CUDA による高速な Burrows-Wheeler 変換の検討

山中 翔[1]　伊野 文彦[2]　萩原 兼一[2]
[1] 大阪大学基礎工学部情報科学科，[2] 大阪大学大学院情報科学研究科

1　はじめに

Burrows-Wheeler（BW）変換 [3] とは，文字列に対する可逆変換の 1 つであり，文字列データに対して圧縮率を向上するための前処理として用いられている．この変換は圧縮率を高めるために，同じ文脈の直前には同じ文字が出現しやすいという言語の性質に着目する．また，BW 変換の特長として，変換後の文字列に対する検索が容易であることが挙げられる．生物情報学の分野では，この特長を用いて遺伝子配列の比較を高速化している．

近年，遺伝子配列のデータサイズは数百 MB を超えていて，遺伝子配列の数も増大している．したがって，BW 変換を高速化することは重要である．これまでにアルゴリズムやデータ構造の工夫により CPU 上で高速化を果たした研究は多い．しかし，アクセラレータによる高速化については定かでない．そこで，本研究はアクセラレータとして台頭しつつある Compute Unified Device Architecture（CUDA）互換の GPU を用い，BW 変換の高速化手法を検討している．

2　Burrows-Wheeler（BW）変換

BW 変換は，長さ n の文字列を入力とし，同じ長さの文字列を出力する．まず，文字列を巡回シフトし，長さ n の文字列を n 個ほど得る．次に，これらの文字列を辞書順にソートし，各文字列の最後の文字を順に並べた文字列を出力とする．BW 変換を高速化するために，既存手法は実行時間の 90%以上を占めるソートを高速化している．

文字列に対する辞書順ソートの代表的なものとして Ternary Quicksort（TQ）[1] がある．TQ は値に対する通常の Quicksort を文字列向けに拡張したものである．TQ は，再帰を用いて次の 2 ステップを処理する．まず，ピボットとなる文字を決定する．次に，そのピボットを各文字列の先頭文字と比較し，文字列の集合を 3 つの部分集合（セグメント）に分割する．部分集合の各々は，ピボットより大きいもの，小さいもの，および等しいものからなる．等しいものに対しては，再帰呼出のたびに比較対象の文字を先頭から順にずらしていく．

3　CUDA による BW 変換

GPU-Quicksort（GQ）[2] を拡張して実装する．GQ は通常の Quicksort を GPU 上に実装したものである．セグメントをブロック分割し，TB にピボットとブロック 1 つのデータの比較を割り当てる．TQ は入力を 3 つのセグメントに分割するので，GQ を実装方法はそのまま TQ に適用できない．

GQ は同じ深さの再帰呼出を 1 回のカーネル実行により処理する．TQ はセグメントを 3 分割するので，ピボットと等しい比較文字を持つ文字列の出力位置が定まらない（図 1）．この出力位置を決定するにはスレッドブロック（TB）間の同期が必要になる．これは出力位置の決定にはセグメント全体の比較結果が必要であり，また 1 つのセグメントに対して複数の TB が割り当てられていることが原因である．しかし，CUDA において TB を同期させる手段はない．

提案手法ではカーネルを 2 つに分割して出力位置を決定する．前半のカーネルは，ピボットと等しい比較文字を持つ文字列の出力開始位置を決定する．次に後半のカーネルは，ピボットと比較する文字との比較結果に従い文字列を配列に出力する．各比較結果による出力開始位置は配列の先頭，末尾，並びに前半のカーネルで決定した箇所となる．最後に，出力結果を比較結果に基づき 3 分割する．

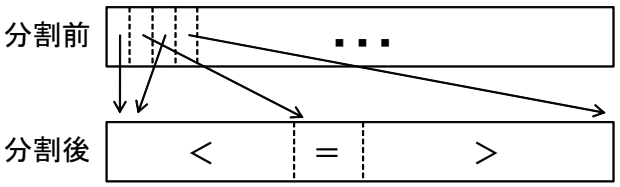

図 1: TQ における文字列集合の分割

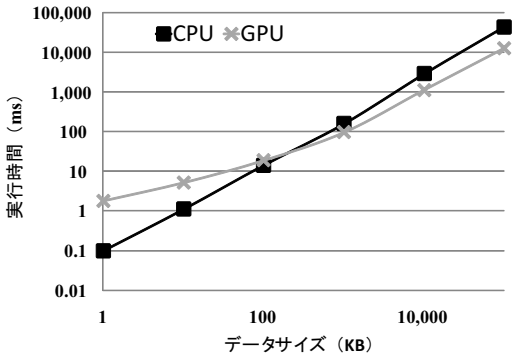

図 2: 実験結果

4　評価実験

提案手法を評価するために，実行時間に関して CPU 実装と比較した．実験に用いた PC は Core2 Duo E6850 および GeForce GTX 470 を持つ．入力のデータサイズを 1KB から 100MB まで変えながら，実行時間を測定した（図 2）．ここで，実行時間は主記憶およびビデオメモリ間のデータ転送に要する時間を含む．入力データが 100KB を超える場合，提案手法は CPU より高速である．速度向上率は 100MB のときに 3.5 倍に達している．

提案手法の問題点として，分割後の両カーネルでは同じ処理が存在している箇所があり実行時間が増えるという点がある．そこで，後半のカーネルに対する前半のカーネルの実行時間の比率 σ を調べた．比率 σ は入力データのサイズが増大するにつれて減少している．データサイズが 1KB のとき 50%，100MB のとき 22%である．

今後の課題としては，新たな出力方法による更なる速度向上が挙げられる．

参考文献

[1] J. L. Bentley and R. Sedgewick. Fast Algorithms for Sorting and Searching Strings. In *Proc. SODA'97*, pp. 360–369, Jan. 1997.
[2] D. Cederman and P. Tsigas. GPU-Quicksort: A Practical Quicksort Algorithm for Graphics Processors. *J. Exp. Algor.*, Vol. 14, Article No. 1.4, Feb. 2009.
[3] B. M and D. J. Wheeler. A Block-sorting Lossless Data Compression Algorithm. Technical Report SRC-RR-124, Digital Systems Research Center, May 1994.