マルチコア CPU 上での マルチプルアラインメントの並列処理
Parallel Processing of Multipule Alignment on a Multi-core CPU

平原　海詞
Kashi Hirahara
広島市立大学情報科学研究科
Email: Kaishi@de.info.hiroshima-cu.ac.jp

田村　慶一
Keiichi Tamura
広島市立大学情報科学研究科
Email: ktamura@hiroshima-cu.ac.jp

北上 始
Hajime Kitakami
広島市立大学情報科学研究科
Email: kitakami@hiroshima-cu.ac.jp

Abstract-**Multiple sequence alignment is one important process that is used in various scenes of bioinformatics. Multiple sequence alignment is a time consuming process. There are a lot of parallelization method of multiple sequence alignment. In this paper, we propose a method for multiple sequence alignment in parallel on a multi-core CPU.**

I.　はじめに

マルチプルアラインメントは，複数の文字配列の類似する部分を並べ合わせる操作で，バイオインフォマティクスの様々な場面で利用されている重要な処理のひとつである．マルチプルアラインメントは，文字配列数が多くなると計算時間が大きくなるため，これまで多くの並列計算機上で並列化が行われている．

本研究では，マルチコア CPU 上でのマルチプルアラインメントの並列化手法を提案する．近年，計算機の CPU のマルチコア化が進んでおり，マルチコア CPU を用いて，マルチプルアラインメントを高速化する手法を開発することは重要な課題である．具体的には，累進法の第 1 段階であるペアワイズアラインメント処理の並列化を，タスク分割並列化手法，パイプライン並列化手法，ブロック分割並列化手法によって行い，高速化を図る．

タスク分割並列化手法は単純な方法であるがメモリの使用量が多く，パイプライン並列化手法はタスク分割と比べると処理がやや複雑で中程度のメモリを使用する手法であり，ブロック分割並列化手法はメモリの使用量を小さくする代わりに処理が複雑となっている．本論文では 3 つの手法を実際に実装し，性能評価を行った結果を報告する．

本論文の構成は以下の通りである．第Ⅱ章で関連研究について述べ，第Ⅲ章で累進法によるマルチプルアラインメントについて説明する．第Ⅳ章で提案手法を示し，第Ⅴ章で実験・評価を行う．第Ⅵ章で本論文のまとめを行う．

II.　関連研究

マルチプルアラインメントは非常に多くの計算時間を必要とする処理として知られており，アルゴリズムを工夫することによる高速化や並列計算機上で並列化に関する研究が行われてきた（文献[1]）．

文献[2]では，分散メモリ並列計算機である PC クラスタ上での累進法の反復改善過程を並列化する手法を提案している．文献[2]ではマスタ・ワーカモデルを用いて並列化を行っている．文献[3]では文献[2]の手法に加えて，A*アルゴリズムを用いて，単一のペアワイズアラインメント処理についてスコア計算が不要な領域を枝刈りすることで，処理の高速化を図っている．これ以外にも，マルチプルアラインメントのアルゴリズムとして広く利用されている ClustalW や T-Coffee なども PC クラスタ上で並列化（文献[4]，文献[5]）が行われている．

一方，単精度での計算速度の優秀さから，グラフィックス処理の高速化を実現する GPU グリッドを用いてアラインメント処理を行うという試み（文献[5]，文献[6]）もある．GPU は大量データに対する計算に特化し単純な計算のみを行う多数のコアを用いたモジュールである. GPU が持つコアは SIMD 型の演算装置であり，ベクトル化を行うことにより，スコア計算の処理速度を向上させることができる．ただし，これらの GPU グリッド上での研究は 2 つの文字配列間のアラインメント処理であるペアワイズアラインメント単体の処理を大量に行うことのみを研究対象としており，複雑な制御を苦手とする GPU モジュールのみではマルチプルアラインメントをすべて並列化することは困難である．

