Speeding up Genotyping through GPU Acceleration

Master's Thesis

Jørgen Wictor Henriksen

15th of May 2023

Abstract

Contents

1 Introduction

2 Background

This section will explain necessary preliminaries for the further topics in this thesis.

2.1 Graphical Processing Units

Graphical Processing Units (GPUs) are massively parallel processing units designed for high-throughput parallel computations. This is as opposed to *Central Processing Units* (CPUs), which are designed to quickly perform many serial computations. GPUs were originally developed to accelerate computations performed on images, a highly parallel task where it is commonplace to have millions of relatively small independent computations that must be performed quickly in a single memory buffer. Although GPUs have mainly been used for graphical computations, they have in recent years been adopted in other areas as well with the introduction of the *General Purpose Graphical Processing Unit* (GPGPU) (**GPGPU cite**). The concept of the GPGPU is to use a GPU, which is designed for computer graphics, to perform computations in other domains where CPUs are typically used. Fields such as artificial intelligence (**AI accelerated by GPU cite**) and the broader scientific computing have enjoyed great utility from GPUs, using them to accelerate embarassingly parallel problems, *e.g.*, matrix operations. Despite being similar in power consumption, a GPU can provide much higher instruction throughput and memory bandwidth compared to its CPU competitors. These capability advantages exist in GPUs because they were specifically designed to perform well with regards to these dimensions.

As of 2023, Nvidia control the vast majority of the GPU market share, with only *Advanced Micro Devices* (AMD) and Intel as current serious competitors (**try to find a serious cite**). Furthermore, Nvidia GPUs with their CUDA programming model is considered to be the standard for scientific computing today (**cite**). Although most of the GPUs manufactured by different GPU manufacturing companies are very similar in both architecture and compute models, the term *GPU* will for the remainder of this thesis specifically refer to Nvidia GPUs, as the work presented in this thesis was developed and tested using only Nvidia GPUs.

2.1.1 GPUs in Computers

Two main computer GPU setups are prominent today: *integrated* graphical processing units (iGPU), and *discrete* graphical processing units (dGPU). iGPUs are GPUs integrated onto the same die as a computer's CPU, where the two share the same physical *Random Access Memory* (RAM) unit. dGPUs are dedicated GPU devices that are physically distinct from the host computer's CPU and RAM, and have their own physical RAM. dGPUs are significantly more powerful in terms of compute throughput when compared to iGPUs. However, having their own physical RAM introduces an overhead; Memory buffers with input data have to be copied to the dGPU's RAM before processing, and results have to be copied back from the dGPU's RAM to the host RAM.

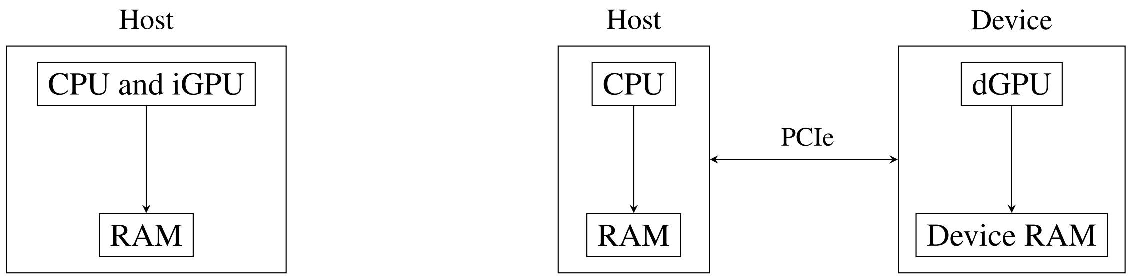

Figure 1: **Left**: A computer setup with a CPU and an iGPU sharing the same die and the same physical RAM. **Right**: A computer setup where a dGPU is connected over PCIe and the dGPU has its own physical RAM, adding the overhead of copying data both to and from the host computer when utilizing the GPU.

For the work presented in this thesis, only dGPUs were utilized. Therefore, the term GPU will from here on out be referring to a dGPU and not an iGPU. This means that all GPU implementations discussed in this thesis will include copying memory back and fourth from the *host* (CPU) RAM and the *device* (GPU) RAM. It is also possible for a single computer to have several connected GPUs, allowing for further parallelization of both memory transfers and compute, however this was not utilized in this thesis' work.

