ハイパフォーマンス
コンピューティング 52−18
（1994．7．23）

ワークステーションクラスタを用いた ホモロジー解析

坂田 聡子、 日向寺 祥子、 長嶋 雲兵、 佐藤 三久*、 関口 智嗣*、 細矢 治夫

お茶の水女子大学、 *電子技術総合研究所

Email:{*sakata,sachiko,umpei,hosoya*}*@is.ocha.ac.jp*, {*msato,sekiguti*}*@etl.go.jp*

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。このホモロジー解析を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）において、必要となる主記憶容量と計算時間は配列の長さNの2乗に比例するため、計算できる配列の大きさの限界は主記憶によって決まる。本研究ではその計算可能次数の拡大及び計算時間の縮小を図るため、この方法の並列化を行なった。

Toshiba AS4040（SUN4 ipc）30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、データ列の計算可能次数上限を大幅に拡大することが可能となった。さらに、一台による実行時間から期待される台数倍の性能が並列化により得られた。

Parallel Homology Analysis using Workstation Cluster

Satoko Sakata, Sachiko Hyugaji, Umpei Nagashima,
Mitsuhisa Sato*, Satoshi Sekiguchi* and Haruo Hosoya

Ochanomizu University, *Electrotechnical Laboratory

Email:{*sakata,sachiko,umpei,hosoya*}*@is.ocha.ac.jp*, {*msato,sekiguti*}*@etl.go.jp*

Homology analysis of protein has an important role in biology. In the homology analysis, required memory size and computing time proportionally increase with the square of the number of amino acids in protein. Therefore the size of protein for homology analysis is limited by the available main memory size. Namely, the upper limit is usually less than about 10^4 on an available scalar computer.

In order to expand the limit of size and to carry out the analysis effectively, we parallelized the program for homology analysis using a message passing library : TCGMSG, on workstation cluster where 30 Toshiba AS4040s are connected by Ethernet. By parallelization, the possible size became several hundred times larger than that by the sequential version of program and linear speed-up has been observed.

1 はじめに

ホモロジー(homology)とは、類似性、相同性という意味の言葉である。ホモロジー解析とはデータ間の類似性の判断を行なうもので、これまで主に、生物学の分野でタンパク質の性質や構造を予測する手段として盛んに行なわれてきた。このアミノ酸の塩基配列の類似性を判断する手法として用いられてきたのが、ダイナミック・プログラミング法 (Dynamic Programming method : DP) である。DP はアミノ酸の塩基配列のホモロジー解析を定量的に行なうのに使われている方法 [1][2] で、その特色は類似度を示すホモロジー・スコアの計算により類似性の定量化が可能になることにある。DP は、情報科学の分野における動的計画法に基づき Needleman、Wunsch が提唱し Sellers により修正されたので Needleman - Wunsch - Sellers (NWS) アルゴリズムと呼ばれることもあるが、一般的にはこれを DP と呼んでいる。

この DP においては、類似度を定量化するホモロジー・スコア計算に必要な文字列の長さ N に対して、N の 2 乗に比例する主記憶および CPU 時間が必要となる。このように N の上限がメモリにより制限されるため、一台の計算機で実行することを仮定すると巨大なデータ列のホモロジー解析は不可能となる。

本論文では、DP の計算可能サイズの拡大及び計算の高速化を図るため、この方法の並列化を行なった。並列化により、メモリの有効利用ができ、逐次プログラムに比べ台数以上の大きな問題を解くことが可能となった。さらに並列化した DP 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、期待される線形の性能向上が観測されたので、報告する。

2 ダイナミック・プログラミング法

ホモロジー解析とは、より具体的には、配列間の類似性を表すホモロジースコアを計算し、スコアの良いものを選び出す操作のことをいう。この操作を行なうのによく使われるのが以下に示すダイナミック・プログラミング法 (DP) である。

DP では、ホモロジー・スコア行列 F と経路図を考える。図 1 に示す経路図に書き込まれた経路は、それぞれが特定の並置に対応している。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。

(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

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

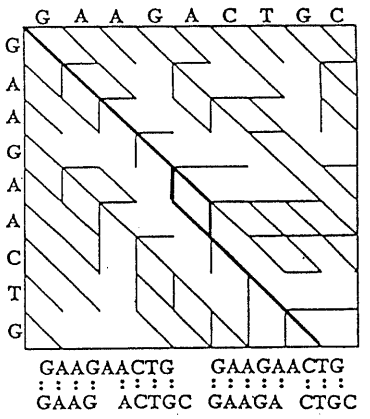

図 1: ホモロジー・スコア行列 F とその経路図

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訳であるが、その時の計算方法は以下のようになる。

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

$$where \begin{cases} \delta : \text{一致の重み } 1, \quad \text{不一致の重み } -2 \\ \beta : \text{挿入（欠失）の重み } -3 \end{cases}$$

