GPU Accelerated API for Alignment of Genomics Sequencing Data

Nauman Ahmed, Hamid Mushtaq, Koen Bertels and Zaid Al-Ars
Computer Engineering Laboratory, Delft University of Technology, Delft, The Netherlands
{n.ahmed, h.mushtaq, k.l.m.bertels, z.al-ars}@tudelft.nl

***Abstract*—Sequence alignment is a core step in the processing of DNA and RNA sequencing data. In this paper, we present a high performance GPU accelerated set of APIs (GASAL) for pairwise sequence alignment of DNA and RNA sequences. The GASAL APIs provide accelerated kernels for local, global as well as semi-global alignment, allowing the computation of the alignment score, and optionally the start and end positions of the alignment. GASAL outperforms the fastest CPU-optimized SIMD implementations such as SSW and Parasail. It also outperforms NVBIO, NVIDIA's own CUDA library for sequence analysis of high-throughput sequencing data. GASAL uses the unique approach of also performing the sequence packing on GPU, which is over 200x faster than the NVBIO approach. Overall on Tesla K40c GASAL is 10-14x faster than 28 Intel Xeon cores and 3-4x faster than NVBIO with a query length of 100 bases. The APIs are included in an easy to use library to allow integration into various bioinformatics tools.**

***Keywords*-sequence alignment; GPU acceleration; software API; NGS**

I. Introduction

Next generation sequencing (NGS) technologies have greatly reduced the DNA and RNA sequencing cost [1]. This has enabled scientists to use sequencing for a variety of applications, ranging from medicine to nutrition. Sequencing determines the exact order of nucleotides in a DNA and RNA sample. Sequencing machines provides sequencing information in a computer readable data format, which is processed by computer programs to extract meaningful result.

NGS data analysis programs perform a series of computational steps on sequenced data. *Sequence alignment* is an important step in many core computer programs designed for processing sequencing data including DNA and RNA read mappers [2], [3], assemblers [4], [5], and sequencing error correction programs [6]. Sequence alignment is the process of editing two or more sequences using gaps and substitutions such that they closely match each other.

NGS machines generate hundreds of gigabytes of data that needs to be processed in a timely manner for the desired purpose. Therefore, a library of sequence alignment algorithms that is fast and highly optimized will help developers to design fast and accurate sequence analysis programs. Moreover, such a library can be easily integrated in various programs transparently. In sequence analysis programs, millions of independent pairwise alignments need to be performed. This is an ideal situation for parallel programming where multiple threads can be launched, each performing a single or a set of pairwise alignments. GPUs are capable of executing the same instructions on thousands of threads concurrently. In this paper, we present GASAL, a GPU accelerated library for pairwise sequence alignment of DNA and RNA sequences. GASAL contains functions for all three types of alignment algorithms i.e. local, global and semi-global. One-to-one as well as all-to-all and one-to-many pairwise alignments can be performed. Apart from NVBIO [7], NVIDIA's own CUDA library for sequence analysis of high-throughput sequencing data, there is no GPU accelerated library for pairwise alignment that exploits the massive parallelism offered by GPUs to deliver good performance. In addition to performing the alignment step on GPU, we used the novel approach of performing sequence *packing* on GPU as well, which shows large speedups as compared to performing the same operation on CPU. The library returns the score and the end position of the alignment with the option of calculating the start position of the alignment as well. GASAL is publicly available and can be easily downloaded from GitHub [8] which also contains the usage instructions. GASAL outperforms current state-of-the-art CPU-optimized implementations of sequence alignment algorithms as well as NVBIO.

This paper is organized as follows. Section II presents the motivation for GASAL. Section III describes the GASAL implementation. Section IV discusses the experimental results. Section V concludes the paper.

II. Motivation

GPU acceleration of sequence alignment has been the topic of many research papers. Most of the previous work, such as [9], [10] and [11], was focused on developing search engines for databases of protein sequences. Alignment of DNA and RNA sequences during the processing of high-throughput NGS data poses a different set of challenges than database searching as described below.

