システムソフトウェアと
オペレーティング・システム　89－9
シ　ス　テ　ム　評　価　2－9
（２００２．２．１４）

分散共有メモリシステムにおけるWaveFront法を用いた タンパク質のホモロジー解析の並列化

奥野 一矢†　横手 聡††　齋藤 彰一†
上原 哲太郎†　國枝 義敏†　福田 晃†††
†和歌山大学システム工学部情報通信システム学科
††和歌山大学大学院システム工学研究科
†††九州大学大学院システム情報科学研究院

概要　タンパク質のホモロジー解析とは，複数のタンパク質の間に存在する類似性を検出をする解析の一種である。これは，ダイナミックプログラミング法（Dynamic Programming method: DP）を用いてタンパク質を構成しているアミノ酸の配列のペアワイズアライメントより求めることができる。しかし，DPはアミノ酸配列のサイズがN倍になると計算時間は N^2 に比例して増大する。したがって，DPを並列化することにより処理を高速化することが重要である。本論文では，ソフトウェアDSM上でDPを並列実行させるためにホモロジースコア行列を分散共有メモリに適切な形式に変換した。この変換行列にタイリングを行い，並列実行した。

Parallelization of Homology Analysis of Protein on Distributed Shared Memory System by Wavefront Method

Kazuya Okuno†　Satoshi Yokote††　Shoichi Saito†
Tetsutaro Uehara†　Yoshitoshi Kunieda†　Akira Fukuda†††
†Department of Computer and Communication Sciences,
Faculty of Systems Engineering, Wakayama University
††Graduate School of Systems Engineering, Wakayama University
†††Graduate School of Information Science and Electrical Engineering, Kyushu University

Abstract　Homology Analysis of Protein detects similarity between some proteins. This similarity can be obtained from Pair-wise Alignment between two Amino acid sequences which consists are elements of proteins by using Dynamic Programming (DP) method. In DP algorithm, both required memory size and computing time increase proportionally with the square of the number of the elements in comparing amino acids sequences. Therefore, this paper shows a new parallelized DP algorithm that can carry out this analysis effectively. In order to parallelize, the homology score array is converted to such form that can execute on Software Distributed Shared Memory system. Finally this paper also shows comparison on execution times of the parallel execution before and after the conversion of the array.

1 はじめに

ヒトゲノム計画は1980年代に始まった生命科学分野のプロジェクトである。ヒトゲノム計画とはヒトゲノム中にあるヒトの生命活動の制御情報や全遺伝情報を格納しているヒトDNAを解析する計画である。ヒトDNAは文字数で表すと約三十億にもなる。ヒトゲノム計画に限らず生物のゲノム情報を解析することは医学, 農学, 薬学の分野の大きな発展に寄与することになり, それ以外にも生物の生命の起源や遺伝現象の理解, 個体の発生など, 生命現象の最も基本的な問題の理解に大きく貢献することができる。近年は, ポストゲノム時代に入ったといわれる。ゲノム情報を解析する以外のアプローチとして, DNAから転写されたRNAやRNAから翻訳されて生成されたタンパク質の解析がある。

しかし, ゲノム情報の解析やタンパク質の解析は, 大規模なデータの解析に, 膨大な計算力と計算時間を要する。そのためヒトゲノム計画は計画当初から新しい情報処理技術や情報処理インフラメントストラクチャの必要性が訴えられてきた。安価にハイパフォーマンスの計算機環境を手に入れる方法の一つに分散共有メモリを用いた並列処理がある。これは安価な計算機をネットワークで介し, それぞれがメモリの一部を共有することで, それぞれの計算機が協調して計算する機構である。本論文では分散共有メモリを用いてタンパク質のホモロジー解析を行う手法について述べる。

