様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　５ 年　 ６ 月 １６ 日現在

機関番号：１２６０２
研究種目：研究活動スタート支援
研究期間：2021～2022
課題番号：２１Ｋ２１２８１

研究課題名（和文）GPUを用いたオミックスデータの高速解析基盤開発

研究課題名（英文）Development of GPU framework for high-speed analysis of omics data

研究代表者
伊東　聡（Ito, Satoshi）

東京医科歯科大学・M&Dデータ科学センター・助教

研究者番号：３０５２５３５８

交付決定額（研究期間全体）：（直接経費）　　1,600,000円

研究成果の概要（和文）：全ゲノム解析パイプラインをGPU上で実装するために必要な機能の開発を行った。特に、MarkDuplicatesとI/Oを中心に開発を行った。具体的には、
1. 重複判定アルゴリズムを精査し、ペア情報を使用しない判定アルゴリズムを採用した。結果、ファイル内で離れた位置にあるペア情報の取得を排除し、データローカリティを確立、飛躍的な速度向上を実現した。
2. 長時間を要する～数十バイト単位での入出力を排除し、大きなバッファ単位での入出力ルーチンを作成した。100MB程度のバッファ単位I/Oを行い、格納するBGZFデータは一気に圧縮・解凍を行うようにした。その結果、GPU上での高速処理を達成した。

研究成果の学術的意義や社会的意義
スーパーコンピュータやクラウドでの計算が期待するほどの劇的な高速化を得にくい現状において、ParabricsとDRAGENシステムだけがGPU/FPGAによる圧倒的な高速解析を実現した成功例である。一方で、これらは改変不可能なブラックボックスであり、範囲外の解析ではその恩恵を享受できない。本ソフトウェアは、ブラックボックス部分を開発者自身で組み上げることを可能にする環境である。BAM作成時点で既存解析フローが適用できない研究が散見されており、そのような分野に対する高速ソフト開発が加速することが期待できる。

研究成果の概要（英文）：We developed the basic functionality required to implement a whole genome analysis pipeline on a GPU. In particular, We focused on MarkDuplicates and I/O, which are difficult to parallelize and require the most computation time.
1. We employed a judgment algorithm in MarkDuplicates that does not use pair read information. This improvement eliminate the acquisition of pair read information at distant locations within a file, establishes data locality in the judgment process, and dramatically improves processing speed.
2. We developed a new I/O functions that read/write data using a large size buffer to remove small data size read/write. The buffer size is 100-200MB and the BGZF data for read/write are compressed/decompressed all at ones.

研究分野：計算科学

キーワード：全ゲノム解析　GPU　HPC

１．研究開始当初の背景

次世代シーケンサーの登場とその著しい性能向上によって、人の全ゲノムシーケンスは現在、500 ドル 1 日程度まで安価かつ高速に実施できるようになった。このような背景から、これまで不可能であった希少疾患の研究やクリニカルシーケンス実現への期待が高まっている。クリニカルシーケンスでは日々訪れる大量の患者のデータ解析が求められる。希少疾患も同様で、検体数が多いほど統計的に優位な研究が期待できるため、要求される検体数は非常に多い。検体数に加え、1 検体あたりのデータ量も増加傾向にあることから、全体として解析に必要な計算機資源の量は爆発的に増えていることが問題である。

本分野において最も使用頻度の高い計算機システムは AWS を始めとするクラウドサービスである。スーパーコンピュータは前述の要求を満たす理論性能を持っている。しかしながら、ジョブスケジューラの問題（グリッドエンジン非搭載）や運用の面で研究者にとって必ずしも良い環境とは言えないマシンも多い。しかし、既存のデータ解析基盤であるクラウドシステムでは、データの転送時間および解析に要する時間が共に大幅に増えることが新たな問題である。