近年，CPU のマルチコア化が年々進んでおり，マルチコア CPU の資源を有効に活用するためにマルチコア CPU 上で効率的な並列化手法を考えることは，重要な課題のひとつとなっている．本研究では，このマルチコア CPU 上での累進法の並列化手法の開発を目指している．マルチコア CPU 上ではメモリやキャッシュを複

数のコアで共有するため，共有資源を複数のコアで奪い合いをしないようなデータ構造や処理方式を考える必要がある．

本研究では，このマルチコアプロセッサ上での累進法の並列化手法の開発を目指している．メモリ使用量，処理の複雑さ等の観点から，タスク分割並列化手法，パイプライン並列化手法，ブロック分割並列化手法の 3 つの手法について比較・検討を行う．

III. 累進法によるマルチプルアラインメント

本章では，本研究が対象としているマルチプルアラインメントと累進法によるマルチプルアラインメントについて説明する．

A. マルチプルアラインメント

バイオインフォマティクスにおけるマルチプルアラインメントは，タンパク質・アミノ酸残基を表す複数の文字配列から類似した部分文字配列を抜き出し，文字配列同士の相同性を比較することが目的である． 図 1 にマルチプルアラインメントの例を示す. 図 1 に示すように文字間にギャップと呼ばれる文字’-’を挿入し，s_1～s_4 の間で類似する文字を並べ合わせている．

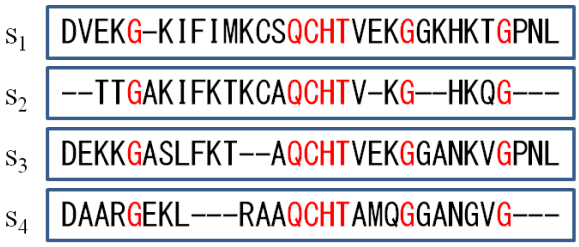

図 1 アラインメント処理を行った後の文字配列

B. 累進法

マルチプルアラインメントの問題は多次元の動的計画問題として定義することができる．文字配列の個数を N とし，配列長の大きさを平均 n とすると，計算量は $O(2^N n^N)$ となることが知られており，N=4 以上となると非常に計算時間がかかり動的計画法をそのまま解くことは困難である．そこで，厳密解でなく，近似解を求めるヒューリスティックが数多く提案されており，累進法は Clustal-W などをはじめとして様々な実用的なアルゴリズムとして利用されている．累進法によるマルチプルアラインメントは，次の 3 つの段階により行われる．

(1) すべての文字配列間でペアワイズアラインメントを行い，距離行列を作成する．(第一段階)
(2) 距離行列から案内木を作成する． (第二段階)
(3) 案内木を用いて，複数の文字配列間でアラインメント処理を行う． (第三段階)

累進法では (1) の一段階目が最も計算時間を必要とするため，本研究では，(1) のペアワイズアラインメントをマルチコア CPU 上で並列化する．

C. ペアワイズアラインメント処理

長さ m の文字列 x と長さ n の文字列 y が与えられたとする．これらの文字列から作成される経路行列の各格子点の評価関数を累積距離 $D_{i,j}$ とすると，$D_{i,j}$ は左上方向，上方向，左横方向の 3 つの経路からアラインメントのスコアが最大となるものを選択する．左上方向からくる場合，文字が一致すれば d が加算され，文字が不一致またはギャップ記号ならば d が減算されるというスコア関数 GS を用いる．また，上方向もしくは左横方向からくる場合は，必然的に d が減算される．これらを式で表すと，

$$D_{i,j} = \max \begin{cases} D_{i,j-1} - d \\ D_{i-1,j} - d \\ D_{i-1,j-1} + GS(x_i, y_j) \end{cases}$$

となる．x_i は文字列 x の i 番目の文字，y_i は文字列 y の j 番目の文字を表している．累進距離表の左上端から順に Di,j を計算していき，右下端に到達したときの累進距離が 2 つの文字列の最適アラインメントスコア (アラインメント処理の結果) である．表の初期値は，$D_{0,0}$=0, $D_{0,j}$ =-d*j (j=1~n), $D_{i,0}$=-d*i(i=1~m)で与えられる．