タンパク質は, 多くのアミノ酸の結合によって構成されている。タンパク質のホモロジー解析とは, タンパク質を構成するアミノ酸配列の類似度を調べることである。これは, 類似度の高いタンパク質は, それぞれタンパク質の持つ機能が類似している可能性が高いという考えからである。アミノ酸配列の類似度を調べる方法をアライメントとよぶ。二本のアミノ酸配列のアライメントを特にペアワイズアライメント, それ以上のアライメントをマルチプルアライメントとよぶ。アミノ酸配列のペアワイズアライメントによく使われる手法に, ダイナミックプログラミング法[1]（以下, DPという）がある。DPはホモロジースコアと呼ばれる値を求めることでタンパク質のホモロジー解析を定量的に行うことができる。DPはホモロジースコアを計算するのに必要な文字列の長さNに対して N^2 に比例する主記憶およびCPU時間が必要となるアルゴリズムである。

本稿では, ソフトウェアDSMライブラリFagus[2]を用いてDPを並列化した。さらにタイリング[3]を用いることで並列演算の最適化を行い、計算を高速化した。特にDPで計算対象である二次元配列の変換を行い, 変換の前後で実行速度を比較した。

2 ダイナミックプログラミング法

DPは情報科学の分野で利用されてきたアルゴリズムである。DPでは, その計算対象となるデータを行列（以下DP行列という）で表す。DP行列は格子状の経路と対角線状の経路からなり, 各経路にはそれを通るためのコストが設けられている。DPでは, DP行列の始点（左上）から終点（右下）へ, 最も小さなコストで移動できる経路を求める。つまり, DPは与えられた解空間の最適解を探索する解探索アルゴリズムの一種である。ホモロジー解析で用いられるDPでは, コストではなくホモロジースコアを設定しており, ホモロジースコアの最大値を得る経路を求める。

次に, ホモロジー解析におけるDPについて, 具体例を用いて説明する。この例では, 二つのアミノ酸配列をそれぞれA, Bとする。配列AはGKFD, 配列BはGFVDとする。図1に示す経路図は, DPの解空間である。各経路のうち横方向と縦方向は, それぞれアミノ酸の欠失挿入を意味し, 対角線方向は経路に対応する配列中のアミノ

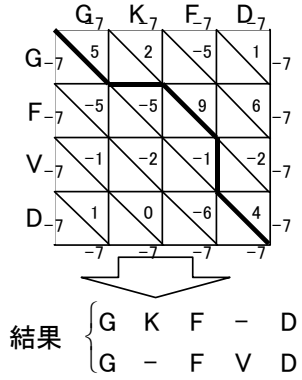

図1 DPの経路図（DP行列）

$$F = \begin{array}{c} \\ G \\ F \\ V \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} G & F & V & D \end{array} \\ \begin{pmatrix} 0 & -5 & -10 & -15 & -20 \\ -5 & 8 & 3 & -2 & -7 \\ -10 & 3 & 4 & 11 & 6 \\ -15 & -2 & 0 & 6 & 7 \\ -20 & -7 & -3 & 1 & 14 \end{pmatrix} \end{array}$$

図2 ホモロジースコア行列F

酸が対になっていることを意味する。図1の太字の経路は, 最適解つまりペアワイズアライメントの結果を示している（この例ではVが挿入されてKが欠失された）。各経路を決定する方法を次に述べる。

図2に, 配列AとBによるホモロジースコア行列Fを示す。まず，経路の始点のホモロジースコアF(0,0)を0とする。いずれかの座標が0であるイタレーションのホモロジースコアF(i,0)とF(0,j)は, それぞれ一つ前のイタレーションのホモロジースコアF(i-1,0)とF(0,j-1)に，ギャップコストと呼ばれる負のスコアを加算する。これら以外のイタレーションのホモロジースコアF(i,j)は,一つ前のイタレーションによって次の三種類が計算され, その中の最大値となる。

(a) F(i-1,j)とギャップスコアとの和
(b) F(i,j-1)とギャップスコアとの和
(c) F(i-1,j-1)と, 配列Aのi番目の要素と配列Bのj番目の要素との類似度を表すスコア $s(A_i, B_j)$ との和

これらを式で表すと以下のようになる（Gはギャップコストを表す）。

$$\begin{cases} F(0,0) = 0 & (1) \\ F(i,j) = \begin{cases} \max \begin{cases} F(i-1,j-1) + s(A_i, B_j) \\ F(i-1,j) + G, \\ F(i,j-1) + G \end{cases} & (2) \\ i = 0 \text{のとき} \\ F(0,j-1) + G & (3) \\ j = 0 \text{のとき} \\ F(i-1,0) + G & (4) \end{cases} \end{cases}$$

