動的負荷分散を用いたDPマッチングの並列処理

酒井 秀宣　　松尾 啓志　　岩田 彰

名古屋工業大学 電気情報工学科
名古屋市昭和区御器所町
Email: matsuo@elcom.nitech.ac.jp

あらまし

DP マッチングの処理速度を高速化する上で，ワークステーションクラスタ上でタイリング法を用いた並列処理は有効な手法である．しかし，従来法では計算機の負荷変動，偏りが問題となり，全体の処理速度に大きく影響していた．そこで，本研究では各計算機の負荷変動を考慮し，計算機の処理能力に応じたタイルサイズを割り当てる事により，効率よく負荷分散を行う動的負荷分散法を提案する．さらに，ワークステーションクラスタ上で実験を行い，本手法の有効性を明らかにする．

キーワード: DP マッチング，並列処理，タイリング法，動的負荷分散，ワークステーションクラスタ

Dynamic Load Balancing Method for the Parallel DP Matching

Hidenori Sakai, Hiroshi Matsuo and Akira Iwata

Department of Electrical and Computer Engineer, Nagoya Institute of Technology
Gokiso-cho,Showa-ku,Nagoya
Email: matsuo@elcom.nitech.ac.jp

ABSTRACT

The parallel dynamic programming algorithm named "Tilting method" is proposed for workstation cluster. But this algorithm does not assume the load unbalancing problem of each workstation. However, load unbalancing problem causes the performance decrease for this algorithm. In this paper, new two load balancing algorithms are proposed that correspond with network communication latency. And the processing performance using this dynamic load balancing algorithm are evaluated by a workstation cluster system.

Keywords: DP Matching, Parallel Processing, Tiling Method, Dynamic Load Balancing, Workstation Cluster

1　はじめに

パターン長が異なる 2 パターン間の類似度を動的計画法 (DP:Dynamic Programing method) によって定量化する DP マッチングは，音声認識やアミノ酸の塩基配列の類似性の判定など幅広い分野で利用されている．また今後はゲノム計画など，より大きなパターンのマッチングでの利用が考えられる．しかし，動的計画法はマッチングするパターンサイズが大きくなるにつれ，その 2 乗に比例した行列計算が必要となる．従って 1 台の計算機で逐次処理を行うには限界がある．

大規模な数値解析ではこのような問題の有力な解決手段の一つとして並列処理による高速化が考えられる．動的計画法に関しても，共有メモリ型マルチプロセッサ上におけるバンドアルゴリズムを用いた並列化 (同期型，無通信)[2]，ワークステーションクラスタ上でタイリングアルゴリズムを用いた並列化 (メッセージパッシング) [1] などが報告されている．これらの従来手法では計算中の各計算機の負荷は常に一定と考え，問題を等分割して，各計算機に割り当てていた．しかし，ワークステーションクラスタの様なネットワーク接続された複数台の計算機を利用する場合，各々の計算機負荷は常に変動し，また偏りがある．このような場合，高負荷の計算機がボトルネックとなり全体の処理速度に大きく影響する．

本研究では，計算中の各計算機の負荷変動を考慮し，タイルサイズを動的に変更する事により効率良く負荷分散を行う 2 種類の動的負荷分散法を提案する．さらに，ワークステーションクラスタ上で実験を行い，本手法の有効性を明らかにする．

2　DP マッチング

パターン間の類似性を，動的計画法 (DP) に基づいた定量化式によって数値化するパターンマッチングの一手法を DP マッチングと呼ぶ．パターン (M, L) 間のある並置に対して類似性を定量化するために，パターンの一致，不一致，欠失に対して適当な重み (β, γ) を与え，その和を算出する手法である (図 1)．パターン間の類似度は，この和から求められる類似度行列 F の要素の最大値と定義し，最大値までの経路が 2 パターン間の最良の並置となる．

$$
\begin{aligned}
&dp(M, L)\{\\
&\ similarity = -\infty;\\
&\ \gamma = -3;\\
&\ for(i = 1; i \leq L; i++)\\
&\quad for(j = 1; j \leq M; j++)\{\\
&\qquad if(M[j] == L[i])\beta = 1\ else\ \beta = -2;\\
&\qquad F(i, j) = max\{F(i-1, j-1) + \beta,\\
&\qquad\quad F(i-1, j) + \gamma, F(i, j-1) + \gamma;\\
&\qquad\quad similarity = max\{F(i, j), similarity\};\\
&\quad \}\\
&\ return\ \ similarity;\\
&\}
\end{aligned}
$$