IV. 提案手法

本章では，マルチコア CPU 上でのペアワイズアラインメントの並列化手法について述べる．本研究では，累進法の第一段階の並列化手法として，タスク分割並列化手法，パイプライン並列化手法，ブロック分割並列化手法を提案する．タスク分割並列化手法は最も単純な方法であるが，コア数の 2 倍の文字配列が同時にメモリからキャッシュに呼び出されるため，コア間でメモリからキャッシュへの読み出しの競合が起こる可能性がある．この競合を防ぐため，読み込むアラインメント領域を帯状に分ける手法がパイプライン並列化手法である．パイプライン並列化の領域をさらにブロックとして分割するのがブロック分割並列化手法である．

A. タスク分割並列化手法

ペアワイズアラインメントにおいて，比較する文字配列の総数を n とすると，アラインメント処理を行う文字配列の組み合わせの総数は ${}_nC_2$ となる．この配列の組み合わせひとつひとつをタスクとして，タスクプールに保存しておく．タスク分割並列化手法では，1 つのペアワイズアラインメントをタスクとして複数のタスクを複数のコアで実行していく．タスクプールに全てのタスクを詰め終えたら，スレッドを生成する．各スレッドはタスクプールへタスクを取りに行き，当該するタスクの処理を行い，処理を終えたら再びタスクを取りに行く，という動作を繰り返す．これをタスクプールにタスクがなくなるまで繰り返す．

B. パイプライン並列化手法

単純に処理を分担させるタスク分割では，ペアワイズアラインメントを行うための文字配列 2 つ分の領域がスレッドが動作しているコア数分必要となる．文字配列のサイズが大きくなると，コア間で共有しているキャッシュミスが増加する可能性が高い．

そこで，パイプライン並列化手法ではひとつのペアワイズアラインメントをスレッド数で分割し，分割された領域に対する領域内のスコアの計算を各スレッドが分担して行う．

図 2 と図 3 にスレッド数が 4 の場合の例を示す．ひとつのペアワイズアラインメントのスコア計算を 4 つの領域に分割する．この例では，領域 A～D の 4 つの領域に分割している．スレッド 1 が領域 A を担当し，スレッド 2 が領域 B を担当し，スレッド 3 が領域 C を担当し，そして，スレッド 4 が領域 D を担当する．

最初にスレッド 1 が領域 A のスコア計算を行い，その最後の行の結果をスレッド 2 に渡し，スレッド 2 は領域 B のスコア計算をするというように流れ作業でペアワイズアラインメントを行う．この流れが途切れないように，スレッド 1 は次々に新しいペアワイズアラインメントの領域 A のスコア計算を行い，結果をスレッド 2 へと渡していく．

