GPGPU を用いた並列 DP マッチングによる大域アラインメントの高速化

山森　一人 [a)] ・ 河野　忠明 [b)] ・ 吉原　郁夫 [c)] ・ 相川　勝 [d)]

High-speed Global Alignment by Parallel DP Matching assisted with GPGPU

Kunihito YAMAMORI[a)], Tadaaki KAWANO[b)], Ikuo YOSHIHARA[c)] , Masaru AIKAWA[d)]

Abstract

Recent advance of genomic analysis technologies gives us a lot of DNA sequence data. Many researchers apply these data for post-genomic investigation. To find the optimum global alignment is investigated as one of the most important tasks of post-genomic investigation because it is the first step of there researches. Dynamic Programming (DP) matching is a major way to find the optimum alignment between sequences. However, DP matching requires $O(N^2)$ memory space to align the sequences consisting of *N* bases. To reduce memory space, divide and conquer algorithm is used with DP matching, but it leads longer computation time. In this paper, we propose a high-speed method to obtain the optimum global alignment by parallel DP maching assisted with GPGPU(General Purpose computing on Graphics Processing Unit). In this method, we utilize many stream processors like an arithmetic pipeline. Experimental results showed that our method achieved about five times faster than that by CPU.

Keywords: Dynamic Programming, global alignment, DNA sequence, Graphical Processing Unit (GPU), Compute Unified Device Architecture(CUDA)

1. はじめに

ヒトをはじめとする様々な生物種のゲノム解析プロジェクトにより、大量のゲノム情報を得ることができるようになった。現在では、得られたゲノム情報を利用して遺伝子の機能や役割を解明するポストゲノムの研究が盛んに行われている [1)]。ポストゲノム研究の一つとして、塩基配列の類似性に着目してグローバル（大域）アラインメントを求め、解析に利用する研究がある。

アラインメントを求めるには、動的計画法（Dynamic Programming：DP）を利用した DP マッチングが用いられる [1)]。しかし、DP マッチングを素直に実装すると塩基配列長の二乗のメモリ領域が必要となる。

Kleinberg ら [2)]は、分割統治法によりメモリ領域を削減する手法を提案しているが、代わりに計算時間がが長くなる問題がある。

一方、高速処理機構の一つとして近年 GPU （Graphics Processing Unit） [3)]が注目されている。GPU とは、コンピュータシステムにおけるグラフィックス処理に特化した半導体チップである。CPU を超える GPU の演算性能に着目し、GPU の並列アーキテクチャを画像処理以外の目的に応用する GPGPU（General Purpose comput- ing on Graphics Processing Units）の研究が盛んである。本論文では、分割統治法を用いた塩基配列の大域アラインメントにおいて、GPU 内の多数のプロセッサをパイプライン的に用いて並列処理を行うことで高速化することを目的とする。

[a)]情報システム工学科准教授

[b)]情報システム工学科

（現在、株式会社インターネットイニシアティブ）

[c)]情報システム工学科教授

[d)]宮崎大学工学部教育研究支援技術センター技術職員

2. 塩基配列のアラインメント

2.1 塩基配列とアラインメント

DNA（Deoxyribonucleic Acid ： デオキシリボ核酸）は、遺伝情報をコーディングする生体物質のことである。DNA を構成するアデニン（A）、シトシン（C）、チミン（T）、グアニン（G）の並び方を決定することをシークエンシングという。シークエンシングにより得られた塩基配列を利用して、遺伝子の機能特定や生物種ごとの類似部、非類似部の特定などのゲノム解析が可能になる。

2 本の塩基配列に対して、塩基の間にギャップを入れ、それぞれの塩基座における塩基同士を出来るだけ一致させることをペアワイズアラインメント（以下、アラインメントと呼ぶ）という。遺伝子に欠損や挿入などによって変異が起きていても、アラインメントを行うことにより塩基配列の類似部や非類似部を発見できる。つまり、アラインメントは塩基配列の意味を知る研究の第一歩である。

アラインメントを行って塩基配列の類似部や非類似部を見つけるには、2 本の塩基配列が最も類似するようなギャップの挿入パターンである、最適アラインメントを見つける必要がある。最適アラインメントは、表 1 に示すよう

に塩基の一致に“0”、不一致に“2”、ギャップの挿入に“3”のペナルティを与え、ペナルティを足してペナルティスコアを求め、これを最小化することで得ることができる。

表 1. ペナルティ計算テーブル.