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。ホモロジー・スコアの値は、比較するデータ文字列の相似性の度合に比例し、また、データ文字列間の距離はホモロジー・スコアの逆数を取ったもので表すことができる。従って、ホモロジー・スコアはデータ文字列間の相似性の指標となる。

3　ダイナミック・プログラミング法の並列化

DP の計算可能データ数の拡大及び計算時間の縮小を図るため、先に説明した DP の並列化を行なった。この２つの目的を実現させるためには、プロセス間の通信回数を極力少なく、また計算の最大並列度が長時間保たれるように、なおかつ有限サイズのメモリを有効に利用することが必要となる。すなわち、行列の縦の長さ（片方の文字列の長さ）を L、横の長さ（もう片方の文字列の長さ）を M （$M \geq L$）とするとき、L をプロセス数 P で分割し、M を P に粒度制御パラメータ α （$\alpha \geq 1$）を乗じた数 αP で分割して、行列のブロック化を行なう（図２に $\alpha = 2$、$P = 4$ の場合の分割と実行を模式的に示した）。これにより、各プロセスの記憶領域は、もとの行列の大きさ $(L \times M)$ の $1/(\alpha P^2)$ になる。

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、並列化によりブロック分割された各ブロック（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、各ブロックの値を求める時には、上のブロックの下の隅、左のブロックの右の隅、そして左上のブロックの右下の隅の値があれば良い。つまり、各ブロックは、上のブロックの一行、左のブロックの一列のデータのみがあれば計算ができるので一つの反対角線方向のブロックは全部独立に並列して計算できる。よって、プロセスがブロック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ブロックの計算を複数のプロセスにより同時に行なうことができる。従って、最大並列度は P であり、αP と P の差に対応する処理ステップ数だけ実行される。

このとき、粒度を制御するのは、各ブロックの分割に影響を与える粒度制御パラメータ α である。最大の並列性の保持ということを考慮すると、１ブロックの大きさはできるだけ小さい方が望ましいので、α を大きくする必要がある。一方プロセス間の通信回数対計算回数の比を小さくするには、α を小さくして一回の通信当たりできるだけ沢山の計算をするよう１ブロックの大きさを大きくし、通信コストを減らすのが望ましい。前者の場合、最大並列度が何ステップにもわたって保たれるが、通信回数が α に比例して増大する。一方後者の場合、通信コストは少なくなるが、最大並列度でのステップ数も減少する。これらのことを考慮して、使用可能な全てのプロセッサを利用できるような高さを持ち、なおかつ計算初期及び終期の非効率さをなるべく押えることのできるような幅を持った、ブロック単位の行列を作るよう、α を決定しなくてはならない。

4　並列化の効果

DP について並列化を行なった結果を以下に示す。データはいずれもランダムにアルファベット２６文字を並べたものを利用した。

用いたメッセージパッシングライブラリはTCGMSG [3] 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理学部情報科学科の情報教育用計算システム、Toshiba AS4040 (SUN4 ipc) ／ 22MB、30台である。これらはイーサネットで相互結合され、NFSでファイル共用がなされている。ここでは、1プロセッサ上で1プロセスのみを実行させた。つまり、プロセス数 P = プロセッサ数である。

図3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な問題のサイズの上限の変化を表している。ここでは、$\alpha \geq 30,000$ とした。各プロセッサの負担する行列の領域をブロック化により縮小したので、求めるべき行列全体の計算可能次数は並列化により大幅に上昇したことがわかる。30台のプロセッサによる計算可能な行列の次数は約 10^7 であり、これは通常の大規模文字列及び生物学における塩基配列のホモロジー解析に十分対応できる大きさであるといえる。巨大な塩基配列のホモロジー解析は一般に専用並列計算機を用いて行なわれることが多いが、このように少ない記憶容量のワークステーションクラスタでも実行可能なことが示された。つまり、一台でのみ実行することを想定したDPのプログラムに比べ、並列化に対応したDPのプログラムはホモロジー・スコア行列の次数を大幅に拡大できるということである。(しかし、数千次数程度の短いデータ文字列を扱う場合、実行時間が短いのは、ブロック化を行なわず一台でのみ実行することを想定したプログラムである。従って、ブロック化の手法は、データ文字列のサイズが大きい時により有効である。)

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4に示す。このとき、データサイズは $L = M = 50,000$ であり、$\alpha = 10$ とした。期待される線形の性能向上をほぼ実現しており、しかもプロセッサ数が増すにつれ向上度が大きくなっている。元の行列を αP^2 個のブロックで分割するので、各ブロックの大きさは αP^2 に従って小さくなり通信回数も増えるが、P の増加に伴い最大並列度の実現されるステップ数も増加するので両者の釣合がとれ、このように僅かではあるが台数倍以上のスピードアップが実現された。