2010 年代後半になり、これらの問題を踏まえた新たな解析システムが登場した。一つが FPGA による高速解析パイプライン DRAGEN であり、もう一つが GPU を用いた高速解析パイプライン Parabrics である。どちらのシステムも GATK を高速化したものであり、CPU での解析に比べ 10～20 倍の高速解析が可能である。DRAGEN はクリニカルシーケンスに最適化した発展を遂げ、シーケンサーと一体化した DRAGEN プラットフォームを商用展開しており、米国 Broad Institute をはじめ、各国の大規模病院においてクリニカルシーケンスを支える基盤システムとしての地位を確立しつつある。Parabrics については、GPU という汎用的装置を利用することから FPGA に比べて利用可能ユーザが非常に多く、また、クラウドサービスを介して GPU を使用可能な環境を容易に利用できることから、研究分野での需要が異常に高くなっている。

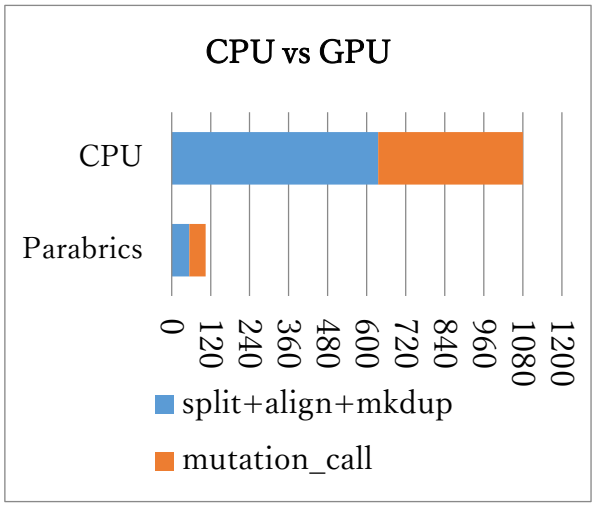

図１　GPU による高速パイプラインの解析例

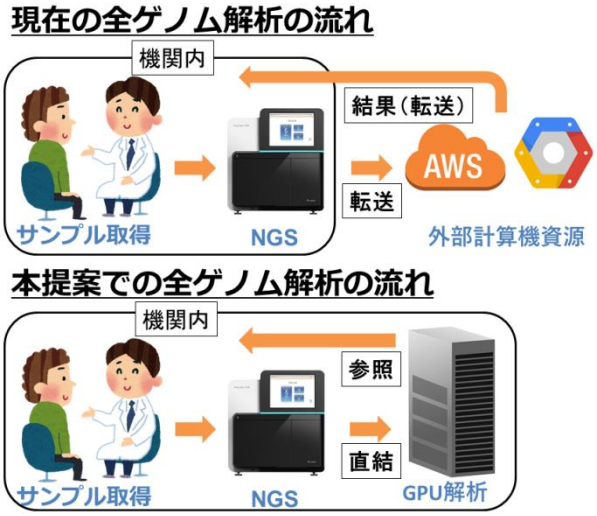

図２　本研究の目的

２．研究の目的

高速解析が可能な DRAGEN および Parabrics にも問題点はある。それは組み上げられたシステムの内容を改変できないことである。両者ともに商用のパッケージソフトウェア（DRAGEN の場合は FPGA 組み込み装置を含む）である。そのベースは GATK パイプラインであり、Germline 等の mutation コールを実施する場合には最適であるが、それ以外の解析の場合、本システムの一部結果のみを用いることしかできない。また、そもそも GATK のワークフローとは異なる解析が必要な場合には部分利用も望めないため、FPGA/GPU の恩恵を享受できない。

