Langdon and Lam *BioData Mining* (2017) 10:28
DOI 10.1186/s13040-017-0149-1

BioData Mining

SHORT REPORT Open Access

# Genetically improved BarraCUDA

W. B. Langdon[1*] and Brian Yee Hong Lam[2]

*Correspondence:
w.langdon@cs.ucl.ac.uk
[1]Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK
Full list of author information is available at the end of the article

**Abstract**

**Background:** BarraCUDA is an open source C program which uses the BWA algorithm in parallel with nVidia CUDA to align short next generation DNA sequences against a reference genome. Recently its source code was optimised using "Genetic Improvement".

**Results:** The genetically improved (GI) code is up to three times faster on short paired end reads from The 1000 Genomes Project and 60% more accurate on a short BioPlanet.com GCAT alignment benchmark. GPGPU BarraCUDA running on a single K80 Tesla GPU can align short paired end nextGen sequences up to ten times faster than `bwa` on a 12 core server.

**Conclusions:** The speed up was such that the GI version was adopted and has been regularly downloaded from SourceForge for more than 12 months.

**Keywords:** GPGPU, Parallel computing, Genetic improvement, Double-ended DNA sequence, Nextgen NGS

## Background

### Why run bioinformatics on gaming machines

Bioinformaticians have seized the advantages of using computer graphics hardware (GPUs) [1], particularly those made by nVidia. Amongst other software tools, nVidia's CUDA gives the ability to run C/C++ programs on nVidia GPUs. CUDA versions of several popular Bioinformatics applications have been written. In particular BarraCUDA [2], which aligns short noisy DNA sequences against one of the increasing number of reference genomes. It can align human DNA sequences when run on nVidia consumer GPU cards with more than 4 GB of memory, e.g. the GeForce GT 730. In fact it has been run on cards costing less than $100 up to $325 million super computers.

Recently [3] we presented an approach in which a small part of the manually written code had been optimised by a variant of genetic programming [4, 5] to give a huge speed up on that part. (The raw graphics kernel can process well over a million DNA sequences a second ([3], Fig. 1)). The next section describes BarraCUDA and other programs ("Alternative tools" section) for aligning large numbers of next generation DNA sequences, whilst "Scalability" section considers factors affecting their performance and "Genetic improvement (GI)" section contains a very quick introduction to the genetic improvement of software (GI) technique used to create the current version of BarraCUDA. "Programs, DNA sequences and parallel operation under multi-core Unix" section gives details of the programs and DNA benchmarks. This is followed ("Results"

section) by the overall performance changes genetic improvement [6–12] gives and comparison with `bwa`. (See particularly Table 3.)

## Introduction

### NextGen DNA sequence alignment

Both version of BarraCUDA and `bwa` use the Burrows-Wheeler algorithm. This requires the reference genome to be converted offline into an index file. The whole of the index must be kept in memory. Fortunately modern GPUs have high capacity and high bandwidth to their on-board memory (see last column in Table 1).

Typically the Burrows-Wheeler algorithm scales linearly with the length of the DNA sequences to be looked up. This makes it more suitable for shorter sequences than for longer ones. Ideally sequences should not exceed 100 bp, however we have recently demonstrated BarraCUDA on paired end epigenetic data of 150 bp [13]. (Above 150 bp, BarraCUDA issues a warning and ignores the remainder of the string.)

Taking The 1000 Genomes Project as an example, ([14], Fig. 4) shows some sequence lengths are much more common than others. In Section Results we report tests on paired end data comprised of 36 bases per end and of 100 bases per end. Both are common in The 1000 Genomes Project (in fact the most popular is 101 bases).

### Alternative tools

Highnam et al. [15] report the accuracy of four popular CPU based alignment tools. Three are open source Bowtie2 [16] `bwa` and BWA-MEM [17] whilst Novoalign3 is commercial). They says the tools' accuracy lies between 91% (Bowtie2) and 98% (BWA-MEM). BarraCUDA is towards the top of this range, see "GCAT" data in the last column in Table 3. Lenis and Senar [18] tuned performance for four open source CPU aligners (Bowtie2, BWA-MEM, GEM [19] and SNAP [20]) on a 64 core AMD Opteron Processor 6376, 128 gigabyte computer. They report ([18], Table 4) BWA-MEM and GEM 3.0 give similar speed but GEM 3.0 is the fastest of the four at 383 000 sequences per second. (Notice this is for single ended 100bp next generation DNA sequences, NGS, see list of abbreviations).