	ペナルティ
一致	+0
不一致	+2
ギャップ挿入	+3

最適アラインメントを求める方法の 1 つに、動的計画法を利用した DP マッチングがある。図 1 は、2 本の塩基配列 GTATG と GATG の最適アラインメントを DP マッチングで求めた例である。2 本の塩基配列を縦と横に並べ、2 つの 2 次元配列$P[m][n]$、$B[m][n]$を用意する。ここで、m、nはそれぞれの塩基配列の長さである。また、2 本の塩基配列のある 1 塩基の位置をそれぞれ$i(1 \le i \le m)$、$j(1 \le j \le n)$で表す。$P[\ ][\ ]$はペナルティスコアを保存する配列であり、式(1)、式(2)で計算されたペナルティスコアを格納する。

なお、式(2)におけるgはギャップ挿入に行うペナルティである。$B[\ ][\ ]$はバックトレース情報を保持する配列で、その内容は式(3)により定められる。すべてのペナルティスコアを求めた後、$B[m][n]$から$B[1][1]$に矢印をたどっていくことで最適アラインメントが求められる。

$$P[1][1] = d(1,1). \tag{1}$$

$$P[i][j] = min\begin{cases} P[i-1][j-1] + d(i,j), & -① \\ P[i-1][j] + g, & -② \\ P[i][j-1] + g, & -③ \end{cases} \tag{2}$$

$$1 \le i \le m, 1 \le j \le n,$$

$$d(i,j) = \begin{cases} 0, & if\ Match, \\ 2, & if\ Mismatch. \end{cases} \tag{3}$$

	G	T	A	T	G
G	(N,0,N)	(3,N,N)	(6,N,N)	(9,N,N)	(12,N,N)
A	(N,N,3)	(6,2,6)	(5,3,9)	(6,8,12)	(9,11,15)
T	(N,N,6)	(9,3,5)	(6,4,6)	(7,3,9)	(6,10,18)
G	(N,N,9)	(12,8,6)	(9,5,7)	(8,6,6)	(9,3,9)

・括弧内は、それぞれ、(左,斜め上,上)方向からのペナルティスコアを表す
・○はその時点でのペナルティスコアの最小値
・N は No-Score(1 つ前がないことを表す)

・括弧内は、それぞれ、（左、斜め上、上）方向からのペナルティスコアを表す
・○はその時点でのペナルティスコアの最小値
・N は No-score（１つ前がないことを表す）

図 1.. DP マッチングの例.

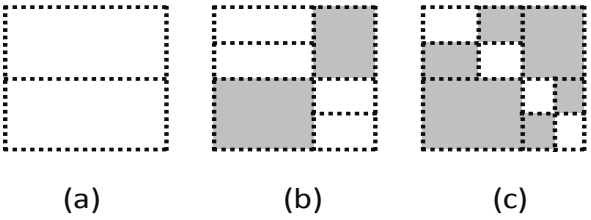


図 2. 分割統治法による DP マッチング対象領域の再帰的縮小.

$$B[i][j] = \begin{cases} "\nwarrow ", & if\ ①, \\ "\leftarrow ", & if\ ②, \\ "\uparrow ", & if\ ③. \end{cases} \tag{4}$$

$$1 \le i \le m, 1 \le j \le n,$$

2.2 分割統治法を用いたメモリ使用量削減

一般に、DP マッチングではバックトレース情報の保存のために$O(mn)$の領域が必要になる。塩基配列の長さが長くなれば、メモリ上に領域を確保することが難しくなる。そこで、Kleinberg ら [2]は DP マッチングに分割統治法を適用して、ペナルティスコア計算時のメモリ使用量を削減する手法を提案した。図 2 は DP マッチングに分割統治法を適用した例である。まず図 2 (a)のように計算領域を上下に 2 分割し、左上と右下からペナルティスコアの計算を行う。ペナルティスコアは直前の 1 行分だけを保存して、使用メモリ領域を削減する。分割線の上下のペナルティスコアを求めた後、求めた上下のペナルティスコアを足して最小値となった位置を分割点とし、図 2 (b)のように計算領域を左右に分割する。この分割点は最適アラインメントが通る点であり、図 2 (b)の右上、左下の領域ではペナルティスコア計算を省略できる。その後、図 2 (c)のようにペナルティスコアを計算する領域を再帰的に小さくしていき、オンメモリで DP マッチングが可能になるまで分割を行う。Kleinberg らの手法では、適切な分割点をあらかじめ求めることによって経路保存に使うメモリ使用量を削減する。しかし、必要最小限のペナルティスコアしか保存

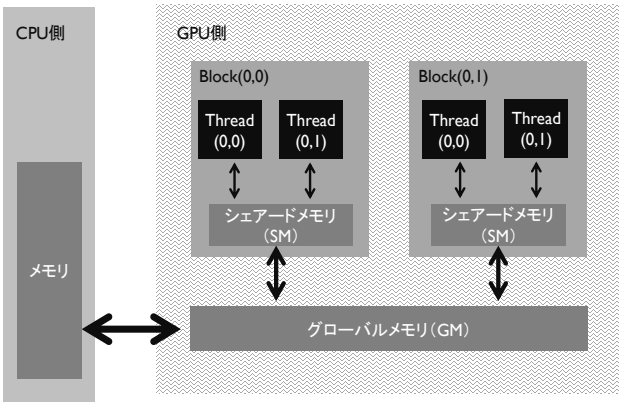
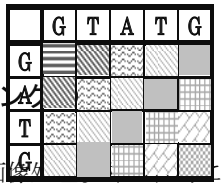


図 3. グラフィックボードのハードウェア構成.