1) The sequences to be aligned in NGS processing are generated specifically for each experiment. In contrast, in database searching, the database of sequences is known in advance and can be preprocessed for higher performance.
2) In database search programs, one-to-all alignment is performed in which a query sequence is aligned against all the sequences in the database (may be regarded as target sequences), whereas the processing of NGS data requires one-to-one (e.g., in read aligners) one-to-all

(e.g., in multi sequence alignment) is performed. Due to this, common performance improvement techniques in database search programs, such as using *query profile*, are not feasible in NGS data alignment.

3) Sequence alignment is only a step in a bigger NGS data processing program rather than a standalone program, introducing new challenges in integrating the alignment step to the rest of the pipeline.

Despite these challenges, GPU acceleration of NGS data alignment can take advantage of the following two observations:

1) In database search programs, each base of the sequence has to represent 20 different possible amino acid values and hence require 8-bits to represent each base. DNA and RNA sequences are made up of only 5 bases (A, C, G, T/U and N for "unknown") and can be represented with a minimum of 3-bits. In this paper, we use an easy-to-implement 4-bit representation. This reduces the memory requirement by half as compared to database searching (which uses 8 bits) and also reduces the GPU global memory accesses.
2) The variation in sequence length is much smaller in NGS data alignment as compared to database sequences, making load balancing much easier.

Due to these differences, GPU accelerated database search cannot be used to accelerate the alignment step in NGS data processing programs. gpu-pairAlign [12] and GSWABE [13] present only all-to-all pairwise local alignment of sequences with the goal of computing the actual alignment (i.e., exact position of matches, mismatches and gaps in the alignment). All-to-all alignment is easier to accelerate on GPU. Since, only one query sequence is being aligned to all other sequences, the query sequence may reside in the GPU cache, substantially reducing global memory accesses. On the other hand, in one-to-one alignment each query sequence is different limiting the effectiveness of caching these sequences. In many NGS data processing applications, one-to-one pairwise alignment is required (e.g., DNA read mapping). In DNA read mapping, local alignment takes a substantial percentage of the total run time. For example, in the BWA-MEM DNA read aligner the local alignment takes about 30% of the total execution time with 150bp (or base pairs) simulated reads (see Section IV-B), while calculating only the score, start and end position. Global alignment is used to compute the actual alignment for generating the output CIGAR. But, this computation takes around 5% of the total execution time.

None of the previously published research efforts have developed any GPU accelerated sequence alignment library that can be easily integrated in other programs that require to perform pairwise alignments. NVBIO [7] is the only public library that contains GPU accelerated functions for the analysis of DNA sequences. Although this library contains a GPU accelerated function for sequence alignments, its performance is limited. Therefore, in this paper we present a GPU accelerated library for pairwise alignment of DNA and RNA sequences, GASAL. The library contains functions that enable fast alignment of sequences and can be easily integrated in computer programs developed for NGS data analysis. Functions for all three types of alignment algorithms (i.e., local, global and semi-global) are available in GASAL. One-to-one as well as all-to-all and one-to-many pairwise alignments can be performed.

