統計数理 (1995)
第43巻 第2号 259–271

「特集 計算統計学の発展」
[原著論文]

ワークステーションクラスタによる分散並列処理

——巨大分子の状態密度の計算と巨大文字列間のホモロジー解析——

お茶の水女子大学* 長嶋 雲兵・日向寺祥子[1]
坂田 聡子・細矢 治夫
電子技術総合研究所** 関口 智嗣・佐藤 三久

(1995年5月 受付)

1. はじめに

高速マイクロプロセッサの発展の恩恵をまともにうけ，ワークステーションはその利用技術の向上および性能価格比に優れる点などがその普及を急速に促してきた．研究室内で数十MIPSというCPU能力を持つワークステーションを独占して利用できるような環境になったが，一方で科学技術計算の応用では依然としてその要求する演算能力は1台のワークステーションでは十分といえない．また，ワークステーションによっては十分に利用されていない物もあり，これらの遊休しているワークステーションをネットワークを通じて有効に活用し並列分散処理を行なうことにより，全体としての高速性能を獲得しようというのがワークステーションクラスタの考え方である．

現在では，最新のワークステーションでクラスタ形成し，スーパーコンピュータシステムに比肩する性能を発揮するようにもなっている．またワークステーションクラスタは価格性能比に優れているばかりでなく自由な構成が取れること，拡張が容易なことから，現在Cray CS6400やIBM SP1/2をはじめ，いくつかの汎用並列計算機システムでワークステーションクラスタと同じ方式を採用したものが製品化されている．

並列分散処理の特長は，計算方法や計算アルゴリズムを工夫することにより，計算機1台のもつ性能に比べ，飛躍的な計算効率向上を安価に実現させることができることである．並列分散処理の効果は，命令列を数台の計算機を用いて分散並列処理し，1) 1台あたりの実行命令数を減少することによる高速化（演算速度の台数効果）に加え，2) 台数倍の主記憶が簡単に利用でき（主記憶領域の台数効果），さらに3) 1台あたりの問題が小さくなることによるキャッシュヒット率の向上による性能の向上（スーパーリニアスピードアップ）が期待できることにある．

イーサネットやFDDIによって物理的に結合された計算機をあたかも一つの並列分散処理システムとして用いるためのソフトウエアは，メッセージパッシングライブラリであり，PVMなどが広く用いられている．本研究で用いられたメッセージパッシングライブラリは化学者の利用を想定して作られたHarrison (1991) によるTCGMSGであり，関口 他 (1993)

* 理学部 情報科学科：〒112 東京都文京区大塚2-1-1.
** 情報アーキテクチャ部：〒305 茨城県つくば市梅園1-1-4.
[1] 現 東海大学電子計算センター：〒259-12 神奈川県平塚市北金目1117.

によれば，他のメッセージパッシングに比べ簡単に使えるライブラリである．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，TOSHIBA AS4040（SUN4 IPC 25MHz/22Mbyte memory；1.8Mflops（linpack 100×100)），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．

プログラムの並列化は，ベクトル計算機の利用においてもそうであったように，もともとの計算アルゴリズムを並列分散処理用に書き換えるのが一番効率がよいことはいうまでもないが，ベクトル計算機上で非常に効率よくベクトル化されているものであれば，比較的簡単に並列化することができる．並列化のための計算機のモデルとしては，主記憶が共有であるか，分散であるかの違いがあり，前者は，プログラミング手法が従来のベクトル化と同様なものが使えるため非常に簡単である．ただしワークステーションクラスタのような物理的に分散メモリシステムをもつ並列分散処理システムでは通信コストが大きくなり計算能率が低下することがある．他方，後者はプロセッサ間の通信を明示的にプログラムしなければならないが，主記憶に関する台数効果の恩恵を受けることができる．

本稿では，お茶の水女子大学の情報教育システムを用いた日向寺 他（1993）による巨大分子の状態密度（density of states: DOS）の計算と坂田 他（1995）による大次元文字列のホモロジー検索の並列分散計算をおこなった例を紹介し，3つの並列分散処理の特長を明らかにする．