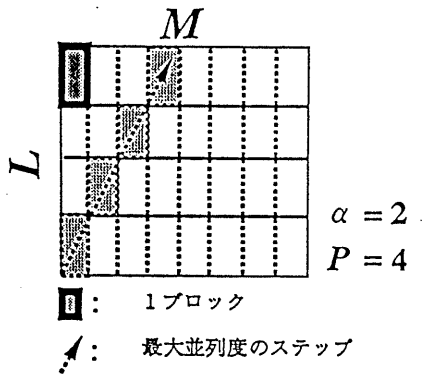

図 2: ホモロジー・スコア行列のブロック化

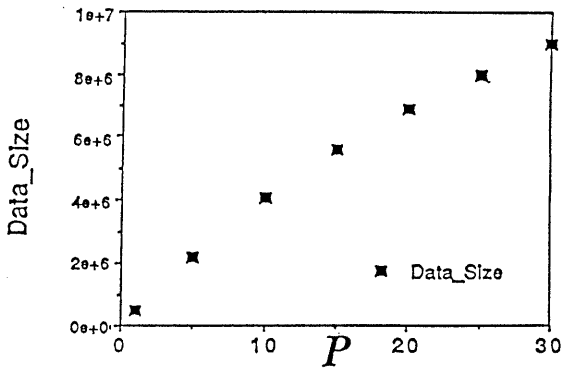

図 3: 並列化による行列計算可能次数の拡大

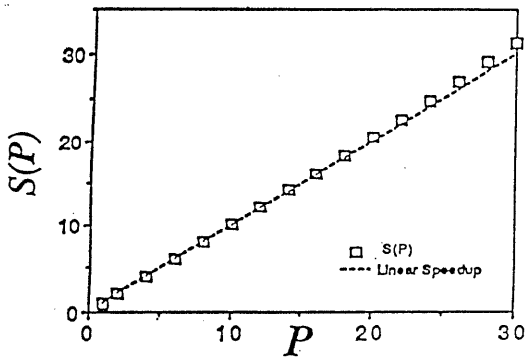

図 4: 並列化によるプロセッサ数 P と速度向上率 $S(P)$ の関係

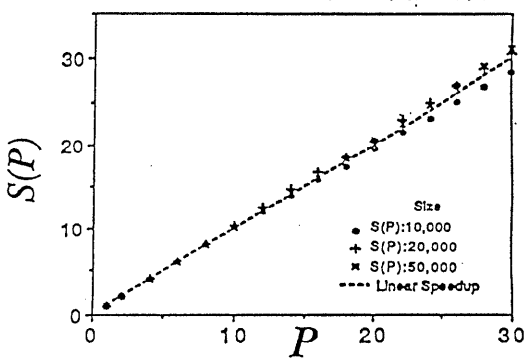

図 5: サイズを大きくしたときの速度向上率の変化

データのサイズを変えた時の実行時間に対する影響を示した図を図5に示した。ここで示したデータサイズは、$L = M = 10{,}000$、20,000、50,000であり、また$\alpha = 10$とした。どのサイズに対してもほぼ線形の性能向上が得られ、データのサイズが大きくなるにつれ多数のプロセッサによるスピードアップ効果がより大きく現れている。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。一方サイズの小さいデータ文字列に対しては、２０台を境に台数倍の性能より下回っている。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、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だと考えられる。

プロセッサ数に比例させて問題のサイズ（文字列長）を大きくした時の実行時間へ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。ここで、$\alpha = 10$とした。表1に示したようにプロセッサ数に比例させて問題のサイズ

Size （L,M の長さ）	P	Elapsed time
2000	1	34.3
4000	2	68.2
6000	3	102.2
8000	4	136.3
10,000	5	170.6
12,000	6	207.3

表 1: 問題のサイズの影響

を大きくすると、実行時間はSizeにほぼ比例する。これは、サイズの拡大に従い、各プロセッサの計算量および通信回数が増加するが、計算時間の実行時間に占める割合が圧倒的に大きく、またキャッシュ効果によりほぼ台数倍の実行時間になることを示している。

表2は、一定サイズの行列に対して粒度制御パラメータαとプロセッサ数Pを変化させた時の実行時間の変化を表している。ここでは、$L = M = 2000$の行列についての測定結果を示す。紙面の都合上Pが２０台の時までのデータを載せたが、２０台以上では$\alpha = 1$の時が常に実行時間が最小であった。比較的少ない台数で実行させた時はαの値が大きいほど実行時間は速くなるが、プロセッサ数が増えるにつれαの値が小さいほど実行時間が速くなるという傾向があることが分かる。