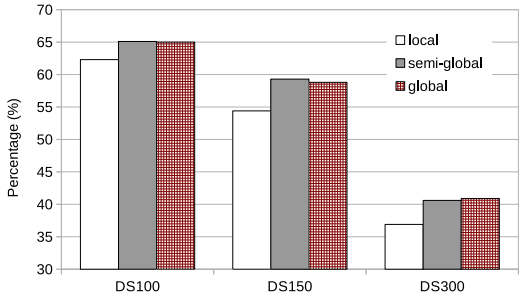

Fig. 1. NVBIO data packing time as percentage of total execution time

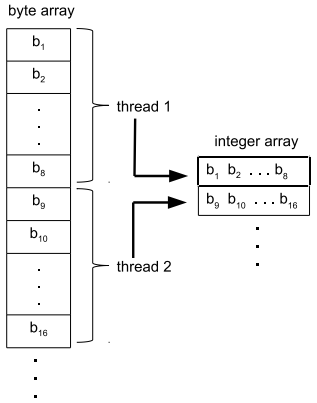

Fig. 2. Packing the sequences on GPU. $b_1, b_2, \ldots,$ are the bases

III. GASAL Implementation

In this paper, we have developed GASAL, a GPU accelerated library for pairwise sequence alignment. The library function returns the alignment scores, end position and optionally the start position of the alignment. The sequences are first *packed* into unsigned 32-bit integers, followed by performing the alignment. This section first describes the packing step, followed by the alignment step.

A. Data packing

Sequences are received in the form of batches in which each base is represented by one byte. However, DNA and RNA sequences are made up of only 5 nucleotide bases, A, C, G, T/U (T in case of DNA and U in RNA) and N (unknown base), 4 bits are enough to represent each symbol of a sequence. As the registers in the GPU are 32-bits wide and each thread can read from or write to at most 4 bytes to the memory, a sequence can be packed in arrays of 32-bit unsigned integers in which each base is represented by 4 bits. This will reduce the amount of memory communication inside the GPU kernel.

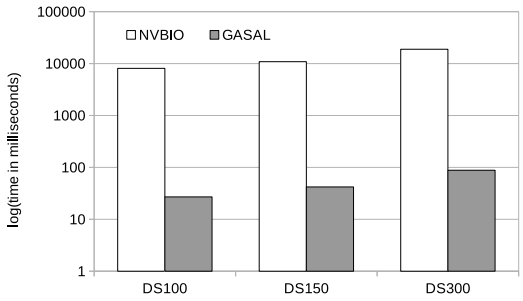

Fig. 3. Data packing time, GASAL vs NVBIO. Y-axis is log-scale

NVBIO also packs the DNA and RNA sequences on CPU using 4 bits for each base. As the the number of sequences in a batch is quite large and the length of each sequence can be hundreds of bases, packing the data on the CPU is very slow.

Figure 1 shows the percentage of data packing time in the total execution time for local, global and semi-global alignments in NVBIO. The input dataset and GPU platform are described in Section IV-B. Figure 1 shows that in NVBIO considerably large percentage of time in data packing. For 100, 150 and 300bp reads, packing consumes up to around 65%, 60% and 40% of execution time for all types of alignments. Hence, NVBIO spends a large portion of the total execution time in preparing the sequences to be aligned on GPU. Based on this observation, we further improve the performance by accelerating the data packing process on GPU. Unpacked batches of sequences are transferred to the GPU global memory before the data packing kernel is launched. An unpacked batch is basically a concatenation of sequences in which each base is represented by 8 bits. Figure 2 shows the process of data packing on GPU. Each GPU thread loads eight bases of a sequence at a time from global memory. Each base is converted from 8-bit to 4-bit representation and then packed into an unsigned 32-bit integer which is written back to global memory. Figure 3 shows the speed up achieved by packing the data on the GPU for the data sets. Figure 3 shows that our novel approach of packing the sequences on GPU in GASAL is very fast as compared to sequence packing performed by NVBIO on CPU. GASAL is at least 200x faster than NVBIO. Only few tens of milliseconds are required to pack the sequences in GASAL. Therefore, the data packing time is completely eliminated.