2.1.2 Programming Model and CUDA

Modern GPUs can in effect be considered to be massive *Single Instruction Multiple Data* (SIMD) machines. Strict flow control is therefore important; The same set of instructions should run in the same order for maximum utilization of the GPU's capability.

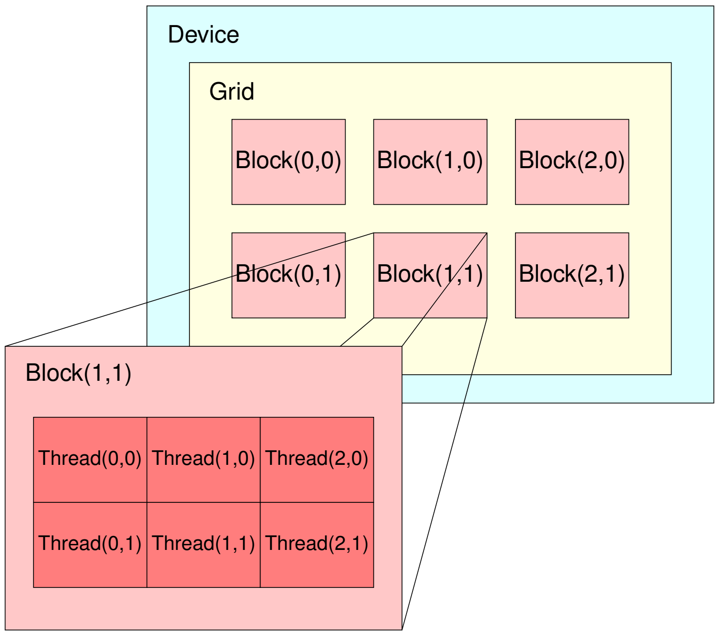

Figure 2: An overview of the CUDA programming model:

2.2 Biology

2.2.1 DNA and Chromosomes

Deoxyribonucleic acid (DNA) is a type of molecule that carries the genetic instructions for the development, function, and reproduction of all known living organisms. The molecule is composed of two connected strands that wind around each other into a double helix shape. Each strand is made up of a sugar and phosphate backbone, with each sugar carrying one out of four bases, often referred to as *nucleotides*. The four DNA nucleotides are: *cytosine*, *guanine*, *adenine* and *thymine*, and they are typically referred to as simply C, G, A and T respectively. Furthermore, chemical bonds form between complementary pairs of nucleotides: AT/TA and CG/GC, meaning that A and T, and C and G are complementary nucleotides. The two strands are connected through the bonds formed between these nucleotides. Since the two strands are sequences of complementary nucleotides, the two strands do in fact represent the same genetic information. In other words, one strand can be determined by reversing the other and exchanging each nuclotide by its complement.

DNA is organized into structures called chromosomes. Humans have 46 chromosomes made up by two sets of 23 chromosomes where each chromosome occurs twice. One is inherited from the male parent and the other is inherited from the female parent.

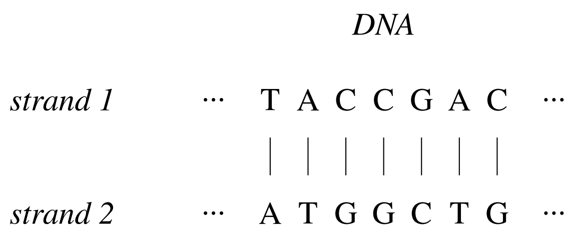
DNA
strand 1 ··· T A C C G A C ···
strand 2 ··· A T G G C T G ···

Figure 3: A conceptual representation of a DNA molecule made up of two strands. The strands are composed of nucleotides forming base pairs where A (adenine) and T (thymine), and C (cytosine) and G (guanine) are complements of each other.