図 1: 定量化式

3　DP の並列化

DP マッチングは，マッチングするパターンサイズの 2 乗に比例した大きさの類似度行列の計算を行う．つまり，サイズの 2 乗に比例した計算時間を必要とするため，大きなパターンサイズのマッチングを 1 台の計算機で逐次処理するには限界がある．そこで，従来よりこのような問題の解決手段として並列処理による高速化が提案されている．

3.1　DP の並列性

まず DP の並列性について考える．DP を用いた 2 つの 1 次元パターン M，L の類似度行列 F の計算を逐次ループで記述すると，図 1 のような 2 重ループ構造となる．ループ内の繰り返しである要素計算は，図 2 に示すような 2 次元空間を構成する．行列要素の参照に伴うデータの流れにより，要素計算では矢印で示される依存関係がある．すなわち，行列 F の各要素 $F(i, j)$ は $F(i-1, j)$，$F(i, j-1)$，$F(i-1, j-1)$ の 3 つの要素を必要とする．$F(i, j)$ の計算に必要な 3 つの要素のうち，$F(i-1, j)$，$F(i, j-1)$ を求める計算時に $F(i-1, j-1)$ の計算結果を必要とする．つまり，$F(i, j)$ の計算には左の要素 $F(i-1, j)$ と上の行の要素 $F(i, j-1)$ の 2 つの要素が揃えば計算可能となる．図 2 に示す様に，イタレーション空間を i 次元で分割して行単位に計算機 P_1，P_2，P_3，…と割り付けることを考える．計算機が

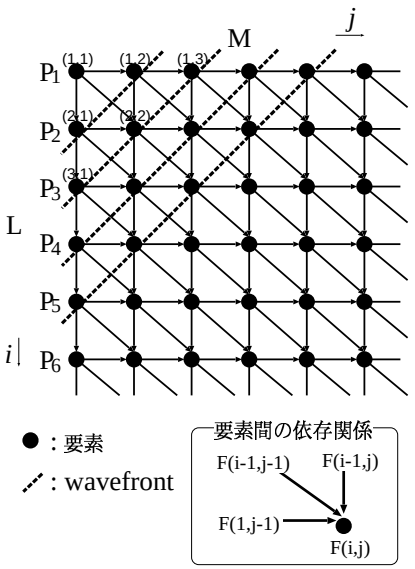

図 2: 要素間の依存関係

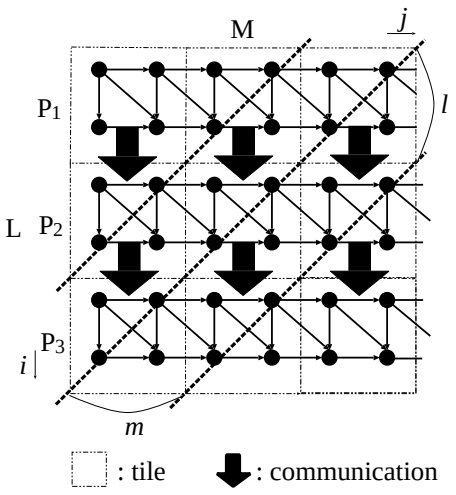

図 3: タイリング法

各行で $F(i,j)$ の計算に必要な 2 つのうち，左の要素 $F(i-1,j)$ は自分自身が計算を行っている．従って要素 $F(i,j-1)$ が上の行からリレー転送式に送られてくる場合，全体としては反対角線方向にウェーブフロント的に依存性が解消される．図 2 では破線がウェーブフロントを表し，これに並行な反対角線方向の要素間には依存関係が存在せず，並列処理が可能となる [3]．最初のステップで P_1 が要素 $F(1,1)$ を計算した結果を計算機 P_2 に転送する．次のステップでは $(1,2)$ を P_1，$(2,1)$ を P_2 が並列に実行し，計算結果をそれぞれ下の行の計算機に転送する．以下同様に，要素参照で依存関係のない反対角線方向の要素計算を各ステップで並列に実行することができる．

