新しい方法を用いて長嶋がポリアセンの状態密度の計算を行なった[3]ところ、原子数２万くらいになるとやっと収束し、その形は解析解とほぼ一致した。

今回本方法を、イーサネットで結合したワークステーションクラスタで並列実行したところ、非常に効率良く並列化され、期待される線形の性能向上以上の高い並列化の効率（スーパーリニアスピードアップ）が観測されたのでそれを報告する。

2．新しい状態密度の計算方法

本方法はマトリックスの非対角要素を、バネでつながれた力学モデルのバネ定数と見立て、その系の固有振動の周期的外場に対する応答をシミュレーションするものである。それは固有振動状態を持つ系が、周期的な外場との共鳴条件を満足することを利用している。つまり、バネモデルに周期的な外力を加えバネが共鳴する外力を固有値（エネル

ギー準位）とするものである。

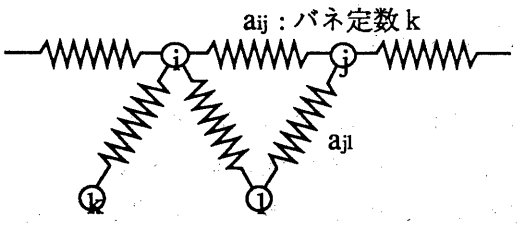


図１　粒子をバネでつないだ力学モデル

上図のような力学モデルに周期的な外力を加えたときの微分方程式は、

$$d^2u_i(t)/dt^2+\Sigma_j a_{ij}u_i(t)=F_i\cos(\Omega t) \qquad \cdots(1)$$

である。ここで$u_i(t)$：粒子 i の変位，$F_i\cos(\Omega t)$：粒子 i に対する外力，a_{ij}：粒子 i と粒子 j の間のバネ定数（ダイナミック行列の要素）である。ただし$\Sigma_j a_{ij}=0$。

ここで粒子 i の変位は $u_i(t)=\Sigma_h Q_h(t)e_i(h)$ のように展開できる。ただし$Q_h(t)$：基準振動の大きさ，$e_i(h)$：基準振動の方向ベクトルである。そのため(1) 式は次のように変形することができる。

$$d^2Q_h(t)/dt^2+\Omega_h^2Q_h(t)=\Sigma_i F_i\cos(\Omega t)e_i(h)$$

系のエネルギーは$E_h=0.5(dQ_h(t)/dt+\Omega_h^2Q_h(t)^2)$以下のよう書くことができるので

$$E_t=0.5\,\Sigma_h(\Sigma_i F_i\cos(\Omega t)e_i(h))^2\sin^2(\omega t/2)/\omega^2$$

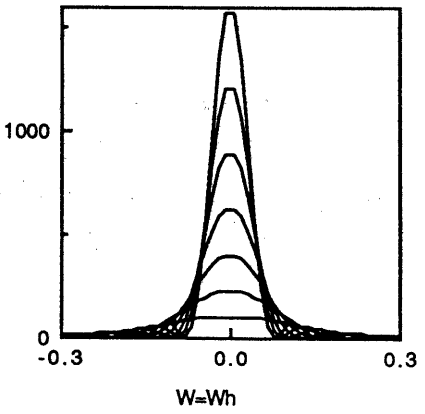
となる。ここで$\omega=\Omega_h-\Omega$，$F_i=F_0\cos k_i$（F_0：定数，k_i：乱数）である。

ここで乱数についてE_tの平均をとることによって固有値を求める。

$$<E>=0.25F_0^2\Sigma_h(\sin^2(\omega t/2)/\omega^2)$$

ここで$\Omega t>>1$かつ$4\pi N/(t\Omega_{max})>>1$ $\cdots(2)$

のとき、下図のように $\sin^2(\omega t/2)/\omega^2\sim\delta(\omega)$ となるから



$$<E>=\pi tF_0^2\Sigma_h\delta(\omega)/8$$

と書くことができる。

ここで状態密度の定義は$g(\omega)=\Sigma_h \delta(\omega)/N$であるので両者を比較すると

$$g(\omega)=8<E>/\pi tF_0^2N \qquad \cdots(3)$$

となる。

一般の行列Aは非対角行列B対角行列Cの和で表されるので、(1)式のかわりに

$$d^2u_i(t)/dt^2+\Sigma_j b_{ij}u_i(t)+c_iu_i(t)=F_i\cos(\Omega t) \quad \cdots(4)$$