2. 巨大分子の状態密度（density of states: DOS）計算の並列分散処理

小さな分子の電子のエネルギー準位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，大きな系の電子構造を議論するには，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，DOSの計算は，小さな系のハミルトニアン行列を対角化し，個々の固有値を求め，それに経験的に仮定したバンド幅をかけることによってそのDOSとしていた．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，従来の方法では巨大次元の行列についてのDOS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DOSの計算は容易ではない．

Williams and Maris（1985）とYakubo and Nakayama（1987）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強制振動法を開発した．この方法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．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）分布とするものである．この方法によって，従来の計算方法では実行できなかった巨大分子のDOSの計算が可能となった．強制振動法は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．この強制振動法を用いて長嶋（1991）がポリアセンのDOSの計算を行なったところ，原子数2万くらいになるとやっと収束し，そのバンド構造はサイズ無限大における解析解とほぼ一致した．

次節では，強制振動法によるDOSの計算アルゴリズムとその並列化について簡単に述べる．

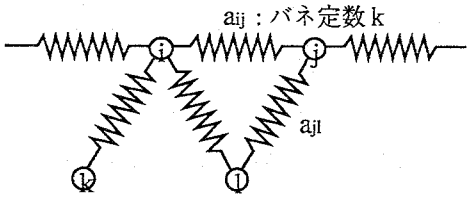

図1. 粒子をバネでつないだ力学モデル.

2.1 強制振動法による DOS の計算とその並列化

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計算する．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．

図1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，

$$d^2u_i(t)/dt^2+\Sigma_j a_{ij}u_i(t)=F_i\cos(\Omega t) \tag{2.1}$$

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（2.1）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

$$E_h=0.5(dQ_h(t)/dt+\Omega_h^2Q_h(t)^2)$$

であり，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．

$$\langle E\rangle=0.25F_0^2\Sigma_h(\sin^2(\omega t/2)/\omega^2)$$

ここで

$$\Omega t\gg 1 \quad \text{かつ} \quad 4\pi N/(t\Omega_{\max})\gg 1 \tag{2.2}$$

のとき，$\sin^2(\omega t/2)/\omega^2$ は t が大きくなるにつれてデルタ関数 $\delta(\omega)$ 的になるので

$$\langle E\rangle=\pi t F_0^2\Sigma_h\delta(\omega)/8$$

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

$$g(\omega)=8\langle E\rangle/\pi t F_0^2N \tag{2.3}$$

となる．

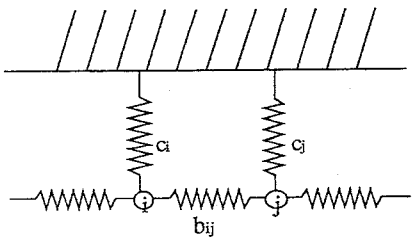

図2. 一般の行列に対する力学モデル.

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, (2.1) 式のかわりに

$$d^2u_i(t)/dt^2+\Sigma_j b_{ij}u_i(t)+c_iu_i(t)=F_i\cos(\Omega t) \tag{2.4}$$

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. 実際の計算はある与えられた Ω に対して (2.1) 式または (2.4) 式に対する初期値問題を解き, (2.2) 式の条件を満たしたときに

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

を計算し, (2.3) 式より $g(\omega)$ を計算する.

このように外場 Ω に対してデータ依存性が全くなく, 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. 以下に TCGMSG を用いて並列化されたプログラムを模式的に示す. 並列計算機のモデルとしては, 共有メモリモデルを採用した. ここで用いられたプログラムの並列化技法は, サイクリック並列化と言われるもので, もともとのソースの書き換えが最小ですみ, 同一機種の演算プロセッサで構成される並列分散処理システム向きの方法である.

```
並列化されたプログラム
  Initialization(N, NW, Wmin, Wmax) …ポリエンの長さ, 外場の個数,
                                     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 …プロセッサの起動
  get Nproc and Me …Nproc, Me: プロセッサの個数と自分のプロセッサ番号

並列計算の開始
  do i = 1, NW
  if(mod(Nproc, i). eq. Me) then
    Calculate DOS …i番目の外力に対する初期値問題の計算
    endif
  enddo

並列計算の終了
  call dgop(ED, WK) …計算結果の収集 (交換)
  call pend …並列化終了
  call fexit …プロセッサの終了処理
```

