BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 61, No. 4, 2013
DOI: 10.2478/bpasts-2013-0106

G-DNA – a highly efficient multi-GPU/MPI tool for aligning nucleotide reads

W. FROHMBERG[1], M. KIERZYNKA[1,2*], J. BLAZEWICZ[1,3], P. GAWRON[1,4], and P. WOJCIECHOWSKI[1]

[1] Institute of Computing Science, Poznań University of Technology, Poznań, Poland
[2] Poznań Supercomputing and Networking Center, Poznań, Poland
[3] Institute of Bioorganic Chemistry, Polish Academy of Sciences, Poznań, Poland
[4] Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Luxembourg

Abstract. DNA/RNA sequencing has recently become a primary way researchers generate biological data for further analysis. Assembling algorithms are an integral part of this process. However, some of them require pairwise alignment to be applied to a great deal of reads. Although several efficient alignment tools have been released over the past few years, including those taking advantage of GPUs (Graphics Processing Units), none of them directly targets high-throughput sequencing data. As a result, a need arose to create software that could handle such data as effectively as possible. *G-DNA* (GPU-based DNA aligner) is the first highly parallel solution that has been optimized to process nucleotide reads (DNA/RNA) from modern sequencing machines. Results show that the software reaches up to 89 GCUPS (Giga Cell Updates Per Second) on a single GPU and as a result it is the fastest tool in its class. Moreover, it scales up well on multiple GPUs systems, including MPI-based computational clusters, where its performance is counted in TCUPS (Tera CUPS).

Key words: DNA assembly preprocessing, sequence alignment, GPU computing.

1. Introduction

Software tools that are currently used for *de-novo* assembly may be, in general, divided into the ones that use the Eulerian path approach and those based on the classical overlap-layout-consensus. Although the former is nowadays more popular, mainly due to its speed and ability to handle large data sets, the latter concept still remains in the point of interest. In fact, some of its unique properties are the cornerstone of a few established solutions, e.g. [1–4]. However, it is the high time consumption that often limits their applications, especially in the context of the constantly increasing number of biological sequences. Therefore, it is of great importance to address this issue by identifying steps that potentially could be accelerated. The fundamental phase of the methods based on the overlap-layout-consensus is, as the name suggests, computing the so called overlaps, i.e. shifts between given pairs of sequences and corresponding alignment scores. The most accurate results may be obtained here using an exact method that is the semi-global version of the Needleman-Wunsch algorithm [5] (NW), although it is often replaced by some faster heuristics.

In this paper, we present a new software tool that has been designed to overcome the outlined problem. It uses NW to compute the overlaps between each pair of sequences from a given list. In contrast to the already available implementations, e.g. [6–9], our solution is highly optimized for reads from modern sequencers (Roche/454, Illumina/Solexa and AB/SOLiD) and, unlike cited works, directly targets the described problem. The software may prove to be useful especially in processing data from Roche/454, where the overlap-layout-consensus method is the most commonly employed.

2. Related works

The growing number of sequences in biological databases, including also protein databases [10–12], was the reason for many scientists to develop efficient software computing pairwise alignment. Some of the implementations, especially those taking advantage of modern GPUs, have become specific gems in the world of high performance computing [13]. The choice of computational architecture of this kind was not incidental though, as its great potential has already been demonstrated in many other works related to scientific simulations [14, 15], databases [16, 17] or optimization problems [18]. Historically, the first implementation of the Smith-Watermann algorithm using CUDA-capable GPUs was developed by Manavski S. et al. [19]. At the time SW-CUDA was able to achieve up to 3.5 GCUPS on two NVIDIA GeForce 8800GTX graphics cards. Another milestone was reached by Ligowski L. et al. [6], who employed the shared memory to attain up to 14.5 GCUPS on two GPUs of the GeForce 9800 GX2. Liu Y. et al. [20], in turn, developed the CUDASW++ algorithm achieving up to 16GCUPS on a dual-GPU GeForce GTX 295. This approach was further explored by its authors resulting in an optimized SIMT and partitioned vectorized algorithm CUDASW++ 2.0 [7] with a very good performance

*e-mail: michal.kierzynka@cs.put.poznan.pl

of up to 17 GCUPS on a GeForce GTX 280 and 30 GCUPS on a dual-GPU GeForce GTX 295. On the CPU side, one solution that deserves special attention is SWIPE developed by Rognes [9]. According to the author, the software is able to run at 106GCUPS on two six-core high-end CPUs, which is a significant increase compared to the previously known CPU, GPU, Cell/BE and even FPGA based solutions. However, all of the aforementioned software tools were designed to perform well in the database scan scenario only where one sequence is aligned with all the others. This means that their performance drops dramatically if applied to the scheme described in the introductory section, in which only selected pairs of sequences are aligned. On the other hand, there are also some interesting GPU tools on the market that perform pairwise sequence alignment with the backtracking step [21, 22], sequence mapping [23] or even multiple sequence alignment [24]. Yet, none of them has been suitable to be used effectively in the next-generation *de-novo* assembly problem. Therefore, to fill this gap we propose *G-DNA*, which is a software tool well-tailored for the outlined problem.

