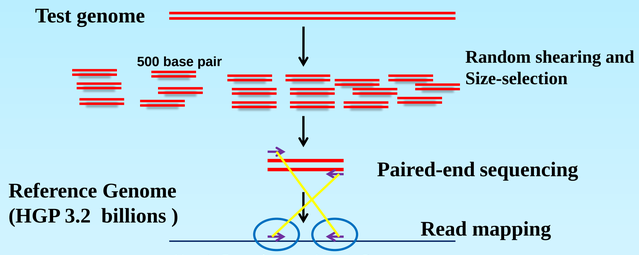
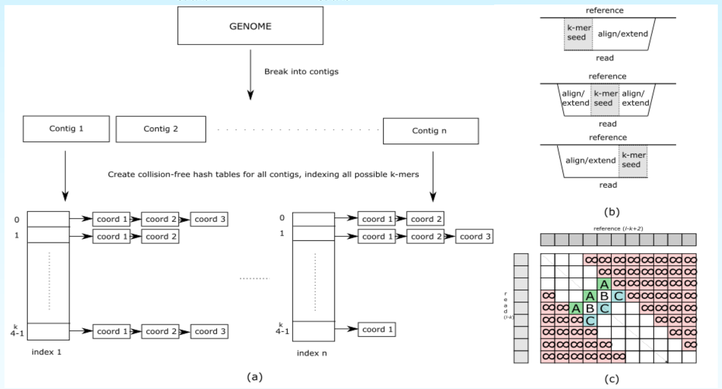
Massively Parallel Mapping of Next Generation Sequence Reads Using GPUs

Azita Nouri, Reha Oğuz Selvitopi, Özcan Öztürk, Onur Mutlu, Can Alkan
Bilkent University, Computer Engineering Department, Turkey
Carnegie Mellon University, Electrical and Computer Engineering Department, USA

Motivation

DNA sequence alignment problem is a character-level comparison of DNA sequences obtained from one or more samples against a database of reference genome sequence of the same or a similar species. This presents a computational problem since the analysis of data requires the comparison of >1 billion short (100 characters, or base pairs) "reads" against a very long (3 billion base pairs) reference genome.





Background

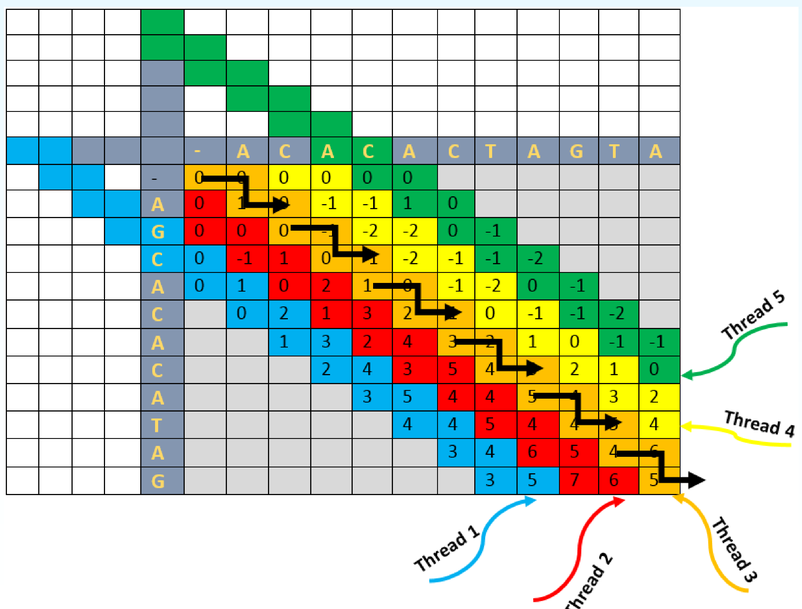
Most of the available algorithms for read mapping are CPU-based, and they require very long running times (30-100 CPU days per genome). GPUs-based mapper have different approaches for DNA alignment, and are limited in performance gains; or they are developed for slightly different problems such as protein alignment.

Our goal is to develop and implement a GPGPU-friendly algorithm based on Levenshtein's edit distance 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

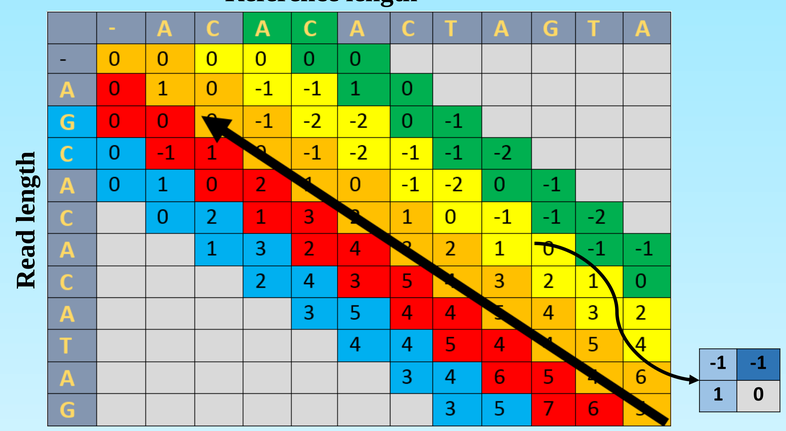
Summary & Contributions

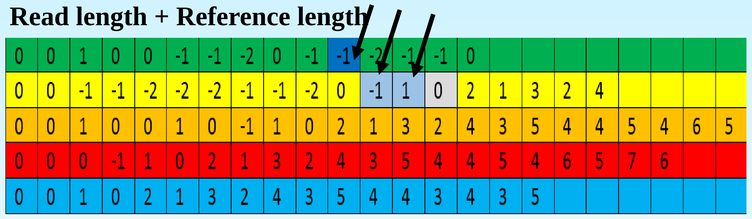
- Map a time-consuming application to massively parallel GPU architectures.
- Move the compute-intense parallel verification step to the GPGPUs.
- Collect the reads in a buffer, then pass to the GPGPU for millions of simultaneous alignments.
- Determine the number of alignments automatically by considering the characteristics of the GPGPU.
- Adjust the number of threads used per alignment dynamically based on the maximum allowed error threshold set by the user.
- Ability to be merged with any existing and future hash-table based read mapping applications.
- Ability to be used for various configurations like different read sizes, reference genome size and error allowance.
- Reduce host to GPU transfer time significantly by placing all relevant data to the GPU global memory in the initialization step.
- Develop dynamic programming backtracking in GPU, bypassing CPU-based post processing all together, except for I/O operations.
- Use 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.

Alignment GPU Algorithm



Backtracking GPU Algorithm





Speedup (tentative results)