そこで本研究では、GPU を用いた解析パイプラインの開発を支援するフレームワークを開発する。GPU 用プログラムの開発には並列化、高速化に関する専門知識や CUDA 等の GPU 専用言語が必要であり、これらは医学／生物学者にとっては大きすぎる障害である。一方で、計算科学/計算機科学者がオミックス解析の実体を知らずにソフトウェアのみを見ての並列化や高速化を実施することは困難な構造をしているパイプラインが殆どのため、Smith-Warterman 等のアルゴリズム部分だけを GPU 化したライブラリが先行研究の大部分を占めている。本フレームワークを用いることにより、それらの専門知識を持たない医学／生物学者による GPU 用パイプラインの開発が可能となる。

３．研究の方法

フレームワークの目的を達成するためには、(a)ユーザがシーケンスデータ（SAM/BAM）を GPU 上で取得・改変できるような機能（関数等）や手段（テンプレート）を用意すること、(b)それらが GPU 上で十分な性能（高速処理）を発揮すること、の 2 点が必要である。(a)に関しては、既存の SAM/BAM ハンドリングツールの代表である SAMtools をベースに I/O 機能の調査と GPU 実装を、(b)に関しては MarkDuplicates の実装を通して CPU に対する高速性を、それぞれ実現する。

４．研究成果

まず初めに、I/O 機能の実装のため、SAM/BAM ファイルフォーマットとそのプログラム内でのデータ構造について調査した。SAM（Sequence Alignment/Map Format）は全ゲノム解析パイプラインで利用される主要なデータフォーマットである。シーケンサの出力データフォーマットのみ別（Fastq）であるが、マッピング後の大部分と処理後のデータを取り扱うツールのほぼすべてが、この SAM またはそのブロック GZIP 圧縮（BGZF）形式である BAM をデータフォーマットとして採用している。

ユーザがシーケンスデータを取り扱う場合、リード単位でのデータ構造を操作することになる。一方、BAM フォーマットでは圧縮前のデータを指定サイズごとに切り出し、ヘッダ情報を付加した上で GZIP 圧縮し、そのブロックを連結することで一つのファイルを構成している。

Field	Description	Type	Value
magic	BAM magic string	char[4]	BAM\1
l_text	Length of the header text, including any NUL padding	uint32_t	$< 2^{31}$
text	Plain header text in SAM; not necessarily NUL-terminated	char[l_text]	
n_ref	# reference sequences	uint32_t	$< 2^{31}$
List of reference information (n=n_ref)			
l_name	Length of the reference name plus 1 (including NUL)	uint32_t	*limited*
name	Reference sequence name; NUL-terminated	char[l_name]	
l_ref	Length of the reference sequence	uint32_t	$< 2^{31}$
List of alignments (until the end of the file)			
block_size	Total length of the alignment record, excluding this field	uint32_t	*limited*
refID	Reference sequence ID, $-1 \leq$ refID $<$ n_ref; -1 for a read without a mapping position	int32_t	[-1]
pos	0-based leftmost coordinate (= POS − 1)	int32_t	[-1]
l_read_name	Length of read_name below (= length(QNAME) + 1)	uint8_t	
mapq	Mapping quality (=MAPQ)	uint8_t	
bin	BAI index bin, see Section 4.2.1	uint16_t	
n_cigar_op	Number of operations in CIGAR, see Section 4.2.2	uint16_t	
flag	Bitwise flags (= FLAG)[29]	uint16_t	
l_seq	Length of SEQ	uint32_t	*limited*
next_refID	Ref-ID of the next segment ($-1 \leq$ next_refID $<$ n_ref)	int32_t	[-1]
next_pos	0-based leftmost pos of the next segment (= PNEXT − 1)	int32_t	[-1]
tlen	Template length (= TLEN)	int32_t	[0]
read_name	Read name, NUL-terminated (QNAME with trailing '\0')[30]	char[l_read_name]	
cigar	CIGAR: op_len<<4|op. 'MIDNSHP=X'→'012345678'	uint32_t[n_cigar_op]	
seq	4-bit encoded read: '=ACMGRSVTWYHKDBN'→ [0, 15]. See Section 4.2.3	uint8_t[(l_seq+1)/2]	
qual	Phred-scaled base qualities. See Section 4.2.3	char[l_seq]	
List of auxiliary data (until the end of the alignment block)			
tag	Two-character tag	char[2]	
val_type	Value type: AcCsSiIfZHB, see Section 4.2.4	char	
value	Tag value	(by val_type)	