を解けばよい。ここで$\Sigma_j b_{ij}=0$, $c_i=a_{ii}-b_{ii}$である。この場合の力学モデルは図2に示す通りである。

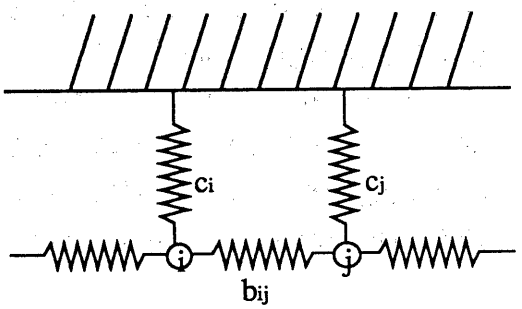


図2　一般の行列に対する力学モデル

実際の計算はある与えられたΩに対して(1)式または(4)式に対する初期値問題を解き、(2)式の条件を満たしたときに

$$<E>=0.5(\Sigma_i(du_i/dt)^2+\Sigma_{ij}a_{ij}(u_i-u_j)^2)$$

を計算し、(3)式より$g(\omega)$を計算する。

このように外場Ωに対して完全に並列化が可能であるので、Ωに対する並列化を行なった。

並列化されたプログラムを模式的に図3に示す。

3．並列化の効果

3.1ワークステーションクラスタとメッセージパッシングライブラリ

用いられたメッセージパッシングライブラリはＴＣＧＭＳＧ[4]であり、ワークステーションクラスタの機器構成はお茶の水女子大学情報科学科の情報教育用計算機システム、ＳＵＮ４／２２ＭＢ、２８台である。これらはイーサネットで相互に結合され、ＮＦＳでファイル共用がなされている。

図３　並列化されたプログラム

```
Initialization(N,NW,Wmin,Wmax)   …ポリエンの長さ、固有値検出範囲の設定
call pbeginf                     …並列化のための環境設定
call evon
get Nproc and Me
do i=1,NW
if(mod(Nproc,i).eq.Me) then
  Calculate DOS                  …初期値問題の計算
endif
enddo
  call dgop(ED,WK)               …計算結果の収集
call pend                        …並列化終了
call fexit
```

３.２並列化の効果

図３に示すように本プログラムは計算の最後にデータの交信が行なわれるので、データが少ないところでは通信に関する負荷は極めて小さい。

図４にプロセッサ台数に対する、並列化のプロセス生成と２０、４０、６０ワードのデータ転送にかかる時間を示した。それらはプロセッサの台数に対して線形に増加するが、転送ワード数による顕著な変化は見られない。また、図５はプロセッサ台数に対する、並列化のプロセス生成のみにかかる時間のグラフであるが、図４と比較するとほとんど差が見られない。これらは先にも述べたように、本計算では転送ワード数が比較的少ないためである。

図６にプロセッサ台数に対する並列化されたプログラムをそのまま実行した時間（プロセス生成とデータ転送にかかる時間を含む）のグラフを示す。プロセッサの数が２０台くらいまでは、１台で計算したときの結果より期待される時間を上回る高い効果（スーパーリニアスピードアップ）が得られていることがわかる。[5] また１０台あたりからグラフが横這いになっているのは、図４で示したプロセス生成およびデータ転送の時間が実際の計算時間に比較して多くかかっているためである。

図７にはそのプロセス生成およびデータ転送の時間を実行時間から除いたもの、すなわち実際の計算時間のみを、プロセッサの数に対して示した。プロセス生成およびデータ転送の時間の影響が消え、ほぼ直線状に並列化の効果がのびている。台数の多いところでのデータのばらつきは短い測定時間の逆数をとっているため、わずかな測定のずれが大きく現れているものと思われる。

