Ultra-fast genotyping of SNPs and short indels using GPU-acceleration

Authors

- **Jørgen Wictor Henriksen**
  Biomedical Informatics research group, Department of Informatics, University of Oslo, Oslo, Norway

- **Ivar Grytten**
  Biomedical Informatics research group, Department of Informatics, University of Oslo, Oslo, Norway

- **Knut Dagestad Rand**
  Biomedical Informatics research group, Department of Informatics, University of Oslo, Oslo, Norway; Centre for Bioinformatics, University of Oslo, Oslo, Norway

- **Geir Kjetil Sandve** ✉
  Biomedical Informatics research group, Department of Informatics, University of Oslo, Oslo, Norway; Centre for Bioinformatics, University of Oslo, Oslo, Norway; UiORealArt Convergence Environment, University of Oslo, Oslo, Norway

✉ — Correspondence possible via GitHub Issues or email to Geir Kjetil Sandve <geirksa@ifi.uio.no>.

Abstract

As sequencing costs have steadily decreased in recent years, having efficient methods for variant discovery and genotyping has become increasingly important. Recent methods have shown that genotyping can be done efficiently and accurately using *alignment-free* methods that are based on analysing kmers from sequenced reads. In recent work, we have presented the KAGE genotyper, which uses an efficient pangenome representation of known individuals in a population to further increase accuracy and efficiency.

We here present *GKAGE*, a new and improved version of KAGE that utilises the *Graphical Processing Unit* (GPU) to further increase the efficiency by counting and analysing large amounts of kmers in parallel. We show that GKAGE is able to genotype an individual up to a magnitude faster than KAGE while producing the same output, which makes it by far the fastest genotyper available today. GKAGE can run on consumer-grade GPUs, and enables genotyping of a human sample in only a matter of minutes without the need for expensive high-performance computers. GKAGE is open source and available at https://github.com/ivargr/kage.

Introduction

The cost of sequencing a full human genome has fallen drastically in recent years, and consumers can now get their whole genome sequenced for a few hundred dollars, a fraction of what was the price only a few years ago. As millions of genomes are likely to be sequenced in the coming years, there is an ever-increasing need for efficient methods for analysing this genomic data. At the core of such analysis is *variant detection*, determining which genetic variants a sample has based on the sequenced reads.

Recent methods [1,2,3] have shown that detecting variants in a human sample can be performed efficiently and with high accuracy by *genotyping* the sample against an existing database of known human variation. Such methods use prior knowledge from e.g. the 1000 Genomes Project [4] about where in the genome individuals may have variation, and for each such genetic variant uses the genomic reads from the sample to infer the most likely genotype for the given sample. While such methods traditionally have been based on aligning reads to a reference genome, which is slow, recent *alignment-free methods* have shown that drastic speedup can be achieved by instead analysing kmers from the sequenced reads. In a recent publication [1] we proposed a highly efficient alignment-free method, KAGE, that used prior knowledge from a population to achieve high genotyping accuracy while being more efficient than other alignment-free genotypers.

Alignment-free genotypers, like KAGE, generally work by using an index (e.g. a hashmap) that enables "mapping" of kmers from input reads to alleles of known genetic variants. The genotyping is then performed by counting how many reads support the various alleles of the variants of interest. Since all kmers from the input reads are counted independently, this process can easily be parallelized. While existing alignment-free genotypers uses the central processing unit (CPU) for parallel kmers counting, we hypothesise that this can be done much more efficiently by instead using the *Graphical Processing Unit (GPU)*, which is able to perform massive amounts of computations in parallel. Recent work has also shown that the GPU can greatly improve the efficiency of kmer counting in general [6].

We here present GKAGE, a GPU-accelerated version of KAGE, which to our knowledge is the first genotyper to utilise the GPU. GKAGE uses the GPU to process genomic reads, extract kmers and count how many kmers support alleles of genetic variants. We show that GKAGE gives significant speedup (up to 8X) over the original KAGE genotyper while producing the exact same output. GKAGE has been implemented so that it is able to run even on standard consumer GPUs, and is able to genotype a whole human sample in a matter of minutes.

Results

GKAGE is a GPU-accelerated version of the recently published KAGE genotyper. GKAGE uses CUDA-powered GPUs to efficiently parse and encode kmers from reads and genotype a set of known SNPs and indels based on the kmer counts. GKAGE produces identical output as KAGE with reduced runtime on systems that enable GPU-acceleration and support the CUDA-interface. The software is open source and available at https://github.com/ivargr/kage. As part of GKAGE, we have also implemented a static GPU hashmap for counting kmers through a Python interface, available at https://github.com/jorgenwh/cucounter.

We have recently shown that KAGE is an order of magnitude faster than existing genotypers while giving better or comparable accuracy [1]. We thus only benchmark GKAGE against KAGE to show the effect of GPU-acceleration. We do this by running GKAGE and KAGE on a human whole genome sample (15x coverage) on two different systems:

1. A high-end server with an AMD EPYC 7742 64-Core CPU and two NVIDIA Tesla V100 GPUs. KAGE was run using 16 cores and GKAGE was run using one GPU.

2. A regular desktop computer with an 11th Gen Intel(R) Core(TM) i5-11400F @ 2.60GHz CPU and a NVIDIA GTX 1660 super GPU. KAGE was run using 6 cores.

Table 1 shows the runtimes on these two systems. GKAGE is approximately 5x faster on the high-end system and 8x faster on the desktop computer.