3. Methods and implementation

Our implementation of NW takes a set of nucleotide reads and a list of previously selected pairs to align as an input. For each given pair of sequences it computes the overlap parameters. To achieve satisfactory efficiency of the solution, i.a. high ratio of computations to data transfer, a number of coarse and fine grained optimizations has been introduced. One of them is specially designed bitwise data compression that enables the algorithm to use packed sequences as they are, i.e. without expensive decompression phase. In our implementation each residue uses as few bits as it is required by the cardinality of a given input alphabet. To illustrate, in the case of four residues (A, C, G, T/U), 16 of them are stored in one 32-bit word, whereas 10 nucleotides can be packed within the same space when extra symbols are used, e.g. N indicating uncertain read.

As the NW algorithm is based on the dynamic programming (DP), an efficient method of processing 2-dimensional matrices on GPU needed to be developed. We adapted the idea described in [21], modified, however, in the way that benefits most from the compression. The previous work has been mainly optimized to align each possible pair of amino acid sequences from a given input set and to perform the backtracking step. Here, we follow another memory access pattern as only selected pairs of nucleotide sequences are aligned and no backtracking is needed. Also the scoring scheme is somewhat different, since the user defines only one gap penalty value and a desired substitution matrix. No affine gap penalties are computed, because modern sequencers are more likely to misread individual residues rather than longer segments.

Moreover, a number of low-level improvements have been introduced to the software. For example, to minimize the number of expensive conditional instructions, we placed special emphasis to unroll each possible loop in the GPU kernel code. This, in a nutshell, makes the code specific to a given sequence length. To overcome this problem and handle reads of various lengths, we prepared a number of template-based kernels. As a result, we ended up with 28 GPU functions that cover all possible cases. Other optimizations, despite being interesting, are out of the scope of this paper.

4. Tests and conclusions

To evaluate the performance of the proposed implementation we used real biological sequences from a variety of modern sequencers. In the case of SOLiD, 3.4M reads of the same length (46bp) representing genome data from *Streptococcus suis* bacterium were used. Likewise, Illumina GA IIx produces reads of equal length, here 34M sequences of 120bp from *Clonorchis sinensis* became the second input set. In contrast, Roche 454 pyrosequencing genome data from *E. coli* were of variable length with 235bp on average, 436k reads in total. The purpose of the last data set, coming from Roche 454 GS FLX Titanium, was to illustrate the peak performance of the implementation, hence only very long reads were used (avg. 1020bp). Tests were performed on the following hardware: CPU: Intel Core 2 Quad Q8200, 2.33GHz, GPU: 2 $\times$ NVIDIA GeForce GTX 580 with 1.5GB of RAM, main RAM: 8GB.

Figure 1 shows that the efficiency of the software grows with increasing sequence length, and thus *G-DNA* performs best for Roche/454 data. Yet, even for relatively short sequences, e.g. from Illumina, the algorithm achieves very good performance. Also the multiple GPUs support works very well thanks to our load balancer, resulting in nearly 2-fold speedup when 2 GPUs are in use. To illustrate, the results of 112, 160, 165 and 177 GCUPS translate into 53M, 11M, 3M and 170K sequence pairs aligned per second for data from SOLiD, Illumina, 454 and 454 Titanium, respectively. However, one should bear in mind that the latter measure might be confusing as it depends heavily on the length of sequences favoring the short ones. Hence, in further comparisons we will use more intuitive measure of GCUPS.