Luo et al. [21] compare SOAP3-dp [22] and their own MICA on what was at the time the world's fastest computer. The Tianhe-2 contains 48 000 Intel Xeon Phi 31S1P many integrated core (MIC) co-processor boards. They report that SOAP3 [23] running on GTX 680 [24] is the fastest of the 13 open source aligners they benchmarked. (nVidia's GTX 680 GPU has 1536 1.06 GHz cores and they claim a memory bandwidth of 192.26 GB/s. Luo et al. give performance for MICA, SOAP3-dp, SOAP, Bowtie2 (3 settings), `bwa`, SeqAlto [25] (2 settings) CUSHAW2 [26] and GEM (3 Settings)). Luo et al.'s

**Table 1** Parallel computer graphics hardware

| GPU | | Compute level | | MP | Total cores | Clock | Memory | |
|---|---|---|---|---|---|---|---|---|
| GT 730 | 2014 | £54 | 2.1 | 2× | 48 = 96 | 1.40 GHz | 4 GB | 23 GB/s |
| Tesla K20 | 2012 | £2905 | 3.5 | 13× | 192 = 2496 | 0.71 GHz | 5 GB | 140 GB/s |
| Tesla K80[a] | 2014 | £6261 | 3.7 | 13× | 192 = 2496 | 0.82 GHz | 11 GB | 138 GB/s |

Fourth column is CUDA compute capability level. Each GPU chip contains 2 or 13 identical independent multiprocessors (MP, column 5). Each MP contains 48 or 192 stream processors (total column 7). Onboard memory size and bandwidth are given in the right most two columns. Technical report [36] has full details

[a]K80 is a dual GPU, Original total list price is followed by performance data for one half

Table 3 [21] suggests that SOAP3 on a GTX 680 processes 45 500 paired end simulated NGS DNA sequences per second. However the accuracy (sensitivity) of 97.77% is the lowest of the 13 tools in ([21], Table 3) whereas SOAP3-dp (99.66%) is the highest. Apart from their own MICA, the other ten tools are all benchmarked on Intel i7-3730k, 6-core 3.2 GHz CPUs. For example (depending upon parameter settings), ([21], Table 3) suggests GEM processes from 14 400 to 20 100 sequence/sec.

nvBowtie https://nvlabs.github.io/nvbio/nvbowtie_page.html was written by nVidia on top of their nvbio CUDA templates to emulate Bowtie2 on their GPUs. They claim similar accuracy and on real 100 bp paired end NGS data they claim a single dual K80 Tesla gives a 2.1 speed up compared to Bowtie2 using 20 threads on an Intel Xeon E5-2690 v2 CPU.

The above tools are based on compressing the reference genome into a prefix index using the Burrows-Wheeler transform (which gives `bwa` its name). This has the advantage that typically, e.g. for human data, the whole index can be fitted into a GPU's memory or indeed into many laptops and personal computers. However it means at best finding each NGS short read (of $n$ bp) takes $O(n)$ time. (Typically, where the NGS data are noisy or the DNA truly differs from the reference genome, e.g. due to SNPs or indels, these tools back up and perform some type of heuristic guided truncated tree search of the index. This can greatly increase run time.) Arioc [27] takes a different approach.

Arioc [27] uses hash techniques. In principle hashing allows constant time ($O(1)$) access but in practise search is complicated by the need to deal with inexact matches and the size of the hash table. Wilton et al. [27] say for the human reference genome Arioc needs about 65 GB of RAM on the host computer. Sixty five gigabyte is far more than current generation GPUs and so Arioc takes care to optimise loading it in parts into GPUs. They compare Arioc performance against four Burrows-Wheeler based tools, two CPU based (Bowtie2 and BWA-MEM) and two GPU based (SOAP-dp and nVidia's NVBIO from http://nvlabs.github.io/nvbio/) Wilton et al. [27] run the three GPU based aligners on an nVidia K20 Tesla. They use a 6 dual 2.93 GHz cores (24 threads of execution) workstation with 144 GB of system memory for Bowtie2 and BWA-MEM. They say "Arioc demonstrated up to 10 times higher throughput across a wide range of sensitivity settings." ([27], Fig. 8) suggests Arioc can process well in excess of 200 000 100 bp paired end simulated NGS DNA sequences against the YanHuang genome per second, although speed falls by more than a factor of ten to increase accuracy from ≈93.4% to ≈95.6%. Surprisingly ([27], Fig. 8) suggests BWA-MEM accuracy is at best 93.2% which is well down on other benchmarks, e.g. those reported by Highnam et al. [15].

#### Scalability

Typically next generation sequencing (NGS) alignment tools (such as `bwa`, both versions of BarraCUDA and those mentioned in the previous section) have a start up overhead. Once started, data are often processed at a constant rate. (I.e. run time is linear in number of NGS DNA sequences.) However speed may be dramatically affected by user selected options and the quality of the data. For example, some tools allow the user to force all potential matches to be reported. Naturally this can considerably reduce the aligner's speed. Similarly poor data can force the aligner to do more internal back tracking, which can similarly reduce the alignment rate (particularly if the user changes parameters to compensate for the noisy data).