これらの計算の結果に基づき, 各イタレーションにおける最適な経路をDP行列内のポインタに記憶する。全イタレーションを計算した後に終点から始点に向かってポインタを参照しながらトレースバックをすることで, 全体の最適解を求める。なお，$s(A_i, B_j)$，つまり，アミノ酸配列の類似度には, アミノ酸の類似度を考慮したアミノ酸置換行列を用いる。本研究ではDayhoffマトリックス[2]を採用した。また，アミノ酸置換置換行列以外に, アミノ酸の挿入と欠失の場合に与えるギャップコストも決定しなければならない。ギャップコストを大きくすると結果に含まれるギャップの個数が減り, ギャップコストを小さくすると結果に含まれるギャップの個数が増大するために, ギャップコストは出力を左右するパラメータの一つである。本論文ではギャップコストを−5に設定している。

3 DPの並列化

並列計算を高速に行うためには, ノード間の通信回数を極力減らし, かつ計算の最大並列度が長時間保たれるようにすることが必要である。本研究では分散共有メモリ型並列システム上でDPの並列化を行った。

まず, 逐次実行の場合のホモロジースコアの計算プログラム(二重ループ構造)を次に示す。

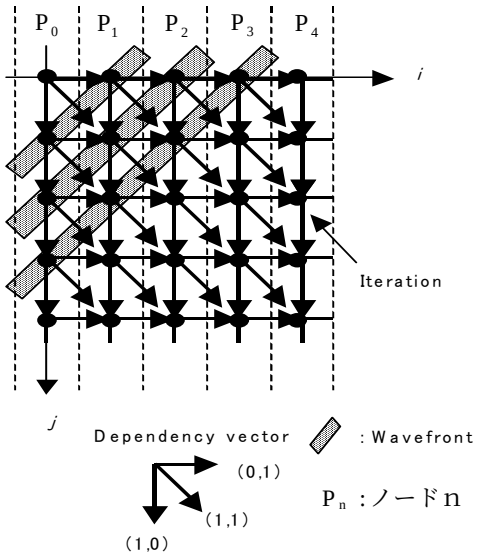

図3 DPにおけるイタレーション空間

```
for(j=0j<=N;j++)
  for(j=0;j<=M;j++)
    F[i][j]＝max{F[i－1][j－1]＋s(x, y)，
                F[i-1][j]+G,F[i][j-1]+G};
```

この二重ループが表すイタレーションの集合は，図3に示す二次元配列のイタレーション空間を構成する。以下，本章では，DPの並列化に用いるウェーブフロント法とそのタイリングについて述べる。

3.1 ウェーブフロント法

DPを並列化するためには, DPのイタレーション間の依存関係を解決しなければならない。DPは，一般的にイタレーションF[i][j]を計算するためにF[i][j-1], F[i-1][j], F[i-1][j-1]の値を参照する（図3参照）。つまり，F[i][j]はF[i-1][j]，F[i][j-1]，F[i-1][j-1]と依存関係があるといえる。この依存関係は，三種類の距離依存ベクタ(1,0)，(0,1)，(1,1)で表すことができる。iループとjループのいずれにおいてもループ運搬依存があるため，do all型の並列実行はできない。よって，この二重ループはdo across型ループである[3]。つまりこの二重ループは，次に示す二つの条件を満たす。