3.2 タイリング法による並列化

前節では並列化を行う際に，要素計算毎に計算機を割り当て，計算結果を転送した．しかし，並列計算にワークステーションクラスタの様なネットワーク接続された複数の計算機を用いると，計算機内の計算時間に比べて計算機間の通信時間が非常に大きくなる．また，計算機の数が有限であることを考えると，この割り当ては実用的ではない．そこで，お茶の水女子大の坂田らはタイリング法と呼ばれる，複数のイタレーションをまとめた「タイル」を単位として演算と通信を行う手法を用いて並列処理を行っている [1]．タイルは類似度行列を矩形で分割した部分行列で構成される (図 3)．行列 F において，各要素計算が 3 つの要素に依存していたのと同様に，各タイルについても同じ関係が成り立つ．すなわち，反対角線方向のタイルは全て各計算機が独立に並列計算できる．タイルを単位として通信を行えば，タイリングしない場合に比べて通信回数を少なくすることができる．また，必要な計算結果のみを送信すればよいので通信量も少なくできる．

タイルサイズは，類似度行列を行方向に計算機数 P で等分割し，$l(=L/P)$ 行ごとに計算機を割り当てた場合．列方向のタイル幅を最小で 1，最大で M/P に分割し変更する事ができる．最大の並列性の保持を考慮すると，タイルサイズはできるだけ小さくして，計算初期および終期の非効率さを押さえる必要がある．一方通信コスト (回数) を押さえるには，タイルサイズをできるだけ大きくして，通信回数対計算回数の比を小さくする必要がある．

4 タイリング法の動的並列化

複数の計算機を利用して並列計算を行う場合，同じ負荷の計算機を複数台利用する事ができれば理想的である．しかし，ワークステーションクラスタの様な環境

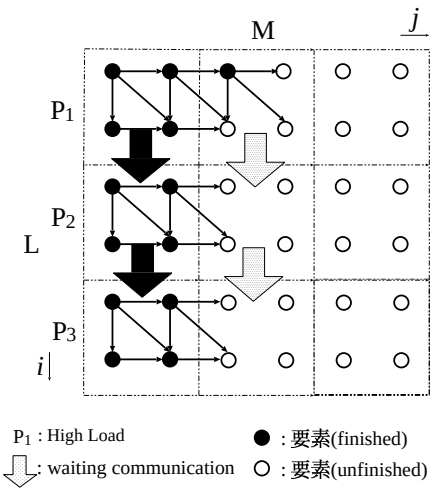

図 4: タイリング法の問題点

では，各計算機の負荷は常に様々なユーザーのプロセスによって変動し，また偏りがあると考えられる．従来のタイリング法は各計算機の処理速度は同一で常に一定と考え，類似度行列をを計算機数で等分割したタイルを割り当てていた．このため計算機間に負荷の偏りがある場合 (図 4)，負荷の高い計算機 P_1 は，他の計算機と比較してタイル計算に時間が必要となり，次の計算機 P_2 への送信が遅れる．計算機 P_2 はタイル計算に必要な計算結果を受けとれないので，計算が始められず，さらに次の計算機 P_3 への送信が遅れる．このように，各計算機は依存するタイルの処理が終了しないと，タイル処理を開始することができないので，結果的にシステム全体が負荷の高い計算機の処理速度に左右されてしまう．