B. GASAL alignment kernels

After packing the sequences, the sequence alignment kernel is launched to perform pairwise alignment of the sequences. The algorithm for the local alignment kernel is shown in Algorithm 1. To find an alignment the cells of three identical H, E and F matrices, shown in Figure 4, are computed. But, computing a cell requires the values from the top, left, and top-left cells. Therefore, there is no need to store the whole matrix. Only a complete row of cells and a column of the tile, shown shaded in color in Figure 4 are stored, reducing the memory requirements. Each GPU thread executes the same kernel to perform a pairwise alignment of the assigned sequences. Since, the packed sequences already reside in the global memory, there is no need to perform any transfer of data from host to GPU to launch the kernel. The kernel uses the tiled approach (shown in Figure 4), to reduce the number of global memory accesses while computing the local alignment. In the tile-based scheme, all the cells in the tile are computed by accessing the global memory for retrieving the bases of S_1 and S_2 only once. This is possible due to packing of data. Figure 4 shows a 4 x 4 tile. Since, we have packed 8 bases in one unsigned 32-bit integer, our tile is 8 x 8. Each tile is computed column-wise. After a tile is computed the adjacent tile on the right is computed next. For example, the 4 x 4 tile shown in Figure 4 is the sixth in the order of computation.

Algorithm 1: The local alignment kernel to find the score and end positions of the alignment

Input: Two sequences S_1 and S_2_pack packed in sets of unsigned 32-bit integers S_1_pack and S_2_pack in which each base occupies 4 bits: S_1_pack, S_2_pack; length of S_1 $|S_1|$ and S_2 $|S_2|$; score for match $match\_sc$, mismatch penalty $miss\_sc$, gap open penalty $gapo$ and gap extension penalty $gape$

Output: maximum score max_sc and end position end_pos

Function GASAL_LOCAL(k) **begin**
  Initialize H and E arrays of size of $|S_1|$ containing zeros
  $max\_sc \leftarrow 0$
  $end\_pos \leftarrow \{0, 0\}$
  for $j \leftarrow 1$ **to** $|S_2|/8$ **do**
    Initialize h and f and p arrays of size of 9 containing zeros
    $s_2\_reg \leftarrow S_2\_pack[j]$
    $s_2\_idx \leftarrow (j-1) * 8$
    $s_1\_idx \leftarrow 0$
    for $i \leftarrow 1$ **to** $|S_1|/8$ **do**
      $s_1\_reg \leftarrow S_1\_pack[i]$
      $k \leftarrow 32$
      while $k >= 4$ **do**
        `// get 4 bits of s1_reg`
        $s_1\_base \leftarrow s_1\_reg[k, \ldots, k-3]$
        $h[1] \leftarrow H[s_1\_idx]$
        $p[1] \leftarrow H[s_1\_idx]$
        $e \leftarrow E[s_1\_idx]$
        $l \leftarrow 32$
        for $m \leftarrow 2$ **to** 9 **do**
          `// get 4 bits of s2_reg`
          $s_2\_base \leftarrow s_2\_reg[l, \ldots, l-3]$
          $s_2\_idx \leftarrow s_2\_idx + (m-1)$
          $f[m] \leftarrow \max\{h[m] - gapo, f[m] - gape\}$
          $e \leftarrow \max\{h[m-1] - gapo, e - gape\}$
          `// 0 is the padding base`
          if $s_1\_base = 0$ **or** $s_2\_base = 0$ **then**
            $h[m] \leftarrow p[m] + 0$
          else if $s_1\_base = s_2\_base$ **then**
            $h[m] \leftarrow p[m] - match\_sc$
          else
            $h[m] \leftarrow p[m] - miss\_sc$
          $h[m] \leftarrow \max\{h[m], f[m], e, 0\}$
          if $h[m] > max\_sc$ **then**
            $max\_sc \leftarrow h[m]$
            $end\_pos \leftarrow \{s_1\_idx, s_2\_idx\}$
          $p[m] \leftarrow h[m-1]$
          $m \leftarrow m + 1$
          $l \leftarrow l - 4$
        $H[s_1\_idx] \leftarrow h[m-1]$
        $E[s_1\_idx] \leftarrow \max\{e, 0\}$
        $s_1\_idx \leftarrow s_1\_idx + 1$
        $k \leftarrow k - 4$
  return $\{max\_sc, end\_pos\}$