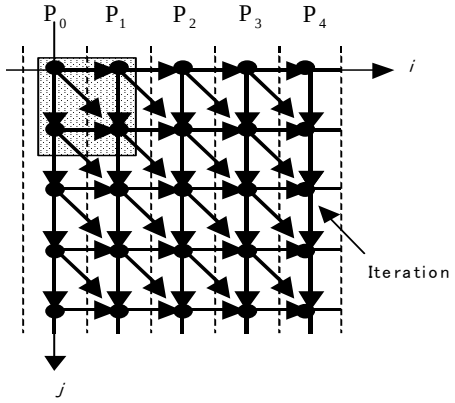

図4 DPのタイリング

(a) 並列実行する次元についてループ運搬依存がある

(b) 並列実行する次元の内側に別の次元を含む

したがって，このアクセスパターンは，ウェーブフロント型ループといえる。ウェーブフロント型ループを並列化するためには, ウェーブフロント法を用いる。

3.2 DPのタイリング

DPの計算をウェーブフロント法により，複数のノードで並列実行する場合を説明する。まず，ノード0が始点のホモロジースコアF(0,0)を計算しバリア同期を実行する。次にノード1がF(0,1)を計算しバリア同期を実行する。それと同時にノード2がF(1,0)を計算しバリア同期を実行する。それぞれの同期処理が完了後，同様に図3のウェーブフロントループ毎にバリア同期を実行しながら並列に処理を進めていく。しかし，このような処理方式では, バリア同期を実行するために必要な時間が, ノード内の演算時間と比べて非常に大きくなる。また，ノード数の増加により，上記の方法では高速化が期待できない。高速化のためには，ノード間の通信にかかるオーバーヘッド，すなわちバリア同期の回数を減らす必要がある。この問題を解決するため, 複数のイタレーションを一つの単位にまとめ, その単位毎にDPを行う。

これをタイリング[3]とよび，複数のイタレーションでできた単位のことをタイルとよぶ。

タイリングにおいて, タイル間で生じる依存関係はイタレーション間での依存関係と同じである。したがって，各タイルはそのタイルの上，左上，左のタイルを参照する。また，一つの反対角線状のタイルは全て独立して並列に実行することができる。つまりタイリングにもウェーブフロント法を適用することができる。

3.3 DPの行列変換

本研究では，我々が開発したFagus(第4章参照)というエントリー一貫性制御のソフトウェアDSMライブラリを使用している。Fagusは，共有変数をアクセスするための同期変数を, 行列の横方向に対してまとめて関連付けすることができる。しかし，ウェーブフロント型ループのアクセスパターンは, 逆対角線方向である。このため，DPをFagus上で並列化するには，ウェーブフロント型ループを横方向のループになるように行列変換する必要がある。

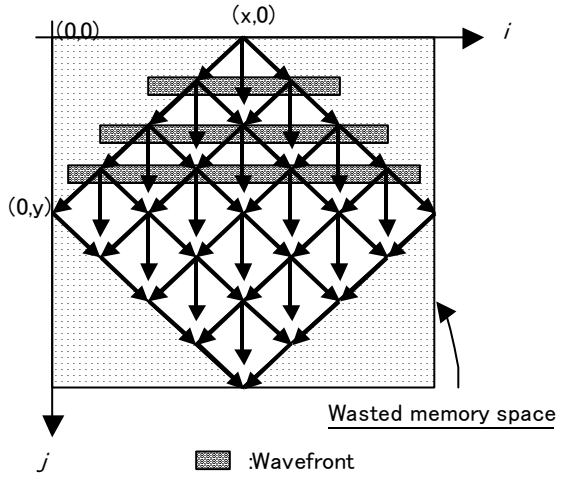

図5 DPの行列変換(1)

そこで，図5のように行列を回転する。太い色つきの線はウェーブフロント型ループによるアクセスパターンである。この行列回転により，ウェーブフロント型ループのアクセスパターンが行列に対して横方向になったことがわかる。この行列回転は，DP行列を－45度回転させて二次元配列にマッピングしたものである。この行列回転により, ウェーブフロント型ループのアクセスパターンとFagusのページマッピングが一致させることができる。なお, 四隅の色付き部分はデータのない空白部分である。さらに計算を容易にするために，さらに図6に示す行列変換を行う。これは図6のすべてのイタレーションを二次元配列の左へ移動させたものである。この変換により，プログラムの実装が容易になるという利点がある。

