FastHASH: A new algorithm for fast and comprehensive next-generation sequence mapping

Hongyi Xin[1], Donghyuk Lee[1], Farhad Hormozdiari[2], Can Alkan[3], Onur Mutlu[1]

[1] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
[2] Department of Computer Science, University of California Los Angeles, CA
[3] Department of Genome Sciences, University of Washington, Seattle, WA

With the introduction of next-generation sequencing (NGS) technologies, we are facing an exponential increase in genomic sequence data. NGS provides low-cost and high-throughput genome sequencing. Yet, it requires enormous computational resources. This is because 1) NGS maps many reads to the *entire* human genome and 2) each read is *short*, making it computationally expensive to search for all potentially-matching locations in the human reference genome, especially in the presence of polymorphisms. A key challenge in making NGS successful in medical and genetic applications is to design algorithms that can process and analyze the enormous amounts of sequence data very fast and energy-efficiently.

To address this ever increasing computational need, several solutions were proposed. Mapping algorithms based on Burrows-Wheeler transform and Ferragina-Manzini index such as BWA and SOAP2 are fast in searching for exact matches of a read in the human reference genome, but are not comprehensive, i.e. they do not search for all potentially-matching locations. On the other hand, hash table based seed-and-extend algorithms, like mrFAST and SHRiMP, are comprehensive but typically much slower. Some of these algorithms use single-instruction multiple-data (SIMD) operations available in modern processors to modestly improve performance. Several other algorithms utilize graphics processing units (GPUs), but they are not optimized to fully take advantage of the massively-parallel GPU architecture.

In this work, we propose a new algorithm, FastHASH, which drastically improves performance over mrFAST, while maintaining comprehensiveness. Our key observation is that mrFAST performs too many costly computations that can be avoided by intelligently restructuring the algorithm to take advantage of the full knowledge of the human reference genome. There are two sources of costly computations in MrFAST that our new algorithm reduces. First, for every read, mrFAST finds out *all* potentially-matching locations in the reference genome. Then, for each possible potentially-matching location, mrFAST performs an expensive string comparison to extract complete differential information between the input read and the reference genome, *even if the location will not match*. FastHASH uses two key ideas to reduce both types of expensive computations. First, it drastically reduces the number of potentially-matching locations considered for string comparison, while still preserving comprehensiveness, by making use of complete information of the reference genome. We call this method Cheap Session Selection. Second, it drastically reduces the number of string comparisons by rejecting obviously-incorrect locations in the early stages of mapping. We call this method Adjacent Filtering.

Our initial CPU implementation of FastHASH provides 40-fold speedup over mrFAST, while preserving the same comprehensiveness. Since FastHASH has little control-flow that significantly diverges, a property essential for efficient GPU implementation, FastHASH is also amenable to a GPU implementation.