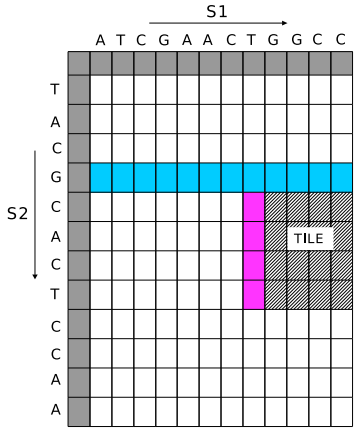

Fig. 4. Identical H, E and F matrix

The kernel returns the alignment score and the end position of the alignment on both sequences. Our library can also compute the start position using the same packed sequences. To find the start position we restart the computation, but now from the end cell, in the backward direction and exit where the score becomes equal to the previously found score. The coordinate of the cells at the exit point gives the starting positions of the alignment.

IV. Experimental results

A. Libraries compared with GASAL

We compared the performance GASAL against the fastest available GPU and CPU libraries. For local alignment GASAL is compared with NVBIO, as well as Striped Smith-Waterman (SSW) [14] which is the fastest implementation of local alignment on CPU and uses SSE2 SIMD instructions. Performance of global and semi-global alignment of GASAL is compared against NVBIO and Parasail [15]. Parasail allows the user to choose between SSE2 and AVX2 SIMD implementations. It also consists of different vectorization approaches namely *scan, striped, diagonal, blocked*. We have used the *scan* approach implemented with AVX2 instructions as it is the fastest for our dataset. For our dataset, SSW is faster than Parasail for local alignment. For SSW and Parasail we varied the number of threads from 1 to 28 to find the fastest running times for these libraries. Therefore the execution time of the SSW and Prasail reported in Section IV-D are the shortest possible execution times on a high-end machine.

NVBIO does not contain any function that only computes the start position of the alignment. The `alignment_traceback` function of NVBIO computes the actual alignment and also returns the start position, but is very slow. To compute the start position with NVBIO we make two copies of each sequence, one in original form and other reversed. The alignment of original sequences is used to compute the score and end position, while the reverse sequence are aligned to compute the start position. Similarly, Parasail also do not compute the start position. Therefore, the original sequences are aligned to obtain score and end position, then both sequences are reversed to calculate the start position. SSW has the inbuilt ability to compute the start position if asked to do so.

TABLE I. Characteristics of the input dataset

Dataset	Read Batch			Target Batch		
	avg. len.	max. len.	No. of seq.	avg. len.	max. len.	No. of seq.
DS100	100	100	10e6	162	177	10e6
DS150	150	150	10e6	260	277	10e6
DS300	300	300	10e6	538	571	10e6

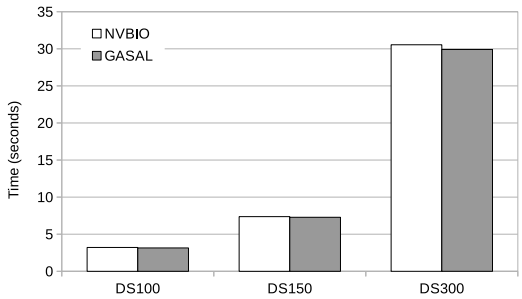

Fig. 5. Alignment kernel execution times for local alignment of NVBIO and GASAL without computing the start position

B. Input dataset and execution platforms

To evaluate the performance of GASAL we considered the case of DNA read mapping. Read mappers have to perform billions of *one-to-one* pairwise alignments between short segments of DNA and substrings of the reference genome. In the rest of the paper, we will refer to the substrings of the reference genome as *target* sequences. The length of the read sequence is fixed, while the length of the target sequence may vary. We simulated reads using Wgsim [16]. The reads are used to generate the corresponding target sequence using BWA-MEM. Three typical read lengths generated by Illumina high-throughput DNA sequencing machines are used: DS100, DS150 and DS300 representing 100, 150 and 300bp, respectively. Table I shows the number of sequences in the read and target batch and the corresponding maximum and average length of the sequences in each batch. Minimum target sequence length in each case is approximately equal to the length of the read.