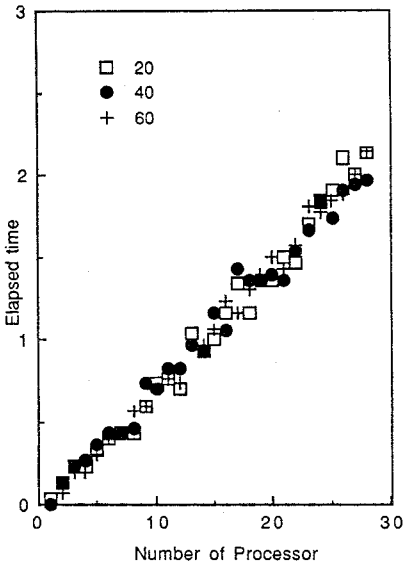

図3. プロセス生成とデータ転送の時間.

ここ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，並列分散処理に特有の通信に関する負荷は極めて小さい．ここで，図3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．本計算では演算中のプロセッサ間のデータ転送が全くなく，演算終了後の転送ワード数も比較的少ないためである．この例では，通信のオーバーヘッドは，ほとんどがプロセッサの起動と停止に費やされている．

2.2 DOS計算の並列分散処理の効果

図4に問題サイズを固定し，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）の逆数をプロセッサ台数に対して示した．実線は，1台のプロセッサから期待される線形の性能向上（リニアスピードアップ）である．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並列分散処理の効果（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．これは，プロセッサ台数が大きくなるに従い，1台あたりの問題のサイズが小さくなるために，キャッシュヒット率が向上し，仮想的に幅広いメモリバンド幅を利用できることに起因する．10台あたりからグラフが横這いになっているのは，図3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5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．この場合破線が期待される線形の性能向上を示している．これでは，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ためである．

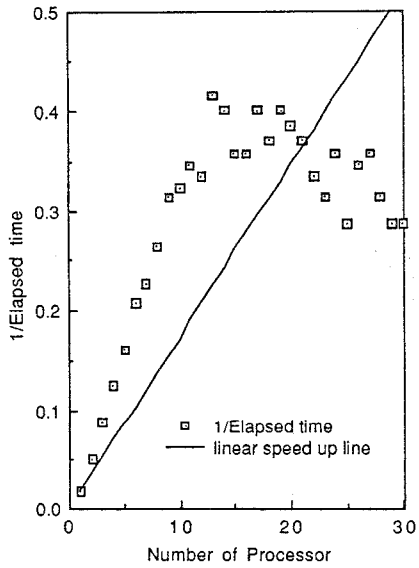

図4. プロセッサ数と実行時間の関係（プロセッサ生成とデータ転送時間含む）.

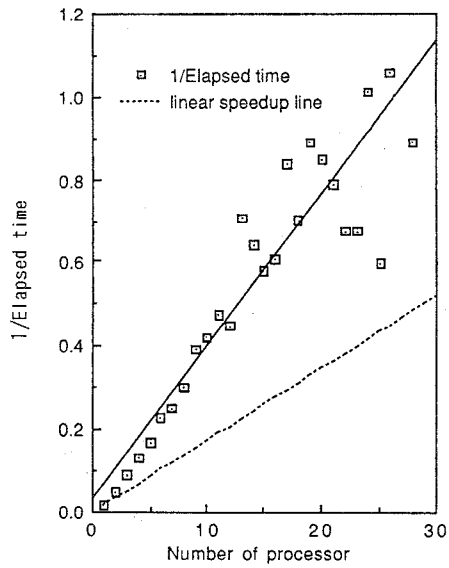

図5. プロセッサ数と実行時間の関係（プロセッサ生成とデータ転送時間含まない）.

表1. 1台あたりの問題のサイズを同じにした場合の経過時間（秒）（プロセス生成およびデータ転送時間は除く）.