そこで本研究では，各計算機の負荷変動を考慮し，処理能力に合わせてタイルサイズを動的に変化させる事により，効率良く負荷分散を行い実行時間の短縮を図る動的負荷分散法を提案する．各計算機の負荷値に，直前のステップでタイル処理に必要とした時間を，タイルサイズで正規化した 1 要素の計算時間を用いる．この値が大きいほど負荷の高い計算機と考え，計算機の負荷値を元に，次ステップの計算機間のタイル処理時間を均一化するようタイルサイズの変更を行う．各計算機が依存するタイル処理の終了待ち時間を短縮し，効率良く利用することによって処理時間の短縮を図る．

動的負荷分散法にもタイリング法のタイルサイズのように，通信コストをパラメータとしたトレードオフが存在すると考え，以下に示す 2 種類の手法を提案する．

- 集中型　(通信コスト 大，負荷分散効果 大)
  スケジューラーが全計算機の負荷情報を用いて，1 ステップ毎に各計算機のタイルサイズを大局的に変更する．

- 分散型　(通信コスト 小，負荷分散効果 小)
  各計算機が計算機自身の負荷情報のみを用いて，タイルサイズを局所的に変更する．

4.1 集中型負荷分散

集中型ではタイル処理を行う計算機以外に，新たに各計算機のタイルサイズを一括して行うスケジューラーを付加する．スケジューラーは 1 ステップ毎に全計算機の 1 要素の計算時間を把握し，各計算機に 1 要素の計算時間に比例したタイルサイズを割り当てる．次ステップのタイル処理時間を大局的に定量化し，待ち時間を小さくすることによって全体の処理速度の向上を図る．

スケジューラーは，動的負荷分散を行うために 1 ステップ毎に以下の処理を行う．

step1 各計算機の計算時間，計算結果の受信
タイル処理を終えた計算機 P_i から，

- 1 要素の計算に必要とした時間 T_i
- 計算結果テーブル

を受け取る．T_i は計算機 P_i のタイル処理時間を，P_i が処理したタイルサイズ $m_i * l$ で正規化した値である．また，計算結果テーブルは P_i が処理したタイルの計算結果のうち，隣接したタイル処理に必要な結果をテーブル化したものである．

step2 サイズ決定
スケジューラーは，各計算機の 1 要素の計算時間 T_i が大きなものほど負荷が高い計算機として，計算機を負荷値の高い順にソートする．負荷の一番低い計算機の計算時間 T_{fast}，負荷の一番高い計算機の計算時間を T_{slow} とする．負荷値が 2 つの計算機の中間の計算機を仮定し，その計算機が基

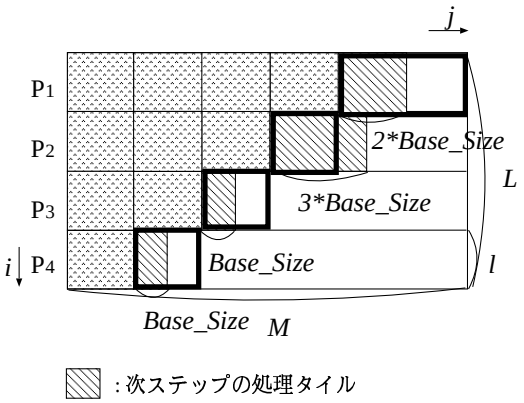

図 5: タイルサイズ変更例

本タイル幅 $Base\_Size$ を処理する時間を次ステップの単位時間として式 (1) より求める．

$$Next\_Step\_Time = \frac{T_{slow} + T_{fast}}{2} * Base\_Size * l \quad (1)$$

各計算機 P_i の次ステップのタイル処理時間を $Next\_Step\_Time$ 定量化するような新たなタイル幅 m_1 を式 (2) より求める．

$$m_i = \frac{Next\_Step\_Time}{T_i * l} \quad (2)$$

しかし，新たなタイル幅 m_i は，依存関係のある計算機 P_{i-1} が直前のステップまでにタイル処理を終え，必要な計算結果テーブルが揃えられている以上の大きさ (図 5 の太枠) を割り当てることはできない．図 5 の例では，依存関係を無視してタイルサイズを割り当てられた計算機 P_2 は，依存している計算機 P_1 が同一ステップで計算しているタイルの計算結果テーブルを必要とする．しかし，同一ステップで他の計算機が計算した結果は，そのステップ終了後にしか受け取ることができず，デッドロックに陥ってしまう．これは集中型負荷分散は計算機間通信をステップ毎に同期を取って行っているからである．依存関係を保つには，P_2 に割り当てるタイルサイズを小さくする手法が考えられる．しかし，結果的にタイル処理を早く終え，他の計算機の処理待ちとなり，負荷分散効果を得られることができない．

