Graphics card processing: accelerating profile-profile alignment

Muhammad Kashif Hanif*, Karl-Heinz Zimmermann

Institute of Computer Technology,
Hamburg University of Technology,
Schwarzenbergstrasse 95e, 21073 Hamburg, Germany

Received 27 April 2012; accepted 29 October 2012

Abstract: Alignment is the fundamental operation in molecular biology for comparing biomolecular sequences. The most widely used method for aligning groups of alignments is based on the alignment of the profiles corresponding to the groups. We show that profile-profile alignment can be significantly speeded up by general purpose computing on a modern commodity graphics card. Wavefront and matrix-matrix product approaches for implementing profile-profile alignment onto graphics processor are analyzed. The average speed-up obtained is one order of magnitude even when overheads are considered. Thus the computational power of graphics cards can be exploited to develop improved solutions for multiple sequence alignment.

Keywords: alignment • progressive alignment • graphics processor card • basic linear algebra subprograms • performance

© Versita Sp. z o.o.

1. Introduction

Sequence alignment is the technique in molecular biology used to compare sequences and to arrange sequences of biomolecules for identifying regions of similarity that are eventually consequences of structural, functional, or evolutionary relationships [6, 10, 20, 22]. The problem of sequence alignment can be tackled by two computational approaches: optimal methods following the paradigm of dynamic programming and heuristic methods. The optimal methods include two basic algorithms. The Needleman-Wunsch algorithm provides a global alignment of two sequences aligning every residue in both sequences. It is most useful when the sequences are similar and of roughly equal length [15]. The Smith-Waterman algorithm yields a local alignment of two sequences in which only part of the residues participate. This method is more utilizable for dissimilar sequences that are suspected to contain regions of similarity [21].

Today, the size of biomolecular sequence databases grows exponentially due to the recent availability of high-throughput sequencing technologies [19]. This upsurge demands for fast alignment techniques rendering the more time-consuming optimal alignment techniques less useful for searching similarities in larger data sets. This is the reason why last heuristic techniques such as BLAST [2] and FASTA [17] are preferred that are an order of magnitude faster than the

* E-mail: muhammad.hanif@tuhh.de
optimal algorithms. However, a downside of heuristic approaches is that they are less sensitive (i.e., missing more homologous) than the optimal ones.

The simultaneous alignment of several sequences results in an NP-complete combinatorial optimization problem [20, 22]. Therefore, a variety of heuristic methods have been developed for the alignment of three or more sequences. The most popular method to generate a multiple sequence alignment is based on trees which are used to describe a relationship between the sequences based on their pairwise comparison. In these progressive methods, the most similar sequences are aligned first and then less related sequences or groups of sequences are successively added to the alignment. Some progressive methods additionally assess the sequences according to their relatedness in order to improve alignment accuracy. The most widely used multiple sequence alignment programs are Clustal [5] and T-coffee [16].

Today, new desktop and notebook computers contain hardware for 3D graphics acceleration called Graphics Processing Units (GPUs). Modern GPUs include many independent floating-point arithmetic units for computing 3D models and other graphical tasks such as video-related functions. This makes GPUs amenable for general-purpose (GPGPU) computations that are traditionally treated by personal computers or workstations. GPUs have already been employed to general purpose computing in several areas such as molecular dynamics, physics simulations, and scientific computing1. Manavski et al. [13] and Munekawa et al. [14] have accelerated the Smith-Waterman algorithm on a GPU gaining moderate performance boosts. Methods to reduce the amount of data transfer and data fetches help to further increase the speed-up. Schatz et al. [18] have provided an implementation of a local sequence alignment algorithm (MUMmer) on a GPU attaining a ten-fold speed-up over a serial CPU version. Similarly, Dzivi [7] has directly implemented the Needleman-Wunsch algorithm and gained performance peaks of an eighty-fold speed-up. All these algorithms follow the wavefront approach utilizing the fact that the anti-diagonals in the corresponding forward table are independent of each other.

Recently, Bassoy et al. [3] have transformed the sequence-profile algorithm into matrix form as a vector-matrix and matrix-matrix product attaining maximum speed-up of 278.5 using NVidia BLAS3 when compared with a native Intel CPU implementation. This huge performance boost is due to the conversion of the alignment problem into a form that matches the vector-processing architecture of commodity GPUs.

In this paper, we provide GPGPU programs for performing profile-profile alignments. This method of alignment is part of progressive alignment allowing to combine two groups of alignments into a single alignment. For this, each alignment is represented by a statistical representative called profile. The basic operation used in profile-profile alignment are scalar products of fixed-length vectors that facilitate the generation of efficient GPGPU code; there are exceptions to this approach [8]. Our implementations run on recent hardware available from NVidia using a new software development kit (CUDA) for GPGPU programming. The performance of our implementations is assessed by comparing it with CPU based computations. The speed-ups achieved by Bassoy et al. [3] for profile-sequence alignment encouraged the implementation of profile-profile alignment algorithm. Although, computation times for profile-profile alignment on CPU might be tolerable since globular proteins are considered in this article. However, computation time on a CPU will be a major factor for genome based profiles which have much longer lengths.

2. Compute unified device architecture

The Compute Unified Device Architecture (CUDA) is the computing engine in NVidia GPUs that is accessible to software developers through standard programming languages like C [1]. CUDA treats the GPU as a compute device that is able to execute a high number of threads in a concurrent manner. CUDA enables the programmer to write C-like functions called kernels. Each kernel is executed by a batch of threads that are organized as a grid of blocks. Each block consists of threads that execute in parallel. Threads in a block can efficiently communicate by using shared memory and in this way can synchronize their execution to coordinate memory access. Each block can be organized as a one-, two-, or three-dimensional array of threads. The maximum number of threads per block is limited. All blocks of a kernel can be grouped into one- or two-dimensional arrays. With the invention of

1 http://gpgpu.org