SSW and Prasail libraries are executed on a high-end machine consisting of 2.4 GHz Intel Xeon processors with a total of 28 two-way hyper-threaded cores and 192 gigabytes of RAM. An NVIDIA Tesla K40c GPU is used to execute GASAL and NVBIO kernels.

C. GASAL alignment kernel performance

Table II shows a comparison of the alignment kernel execution times of NVBIO and GASAL (rounded off to the closest integer). The times listed in the table represent the times spent on only performing the pairwise alignments on the GPU and do not include data packing and data copying time. This table shows that the alignment kernel execution times of NVBIO and GASAL are the same for most alignment types.

TABLE II. Alignment kernel times (in seconds) for NVBIO and GASAL

	DS100		DS150		DS300	
	NVBIO	GASAL	NVBIO	GASAL	NVBIO	GASAL
Local	3	3	7	7	31	30
Local with start	6	6	15	15	61	57
Semi-global	3	3	6	6	25	24
Semi-global with start	5	5	12	12	50	45
Global	3	3	6	6	25	25

TABLE III. Total execution times (in seconds) for SSW, Parasail, NVBIO and GASAL. SSW is used for local alignment, while Parasail is used for global and semi-global alignment.

	DS100			DS150			DS300		
	SSW/Parasail	NVBIO	GASAL	SSW/Parasail	NVBIO	GASAL	SSW/Parasail	NVBIO	GASAL
Local	39	13	4	50	20	8	61	51	31
Local with start	43	30	7	50	45	15	67	110	58
Semi-global	42	12	3	48	18	7	74	45	25
Semi-global with start	50	28	6	62	42	12	87	98	47
Global	43	12	3	57	19	7	78	45	26

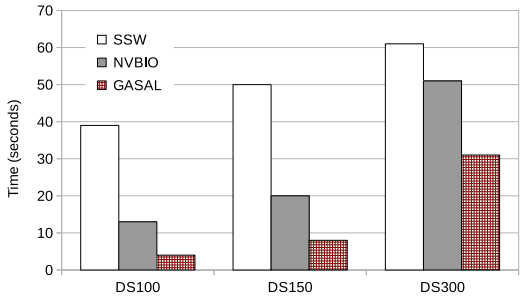

Fig. 6. Total execution times for local alignment of SSW, Parasail, NVBIO and GASAL without computing the start position

GASAL does show a performance advantage for alignments of longer reads (DS300). Figure 5 shows a visual comparison of the local alignment kernel execution times of NVBIO and GASAL (as listed in the first row of Table II). The figure shows that the alignment time increases as expected with increasing read length. It also shows a slight advantage of GASAL as compared to NVBIO. This indicates that the unique data packing approach used in GASAL does not cause any performance loss in alignment kernel execution as compared to NVBIO while completely eliminating the data packing time.

D. Overall speedup

Figure 6 and Table III show the end-to-end performance of SSW, Parasail, NVBIO and GASAL libraries for performing one-to-one pairwise alignments. For NVBIO and GASAL, the total execution time is the sum of data packing, data copying and alignment kernel times.