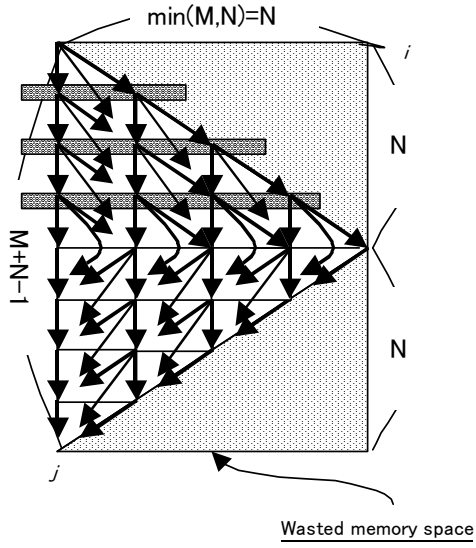

図６　DPの行列変換（2）

行列変換により問題となる点は，元のDP行列と変換後の行列（以下，変換行列という）との間の座標の対応関係である。図6に示すように，変換行列のイタレーション間の依存関係は非常に複雑である。行列計算において，この変換行列の要素を直接指定することは困難である。本システムでは, 計算量の少ない座標変換関数を作成することで対応した。この座標変換関数により，変換前の行列の座標を指定することで変換行列の座標を与えることができる。これにより，少ないオーバーヘッドで, 座標の変換を行うことができる。以上の変換処理により，Fagusにおける共有メモリの関連付けと，DP行列のアクセスパターンを

同じにすることができる。

3.4 ＤＰのメモリ使用量

DPの各イタレーションが保持するデータは，始点からそれぞれのイタレーションまでのホモロジースコアと，トレースバックのためのポインタである。本システムでは，これらの情報を保持するための各イタレーションサイズは6バイトである。

次に，ホモロジースコア行列の横方向のサイズは，行列変換の前後のいずれも，仮想記憶のメモリのページサイズの倍数（4096バイトの倍数）となる。これは，Fagusの仕様によるものである。例えば，6000個のアミノ酸配列を比較する場合，横方向のサイズは，6000×6バイト=36000バイトとなり，ページサイズに合わせると9ページ（36キロバイト）確保する必要がある。ここで，行列変換によってメモリ領域がどのように変化するかについて述べる。長さがMの配列Aと長さがNの配列Bを例に考える（M≧N）。変換前のメモリ空間はM×N=MNに対して，変換行列のサイズは，横方向がmin(M.N)=N，縦方向がM+N－1である。よって確保するメモリ空間は変換前と比べて$(M+N-1)/M$倍となる。このうち未使用のメモリ空間は，図6より(N×N)/2×2=N^2となる。したがって，6000×6000の場合には，36k×6000×(6000＋6000－1)/6000倍となり，36k×11999　421Mバイトが必要となる。

4 分散共有メモリシステムFagus

Fagusは，我々が開発中である自動並列化コンパイラMIRAIの実行時環境として実現したソフトウェア分散共有メモリシステムである。Fagusでは，共有メモリの一貫性制御をEC（RCのエミュレートも可能である）によって行う。ユーザは並列プログラムに，Fagusの機能を提供するライブラリlibFagusをリンクして使用する。

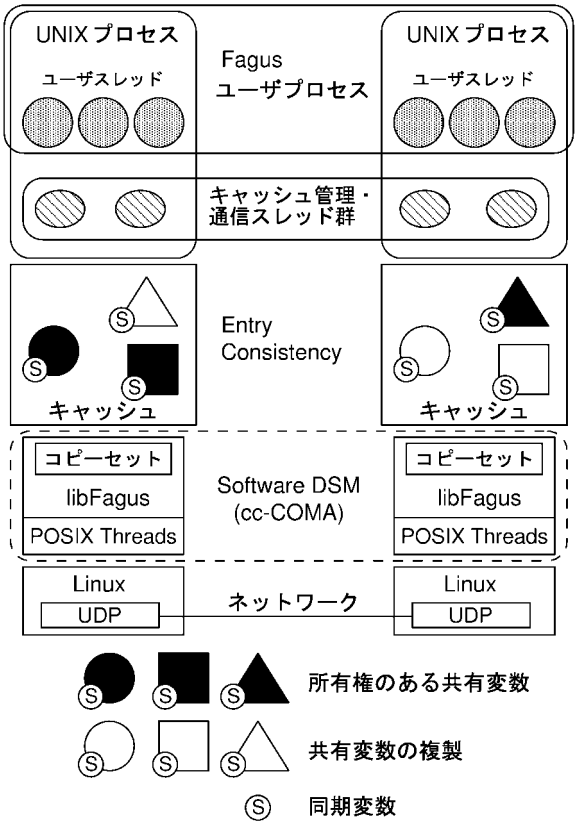

図7 Fagusの全体構成

通信には，POSIXスレッドとUDP/IPを利用して通信を行っている。

現在，FagusはLINUX上で動作しているが，LINUXそのものに変更を加えていないため，POSIXスレッドとUDP/IPが使用可能なUNIX系OSであれば，容易に移植可能である。図7にFagusの全体構成を示す。