In many cases the aligners have strict memory requirements. Typically the computing hardware must have sufficient RAM to hold one or more large indexes. The size of the index typically depends upon the reference genome. Once this requirement is met, there is often little additional advantage if the size of RAM is increased. With CPU based aligners, Lenis and Senar [18] showed performances gains can sometimes be had by placing the data near the CPU using it, even if the data have to be duplicated. Usually GPU based tools use techniques such as non-paged/locked memory on the host to maximise data rates across the PCI bus. (PCI buses are often used to connect GPUs and host computers.) However PCI is typically an order of magnitude slower than CPU–RAM speeds. Therefore, except for Arioc[1] (previous section), GPU based aligners usually copy large data structures onto the GPU only once when they start. For BarraCUDA and similar aligners this means for human data the GPU must have at least 4GB of on board RAM.

GPU based aligners (such as BarraCUDA) typically get their speed by using a GPU thread per sequence. This works well where DNA strings map directly and uniquely to the reference genome. However, e.g. when data are noisy, there may be a need for the search to backtrack. Since when backtracking is invoked is different for each query, this means each GPU thread behaves differently, which means in turn that they diverge. Modern GPUs support thread divergence transparently, however divergent threads impact the GPU's speed. In the case of the GI version of BarraCUDA we deal with this by inserting a non-divergent high speed pre-pass. This handles most cases. DNA strings which do not map simply, fall back to the slower divergent code. Thus, as with other tools, mentioned above, in practise speed will depend upon how noisy the DNA strings are. Although thread divergence is particularly a problems with GPU based aligners, the problem of noisy data needing more complex handling affects all NGS aligners and their speed can always be adversely affected by poor quality DNA sequence data.

#### Genetic improvement (GI)

Genetic improvement is the process of applying search based optimisation techniques, such as genetic programming [5], directly to software [12]. Specifically, we applied grow and graft genetic programming [28] to BarraCUDA's GPU source code. (Full details are given in [3].)

Darwinian evolution is applied inside the computer (see evolutionary cycle in Fig. 1). A grammar describing the CUDA source code and CUDA parameters, and all legal mutations of source code is automatically generated from the manually produced program, and a population of 1000 individuals each defining a set of CUDA parameters and code mutations is created. Each is applied via the grammar to give a new CUDA kernel, which is compiled. The grammar ensures the mutant code is syntactically correct and has a high chance of compiling. Each new kernel is run on ≈160 000 NGS DNA strings and its answers and how long it took are compared with the original code. Kernels which produce equivalent answers and are faster than the original are eligible to be parents of the next generation of mutant kernels. The fastest half of the population are given two children each, one is created by crossover with another fit parent and one by mutation. We cycle through 50 generations. The best of the last generation is tested on millions of DNA sequences not used by the GP.

In [3] we evolved new versions of BarraCUDA's GPU code specifically for two different GPUs (K20 and K40), however it is easier to support a single version. Therefore

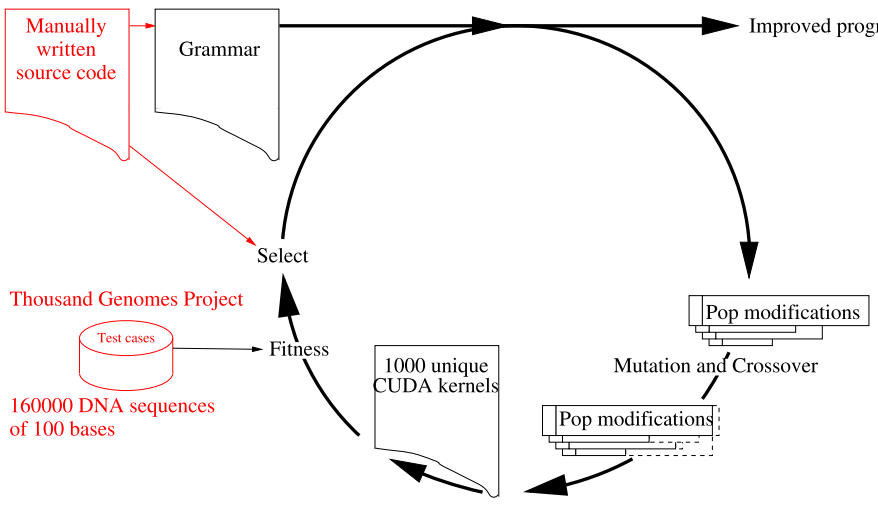


**Fig. 1** Major components of Genetic Improvement (GI)

the two evolved versions were manually reconciled into one, which was released in March 2015. This version has subsequently been maintained in the conventional manual way. Nonetheless this does not preclude re-applying genetic improvement in future. E.g. to re-tune BarraCUDA to new hardware or new types of data, such as longer DNA sequences.