1) Without computing start-position: Figure 6 shows the performance of GASAL for performing the local alignment without computing the start position. For 100bp read length, GASAL performs the alignment in much less time as compared to SSW and NVBIO, giving an overall speedup of 10x as compared to SSW and over 3x as compared to NVBIO. With 150bp reads, GASAL is over 6x faster than SSW and 2.5x faster than NVBIO. For 300bp reads, the speedup over SSW and NVBIO is 2x and 1.6x, respectively. Table III lists execution times for the rest of alignment types without computing start position. For semi-global and global alignments the speedup obtained by GASAL is even higher. It is 14x, 7-8x and 3x over Prasail and 4x, 2.7x and 1.7x over NVBIO for read lengths of 100, 150 and 300bp, respectively. These results indicate that the speedup achieved by GASAL and NVBIO over CPU implementations (SSW and Parasail) decreases with longer reads. This is due to the fact that the CPU implementations use the striped heuristic that limits the computational complexity for longer reads, as compared to the GPU implementations. The results also show that the speedup achieved by GASAL compared to NVBIO decreases with longer reads. The reason for this decreasing speedup over NVBIO with increasing read lengths is the reduction in the data packing percentage (Figure 1) of the overall application as the alignment time continues to increase. GASAL speeds up the data packing while its alignment kernel performance remains similar to that of NVBIO.

2) With start-position computation: Table III also shows the overall execution times with start-position computation. With start-position computation, the speedup of GASAL over NVBIO is nearly the same as without computing start-position, since the alignment time simply doubles for both implementations when the start position is calculated. The executions times of NVBIO are slightly higher than double due to the time required to reverse the sequences. Computing local alignment with start position using GASAL is 6x, 3x and 1.1x faster than SSW for 100, 150 and 300bp reads, respectively. For semi-global alignment, the speedup is 8x, 5x and 1.85x for 100, 150 and 300bp, respectively. The results also show that NVBIO is slower than SSW/Parasail with start-position computation for 300bp read length, while GASAL is still faster than SSW/Parasail. The speedup of GASAL over SSW/Parasail is less with start-position computation as without start-position computation. This is due to the large caches in CPU as after computing the end-position the sequences are already available in cache and do not need to be fetched again from DRAM

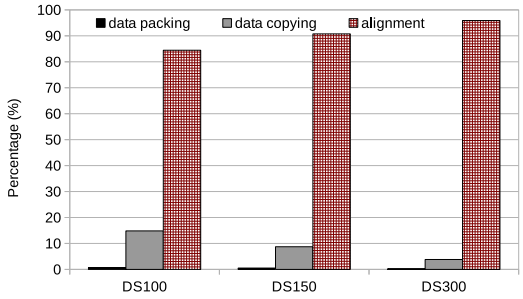

Fig. 7. Percentage of data packing, data copying and alignment kernel times in local alignment of GASAL

resulting in much less time to compute start-position.

Hence, GASAL is 2-4x faster than NVBIO and gives more speedup over state-of-the-art SIMD libraries making it a good choice for sequence alignment in high-throughput NGS data processing. Moreover, GASAL is much easier to use as compared to NVBIO. In NVBIO, the users have to manage data packing and data copying themselves by calling appropriate NVBIO functions. In GASAL, on the other hand, the users call the required alignment function and all the data packing and data copying operations are performed by the alignment functions. Figure 7 shows the percentage of data packing, data copying and alignment kernel time in the total execution time of GASAL while computing the score and end-position for local alignment. It shows that the alignment kernel takes more than 80% of the time for 100bp reads and this percentage goes above 90% for 150 and 300bp reads. Data packing time is negligible. The percentages for semi-global and global alignment are also nearly the same. Therefore, any further improvement in the alignment kernel execution time of GASAL will boost up its performance and directly increase the overall speedup over other implementations.