Size(Mbyte)	Nproc	Elapsed time
1	1	5.8
2	2	5.7
3	3	5.7
4	4	5.9
5	5	5.9
6	6	5.9

図4，図5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起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が，これは従来の逐次型計算機システムではみられない性質であり，分散メモリ並列処理システム特有の現象であるといえる．加えて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．

2.3 巨大分子の DOS 計算の並列化のまとめ

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，ワーク

ステーションクラスタを用いた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．この効果は大きなサイズ問題を並列分散処理することで，1台あたりの問題をより小さくすることができるため，キャッシュヒット率が向上することによる．紙面の都合上実測のデータは示さないが，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る．

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，事実上利用できるメモリバンド幅が増加することにより大規模数値計算の効率よい実行が可能となる．

次の例では巨大次数の文字列間のホモロジー解析の分散処理を分散メモリモデル上で実現した例を示す．

3. 大規模文字列間のホモロジー解析の並列化

3.1 はじめに

ホモロジー解析とはデータ配列間の類似性の判断を行なうもので，これまで主に，生物学の分野でアミノ酸の塩基配列の類似性の判定を行なうのに用いられてきた（Needleman and Wunsch (1970), Sellers (1974)）．このホモロジー検索を，ホモロジースコアを求めることで定量的に行なうダイナミック・プログラミング法（Dynamic Programming method：DP）では，類似度を定量化するホモロジー・スコア計算に必要な行列に対してデータサイズ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，しかも次数の上限がメモリの制限により数千次元であるため，逐次プログラムの形では巨大次数のデータ列のホモロジー解析は不可能である．もっともDiskを用いれば巨大次数のホモロジー解析が逐次型計算機を用いても可能であるが，経過時間が大きくなり，全く実用的でない．そこで，DPの計算可能次数の拡大及び計算時間の縮小を計るため，この方法の分散メモリモデルを用いた並列化を行なった．次節では計算のアルゴリズムを簡単に説明する．

(例)文字列数が9である二つの文字列を
H1: G,A,A,G,A,A,C,T,G
H2: G,A,A,G,A,C,T,G,C　とするとき

$$F=\begin{array}{c} \\ G \\ A \\ A \\ G \\ A \\ A \\ C \\ T \\ G \end{array}\begin{array}{c} \begin{array}{ccccccccccc} & G & A & A & G & A & C & T & G & C \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & -2 & 1 & -2 & -2 & -2 & 1 & -2 \\ 0 & -2 & 2 & -1 & -2 & 2 & -1 & -4 & -2 & -1 \\ 0 & -2 & -1 & 3 & 0 & -1 & 0 & -3 & -5 & -4 \\ 0 & 1 & -2 & 0 & 4 & 1 & -2 & -2 & -2 & -5 \\ 0 & -2 & 2 & -1 & 1 & 5 & 2 & -1 & -4 & -4 \\ 0 & -2 & -1 & 3 & 0 & 2 & 3 & 0 & -3 & -6 \\ 0 & -2 & -4 & 0 & 1 & -1 & 3 & 1 & -2 & -2 \\ 0 & -2 & -4 & -3 & -2 & -1 & 0 & 4 & 1 & -2 \\ 0 & 1 & -2 & -5 & -2 & -4 & -3 & 1 & 5 & 2 \end{pmatrix} \end{array}$$

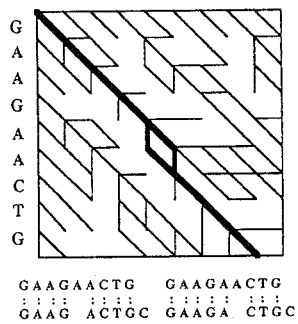

図6. ホモロジー・スコア行列と経路図.

3.2 ダイナミック・プログラミング法：DP

DPでは，図6に示す行列Fと経路図を考える．Fは類似性を定量化する最良のホモロジー・スコアを求める行列である．

図6に示す経路図に太線で示した経路は図の下部に示した並置を示しており，これが最良の並置になっている．経路図で対角線方向の経路は対応する文字が対になっていることを示し，横方向と縦方向の経路は文字の挿入及び欠失を示している．ある並置に対して類似性の程度を定量化するために，文字の一致，不一致，欠失（あるいは挿入）に適当な重みを与えて，その和を取る．二つの配列のホモロジー・スコアは，この和の最大値と定義され，これに最良の並置が一つ対応することになる．