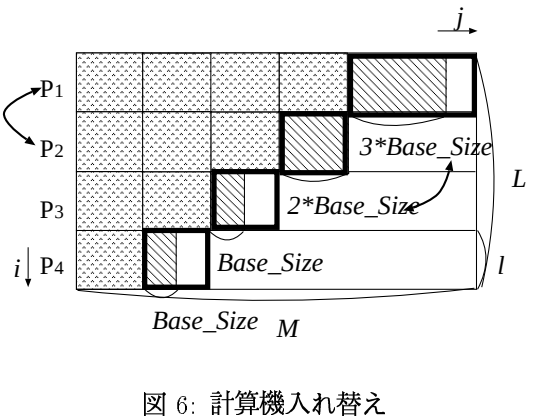

図 6: 計算機入れ替え

step3 計算機間の依存関係の変更

以上の問題点を解決すべく，次ステップで計算機に割り当てるタイルサイズが，各計算機に収まるように計算機間の依存関係を入れ換える (図 6)．

step4 タイルサイズと計算結果テーブルを送信

全計算機の処理タイルサイズが決定された時点で，各計算機へ次ステップのタイルサイズと，タイル処理に必要な計算結果テーブルを送信する．

以上の様に，1 ステップ毎に全計算機のタイルサイズを処理速度に応じて繰り返し変更する事により，大局的な負荷分散を行い，処理速度の向上を図る．しかし，集中型は負荷分散に必要な通信コストが大きい．これは，**step2** でタイルサイズを変更することによって，計算機毎に処理タイルサイズが異なり，直前のステップでの計算結果が次のステップで使われるとは限らない．また，**step3** で計算機を入れ替えることにより，計算機間の依存関係もステップ毎に異なるので，単純に次の計算機へ送信する ”ことが出来ないため，従来は計算機間で受渡していたタイル処理に必要な計算結果テーブルを，スケジューラーを介して行っているからである．

4.2 分散型負荷分散手法

前述の集中型は，1 ステップ毎に全計算機の同期を取って大局的にタイルサイズを変更する事によって，計

算機間の依存関係を入れ換えるなど，負荷分散に必要な通信コストが大きい．そこで，各計算機は計算機自身の負荷情報のみを用いてタイルサイズを変更し，タイルサイズを変更しても計算機間の通信量が変化しない，負荷分散のために特別な通信コストを必要としない分散型負荷分散手法を提案する．

まず，類似度行列を行方向 (L) は計算機数 P で等分割した $l(=L/P)$，j 次元のイタレーションを初期サイズ m で分割してタイルを構成する (図 7)．但しタイル幅を変更しても計算機間の通信量は常に一定であるように，タイルサイズの変更方向と同じ列方向に計算機を割り当てる．タイルサイズの変更は，各計算機 P_i が順に列方向に割り当てられたタイルを P 個処理し終え，新しい列に移動する際に計算機自身の負荷情報のみを利用して，非同期に行う．計算機 P_i の新たなタイル幅 m_i は，直前に処理してたタイルサイズを m'_i，1 要素の計算時間を T_i，1 要素の過去の平均計算時間 Ave_i として式 (3) によりに決定する．

$$m_i \begin{cases} m'_i * INC : T_i < Ave_i * DEC \\ m'_i * DEC : T_i > Ave_i * INC \end{cases} \quad (3)$$

$$DEC(=0.75) < 1.0 < INC(=1.25)$$

過去の平均計算時間と，直前の計算時間を比較して，より直前の計算時間が大きければ，計算機の負荷は高いとみなし，タイルサイズを小さくする．逆に，より計算時間が小さければ，負荷が軽いとみなし，タイルサイズを大きくする (図 7)．