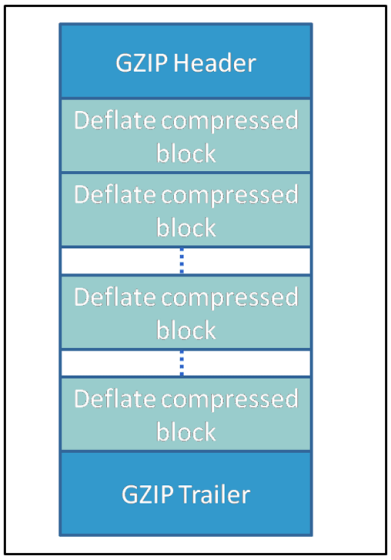

図 3　BAM フォーマット（左）および BGZF データ構造（右）

I/O 機能の実装では、上記の違いに加え GPU へのデータ転送内容を検討する必要がある。すなわち、GPU 転送前に BGZF ブロックの解凍を CPU 側で実施するかどうかである。比較的サイズの大きなデータを用いて SamTools の MarkDuplicates の実行時間を計測したところ、実に 85%もの割合を I/O 処理に要していることが判明した。そこで、SamTools における BAM ファイルの入出力コードを詳細に検討したところ、下記の事実が判明した。

1. BGZF ヘッダの入力/解凍または圧縮/出力から始まり、そのサイズは 4+32 バイトである
2. ヘッダの最後に格納された実データのサイズに合わせてバイナリデータを読み込むが、そのサイズは最大で 64KB 以下である
3. BAM フォーマットではバイトオーダおよび圧縮/非圧縮が不明（非圧縮とは BAM データを格納した構造体の内容をそのままダンプしたもの）なため、BGZF ブロックごとにそれらの判定およびバイトスワップ処理を実施している

上記 3 つの項目はいずれも I/O 処理の負荷を増大させる。このうち、１および２に関してはデータサイズが小さいことが問題であり、要因としては共通である。いずれのストレージシステムにおいても小サイズデータの大量 read/write は I/O 性能低下の要因となるが、スーパーコンピュータ等で一般的な Lustre などの分散ストレージシステムにおいては特に致命的である。3 とそれ以外のどちらに主要因があるかを調べるため、BAM ファイルの単純 I/O（読み込んだデータをそのまま出力する）を行うテストプログラムを作成し、128MB 程度のバッファサイズで一度に大量の BGZF ブロックを読み込み、ついで出力するテストを実施した。中規模（500MB）および大規模（120GB）での実験の結果、圧縮/解凍を伴う I/O ではバッファを用いた一括 I/O を行っても処理時間はほとんど高速化されないことが確認された。よって、I/O の高速化のためには BGZF ブロックの圧縮/解凍から GPU で実施する必要があり、CPU 側では BGZF データをバイ

ナリのまま読み込み、GPU へ転送、その後、GPU で逐次解凍しそのまま GPU 上で BAM 構造体（bam1_t*）型で保持する仕組みを開発した。本機能を用いて I/O テストを実行したところ、10～20 倍の高速化を実現した。

次に、MarkDuplicates の GPU による高速化を行った。I/O 機能の実現によって GPU 上でのリードデータのハンドリングが可能になった。これを用いて、全ゲノムデータ解析パイプラインで共通する最も処理時間の必要な本処理を GPU 高速化することにより、データ構造の利用方法の具体例を示すと共に、実用的なツールとしての有用性も担保することを目的としている。