（データ文字列数+1）次元の行列Fにおいて，行列の左端の文字列と，上端の文字列を一文字ずつ比較していくが，その時の計算方法は以下のようになる．

$$F(i,j)=\max\{F(i-1,j-1)+\delta,\ F(i-1,j)+\beta,\ F(i,j-1)+\beta\}$$

where {対角線成分の重みδ：一致の重み1，不一致の重み-2
同行，同列方向の重みβ：挿入（欠失）の重み-3}

初期値を$F(i,0)=F(0,j)=0$として行列全体を計算し，それらのうちで最大値をとるものを最良のホモロジー・スコアとする．この例では5が最良のホモロジー・スコアとなる．

3.3 DPの並列化

DPの計算可能データ数の拡大及び計算時間の縮小を計るため，この方法の並列化を行なった．用いた並列計算機モデルは分散メモリモデルである．

このため，図7に示すような行列のタイリングを行なう．これは，行列の縦の長さをプロ

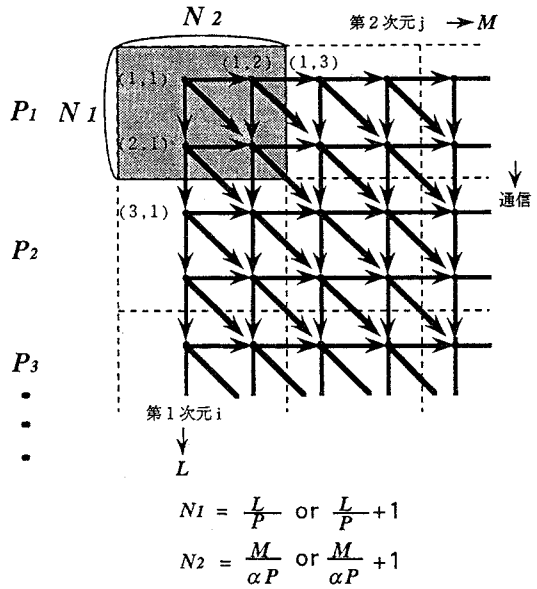

図7. ホモロジー・スコア計算におけるタイリング．

セッサ数Pで分割し，横の長さをPに定数αをかけた数αPで分割するもので，本来必要な記憶領域の縮小を計る．これにより，各プロセッサの負担する記憶領域は$P\times\alpha P$分の1に縮小される．

ホモロジー・スコア行列Fにおいて，各成分$F(i,j)$の値は，3つの値$F(i-1,j)$, $F(i,j-1)$, $F(i-1,j-1)$に依存する．従って並列化によりタイル分割された各タイル（もとの行列の部分行列）についても同じ関係が成り立つので各タイルの値を求める時には，上のタイルの下の隅，左のタイルの右の隅，そして左上のタイルの右下の隅の値があれば良い．つまり，各タイルは，上のタイルの一行，左のタイルの一列のデータのみがあれば計算が出来るので一つの反対角線方向のタイルは全部独立に並列して計算できる．よって，プロセッサがタイル毎に計算を進めていくと考えると，求めるべき行列全体において，同じ反対角線上にあるタイルの計算を複数のプロセッサにより同時に行なうことができる．この方法をタイリングされたウェーブフロント法という（Wolf and Lam (1991)，太田 他 (1994)）．

このように，タイルを単位としたウェーブフロント型の並列実行を行なう方法では，タイリングしない場合に比べて通信回数が少なく，通信起動オーバーヘッドを軽減でき，しかも効率の良い計算を行なうことが可能となる．

また，各マシンはホモロジー・スコア計算においてタイルの大きさに相当するメモリ領域のみを保持すればよいので，本来保持するべき計算領域を縮小することが可能となる．

3.4 DP の並列化の効果

DP の並列化によって得られた効果について述べる．用いた文字列データは，いずれも乱数を発生させ，それらをアルファベット 26 文字に置き換えたものを利用した．

図8はプロセッサ数Pを増加させた時のホモロジー・スコア行列の計算可能次元の変化を表している．各プロセッサの負担する行列の領域をタイリングにより縮小したので，求めるべき行列全体の計算可能次元は並列化により大幅に上昇したことがわかる．30 台のプロセッサに