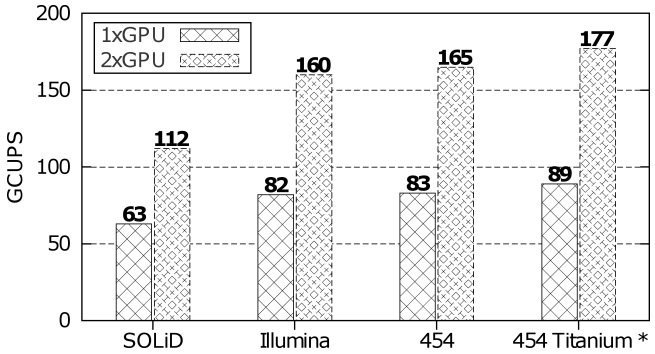

Fig. 1. The performance of the algorithm (in GCUPS) for input sets coming from different sequencing machines. * refers to long reads only

To our best knowledge *G-DNA* is the fastest implementation of NW for nucleotide reads. To compare, state-of-the-art CUDASW++2.0 [7] reaches up to 48 and 83 GCUPS on the

same hardware using one and two GPUs, respectively. Another well-established solution is Farrar's implementation [8], which achieves 20 GCUPS on 8 CPU cores and 16 GCUPS on the Cell B.E. processor. Yet, both of them have recently been outperformed by SWIPE [9] attaining up to 106 GCUPS on 12 SSSE3-compliant CPU cores. Although these methods do not address exactly the same problem as we do (Smith-Watermann instead of NW, database scan rather than alignment of selected pairs), performance-wise they seem to be the right applications to compare with. To give the reader a brief idea, simple, not optimized CPU versions of NW, e.g. needle from the Emboss package [25], implementation of Siriwardena et al. [26] or our own, achieve some 0.02 to 0.03 GCUPS. Therefore, it might be essential to choose a high performance software as the differences in terms of speed are of the order of several thousand.

Additionally, Fig. 2 presents good scalability of our algorithm on a computational cluster. The maximum performance of 1014 GCUPS was achieved in the weak scaling test in which the problem size grows proportionally to the number of GPUs (here 110M alignments for 32 GPUs). 929 GCUPS was the highest result in the case of the strong scaling test, where the problem size was fixed at 55M alignments. The measurements include both computation and inter-node communication times.

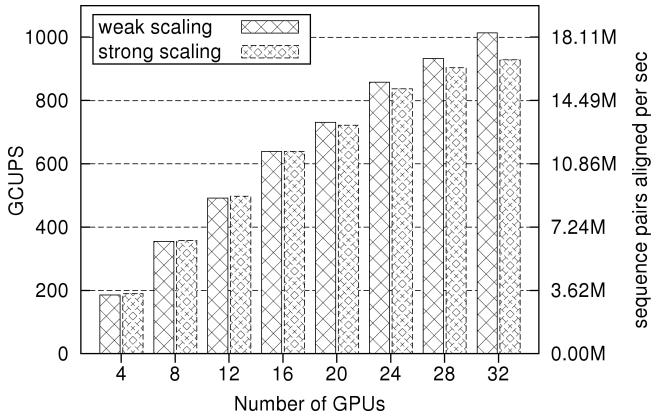

Fig. 2. The efficiency of the MPI version of the algorithm depending on the number of GPUs used (Tesla M2050). Reads were from the 454 sequencer

To conclude, *G-DNA* is an extremely efficient software tool that performs pairwise sequence alignment for selected pairs from a given set of nucleotide reads. It is therefore perfectly suited to be used e.g. in the DNA assembly problem. The software is freely available and may be run on commodity hardware which makes it a perfect tool for the everyday scientific use.

Availability and requirements

- Project home page: http://gpualign.cs.put.poznan.pl,
- Requirements: Linux OS, CUDA >= 4.0, CUDA compliant GPU, make, g++, bison, flex, MPI (for multi-node setup),
- License: GNU GPLv3.

Acknowledgements. This research project was supported by the grants DEC-2011/01/B/ST6/07021 and 2012/05/B/ST6/03026 from the National Science Centre (Poland) and also by the PL-Grid Infrastructure.

REFERENCES

[1] J. Blazewicz, M. Bryja, M. Figlerowicz, P. Gawron, M. Kasprzak, E. Kirton, D. Platt, J. Przybytek, A. Swiercz, and L. Szajkowski, "Whole genome assembly from 454 sequencing output via modified DNA graph concept", *Comput. Biol. Chem.* **33** (3), 224–230 (2009).

[2] J. Blazewicz, W. Frohmberg, P. Gawron, M. Kasprzak, M. Kierzynka, A. Swiercz, and P. Wojciechowski, "DNA sequence assembly involving an acyclic graph model", *FCDS* 38, 25–34, doi: 10.2478/v10209-011-0019-4 (2013).

[3] Forge Genome Assembler http://combiol.org/forge/ (2012).

[4] J. Blazewicz, P. Formanowicz, F. Guinand, and M. Kasprzak, "A heuristic managing errors for DNA sequencing", *Bioinformatics* 18, 652–660 (2002).

[5] S.B. Needleman and C.D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", *J. Mol. Biol.* 48 (3), 443–53 (1970).