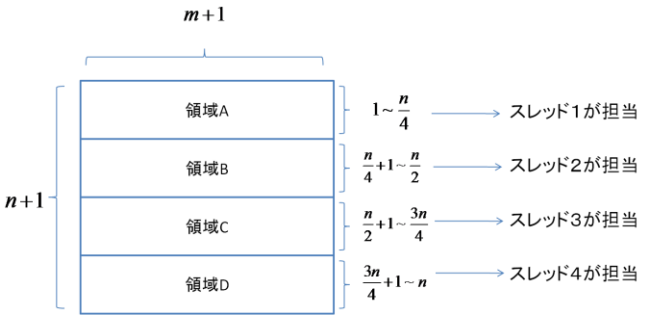

図 2　パイプライン並列化手法：各スレッドの担当領域

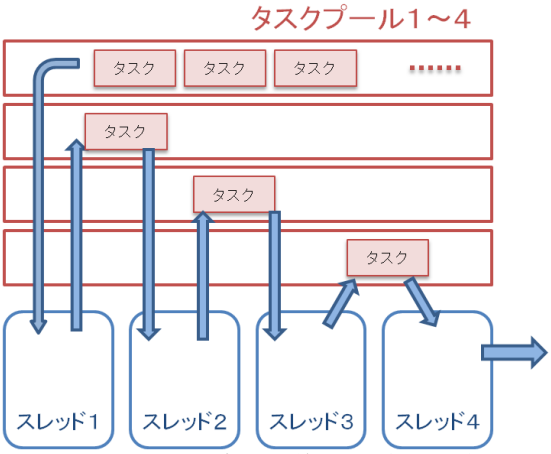

図 3　パイプライン並列化手法の処理例

C. ブロック分割並列化手法

ブロック分割では，計算する領域をブロックに分割して処理を行う．行，列両方について分割を行うため，例えば図 4 の D の処理を行うためには，領域 B，領域 C の処理を待つ必要があり，かつ領域 B，領域 C の必要な情報を統合してアラインメント処理の初期値を作らなければならない．よって，タスクプールに加え，もう一つ統合待ちのプールを作る必要がある．処理を終えたコアは自分の持っているタスクで統合待ちプール内のタスクを検索し，必要があればタスクを統合したのち，改めてタスクをタスクプールに詰め直さなければならない．

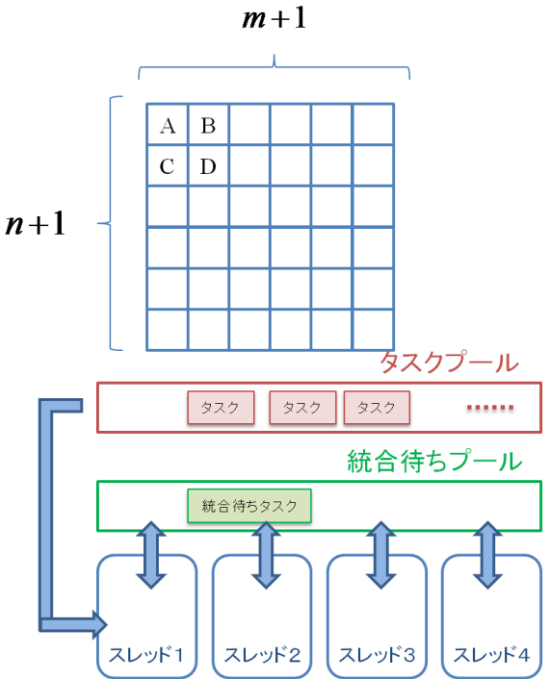

図 4　ブロック分割の担当領域

V.　評価実験

文字列長 100，1000，10000 の文字配列データをそれぞれ 10，100，1000 本用意し，ぞれぞれのデータについて提案手法(1)～(3)によるペアワイズアラインメントの実行時間を測定した．実験で使用した PC は，CPU：Intel （R） Core （TM）2 Quad CPU Q 6700 @ 2. 66GHz, Memory：2GB, HDD：250GB を搭載している．

実験の結果，データセット全般で，タスク分割並列化手法が最もスピードアップが良いことが分かった．これは，パイプライン並列化手法やブロック分割並列化手法では分割した領域間の同期待ちで，スレッド間でのデータのやり取りにオーバヘッドが生じるためだと考えられる．

図 5 に文字列長 10000 が 10 本のデータセットのスピードアップの結果を示す．この結果では，ブロック分割並列化手法が最もスピードアップが良い結果が得られた．また，図 6 は同データセットにおける各手法の総サイクルにおける待ち時間の比率を示したものである．これらの結果を見ると，配列長が長いデータ，すなわち消費するメモリ量が大きい場合には，タスク分割並列化では入力データが大規模になることでキャッシュのミス等が生じ，計算に遅延が生じてしまうのだと考えられる．よって，本実験に用いた 4 コア CPU よりも，さらに多くのコアを用いた実験環境では，パイプライン並列化手法やブロック分割並列化手法の方がより良い結果を得られる可能性がある．