Designers of Bioinformatics and other software are frequently faced with heuristic design choices. Sometimes these can be parametrised so that the choice can be delegated to the users. This can lead to a huge number of frequently opaque command line options. In which case the software designer has to provide sensible defaults. In many cases the designer makes their best guess based on anticipated use and expectations of the computer hardware that will be available. Although best placed to make these choices at the time, pressure of other tasks can make it impractical for them to re-visit these choices should circumstances change. Where parameters were exposed, search can be used to optimise them [29, 30] and if not, new GI techniques, such as deep parameter optimisation [31], might be used to automatically revisit design choices, e.g. in the light of a new use case. In [32] we demonstrated re-tuning legacy code for six different nVidia GPUs covering several generations of their architecture. As well as the comprehensive survey in [11, 12] describes several recent demonstrations in which GI was applied to Bioinformatics tools and other software.

## Method

### Programs, DNA sequences and parallel operation under multi-core Unix

#### `bwa` 0.7.12

`bwa` [17] (0.7.12-r1039 (https://github.com/lh3/bwa/archive/0.7.12.tar.gz)) was down loaded from GitHub and compiled with default settings (i.e. including support for multi-threading).

#### BarraCUDA 0.6.2

For comparison, the previous version of BarraCUDA, i.e. 0.6.2, was compiled with default settings (i.e. again including support for multi-threading).

#### BarraCUDA 0.7.107

BarraCUDA (0.7.107), was down loaded from SourceForge (http://sourceforge.net/projects/seqbarracuda/files/latest/download). Again it was built with default setting (including support for multi-threading). However a second version was built specifically for the GT 730 which was compiled with `-arch 2.1` to support compute level 2.1 (the default is now 3.5 or higher).

#### Reference Genome: UCSC HG19 ucsc.hg19.fasta.gz

The reference human genome [33] was downloaded from the Broad Institute's GATK resource bundle (version 2.8/hg19). It was converted into two indexes. BarraCUDA 0.6.2 converted ucsc.hg19.fasta.gz into an index for itself. Secondly BarraCUDA 0.7.0 converted it into an index for itself and for `bwa`.

#### 36 base pairs: 1000 Genomes project

One of The 1000 Genomes Project [34]'s normal (i.e. not color space encoded) paired end data with 36 DNA bases per end was chosen at random (ERR001270[2]). It contains 14 102 867 sequences. Approximately 5.7% of sequences occur more than once.

#### 100 base pairs: GCAT Benchmark

We used BioPlanet.com's GCAT [15] 100 bp-pe-small-indel alignment benchmark (gcat_set_037, available via http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/gp-code/www.bioplanet.com/gcat/gcat_set_037). It contains 5 972 625 paired end (100 base) sequences. (Less than 0.1% of sequences were repeated.) (We have compared other nextgen tools on GCAT for this journal [35].)

**Example 1** *Example bash command line using process substitution, pipes and input-output redirection to run two "aln" processes (one per paired end) in parallel with "sampe", thus avoiding use of intermediate disk files.*

```
$exe1 sampe -t 24 $hg19           \
    <($exe1 aln -C 0 $hg19 $seq1)     \
    <($exe1 aln -C 1 $hg19 $seq2) $seq1 $seq2 \
  > $sam
```

`$exe1`, `$hg19`, `$seq1`, `$seq2` *and* `$sam` *are the names of bash environment variables.* `$exe1` *is the program, (i.e.* `bwa`*, BarraCUDA 0.6.2 or BarraCUDA 0.7.107),* `$hg19` *is the location of the reference genome index,* `$seq1` *and* `$seq2` *are the files holding the pairs of DNA sequences and* `$sam` *is the output. See also Fig.* 2*.*

## Results

`bwa`, the original BarraCUDA (i.e. version 0.6.2) and the GI version of BarraCUDA (i.e. 0.7.107) were each run five times on both the fourteen million real world paired end DNA sequences from The 1000 Genomes Project ("36 base pairs: 1000 Genomes project" section) and the almost six million GCAT paired end DNA sequences (Section "100 base pairs: GCAT Benchmark"). `bwa` was run on 12 core 2.60 GHz servers (see Table 2) whilst BarraCUDA was run on three GPUs, stretching from £50 low end GT 730 to the top of the range K80 Tesla (see Table 1). The results are summarised in Table 3.