しないため、領域の再分割後、同じ箇所のペナルティスコアを何度も求め直すので処理時間が長くなるという問題がある。



3. GPGPU を用いた DP マッチン

3.1 GPGPU と CUDA

GPGPU とは、GPU の演算資源を画像処理以外の汎用計算に応用する技術であり、CUDA は NVIDIA 社が提供する GPGPU 専用の開発環境である。本論文では、GPU として NVIDIA 社の GeforceGTX580 を使用する。GTX580 は 16 個のマルチコアプロセッサ（MP）を持ち、各 MP 内に 32 個の SIMD 型ストリームプロセッサ（SP）を持つ。従って、GTX580 内の SP は合計で 512 個になる。

CUDA はマルチスレッドプログラミング環境であり, 各 SP でスレッドを平行動作させる。CUDA では、GPU で実行されるコードをグリッドと呼ぶ。グリッドは複数のブロックと呼ばれる単位に分解され、ブロックは MP で並列処理される。各ブロックは複数のスレッドで構成され、MP 内のそれぞれの SP がスレッドを並列実行していく。スレッドとブロックはそれぞれ固有の番号を持っている。この番号を用いて、それぞれのスレッドに固有の処理を並列で実行させることができる。

図 3 に GPGPU を搭載したグラフィックボードの構成を示す。NVIDIA GeforceGTX580 には、ボード上に実装されているオフチップメモリと、GPU 内に搭載されているオンチップメモリがある。オフチップメモリには大容量、低速度のグローバルメモリ、オンチップメモリには小容量、高速度のシェアードメモリがある。グローバルメモリは全ての MP からアクセス可能である。シェアードメモリは各 MP がそれぞれ持っており、同一ブロック内の SP からのみアクセスが可能である。

3.2 GPGPU を用いた DP マッチングの高速化

分割統治法を用いた DP マッチングでは、分割点を再帰的に求めていく過程でペナルティスコアの再計算が必要

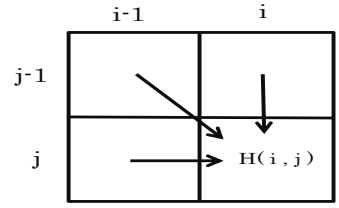
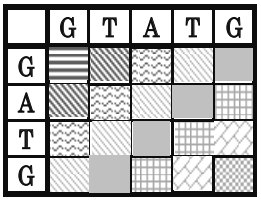


図 4. ペナルティスコア計算に必要な要素.



同じ模様の部分は
同時実行が可能

図 5. 同時計算可能な要素.

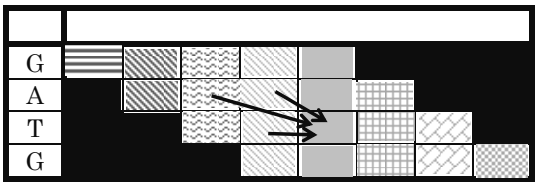


図 6. パイプライン的に表現したペナルティスコア計算.

であり、処理時間が長くなる。そこで本論文では、GPU の SP をパイプライン状に並列動作させることでペナルティスコア計算を高速化し、処理時間を短縮する手法を提案する。本論文では、DP マッチングのペナルティスコア計算部分を GPU に、その他の部分は CPU で計算させる。

図 4 に示したように、ペナルティスコア$H(i,j)$は隣接する$H(i-1,j)$、$H(i-1,j-1)$、$H(i,j-1)$があれば算出できる。従って、図 5 の同じ模様の部分のペナルティスコアは同時に計算でき、並列化可能である。パイプラインによるペナルティスコアの並列計算をわかりやすく説明するために、図 5 を図 6 に書き直す。図 6 において同一列の要素は同時に計算できるので、図 7 のように各行にスレッドを割り当てて並列に実行する。各スレッドは、行方向のn塩基のペナルティスコア計算を行う。図 7 の第k列のペナルティスコアを同時に求めるためには、第$(k-1)$列、第$(k-2)$列のペナルティスコアがあれば計算できる。このため、2 列分のペナルティスコアをそれぞれ 1 次元配列に保存していく。これにより、本来$O(\mathrm{mn})$使用するメモリ領域を$O(2\mathrm{m})$に削減できる。

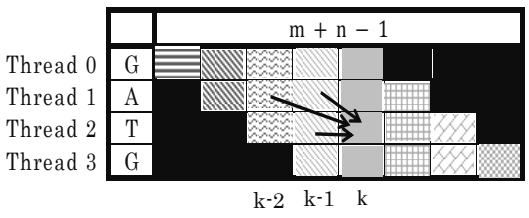


図 7. ペナルティスコア計算時のスレッド割り当て.