図4　プロセス生成とデータ転送の時間

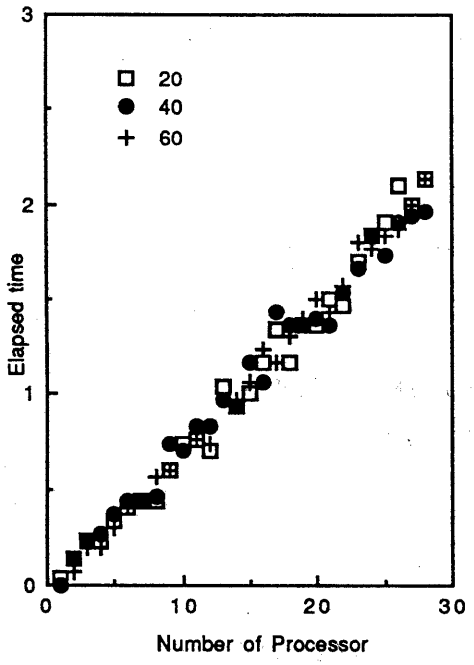


図5 プロセス生成の時間

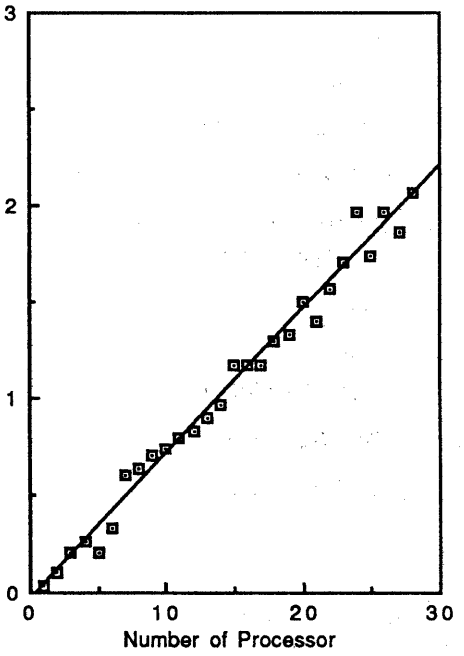


図6　プロセッサ数と実行時間の関係
（プロセッサ生成とデータ転送時間含む）

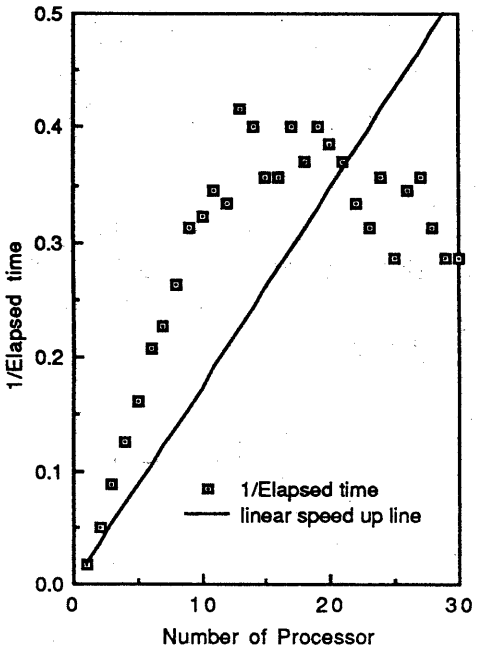


図7　プロセッサ数と実行時間の関係
（プロセッサ生成とデータ転送時間含まない）

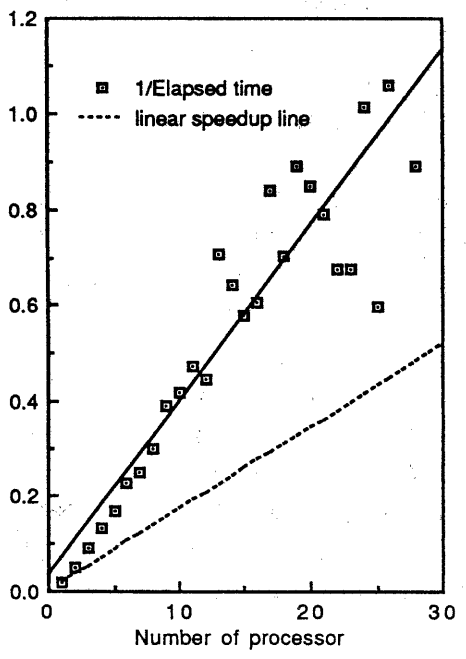


図6、図7でみられるようなスーパーリニアスピードアップが起こる原因として、台数を増やし問題を小さくすることによって、台数が少ないときにおこるキャッシュミス等のメモリアーキテクチャ上の不都合が除かれることがあげられる。