以上の様に各計算機が，局所的にタイルサイズを変更することにより，システム全体の負荷を分散し，処理速度を向上できると考える．

5 結果

提案した 2 種類の動的負荷分散法の有効性を示すため，ワークステーションクラスタ上で実験を行った．実験には SiliconGraphics Indy(R4600PC 133MHz,Memory 64MByte)5 台 (内 1 台は集中型のスケジューラー) を 10Mbps の Ethernet で相互結合したワークステーションクラスタを用いた．計算機間の通信にはメッセージパッシングライブラリ PVM3.3.10 を用いた [4]．マッチングパターンにアルファベット 26 文字をランダムに

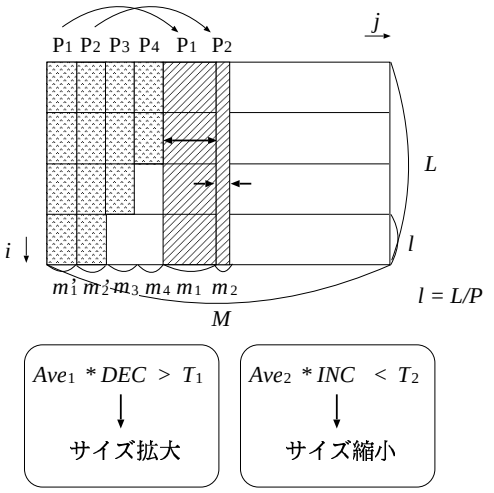

図 7: タイルサイズ変更

10000 個並べたものを利用した．また，タイル処理を行うプロセスは 4 台の計算機上で 1 つ実行させた．各実験結果は 20 回の計測値の平均である．

5.1 タイリングによる並列効果

まず，今回 DP を並列化するのに基本としたタイリング法を用いて，タイルサイズの変化に伴う処理時間を計測した．

実験結果を図 8 に示す．図の横軸はタイルサイズ，縦軸は処理時間を表している．結果のグラフはすり鉢状になっており，タイルサイズ 200 の時が最小の処理時間でタイルサイズが大き過ぎても，小さ過ぎても処理時間が遅くなっている事がわかる．タイルサイズは大き過ぎると，4 台が並列にタイル処理を行える最大の並列度を保てるステップ数が少なくなる．小さ過ぎると，最大の並列度が何ステップにも渡って保たれるが，通信回数の増大が大きくなる．共にタイルの処理時間と通信時間のバランスがとれなくなり，処理時間に影響していると考えられる．