Table 1: Running times of KAGE and GKAGE

	KAGE	GKAGE
Desktop computer	1880 sec	180 sec
High-performance computer	510 sec	108 sec

Methods

Implementation

Here we describe more in detail how GKAGE has been implemented. While GKAGE is implemented as part of KAGE, and in large parts uses the same code, compute-heavy parts have been reimplemented so that the GPU is utilised. GKAGE mainly implements GPU support for two bottleneck components of KAGE that were suitable for GPU-acceleration:

Reading and encoding kmers from a FASTA/FASTQ file is achieved in KAGE by using BioNumPy [7], a Python library built on top of NumPy [8]. BioNumPy uses NumPy to efficiently read chunks of DNA reads from fasta files, encode the bases to a 2-bit representation, and then encode the valid kmers as 64-bit integer representations in an array. GPU support for this step was achieved by utilising CuPy [9], a GPU accelerated alternative to NumPy, as a drop-in replacement for NumPy.

Counting kmers As part of GKAGE, we have implemented a static hashmap for counting a predefined set of kmers on the GPU. The implementation supports parallel and high-throughput hashing and counting of large chunks of kmers simultaneously on the GPU.

Genotyping based on kmer counts is performed using the existing KAGE algorithm, which is already very fast and did not benefit from GPU-acceleration.

Benchmarks

A Snakemake pipeline for reproducing the benchmarking results can be found at https://....

Discussion

We have presented a GPU-accelerated genotyper by enabling GPU-support for the most compute-intensive operations of the existing KAGE genotyper. Our results show that alignment-free genotyping is an ideal problem for GPU-acceleration. While the existing KAGE genotyper already is fast by today's standards, GKAGE is considerably faster, and makes it even easier to run whole-genome genotyping on a typical consumer computer. We believe the results are relevant as whole-genome sequencing is continuously becoming cheaper and more common.

Since the original KAGE genotyper was implemented mainly using the array programming libraries NumPy and BioNumPy in Python, adding GPU-support to the existing code base was possible to do in a clean way by using the CuPy library as well as implementing some custom CUDA kernels with Python wrappers. We thus believe that this work shows that adding GPU-support to existing tools is feasible and can be beneficial in cases where many independent operations are performed on an array of data. As GPUs are becoming cheaper and more available, we thus believe there might be a huge potential in improving the performance of existing tools and methods, which in many cases can be done quite easily through the Python ecosystem with packages such as CuPy [9], Numba [10] and BioNumPy [7].

References

1. **KAGE: fast alignment-free graph-based genotyping of SNPs and short indels**
Ivar Grytten, Knut Dagestad Rand, Geir Kjetil Sandve
Genome Biology (2022-10-04) https://doi.org/grpzvf
DOI: 10.1186/s13059-022-02771-2 · PMID: 36195962 · PMCID: PMC9531401

2. **MALVA: Genotyping by Mapping-free ALlele Detection of Known VAriants**
Luca Denti, Marco Previtali, Giulia Bernardini, Alexander Schönhuth, Paola Bonizzoni
iScience (2019-08) https://doi.org/gmhw28
DOI: 10.1016/j.isci.2019.07.011 · PMID: 31352182 · PMCID: PMC6664100

3. **Graphtyper enables population-scale genotyping using pangenome graphs**
Hannes P Eggertsson, Hakon Jonsson, Snaedis Kristmundsdottir, Eirikur Hjartarson, Birte Kehr, Gisli Masson, Florian Zink, Kristjan E Hjorleifsson, Aslaug Jonasdottir, Adalbjorg Jonasdottir, … Bjarni V Halldorsson
Nature Genetics (2017-09-25) https://doi.org/gbx7v6
DOI: 10.1038/ng.3964 · PMID: 28945251

4. **A global reference for human genetic variation**
, Adam Auton, Gonçalo R Abecasis, David M Altshuler (Co-Chair), Richard M Durbin (Co-Chair), Gonçalo R Abecasis, David R Bentley, Aravinda Chakravarti, Andrew G Clark, Peter Donnelly, …
Nature (2015-09-30) https://doi.org/73d
DOI: 10.1038/nature15393 · PMID: 26432245 · PMCID: PMC4750478

5. **Gerbil: a fast and memory-efficient k-mer counter with GPU-support**
Marius Erbert, Steffen Rechner, Matthias Müller-Hannemann
Algorithms for Molecular Biology (2017-03-31) https://doi.org/gkzhfr
DOI: 10.1186/s13015-017-0097-9 · PMID: 28373894 · PMCID: PMC5374613

6. **GPU Acceleration of Advanced k-mer Counting for Computational Genomics**
Huiren Li, Anand Ramachandran, Deming Chen
2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP) (2018-07) https://doi.org/grp3kq
DOI: 10.1109/asap.2018.8445084

7. **BioNumPy: Fast and easy analysis of biological data with Python**
Knut Rand, Ivar Grytten, Milena Pavlovic, Chakravarthi Kanduri, Geir Kjetil Sandve
Cold Spring Harbor Laboratory (2022-12-22) https://doi.org/grp3k6
DOI: 10.1101/2022.12.21.521373

8. **Array programming with NumPy**
Charles R Harris, KJarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, … Travis E Oliphant
Nature (2020-09-16) https://doi.org/ghbzf2
DOI: 10.1038/s41586-020-2649-2 · PMID: 32939066 · PMCID: PMC7759461

9. **Cupy: A numpy-compatible library for nvidia gpu calculations**
Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, Crissman Loomis
Thirty-first Annual Conference on Neural Information Processing Systems (NeurIPS)

10. **Numba**
Siu Kwan Lam, Antoine Pitrou, Stanley Seibert
Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC (2015-11-15) https://doi.org/gf3nks

DOI: 10.1145/2833157.2833162