[6] L. Ligowski and W. Rudnicki, "An efficient implementation of Smith Waterman algorithm on GPU using CUDA, for massively parallel scanning of sequence databases", *IPDPS 2009, IEEE Computer Society*, doi:10.1109/IPDPS.2009.5160931 (2009).

[7] Y. Liu, D.L. Maskell, and B. Schmidt, "CUDASW++2.0: enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions", *BMC Research Notes* 3, 93 (2010).

[8] M.S. Farrar, "Optimizing Smith-Waterman for the cell broadband engine", *Bioinformatics* 23, 156–161 (2008).

[9] T. Rognes, "Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation", *BMC Bioinformatics* 12, 221 (2011).

[10] J. Blazewicz, P.L. Hammer, and P. Lukasiak, "Predicting secondary structures of proteins. Recognizing properties of amino acids with the logical analysis of data algorithm", *IEEE Eng. Med. Biol. Mag.* 24 (3), 88–94 (2005).

[11] P. Lukasiak, J. Blazewicz, and M. Milostan, "Some operations research methods for analyzing protein sequences and structures", *Annals OR* 175 (1), 9–35 (2010).

[12] P. Lukasiak, M. Antczak, T. Ratajczak, J.M. Bujnicki, M. Szachniuk, R.W. Adamiak, M. Popenda, and J. Blazewicz, "RNAlyzer novel approach for quality analysis of RNA structural models", *Nucleic Acids Res* 41 (12), 5978–5990, doi:10.1093/nar/gkt318 (2013).

[13] W. Hwu, *GPU Computing Gems Emerald Edition*, Morgan Kaufman, Berlin, 2011.

[14] M. Blazewicz, S.R. Brandt, M. Kierzynka, K. Kurowski, B. Ludwiczak, J. Tao, and J. Weglarz, "CaKernel – a parallel application programming framework for heterogenous computing architectures", *Scientific Programming* 19 (4), 185–197 (2011).

[15] M. Blazewicz, I. Hinder, D.M. Koppelman, S.R. Brandt, M. Ciznicki, M. Kierzynka, F. Loffler, E. Schnetter, and J. Tao, 'From physics model to results: An optimizing framework for cross-architecture code generation", *Scientific Programming* 21 (1–2), 1–16 (2013).

[16] W. Andrzejewski, A. Gramacki, and J. Gramacki, "Graphics processing units in acceleration of bandwidth selection for Kernel density estimation", *AMCS* 23 (4) (2013).

[17] W. Andrzejewski and R. Wrembel, "GPU-PLWAH: GPU-based implementation of the PLWAH algorithm for compressing bitmaps", *Control and Cybernetics* 40 (3), 627–650 (2011).

[18] R. Nowotniak and J. Kucharski, "GPU-based tuning of quantum-inspired genetic algorithm for a combinatorial optimization problem", *Bull. Pol. Ac.: Tech.* 60 (2), 323–330, doi: 10.2478/v10175-012-0043-4 (2012).

[19] S. Manavski and G. Valle, "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment", *BMC Bioinformatics* 9 (2), S10 (2008).

[20] Y. Liu, D.L. Maskell, and B. Schmidt, "CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units", *BMC Research Notes* 2 (2009).

[21] J. Blazewicz, W. Frohmberg, M. Kierzynka, E. Pesch, and P. Wojciechowski, "Protein alignment algorithms with an efficient backtracking routine on multiple GPUs", *BMC Bioinformatics* 12, 181 (2011).

[22] J. Blazewicz, W. Frohmberg, M. Kierzynka, and P. Wojciechowski, "G-PAS 2.0 – an improved version of protein alignment tool with an efficient backtracking routine on multiple GPUs", *Bull. Pol. Ac.: Tech.* 60 (3), 491–494, doi: 10.2478/v10175-012-0062-1 (2012).

[23] C. Liu, T. Wong, E. Wu, R. Luo, S. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, R. Li, and T. Lam, "SOAP3: ultra-fast GPU-based parallel alignment tool for short reads", *Bioinformatics* 28 (6), 878–879, doi: 10.1093/bioinformatics/bts061 (2012).

[24] J. Blazewicz, W. Frohmberg, M. Kierzynka, and P. Wojciechowski, "G-MSA – A GPU-based, fast and accurate algorithm for multiple sequence alignment", *JPDC*, doi:10.1016/j.jpdc.2012.04.004 (2012).

[25] *EMBOSS Package*, http://emboss.sourceforge.net/ (2012).

[26] T.R.P. Siriwardena and D.N. Ranasinghe, "Accelerating global sequence alignment using CUDA compatible multi-core GPU", *ICIAFs 2010* 1, CD-ROM (2010).