5.2 動的負荷分散効果

次に，各計算機負荷を変動させ，集中型，分散型による動的負荷分散法の有効性を確認した．初期タイルサイ

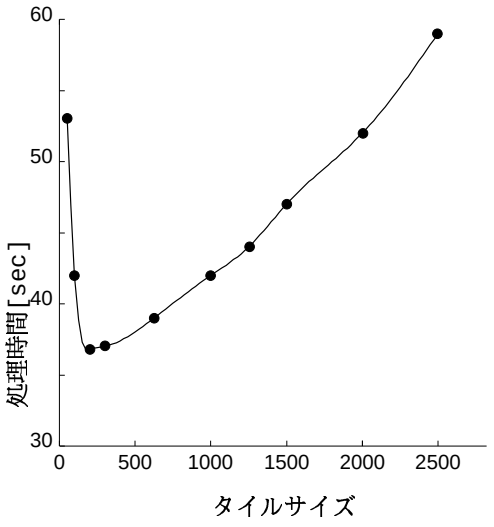

図 8: タイルサイズと処理時間

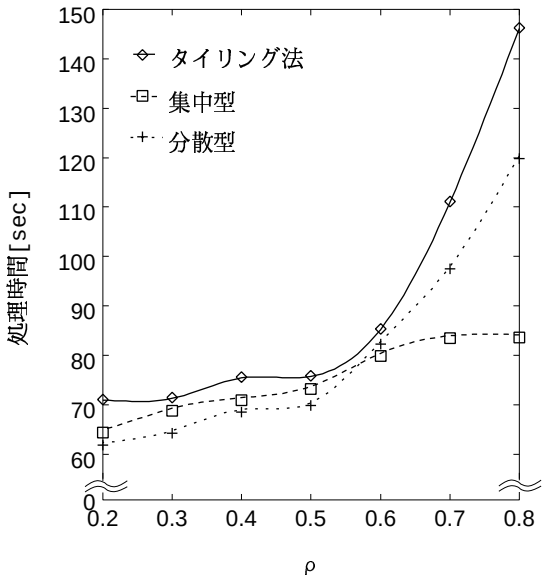

図 9: 負荷変動 (1 台)

ズを負荷のない状態で最適なサイズであった 200 とした．各計算機には外乱として，平均サービス率 μ[task/sec]，平均到着率 λ[task/sec] (共に指数分布に従う) 負荷を与えた．

まず，平均サービス時間 $1/\mu$ は 2.5[sec] に固定，平均到着率 λ は 3 台の計算機 ($P_{2,3,4}$) は 0.2 に固定し，1 台の計算機 (P_1) への平均到着率を 0.08 から 0.04 刻みで 0.32 まで変化させた．図 9 の実験結果は，横軸は外乱を変動させている計算機の利用率 $\rho(=\lambda/\mu)$ で，縦軸は処理時間を表す．

従来のタイリング法の処理時間は，P_1 の ρ が 0.5 までは ρ の値にかかわらず 70 秒台．0.6 を越えると ρ の増加に伴い目立って大きくなっている．これは，ρ が 0.5 未満の時には，外乱の平均到着率を 0.2(ρ=0.5) に固定した $P_{2,3,4}$ の計算機のいずれかが，常に負荷が高くボトルネックとなっているためと考えられる．また，ρ が 0.5 より大きくなるにしたがって P_1 の負荷が他の 3 台より高くなり，ボトルネックとなるためと考えられる．タイリング法に比べ，集中型，分散型，共に動的負荷分散による処理時間の短縮を確認することができる．分散型は，ρ が大きくなるにつれて処理時間も大きくなるのだが，常にタイリング法に比べて 1〜2 割程度の高速化が得られた．集中型は ρ が大きくなっても，他の 2 手法に比べて処理時間への影響が小さい．これは，P_1 の ρ が 0.5 より大きくなり，1 要素の処理時間差が大きくなるに連れ，より負荷の低い $P_{2,3,4}$ の 3 台に大きなタイル処理を行わせ有効に利用しているためである．高負荷時にはタイリング法と比べて約 8 割の高速化が得られた．

また，外乱を変化させる計算機数を 2，3，4 台と増やして実験を行った所，3 手法の差は段々と少なくなってきた．全ての計算機 ($P_{1,2,3,4}$) へ与える外乱の平均到着率を 0.08 から 0.04 刻みで 0.32 まで変化させ，処理時間を測定した結果を図 10 に示す．

ρ が小さく計算機負荷が低い時に集中型の処理時間がタイリング法より大きくなった．これは，集中型の通信コストが動的負荷分散による効果を上回ってしまっているためだと考えられる．それに対して，分散型は動的負荷分散に特別な通信コストを必要としないので，タイリング法に比べて 2 割程度の高速化が得られている．また，ρ が大きくなりシステム全体の計算機負荷が高くなると，分散型はタイリング法との差がほとんどなくなった．これは各計算機が激しく負荷変動する中，集中型に比べて P 倍のタイルサイズ処理毎の間隔でしか負荷分散を行えず，他の計算機の情報を利用せずに負荷分散を行っているために激しい負荷変動に追従できないためと考えられる．逆に集中型は，負荷分散の効果が通信コストを上回り，他手法に比べ 2〜3 割