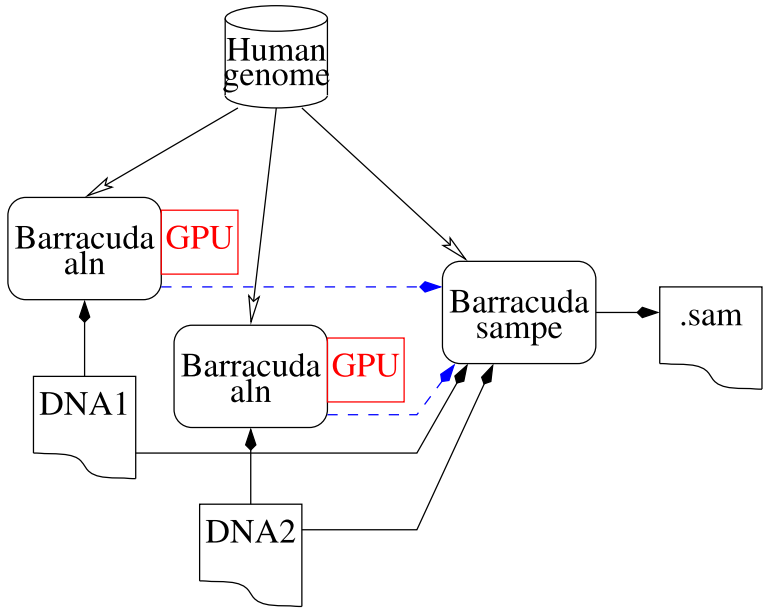


**Fig. 2** Processing paired end DNA sequences. "aln" is run two times (once per end), potentially in parallel, and its alignments are piped or passed via intermediate .sai files (*dashed blue arrows*) into "sampe" (sam (pe) paired end). "sampe" may be run in parallel. It also reads the index of the reference human genome and both ends of each DNA sequence in order to give the combined alignment in sam format. In the case of BarraCUDA, the two "aln" process each use a GPU and "sampe" uses multiple host threads. For `bwa` "aln" uses multiple host threads but "sampe" is single threaded

Apart from the low end GT 730, BarraCUDA is typically between two and ten times faster than `bwa` on a 12 core compute server. Table 3 shows the new version of BarraCUDA is up to three times faster than BarraCUDA 0.6.2 on the real world DNA sequences (36 bp) and typically about 10% faster on the longer benchmark strings (100 bp).

`bwa` "aln" (via pthreads) uses multiple threads. However `bwa` "sampe" does not support multiple host threads. This may explain why `bwa` performs relatively badly on the short 36 bp 1000 Genomes Project data.

We have used large real world and benchmark sequences. However both `bwa` and BarraCUDA are sensitive not only to the length of the DNA sequences but also how noisy they are. Resolving ambiguous matches caused by noise slows them down.

## Conclusions

Depending upon examples, even a £50 GPU running BarraCUDA can be faster than `bwa` on a twelve core 2.60 GHz server. With a top end nVidia Tesla GPU, BarraCUDA can be more than ten times faster than `bwa` on a 12 core server.

**Table 2** Computers. The desktop computer houses one GT 730. The servers are part of the Darwin Supercomputer of the University of Cambridge and hold multiple Tesla K20 or K80 GPUs

| Type | Intel x86 | Effective cores | Clock | Memory |
|---|---|---|---|---|
| Desktop | Core$^{TM}$2 CPU 6700 | 2 | 2.66 GHz | 4 GB |
| Darwin | Xeon CPU E5-2630 v2 | 12 | 2.60 GHz | 62 GB |
| NVK80 | Xeon CPU E5-2670 v3 | 24 | 2.30 GHz | 125 GB |

**Table 3** Mean number of paired end sequences processed per second

| Prog | length | 12 core server[a] | GT 730[b] | 2 × K20 | K80 | | Accuracy % |
|---|---|---|---|---|---|---|---|
| `bwa` | 36 bp | 1900 ± 50 | – | – | – | Mapped reads | 82.05 |
| `bwa` | 100 bp | 4500 ± 20 | – | – | – | GCAT | 98.91 |
| 0.6.2 | 36 bp | – | 3270 ± 2 (1.7 ± 0.05) | 5300 ± 110 (2.8 ± 0.10) | 6500 ± 180 (3.4 ± 0.13) | Mapped reads | 83.17 |
| 0.6.2 | 100 bp | – | 1860 ± 4 (0.4 ± 0.002) | 8700 ± 140 (1.9 ± 0.03) | 11700 ± 100 (2.6 ± 0.02) | GCAT | 97.49 |
| 0.7.107 | 36 bp | – | 7600 ± 6 (4.0 ± 0.11) | 12900 ± 160 (6.8 ± 0.20) | 19900 ± 500 (10.5 ± 0.39) | Mapped reads | 83.01 |
| 0.7.107 | 100 bp | – | 2100 ± 14 (0.5 ± 0.004) | 8800 ± 70 (2.0 ± 0.02) | 12800 ± 270 (2.8 ± 0.06) | GCAT | 98.43 |
| | | | | Improvement ratio Barracuda 0.7.107 over 0.6.2 | | | |
| | 36 bp | – | 2.32 ± 0.003 | 2.43 ± 0.06 | 3.07 ± 0.11 | Mapped reads | –0.16 |
| | 100 bp | – | 1.13 ± 0.01 | 1.00 ± 0.02 | 1.09 ± 0.02 | GCAT | 1.60 |

In (brackets) speed relative to `bwa` 0.7.12. ± gives standard deviation estimated from five runs. There was almost no variation in mapping rate or accuracy reported by GCAT

[a]2.60GHz, see "Darwin" in Table 2

[b]Estimated for two GT 730 GPUs

## Endnotes

[1] Arioc's index considerably exceeds current GPU memories and care must be taken to try and limit the volume of re-loaded data.