プログラマまたは，コンパイラによって生成された並列プログラムは，ユーザによって各ノードへ配置される。実行時には，プロセス内で1つ以上のユーザスレッドが生成され，ユーザスレッドを用いて並列プログラムの各タスクを実行する。共有変数は，Fagusのメモリ割り当て機能を利用することで，プロセスの仮想メモリ空間に割り当てる．ユーザスレッドは、共有変数のアクセス時に必要に応じて、libFagusにキャッシュ制御の指示を与える。

本実験では、Fagusのキャッシュ制御インタフ

ェースである。fagus_barrier()のみを用いて計測を行った。fagus_barrier()は、並列計算に参加する全ノードの共有メモリの更新と、プログラム実行の同期を同時に行う。キャッシュ制御により，他のノードとの通信が必要になった場合は，libFagusの通信モジュールを通じて通信を行う。通信によって共有変数の転送と排他制御を実行し，共有変数の一貫性制御を実現している。

5 評価

5.1 評価環境

実験に使用した配列は, いずれもアミノ酸に相当する20種類のアルファベットをランダムに5000個並べたものである。本実験におけるペアワイズアライメントプログラムは, 特定のデータに対して効果的な結果を得るような工夫をしていない。したがって，どのような入力に対しても一様な計算時間を要する。実験に使用したシステムは，プロセッサがPentiumIII800MHz，メモリが512MB，ギガビットイーサネットで接続された16台のPC互換機である。

評価に用いたプログラムは, ウェーブフロントループ中でのノードの割り当て方法が異なる三種類を用いて実行速度を測定した。三種類のプログラムの概要を以下に示す。

評価1：0番目のイタレーションをノード1，1番目をノード1･･･n台目のイタレーションをノードnと順番にノードに割り付ける。すべてのノードに割り付けた場合には, 再びノード0から順に割り付ける。

評価2：DPの変換行列を縦方向にノードの台数で分割し, 分割された各領域をノードに割り付ける。

評価3：ウェーブフロントループ毎にループの長さを調べ, その長さをノードの台数で分割し，分割された各領域をノードに割り付ける。

5.2 実行結果

図8に評価1の結果を，図9に評価2を，図10に評価3の結果をそれぞれ示す。図8より，評価1ではノードの増加により実行時間が増加している。これは，各ノードの計算結果が行列中に散在してしまうために, データの一貫性を保つためのデータを転送するオーバーヘッドが大きくなったためと考えられる。また，タイルのサイズが増加するにつれて実行時間が短くなるが, ある大きさよりも大きくなると, 実行時間が長くなる。