V.　まとめ

本論文では，累進法の第 1 段階であるペアワイズアラインメント処理のマルチコア CPU 上での並列化手法を提案した．実験の結果から，文字列長が短いとき，タスク分割並列化手法，文字列長が長いとき，ブロック分割並列化手法が有利であることが分かった．これからの課題として，コア数を増やして評価を行うことと，累進法の他の部分の並列化が挙げられる．

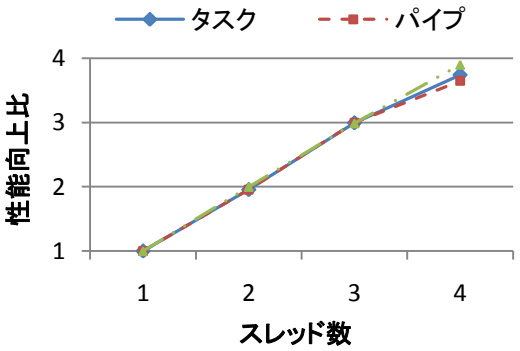

図 5：各手法のスピードアップのグラフ
（入力データ：10000_10 の場合）

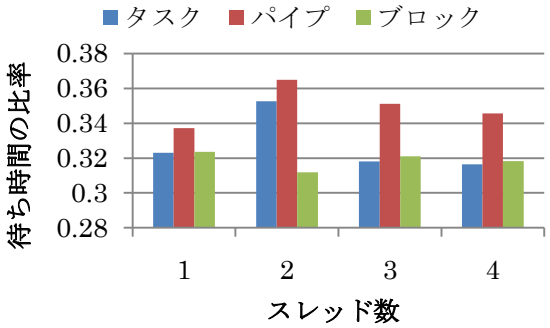

図 6：各手法の総サイクルにおける待ち時間の比率
（入力データ：10000_10 の場合）

謝辞

本研究の一部は，日本学術振興会・科学研究費補助金(基盤研究(C)，課題番号：20500137) ，文部科学省・科学研究費補助金（若手研究(B)，課題番号：23700124）の支援により行われた．

参考文献

[1] C. Notredame: Recent progresses in multiple sequence alignment: a survey, Pharmacogenomics, Vol. 3, pp. 131-144, 2002.

[2] Akiyama, Y., Onizuka, K., Noguchi, T, and Ando, M.: Parallel Protein Information Analysis (PAPIA) system running on a 64-node PC Cluster, Genome Informatics, Vol.8, pp.131-140, 1998.

[3] 十時 泰，秋山 泰，鬼塚 健太郎，野口 保，斉藤 稔，安藤 誠：アミノ酸配列のマルチプルアライメントにおける反復改善過程の並列化と A*アルゴリズムの適用，情報処理学会論文誌数理モデル化と応用，Vol.40, No.SIG9 (TOM2), pp.138-149, 1999. 伊野文彦，小谷裕基，萩原兼一：GPU グリッドによる高速な塩基配列アライメント，情報処理学会研究報告，2007-HPC-111, pp. 73-78, 2007.

[4] 宗川裕馬，伊野文彦，萩原兼一：統合開発環境 CUDA を用いた GPU での配列アライメントの高速化手法，情報処理学会研究報告，2008-HPC-114, pp. 13-18, 2008.

[5] Kuo-Bin Li: ClustalW-MPI: ClustalW Analysis Using Distributed and Parallel Computing, Bioinformatics, Vol.19, No.12, pp.1585-1586, 2003.

[6] Jaroslaw Zola, Xiao Yang, Adrian Rospondek, Srinivas Aluru: PARALLEL-TCOFFEE: A parallel multiple sequence aligner, Proceedings of the ISCA 20th International Conference on Parallel and Distributed Computing Systems, pp.248-25,2007.

問い合わせ先
〒731-3194
広島市安佐南区大塚東 3 丁目 4 番 1 号
広島市立大学　情報科学研究科　知能工学専攻
平原　海詞