[2] ERR001270 is available from The 1000 Genome Project's FTP site. Additionally a copy can be down loaded from http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/gp-code/barracuda_0.7.105/1000.

**Abbreviations**

aln: Alignment process used by both `bwa` and BarraCUDA; bp: Base pair; BWA: Burrows-Wheeler algorithm; BarraCUDA: A parallel computer program which uses nVidia GPUs for aligning DNA particularly sequences produced by short next generation DNA scanners against a reference genome, particularly the human genome; `bwa`: A program for aligning DNA particularly sequences produced by short next generation DNA scanners against a reference genome, particularly the human genome; C: A computer programming language; C++: An object orientated computer programming language based on C; CUDA: Computer uniform device architecture. A set of software tools to enable programs (particularly non-gaming software) to run on nVidia GPUs; GATK: The Genome analysis toolkit suite of computer software written by the Broad Institute; GB: Gigabyte 1 073 741 824 bytes; GCAT: A DNA alignment benchmark; GHz: Gigahertz, $10^9$ clock ticks per second; GI: genetic improvement of software; GPU: Graphics processing unit; GPGPU: General purpose computing on GPUs; GT 730: A domestic gaming GPU designed by nVidia; indel: A DNA mutation which inserts or deletes DNA bases, so changing the length of the DNA sequence; K20, K80: Types of computational accelerators designed by nVidia based on their parallel graphics cards (GPUs); nextgen: Next generation (see NGS); NGS: Next generation DNA sequence; SAM: Sequence alignment/map computer file format; sampe: A software process which resolves pairs of paired end DNA alignments produced by earlier "aln" processes and converts them into SAM text file format

**Acknowledgments**

I am grateful for the assistance of the reviewers, Gareth Highnam of bioplanet.com, Filippo Spiga, Stuart Rankin, Timothy Lanfear, Neil Daeche, Dave Twisleton, John Andrews, Tristan Clark and Justyna Petke.
K20 tesla donated by nVidia (http://www.nvidia.com). K80 runs used the Darwin Supercomputer of the University of Cambridge High Performance Computing Service (http://www.hpc.cam.ac.uk/).

**Funding**

This work was funded by the EPSRC (grant EP/M025853/1). The authors are wholly responsible for all aspects of the study and the manuscript.

**Availability of data and materials**

Please contact author for data requests.

**Authors' contributions**

The authors contributed equally. Both authors read and approved the final manuscript.

**Ethics approval and consent to participate**

Not applicable.

**Consent for publication**

Not applicable.

**Competing interests**

The authors declare that they have no competing interests.

## Publisher's Note



**Author details**

[1] Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK. [2] University of Cambridge Metabolic Research Laboratories, Addenbrooke's Hospital, Cambridge, UK.

Received: 14 May 2017 Accepted: 24 July 2017
Published online: 02 August 2017

**References**

1. Owens JD, Houston M, Luebke D, Green S, Stone JE, Phillips JC. GPU Computing. Proc IEEE. 2008;96(5):879–99. http://dx.doi.org/doi:10.1109/JPROC.2008.917757. Invited paper.
2. Klus P, Lam S, Lyberg D, Cheung MS, Pullan G, McFarlane I, Yeo GSH, Lam BYH. BarraCUDA - a fast short read sequence aligner using graphics processing units. BMC Res Notes. 2012;5(27):. http://dx.doi.org/doi:10.1186/1756-0500-5-27.
3. Langdon WB, Lam BYH, Petke J, Harman M. Improving CUDA DNA Analysis Software with Genetic Programming In: Silva S, Esparcia-Alcazar AI, Lopez-Ibanez M, Mostaghim S, Timmis J, Zarges C, Correia L, Soule T, Giacobini M, Urbanowicz R, Akimoto Y, Glasmachers T, Fernandez de Vega F, Hoover A, Larranaga P, Soto M, Cotta C, Pereira FB, Handl J, Koutnik J, Gaspar-Cunha A, Trautmann H, Mouret JB, Risi S, Costa E, Schuetze O, Krawiec K, Moraglio