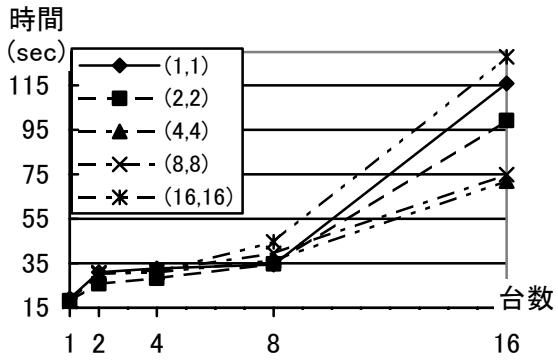

図８　評価1の実行結果

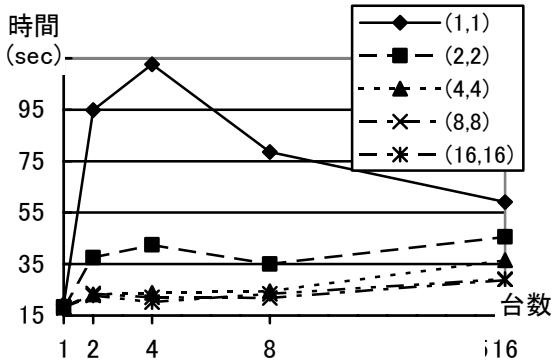

図9 評価2の実行結果

図9より，評価2では，タイルのサイズが小さいほど実行時間が長くなった。これはバリア同期の回数が多くなり, 通信量が増加したためと考えられる。また，ノードの台数が増加しても台数効果が得られないのは, ノードの割り付け方が最大並列性を得られにくいからであると考えられる。

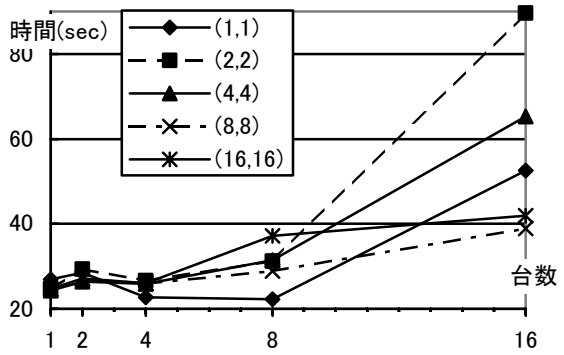

図10 評価3の実行結果

図10より，評価3では，各ループでノードの割り付けのための計算によるオーバーヘッドのため，全体的に計算時間が長くなった。また，タイルサイズの増加により, 全体的に計算速度が向上した。

5.3 考察

全体的に, ノードの増加による台数効果は得られなかった。むしろ台数が増加することで，全体の通信量が増加し, 計算速度の低下につながった。この理由は, 解空間の狭さと無駄な通信が考えられる。解空間が狭いために，1ノードでの実行時間が数秒程度の場合しか評価することができなかった。これは，3.3節で述べたように、行列を変換することで不要なメモリ空間ができてしまい, 有限なメモリ資源を浪費しているためである。さらに, すべてのノードが共有するメモリ空間を変換行列全体としたため, 一部のノードでは不要なデータも，全ノードで送受信してい，通信量の増大につながったためである。

これらを改善するために, 以下のような方法を検討中である。第一に，変換行列の無駄なメモリ使用を無くすために, 変換行列をさらに変換する。第二に, 各ノードが計算するデータをそれぞれのローカルのメモリに格納し, 共有メモリはタイル間で依存関係のある領域のみに限定する。これらにより, 各ノードが変換行列全体をローカルメモリに格納せずに, 利用できるメモリ空間を増大させる。加えて，通信量の削減を可能にする。

6 まとめ

分散共有メモリを用いてDP行列を並列実行した。また，タイリングを用いてバリア同期の回数を減らすことで通信量の低減を行った。しかし，使用したアミノ酸配列の長さが短かったため, 台数効果が通信にかかるオーバーヘッドを上回ることができなかった。今後，利用できるメモリ空間の増大と, 通信量と通信回数の削減を行う予定である。

参考文献

[1]Richard Durbin・Sean R. Eddy・Anders Krogh・Graeme Mitchison著，阿久津達也・浅井潔・矢田哲士訳: バイオインフォマティクス－確率モデルによる遺伝子配列解析，医学出版(2001).

[2]横手聡，齋藤彰一，上原哲太郎，國枝義敏:コンパイラによる制御が可能なDSMシステム「Fagus」の実現,情報処理学会研究会報告2000-OS-85，Vol. 2000,No. 75，pp. 47-54 (2000).

[3]坂田聡子，長嶋雲兵，佐藤三久，関口智嗣，細矢治夫: ホモロジー解析プログラムを用いたワークステーションクラスタ，TMC CM-5, Intel Paragonの性能評価，情報処理学会論文誌, Vol. 37, No. 7, pp. 1440-1450 (1996).

[4]金久實著，ポストゲノム情報への招待，共立出版(2001).

[5]中村春木, 中井謙太著: バイオテクノロジーのためのコンピュータ入門，コロナ社(1995).