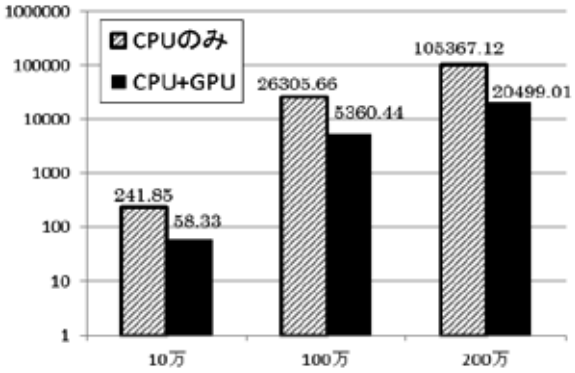


図 8. CPU と GPU での計算時間.

4. 提案手法の評価

4.1 実験・開発環境

実験で使用する塩基配列は、DDBJ（DNA Data Bank of Japan）で公開されているヒト 21 番染色体 GL000151 の塩基配列を利用する。ヒト 21 番染色体塩基配列の 200 万塩基目から 10 万、100 万、200 万塩基をそれぞれ切り出す。それぞれの塩基配列から、約 3%の確率で塩基が変化（置換、挿入、欠損）した擬似塩基配列を作成し、元の切り出した塩基とのアラインメントを求める。

実験・開発環境は、CPU に Intel 社の Core2DuoE6600、主記憶 4GB、GPU に NVIDIA 社の GeforceGTX580、グラフィックメモリ 1.5GB である。実験では、ブロックを 1 つ使用する。従って、32 スレッドでペナルティスコアを並列に実行する。

4.2 実験結果

図 8 に、CPU で逐次処理によりアラインメントを求めた場合と、提案手法で並列にアラインメントを求めた場合の処理時間を示す。図 8 では、縦軸を対数に取っている。また、表 2 に 200 万塩基のアラインメントを行った時の処理時間の内訳を示す。CPU による逐次処理で 200 万塩基のアラインメントを求めるのに要した時間は約 10.5 万秒（約 30 時間）である。同じ条件で GPU による並列処理を併用した場合の実行時間は約 2 万秒(約 6 時間)である。従って、提案手法は CPU による逐次処理に比べて約 5.25 倍の処理速度を達成した。また、提案手法で求めたアラインメント結果は、逐次処理のアラインメント結果と一致した。

表 2. 200 万塩基のアライメント時間の内訳.

	CPU [秒]	GPU [秒]
ペナルティスコア計算	104,944.1	20,041.26
分割点を探す	25.27	25.49
オンメモリ DP マッチング	383.52	384.37
バックトレース	12.22	11.77

5. おわりに

アラインメントを求める手法の一つである DP マッチングはアルゴリズムの性質上、塩基配列長の二乗のメモリ領域が必要となり、塩基配列が長くなればメモリ上に領域を確保することが困難になる。Kleinberg ら [2)]は分割統治法の考えを適用して計算に使うメモリ領域を削減する手法を提案しているが、メモリ使用量が削減できる代わりに、アラインメントを行うための計算量が多くなり、処理時間が増加してしまう。

本論文では GPU のストリームプロセッサをパイプラインとして使用し、ペナルティスコア計算を高速化する手法を提案した。GeforceGTX580 上の 32 個のストリームプロセッサを用いて提案手法を検証したところ、CPU で逐次実行した場合と比べて約 5.25 倍の高速化を達成した。また、提案手法で求めたアラインメントの塩基配列は、逐次プログラムのアラインメントの塩基配列と一致した。

今後の課題は、複数の MP を使用することでさらなる高速化を目指すことが挙げられる。

参考文献

1) 榊原 康文: バイオインフォマティクス概説 -比べることで生命は解明できるか?-, 情報処理, Vol.46, pp.230-238, 2005.

2) J. Kleinberg, E. Tardos: 分割統治法による線形の領域の系列アライメント, in アルゴリズムデザイン, 共立出版, pp.253-258, 2008.

3) 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 計算機アーキテクチャ研究会報告, Vol 19, pp.13-18, 2008.