A, Miller JF, Widera P, Cagnoni S, Merelo J, Hart E, Trujillo L, Kessentini M, Ochoa G, Chicano F, Doerr C, editors. GECCO '15: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. Madrid: ACM; 2015. p. 1063–70. http://dx.doi.org/doi:10.1145/2739480.2754652.
4. Koza JR. Genetic Programming: On the Programming of Computers by Natural Selection: MIT press; 1992.
5. Poli R, Langdon WB, McPhee NF. A field guide to genetic programming; 2008. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk. (With contributions by J. R. Koza).
6. Langdon WB, Harman M. Optimising Existing Software with Genetic Programming. IEEE Trans Evol Comput. 2015;19:118–35. http://dx.doi.org/doi:10.1109/TEVC.2013.2281544.
7. Petke J, Harman M, Langdon WB, Weimer W. Specialising Software for Different Downstream Applications Using Genetic Improvement and Code Transplantation. IEEE Trans Softw Eng. http://dx.doi.org/doi:10.1109/TEVC.2017.2693219.
8. Langdon WB. Genetic Improvement of Programs In: Matousek R, editor. 18th International Conference on Soft Computing, MENDEL 2012. 2nd edition. Brno, Czech Republic: Brno University of Technology; 2012. http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/Langdon_2012_mendel.pdf. Invited keynote.
9. Jia Y, Harman M, Langdon WB, Marginean A. Grow and Serve: Growing Django Citation Services Using SBSE In: Yoo S, Minku L, editors. SSBSE 2015 Challenge Track, Volume 9275 of LNCS. Bergamo; 2015. p. 269–75. http://dx.doi.org/doi:10.1007/978-3-319-22183-0_22.
10. Langdon WB. Genetically Improved Software In: Gandomi AH, Alavi AH, Ryan C, editors. Handbook of Genetic Programming Applications. Springer; 2015. p. 181–220. http://dx.doi.org/doi:10.1007/978-3-319-20883-1_8.
11. Langdon WB, Lam BYH, Modat M, Petke J, Harman M. Genetic Improvement of GPU Software. Genet Program Evolvable Mach. 2017;18:5–44. http://dx.doi.org/doi:10.1007/s10710-016-9273-9.
12. Petke J, Haraldsson SO, Harman M, Langdon WB, White DR, Woodward JR. Genetic Improvement of Software: a Comprehensive Survey. EEE Trans Evol Comput. http://dx.doi.org/doi:10.1109/TEVC.2017.2693219. in press.
13. Langdon WB, Vilella A, Lam BYH, Petke J, Harman M. Benchmarking Genetically Improved BarraCUDA on Epigenetic Methylation NGS datasets and nVidia GPUs In: Petke J, Weimer W, White DR, editors. Genetic Improvement 2016 Workshop. Denver: ACM; 2016. p. 1131–32. http://dx.doi.org/doi:10.1145/2908961.2931687.
14. Langdon WB. Mycoplasma Contamination in The 1000 Genomes Project. BioData Min. 2014; 7(3). http://dx.doi.org/doi:10.1186/1756-0381-7-3.
15. Highnam G, Wang JJ, Kusler D, Zook J, Vijayan V, Leibovich N, Mittelman D. An analytical framework for optimizing variant discovery from personal genomes. Nat Commun. 2015; 6(6275). http://dx.doi.org/doi:10.1038/ncomms7275.
16. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012;9(4):357–9. http://dx.doi.org/doi:10.1038/nmeth.1923.
17. Li H, Durbin R. Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics. 2010;26(5):589–95. http://dx.doi.org/doi:10.1093/bioinformatics/btp698.
18. Lenis J, Senar MA. Euro-Par 2016: Parallel Processing Workshops, Volume 10104 of Lecture Notes in Computer Science In: Desprez F, Dutot PF, Kaklamanis C, Marchal L, Molitorisz K, Ricci L, Scarano V, Vega-Rodríguez MA, Varbanescu AL, Hunold S, Scott SL, Lankes S, Weidendorfer J, editors. Grenoble: Springer; 2016. p. 492–503. http://dx.doi.org/doi:10.1007/978-3-319-58943-5_40. Revised Selected Papers, published 2017.
19. Marco-Sola S, Sammeth M, Guigo R, Ribeca P. The GEM mapper: fast, accurate and versatile alignment by filtration. Nat Methods. 2012;9(12):1185–8. http://dx.doi.org/doi:10.1038/nmeth.2221.
20. Zaharia M, Bolosky WJ, Curtis K, Fox A, Patterson DA, Shenker S, Stoica I, Karp RM, Sittler T. Faster and More Accurate Sequence Alignment with SNAP. 2011. https://arxiv.org/abs/1111.5572v1.
21. Luo R, Cheung J, Wu E, Wang H, Chan SH, Law WC, He G, Yu C, Liu CM, Zhou D, Li Y, Li R, Wang J, Zhu X, Peng S, Lam TW. MICA: A fast short-read aligner that takes full advantage of Many Integrated Core Architecture (MIC). BMC Bioinforma. 2015;16(Supplement 7):S10. http://dx.doi.org/doi:10.1186/1471-2105-16-S7-S10. Selected articles from The 11th Annual Biotechnology and Bioinformatics Symposium (BIOT-2014): Bioinformatics.
22. Luo R, Wong T, Zhu J, Liu CM, Zhu X, Wu E, Lee LK, Lin H, Zhu W, Cheung DW, Ting HF, Yiu SM, Peng S, Yu C, Li Y, Li R, Lam TW. SOAP3-dp: Fast, Accurate and Sensitive GPU-Based Short Read Aligner. PLoS ONE. 2013;8(5):e65632. http://dx.doi.org/doi:10.1371/journal.pone.0065632.
23. Liu CM, Wong T, Wu E, Luo R, Yiu SM, Li Y, Wang B, Yu C, Chu X, Zhao K, Li R, Lam TW. SOAP3: ultra-fast GPU-based parallel alignment tool for short reads. Bioinformatics. 2012;28(6):878–9. http://dx.doi.org/doi:10.1093/bioinformatics/bts061.
24. NVIDIA GeForce GTX 680, The fastest, most efficient GPU ever built. Tech. Rep. V1.0, nVidia. 2012. http://la.nvidia.com/content/PDF/product-specifications/GeForce_GTX_680_Whitepaper_FINAL.pdf. Technology Overview. Accessed 18 June 2017.
25. Mu JC, Jiang H, Kiani A, Mohiyuddin M, Asadi NB, Wong WH. Fast and accurate read alignment for resequencing. Bioinformatics. 2012;28(18):2366–73. http://dx.doi.org/10.1093/bioinformatics/bts450.
26. Liu Y, Schmidt B. Long read alignment based on maximal exact match seeds. Bioinformatics. 2012;28(18):i318–24. http://dx.doi.org/doi:10.1093/bioinformatics/bts414. ECCB 2012.
27. Wilton R, Budavari T, Langmead B, Wheelan SJ, Salzberg SL, Szalay AS. Arioc: high-throughput read alignment with GPU-accelerated exploration of the seed-and-extend search space. PeerJ. 2015;3:e808. http://dx.doi.org/doi:10.7717/peerj.808.
28. Langdon WB, Harman M. Grow and Graft a better CUDA pknotsRG for RNA pseudoknot free energy calculation In: Langdon WB, Petke J, White DR, editors. Genetic Improvement 2015 Workshop. Madrid: ACM; 2015. p. 805–10. http://dx.doi.org/doi:10.1145/2739482.2768418.
29. Williams KP, Williams SA. Genetic compilers: A new technique for automatic parallelisation. L'Alpe d'Hoez; 1996. p. 27–30. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.3499.
30. Hoos HH. Programming by optimization. Commun ACM. 2012;55(2):70–80. http://dx.doi.org/doi:10.1145/2739480.2754648.
31. Wu F, Weimer W, Harman M, Jia Y, Krinke J. Deep Parameter Optimisation In: Silva S, Esparcia-Alcazar AI, Lopez-Ibanez M, Mostaghim S, Timmis J, Zarges C, Correia L, Soule T, Giacobini M, Urbanowicz R, Akimoto Y,