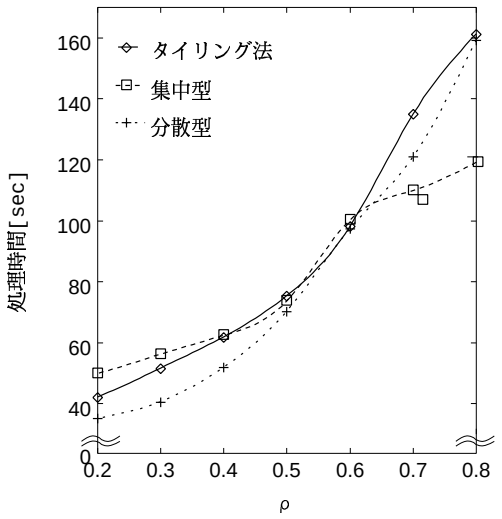

図 10: 負荷変動 (4 台)

程度の高速化が得られている.

6 まとめ

本稿では，従来のタイリング法を用いた DP マッチングの静的負荷分散による並列処理に対して，ワークステーヨンクラスタ上などの各計算機の負荷が変動し，偏りのある場合に，高負荷の計算機がボトルネックとなり効率良く処理を行えないという問題点を提示した．この問題を解決するために，並列処理中に動的にタイルサイズを変更し負荷分散を行う事により，効率良く計算を行い処理速度を向上させる動的負荷分散法を提案した．

動的負荷分散法には，新たにシステム内に付加した負荷分散を行うスケジューラーが収集した各計算機の負荷情報を元に，ステップ毎にタイルサイズを正規化して計算機間の負荷の均一化を図り，システム全体の処理速度を向上させる集中型，各計算機が計算機自身の負荷情報のみを用いて，非同期にタイルサイズの変更を行う事により，計算機間の負荷の均一化を図り，システム全体の処理速度を向上させる分散型の２つを提案した．また提案手法の有効性を示すべく，ワークステーションクラスタ上の計算機に対して負荷変動を与え，従来手法との比較実験を行った．実験の結果，ワークステーションクラスタ上の計算機負荷が高く，大きく偏りがある場合に集中型が 5〜8 割の高速化が得られた．また，分散型は計算機負荷の偏りに関係なく低負荷時には常に 1〜2 割の高速化を得ることができた．しかし，計算機全体の負荷が低い場合に，集中型では動的負荷分散による効果が通信コストを上回り負荷分散のない従来のタイリング法より処理時間が大きくなってしまった．また，計算機全体の負荷が高い場合には，計算機毎にタイルサイズを変更する局所的な負荷分散を行う分散型は，システム全体の激しい負荷変動に追従できず，タイリング法との処理時間差が見られない．

今後の課題として，集中型に対しては，システム全体が低負荷時には負荷分散のための通信コストが小さい方式との切替えを行うハイブリッド方式の検討．また，分散型に対してはタイルサイズ決定に他の計算機の負荷情報を採り入れることによる負荷分散精度の向上などが考えられる．

参考文献

[1] 坂田 , 日向寺 , 長島 , 佐藤 , 関口 , 細矢 : 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, Vol. 36, No. 8, pp. 1987–1994 (1995)

[2] Edmonds, P.,Chu, E. and George, A.: Dynamic Programing on a Shared-Memory Multi Processor, Parallel Computing, Vol. 19, pp. 9–22 (1993).

[3] Rmanujan, J., and Sadayappan, P.: Tiling Multidemensional Iteration Space for Multicomputers, J. Parallel and Distributed Computing, Vol. 16, pp. 108–120 (1992).

[4] Geist, A., Beguelin, A., Dongarra, J., Jiang,W., Mancheck,R. and Sunderam, V.:PVM3 User's Guide and Reference Manual (邦訳 村田 英明: PVM3 ユーザーズガイド & リファレンスマニュアル) (1995)

名古屋工業大学 電気情報工学科 岩田研究室
〒 466-8555 名古屋市 昭和区 御器所町
TEL 052-735-5446 FAX 052-735-5464