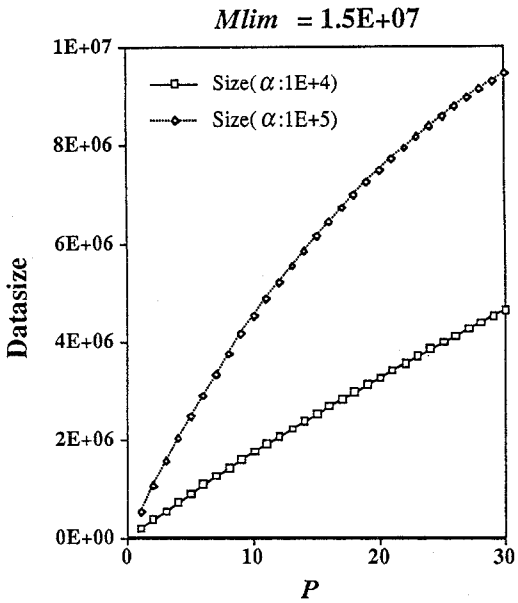

図8. 並列化によるホモロジー・スコア計算可能次元の拡大.

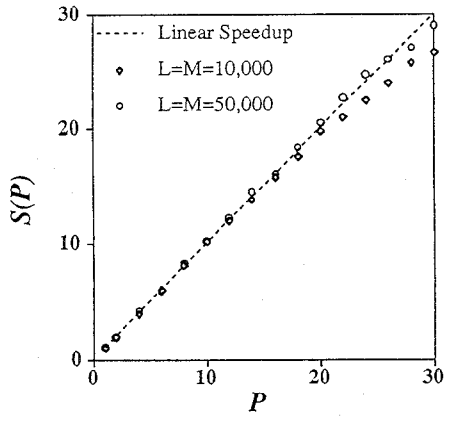

図9. 並列化による問題サイズ固定速度向上率.

よる計算可能な行列の次元は $9M \times 9M$ であり，これは通常の文字列及び生物学における塩基配列のホモロジー検索に事実上十分対応できる大きさである．塩基配列のホモロジー検索は一般に巨大な専用並列計算機を用いて行なわれることが多いが，このように効率の良い並列分散処理を行なえば，少ない記憶容量のワークステーションで構成されるクラスタでも実行可能なことが示された．

次に一定サイズの行列に対するプロセッサ数 P と並列化されたプログラムの速度向上率 $S(P)$ の関係グラフを図9に示す．このとき，データサイズは $L = M = 10{,}000$ および 50,000 であり，$\alpha = 15$ とした．

グラフの横軸はプロセッサ数 P を表し，縦軸は問題のサイズを固定した場合の速度向上率 $S(P)$ を表す．ただし，

$$S(P) = 1\text{台での実行時間} / P\text{台での実行時間}$$

である．破線は $S(P) = P$ という期待される線形の性能向上の直線を示す．$L = M = 10{,}000$ のときは期待される線形の性能向上をほぼ実現しており，またわずかであるが，10台あたりでスーパーリニアスピードアップが観測された．しかし20台を境に台数倍の性能より下回ってくる．この原因としては，プロセッサ数の増加に従い通信回数が計算回数に対して大幅に増加するため，通信の起動コストが実行時間に多大な影響を与え，サイズの小さいデータに対しては，その性能低下が比較的少数のプロセッサの段階で現れるためである．それに対し，データのサイズの大きい場合，つまり $L = M = 50{,}000$ のときは，30台までの領域で線形の性能向上をほぼ実現しており，わずかであるが20台あたりでスーパーリニアスピードアップが観測された．これは，データのサイズが大きくなるにつれその計算回数が大幅に増加するので多数のプロセッサによる並列計算がより有効になるためであり，プロセッサ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．

3.5 DP の並列化のまとめ

図8及び図9に示すようにDPの並列化の二つの目的であるホモロジー検索可能なデータ文字列数の拡大及びその実行時間の縮小は，ワークステーションクラスタを用いた分散並列処理により実現された．これにより，一般の大規模文字列及びもともとホモロジー検索のデータで