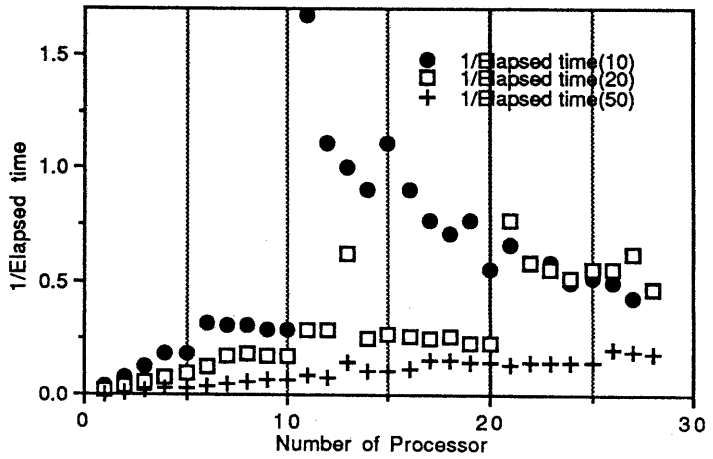
各プロセッサごとに問題のサイズが同じとなるように、プロセッサ数に比例させて問題のサイズを大きくしたときの影響についても測定を行なったので表1に結果を示す。表1ではプロセス生成時間およびデータ転送時間は除いてある。

表1　問題のサイズの影響
（プロセス生成およびデータ転送時間は除く）

Size(相対的)	Nproc	Elapsed time
10	1	5.8
20	2	5.7
30	3	5.7
40	4	5.9
50	5	5.9
60	6	5.9

表1に示したようにプロセッサの数と比例させて問題のサイズを大きくすると、計算時間はほぼ一定の値をとり、各プロセッサでほぼ同じ演算量が実行されていることがわかる。また、行列の次元を大きくしたときの測定も行った。図8には行列の次元を大きくし、問題のサイズを変えたときの実行時間に対する影響を示したものである。大きく性能が向上する点がみられ、問題のサイズが大きくなるとそれはプロセッサ台数の大きなほうにシフトする。これは問題が大きくなることにより、プロセッサ台数の大きなところでキャッシュミス等のメモリアーキテクチャ上のネックが解消されていくことを示している。

図8　データの量が大きいときの問題のサイズの影響



4．まとめ

並列計算による効果として期待されるのは直線的なスピードアップの効果であるが、ＳＵＮ４　ＩＰＣ２８台を用いた今回の実験ではスーパーリニアスピードアップを観測することができた。この効果は大きな問題のサイズを、並列化により小さくすることができるため、キャッシュミス等のメモリアークテクチャ上の不都合が除かれることによる。データの量を大きくしたときにはさらに大きなキャッシュミスがおこり、問題のサイズを大きくすると、キャッシュミスの除かれる台数が大きいほうへとシフトすることが観察できた。

このように並列計算は単純に性能向上が期待されるばかりでなく、メモリアーキテクチャ上の不都合も解消できることを示唆している。すなわちＣＰＵが同時に複数台利用できるとともに、事実上利用できる実メモリが増加することにより大規模数値計算が可能となる。

参考文献

（１）M. L. Williams and H. J. Maris, *Phys. Rev. B*, 31, 4508 (1985)

（２）K. Yakubo and T. Nakayama, *Phys. Rev. B*, 36, 8933 (1987)

（３）長嶋雲兵，情報化学討論会，1991，川口

（４）*ftp.tcg.anl.gov* の pub/tcgmsg/tcgmsg.4.02.tar.Z が最新版

（５）関口智嗣 長嶋雲兵 日向寺祥子，IPSJ SIG Notes, Vol.93, 93-HPC-47, 13 (1993)