Massively Parallel Mapping of Next Generation Sequence Reads Using GPUs

Azita Nouri, Reha Oguz Selvitopi, Ozcan Ozturk, Onur Mutlu, and Can Alkan
azita@bilkent.edu.tr, reha@bilkent.edu.tr, ozturk@cs.bilkent.edu.tr onur@cmu.edu, calkan@cs.bilkent.edu.tr

Categories and Subject Descriptions: Parallel Processing
Keywords: GPGPU programming, CUDA, read alignment

1. Introduction and Motivation

DNA sequence alignment problem can be broadly defined as the character-level comparison of DNA sequences obtained from one or more samples against a database of reference (i.e., consensus) genome sequence of the same or a similar species. High throughput sequencing (HTS) technologies were introduced in 2006 [6], and the latest iterations of HTS technologies are able to read the genome of a human individual in just three days for a cost of $\sim$ \$1,000. However, they also present a computational problem since the analysis of the HTS data requires the comparison of >1 billion short (100 characters, or *base pairs*) "reads" against a very long (3 billion base pairs) reference genome. Since DNA molecules are composed of two opposing strands (i.e. two complementary strings), the number of required comparisons are doubled.

Instead of local alignment of short vs long sequences, heuristics are applied to speed up the process. First, partial sequence matches, called "seeds", are quickly found using either Burrows Wheeler Transform (BWT) [1] followed with Ferragina-Manzini Index (FM) [2], or a simple hash table [8]. Next, the candidate locations are *verified* using a dynamic programming alignment algorithm that calculates Levenshtein edit distance [3], which runs in quadratic time. Although these heuristics are substantially faster than local alignment, because of the repetitive nature of the human genome, they often require hundreds of verification runs per read, imposing a heavy computational burden. However, all of these billions of alignments are independent from each other, thus the *read mapping* problem presents itself as embarrassingly parallel.

Our goal in this project is to develop and implement a GPGPU-friendly algorithm based on Levenshtein's algorithm [3] that can compute millions of dynamic programming matrices concurrently. We implement our algorithms using the CUDA (Compute Unified Device Architecture) platform, and test them using the NVIDIA Tesla K20 GPGPU processors. In this work, we propose a massively parallel, fast, memory-aware Levenshtein edit distance algorithm model for graphics processing units, together with Ukkonen's approximation algorithm [7] to prevent redundant calculations in matrices. Considering the memory limitations and very high number of available threads, our algorithm ensures maximum occupancy on GPGPUs.

2. Background

Most of the available algorithms for read mapping are CPU-based, and they require very long running times (30-100 CPU days per genome). There are few works on read mapping algorithms that utilize GPUs for parallelism, but they either have different approaches for DNA alignment [4], and are limited in performance gains; or they are developed for slightly different problems such as protein alignment [5]. Some GPGPU-based aligners, such as CUSHAW [4], utilize BWT-FM [1, 2], which is not suitable for GPGPUs, since the core of the BWT-FM alignments is binary search that causes many divergent branch operations. Even if BWT-FM is used only on the CPU-side to find seed locations quickly instead of hash tables, this often results in a non-uniform sequence length for the verification step, which in turn causes non-uniform thread utilization and significant branch divergence. Thus, it will be better use hash tables to identify short seeds, followed with millions of concurrent alignments of the same size in GPGPUs that enable full thread synchronization.

3. Our Approach

Basically, our approach is to move the compute-intense but embarrassingly parallel verification step to the GPGPUs. We perform the hash table lookups for seed location on the CPU in $O(1)$ time per seed. We collect the seeds in a buffer, which we then pass to the GPGPU for millions of simultaneous alignments. The number of alignments is automatically determined by considering the characteristics of the GPGPU such as available shared memory, which has an effect on the number of threads, blocks, grid and other variables in GPUs. The number of threads used per alignment is adjusted dynamically based on the maximum allowed error threshold set by the user.

There are several characteristics of our approach that makes our new mapping mechanism beneficial in multiple ways. First, we map a time-consuming application to massively parallel GPU architectures, providing a significant speed-up that will dramatically decrease the time required for DNA sequence analysis. Second, our approach easily can be merged with any existing and future hash-table based read mapping applications. Third, our algorithms can be used for various configurations like different read sizes, reference genome size and error allowance. This provides a high flexibility of processing the outputs of different sequencing instruments since each generates various read sizes and error probability. Fourth, we reduce host to GPU transfer time significantly by placing all relevant data to the GPU global memory in the initialization step. This helps us eliminate the need to re-transfer the same reads for different seed locations, and reference genome segments for different reads. Fifth, we also develop dynamic programming backtracking in GPU, bypassing CPU-based postprocessing all together, except for I/O operations. Finally, by making use of recent CUDA improvements such as dynamic parallelism and Hyper-Q technologies, we are able to re-use early-termination threads and multiple GPUs on the same host more effectively.

We integrated our approach with a popular hash-table based read mapping algorithm, mrFAST [8]. Our initial test results indicate $\sim$300-400-fold speed up in the verification step of read mapping on a single NVIDIA Tesla K20 GPGPU (Figure 1). When fully implemented, we believe our methods will help substantially ameliorate the computational burden of HTS data analysis, a much needed improvement over current methods, especially in the light of the estimations that 1 million human genomes will be sequenced by the end of 2016.[1]

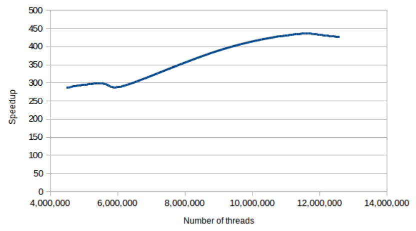

Figure 1: Verification speedup vs. number of threads using 1 million reads under edit distance 3.

References

[1] Michael Burrows and David J. Wheeler. A block sorting lossless data compression algorithm. Technical report, Digital Equipment Corporation, 1994.

[2] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Proc. FOCS*, pages 390–398, 2000.

[3] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[4] Yongchao Liu, Bertil Schmidt, and Douglas L. Maskell. CUSHAW: a CUDA compatible short read aligner to large genomes based on the burrows-wheeler transform. *Bioinformatics*, 28(14):1830–1837, Jul 2012.

[5] Svetlin A Manavski and Giorgio Valle. CUDA compatible GPU cards as efficient hardware accelerators for smith-waterman sequence alignment. *BMC Bioinformatics*, 9 Suppl 2:S10, 2008.

[6] Michael L. Metzker. Sequencing technologies - the next generation. *Nat Rev Genet*, 11(1):31–46, Jan 2010.

[7] Esko Ukkonen. On approximate string matching. In *Proceedings of the International FCT-Conference on Fundamentals of Computation Theory*, pages 487–495, 1983.

[8] Hongyi Xin, Donghyuk Lee, Farhad Hormozdiari, Samihan Yedkar, Onur Mutlu, and Can Alkan. Accelerating read mapping with FastHASH. *BMC Genomics*, 14 Suppl 1:S13, 2013.

[1] http://www.economist.com/news/21631808-so-much-genetic-data-so-many-uses-genes-unzipped