P	$\alpha=1$	$\alpha=5$	$\alpha=10$	$\alpha=15$	$\alpha=20$	$\alpha=25$
4	9.2	8.8	8.8	8.8	8.6*	8.7
8	4.6	4.6	4.8	4.5*	4.6	4.7
12	3.5	3.4*	3.5	3.4*	3.5	3.7
16	2.9*	2.9*	2.9*	3.1	3.1	3.3
20	2.6*	2.7	3.0	3.0	2.9	3.3

表 2: αを変化させた時の実行時間（但しSize = 2000 ; * は同じ台数で実行時間最小値を示す）

ここでαの最適値は計算式より次の様に導かれる [4]。１ブロックを計算の単位とすると、元の行列は$P \times \alpha P$の行列と表すことが出来る。このとき並列実行のステップ数$Step_{par}$は、$2(P-1) + (\alpha P - (P-1))$となり、$P$, αPが十分大きければ$Step_{par} \sim P + \alpha P$となる。一方これらのブロック単位の行列を一台で実行させた時の時間$Step_{seq}$は$P \times \alpha P$であり、このとき速度向上率は、$Step_{seq}/Step_{par} = \alpha P^2/(P + \alpha P) = (\alpha/(1+\alpha))P$である。もし、$\alpha P \gg P$ならば、完全な速度向上が得られる。つまり、ブロックを単位と考えた時、行列全体は行数Pが少なく列数αPの多い方が最大並列度の実行されるステップ数がより多くなるので、αが大きい方が効率的である。しかしαが大きくなると細粒度になるため、通信のためのオーバーヘッドが大きくなる。

この表２の結果では、プロセッサ数Pの大きいときは$\alpha = 1$での実行時間が最小となっている。この場合、配列の長さ$L = M = 2000$と小規模であり、メモリに充分余裕があるため$\alpha = 1$でも計算可能であるが、実際ホモロジー解析の対象となるのは１Ｍ以上の大規模なデータ文字列である。大規模なデータ文字列のホモロジー解析を

限られたプロセッサ数で行なうことを考慮すれば、αの値が必要以上に小さいとメモリ不足となり、実行不可能となる。つまり、データのサイズによってαの値の下限が決定される。従って、サイズの大きい文字列をデータとする場合は、αの値を充分大きくして、各プロセッサの負担する1ブロックの大きさをメモリに入るように小さくする必要がある。

このような手法において通信コストの削減により計算効率を最大限に向上させることは、データ文字列のサイズによりαの値が制限されるので難しいが、並列化のためのブロック分割の際、各ブロックの列数が小さくなると効率はシステムの持つ通信の性能に依存して低下するので、αの値を可能な範囲で調節して、ブロック幅が必要以上に小さくならないよう考慮する必要がある。つまり、αの下限の制限下で、最適のαを用いることで「最大並列度の向上」と「通信コストの削減」のトレードオフをはかることが重要である。

5 結論

DPの並列化の二つの目的、ホモロジー解析可能なデータ文字列数の拡大及びその実行時間の縮小が実現された。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解析のデータであった生物学での大規模塩基配列にもコストの低いワークステーションクラスタ上でDPが適用でき、巨大な文字列のホモロジー解析が可能となった。また、このDPをワークステーションクラスタ上に実現することで、少ない記憶容量のワークステーションから構成されるクラスタでも、台数を増やし、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、なおかつ台数に比例した速度向上をはかることも可能であることが示せた。

このように並列計算は性能向上が期待されるばかりでなく、CPUが同時に複数台利用できるとともに、並列化アルゴリズムにより利用出来る実メモリが増加し、大規模計算を可能とすることができる。

今後は、様々な大規模文字列に対してのDPによるホモロジー解析の適用が考えられるが、中でもベンチマークテストプログラム間のホモロジーについての調査に興味がある。自分の持つ問題に対する計算機の性能をよりよく知るには、自分の行ないたい問題の計算の内容が、どの様なベンチマークテストと相似であるかを知ることが重要である。そのため、プログラム間のホモロジー、中でも性能評価に役立つホモロジーについての定義を確立し、各プログラム及びその構造のホモロジー解析により典型的なプログラムの特徴抽出に応用してみたいと考えている。

謝辞

本研究を行なうに当たり、様々な御討論及び御助言を頂きました基礎生物学研究所の中井 謙太氏および日本電気の妹尾 義樹氏に大変感謝致します。

参考文献

[1] Needleman, S.B. Wunsch, C.D. : J. Mol. Biol., 48 (1970) pp443 - 453.
[2] Sellers, P.H. : SIAM J. Appl. Math., 26 (1974) pp787 - 793.
[3] R. J. Harrison, Int. J. Quantum Chem., 40 (1991) pp847.
TCGMSG is available via electronic mail from ftp.tcg.anl.gov.
[4] Nicholas Carriero, David Gelernter : How to write Parallel Programs., (1990) pp138 - 140.