あった生物学における巨大塩基配列にもDPが安価なワークステーションクラスタ上で適用でき，広大な文字列のホモロジー検索が可能となった．また，このDPをワークステーションクラスタ上に実装実現することで，少ない記憶容量のワークステーションで構成されるクラスタでも，メモリを有効に使うことにより大きなサイズの問題を扱うことができ，なおかつ並列度に応じた速度向上を計ることも可能であることが示せた．

4. まとめ

並列計算による効果として期待されるのは，台数に比例する直線的なスピードアップの効果であるが，SUN4 IPC 28台を用いた巨大分子のDOSの計算と，巨大次数文字列のホモロジー解析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キ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及びメモリバンド幅が増加することにより効率の良い大規模数値計算が可能となる．

ワークステーションクラスタは，非常に価格性能比が良く，また，メッセージパッシングライブラリを用いて並列分散処理システムを構築するため，分散メモリモデルか，共有メモリモデルかというように問題に合わせた計算機モデルを採用できるという利点がある．加えて，現在の汎用並列計算機の多くが，メッセージパッシングを用いた並列分散処理を用いているために，ワークステーションクラスタで開発したプログラムは，汎用並列計算機への移植も容易である．

今後の課題としては，長年逐次型計算機で培われてきた数値解析の技法の並列化が挙げられるが，まだ，その検討は緒についたばかりである．

参考文献

Harrison, R. J. (1991). Portable tools and applications for parallel computers, *International Journal of Quantum Chemistry*, **40**, 847-869.

日向寺祥子，長嶋雲兵，細矢治夫，関口智嗣，佐藤三久（1993）．大規模実対称行列の状態密度の計算とその並列化，*IPSJ SIG Notes*，**93**，93-HPC-48，57-64.

長嶋雲兵（1991）．大規模実対称行列の状態密度の計算とそのベクトル化，情報化学討論会予稿集，川口.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology*, **48**, 443-453.

太田　寛，斉藤靖彦，海永正博，小野裕幸（1994）．ウェーブフロント型ループの超並列計算機向けコンパイル技法，*IPSJ SIG Notes*，**94**，13-20.

坂田聡子，日向寺祥子，長嶋雲兵，関口智嗣，佐藤三久，細矢治夫（1995）．ワークステーションクラスタを用いたホモロジー解析，情報処理学会論文誌，**36**(8)，1987-1995.

関口智嗣，長嶋雲兵，日向寺祥子（1993）．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，*IPSJ SIG Notes*，**93**，93-HPC-47，13-19.

Sellers, P. H. (1974). On the theory and computation of evolutionary distances, *SIAM J. Appl. Math.*, **26**, 787-793.

Williams, M. L. and Maris, H. J. (1985). Numerical study of phonon localization in disordered systems, *Phys. Rev. B*, **31**, 4508-4516.

Wolf, M. E. and Lam, M. S. (1991). A loop transformation theory and an algorithm to maximize parallelism, *IEEE Transactions on Parallel and Distributed Systems*, **2**, 452-471.

Yakubo, K. and Nakayama, T. (1987). Absence of the hump in the density of states of percolating clusters, *Phys. Rev. B*, **36**, 8933-8942.

TCGMSG is available via electronic mail from ftp.tcg.anl.gov .

Parallel Processing on a Workstation Cluster — Computation of Density of State for a Large Molecule and Homology Analysis of Large Character Sequences —

Umpei Nagashima, Sachiko Hyugaji, Satoko Sakata, Haruo Hosoya
(Department of Information Sciences, Ochanomizu University)
Satoshi Sekiguchi, Mitsuhisa Sato
(Department of Computer Architecture, Electrotechnical Laboratory)

Cluster of workstations connected by network is a quite costeffective parallel processing system because many workstations are already connected to network and message passing tools for parallel processing on the cluster are usually obtained as costly as free.

The suitability of parallel processing for carrying out efficient calculations of density of states (DOS) for a large molecule and homology analysis for large character sequences on the cluster has been demonstrated using a message passing library: TCGMSG and a workstation cluster consisted of 28 SUN4 IPCs with 22 Mbyte memory.

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

Key words: Parallel processing, workstation cluster, density of states, homology analysis, dynamic programming, super linear speedup.