Glasmachers T, Fernandez de Vega F, Hoover A, Larranaga P, Soto M, Cotta C, Pereira FB, Handl J, Koutnik J, Gaspar-Cunha A, Trautmann H, Mouret JB, Risi S, Costa E, Schuetze O, Krawiec K, Moraglio A, Miller JF, Widera P, Cagnoni S, Merelo J, Hart E, Trujillo L, Kessentini M, Ochoa G, Chicano F, Doerr C, editors. GECCO '15: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. Madrid: ACM; 2015. p. 1375–82. http://dx.doi.org/doi:10.1145/2739480.2754648.

32. Langdon WB, Harman M. Genetically Improved CUDA C++ Software In: Nicolau M, Krawiec K, Heywood MI, Castelli M, Garcia-Sanchez P, Merelo JJ, Rivas Santos VM, Sim K, editors. 17th European Conference on Genetic Programming, Volume 8599 of LNCS. Granada: Springer; 2014. p. 87–99. http://dx.doi.org/doi:10.1007/978-3-662-44303-3_8.
33. International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. Nature. 2001;409(6822):860–921. http://dx.doi.org/doi:10.1038/35057062.
34. Durbin RM, et al. A map of human genome variation from population-scale sequencing. Nature. 2010;467(7319):1061–73. http://dx.doi.org/doi:10.1038/nature09534.
35. Langdon WB. Performance of Genetic Programming Optimised Bowtie2 on Genome Comparison and Analytic Testing (GCAT) Benchmarks. BioData Min. 2015;8. http://dx.doi.org/doi:10.1186/s13040-014-0034-0.
36. Langdon WB, Lam BYH. Genetically Improved BarraCUDA. Research Note RN/15/03, Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK; 2015. http://arxiv.org/abs/arXiv:1505.07855.