SamTools 等で提供されている MarkDuplicates では、リードの重複判定に Estimate Library Complexity（ELC）を採用している。このアルゴリズムでは、ペアリードの双方に対し、5’-の塩基位置および最初の 5 塩基の内容が一致するものをマーク、マッピングクオリティ最大のもの 1 個をオリジナルとしている。ここで問題となるのは、判定にペアリードの塩基内容の一部を使用していることである。

ELC を用いる場合、リードの 5’-末端位置が同じ塩基である集合に対して、そのペアリードの情報の全てを要求することになる。同じ DNA フラグメントを起源に持つリードであれば、ペアの内容も等しくなるためその情報もファイル内で近しい位置に存在するが、異なるフラグメントであればペアの位置は不明である。特に、非常に長いフラグメントや構造変異等（ex:融合遺伝子）を含むフラグメントを起源としていた場合、ペアの情報は平均的な位置からはかけ離れた場所に存在することになり、その情報取得には多大な時間を要することになる。

そこで､本研究では SamTools0.1.19 までで採用されていた判定アルゴリズムを用いることにした。このアルゴリズムでは、両リードの塩基位置、インサーションサイズ、およびマッピングクオリティの 3 種の情報を元に判定を行っている。これらの情報は各リードに内包され、かつどちらのリードでも同一の情報を保持しているため、判定の一意性を担保しつつ、使用するデータのローカリティを確保することが可能である。小規模データでのテストを実施したところ、本アルゴリズムを実装した MarkDuplicates では最新版 SamTools に対して 1,100 倍以上の高速化を達成した。

最後に、本研究課題におけるアクティビティについて述べる。現在までに github 上でソースコードの公開を実施している。今後、ソートやパイルアップ等の他の機能についても開発を継続していく予定である。一方、当初の計画では学会発表等を行い、フィードバックを得る予定であったが、複数の技術的困難の解決に想定以上の時間を要したため、開発が予定よりも遅延し十分な対外活動を行うことが出来なかった。現在、論文の投稿準備中であり、本年秋ごろに開催予定の研究会にて発表を行う予定である。

```
While Reads exists{
    #pragma omp single nowait
    for each reads sam_read1( b[i] );
    #pragma omp single nowait
    {
         // Make blocks
        while Number_of_reads {
            (snip)
        }
        // Search independent reads
        for each block {
            (snip)
        }
        // Mark duplicates
        for each block {
            (snip)
        }
        for each reads sam_write1( b[i] );
    }
    #pragma omp barrier
}
```

図 4　MarkDuplicates のアルゴリズム部分

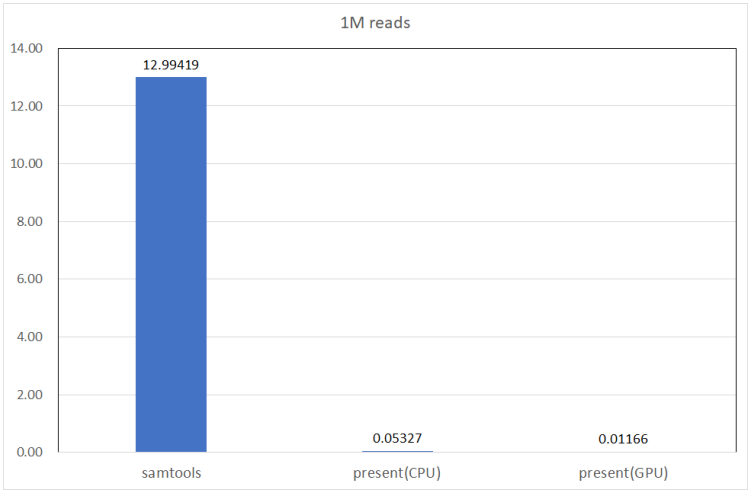

図 5　開発コードの計算時間評価

５．主な発表論文等

〔雑誌論文〕　計0件

〔学会発表〕　計0件

〔図書〕　計0件

〔産業財産権〕

〔その他〕

―

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関