V. Conclusions

In this paper, we presented GASAL, a high performance and GPU accelerated library, for pairwise sequence alignment of DNA and RNA sequences. The GASAL library provides accelerated kernels for local, global as well as semi-global alignment, allowing the computation of the alignment score, and optionally the start and end positions of the alignment. One-to-one as well as all-to-all and one-to-many pairwise alignments can be performed. GASAL uses the novel approach of performing the sequence packing on GPU, which is over 300-200x faster than the NVBIO, NVIDIAs own GPU library for sequence analysis of high-throughput sequencing data approach. The paper compared GASAL's performance with the fastest CPU-optimized SIMD implementations such as SSW and Parasail and NVBIO. Experimental results performed on the Tesla K40c GPU show that GASAL is 10-14x faster than 28 Intel Xeon cores and 3-4x faster than NVBIO with a read length of 100bp without computing start position. For 150bp reads the speedup of GASAL over CPU implementations and NVBIO is 6-8x and 2.5-3x, respectively. With 300bp reads GASAL is 2-3x and 1.6-1.8x faster than CPU and NVBIO, respectively. With start-position computation the speedup of GASAL over NVBIO is the same as without start-position computation. The speedup of GASAL over CPU implementations with start-position computation is 6-8x, 3-5x and 1.1-1.85x for 100, 150 and 300bp reads, respectively. The library provides easy to use APIs to allow integration into various bioinformatics tools. GASAL is publicly available and can be easily downloaded from: https://github.com/nahmedraja/GASAL

Acknowledgment

This work is supported by the Faculty Development Program of the University of Engineering and Technology Lahore, Pakistan.

References

[1] K. Wetterstrand, "DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP)," Available at: www.genome.gov/sequencingcosts, 2016, Accessed [15 October, 2016].

[2] B. Langmead and S. Salzberg, "Fast Gapped-Read Alignment with Bowtie 2," *Nature Methods*, vol. 9, pp. 357–359, 2012.

[3] "NovoAlign," http://www.novocraft.com/products/novoalign/.

[4] X. Huang and S.-P. Yang, *Generating a Genome Assembly with PCAP*. John Wiley & Sons, Inc., 2002.

[5] M. de la Bastide and W. McCombie, *Assembling Genomic DNA Sequences with PHRAP*. John Wiley & Sons, Inc., 2002.

[6] L. Salmela and J. Schröder, "Correcting Errors in Short Reads by Multiple Alignments," *Bioinformatics*, vol. 27, no. 11, pp. 1455–1461, 2011.

[7] J. Pantaleoni and N. Subtil, "NVBIO," https://nvlabs.github.io/nvbio/, 2015.

[8] N. Ahmed, "GASAL," https://github.com/nahmedraja/GASAL, 2017.

[9] Y. Liu, W. Huang, J. Johnson, and S. Vaidya, *GPU Accelerated Smith-Waterman*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 188–195.

[10] Y. Liu, A. Wirawan, and B. Schmidt, "CUDASW++ 3.0: Accelerating Smith-Waterman Protein Database Search by Coupling CPU and GPU SIMD Instructions," *BMC Bioinformatics*, vol. 14, no. 1, p. 117, 2013.

[11] L. Hasan, M. Kentie, and Z. Al-Ars, "DOPA: GPU-based Protein Alignment Using Database and Memory Access Optimizations," *BMC Research Notes*, vol. 4, no. 1, p. 261, 2011.

[12] J. Blazewicz, W. Frohmberg, M. Kierzynka, E. Pesch, and P. Wojciechowski, "Protein Alignment Algorithms with an Efficient Backtracking Routine on Multiple GPUs," *BMC Bioinformatics*, vol. 12, no. 1, p. 181, 2011.

[13] Y. Liu and B. Schmidt, "GSWABE: Faster GPU-Accelerated Sequence Alignment with Optimal Alignment Retrieval for Short DNA Sequences," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 4, pp. 958–972, 2015.

[14] M. Farrar, "Striped Smith-Waterman Speeds Database Searches Six Times Over Other SIMD Implementations," *Bioinformatics*, vol. 23, no. 2, pp. 156–161, 2007.

[15] J. Daily, "Parasail: SIMD C Library for Global, Semi-Global, and Local Pairwise Sequence Alignments," *BMC Bioinformatics*, vol. 17, no. 1, pp. 1–11, 2016.

[16] "Wgsim," https://github.com/lh3/wgsim.