2.2.2 High-Throughput DNA sequencing

2.3 Nucleotide Binary Encoding

DNA nucleotide sequences (described in section 2.2.1) inside computer software is commonly represented simply by a sequence of the 8 bit characters A, C, T and G (or alternatively the lowercase a, c, t and g). This representation is, however, is cumbersome to operate on and requires large amounts of memory to store. To circumvent these issues, a widely adopted technique is to encode the nucleotides into binary form. This leads to much quicker processing of nucleotide sequences and reduces the memory usage needed to store the sequences by 75%. This is achieved by realizing that only 2 bits, giving $2^2 = 4$ possible unique states, is enough to represent all of the four DNA nucleotides A, C, G and T. The binary encoding can be extended further to represent whole nucleotide sequences in binary arrays. For example, an integer array, if interpreted 2 consecutive bits at a time, can represent such a sequence.

3 Thesis Goal

The goal of this thesis is to explore whether state-of-the-art genotyping can be sped up in any significant way by utilizing GPUs. More specifically, this thesis will investigate whether alignment-free genotyping, which presently is significantly faster compared to alignment-based genotyping, can be sped up by the GPU. In order to investigate this, I will attempt to integrate GPU accelerated functionality into a base genotyper, KAGE [1], which is presently the fastest known genotyper that also yields good results. The resulting GPU accelerated genotyping software, GKAGE, will then be benchmarked against KAGE to account for any potential speed up. Finally, I will discuss elements about the work process and implementations, discuss possible future work and conclude the work presented in this thesis.

Nucleotide to 2 bit encoding lookup table

A	$\longleftrightarrow$	00
C	$\longleftrightarrow$	01
G	$\longleftrightarrow$	10
T	$\longleftrightarrow$	11

DNA	A	C	C	T	G	T	A	G
	|	|	|	|	|	|	|	|
2 bit represented DNA	00	01	01	11	10	11	00	10

Figure 4: A lookup table showing how nucleotides can be encoded using 2 bits and a DNA nucleotide sequence represented both as plain characters as well as its 2 bit encoded representation. Recall that computers use 8 bits to represent a single nucleotide with a character, whilst the 2 bit encoding only needs 2 bits to represent a nucleotide.

4 Methods

5 Results

The following section will describe the results yielded from this master's project.

All benchmarking results presented in this chapter were determined by running the associated pieces of software on the following set of computers and then comparing the results.

System	CPU	GPU
High-end compute server	AMD EPYC 7742	NVIDIA Tesla V100 (32GB)
Consumer desktop computer	Intel Core i5-11400F	NVIDIA GTX 1660 SUPER (6GB)

5.1 *K*mer parsing from raw reads

In section () I described how I implemented GPU support for parts of BioNumPy [2] in order to allow for GPU acceleration when when parsing *k*mers from raw reads in FASTA files. In short: BioNumPy reads and parses chunks of *k*mers from FASTA files by 1) reading a chunk of raw bytes from the FASTA file, 2) converting each DNA nucleotide found in the raw read data into 2-bit encoded representations (2.3), and 3) parsing all valid *k*mers of the desired size from the 2-bit encoded reads data.

After copying the raw bytes read from the FASTA file directly to the GPU, steps 2) and 3) could be performed significantly faster on the GPU for large enough chunk sizes.

5.2 *K*mer counting

6 Discussions

6.1 Drawbacks of Graphical Processing Units

While GPUs can be excellent for accelerating many problems in scientific computing, they do come with some notable caveats.

Expensive; power hungry; have experienced significant fluctuation in price and ease of access in the last couple of years (although seemingly mostly because of mining?); can be difficult to make effective solutions for someone inexperienced with the GPU compute models and frameworks; not suitable for every problem, only problems that are possible to rephrase as massively parallel, and it can be difficult to judge whether a problem is suitable before trying

7 Conclusion

References

[1] Ivar Grytten, Knut Dagestad Rand, and Geir Kjetil Sandve. "KAGE: Fast alignment-free graph-based genotyping of SNPs and short indels". In: *bioRxiv* (2021). DOI: `10.1101/2021.12.03.471074`. eprint: `https://www.biorxiv.org/content/early/2021/12/20/2021.12.03.471074.full.pdf`. URL: `https://www.biorxiv.org/content/early/2021/12/20/2021.12.03.471074`.

[2] Knut Dagestad Rand and Ivar Grytten. *bionumpy*. `https://github.com/bionumpy/bionumpy`. 2022.