UCL DEPARTMENT OF
COMPUTER SCIENCE

Research Note

RN/15/03

Genetically Improved BarraCUDA

28 May 2015

W. B. Langdon and Brian Yee Hong Lam

Abstract

BarraCUDA is a C program which uses the BWA algorithm in parallel with nVidia CUDA to align short next generation DNA sequences against a reference genome. 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 CPU.

1 Why Run Bioinformatics on Gaming Machines

The explosive growth in Biological datasets has coincided with a similar exponential increase in computer processing power (known as Moore's Law [1]). Before 2005 the doubling of integrated circuit complexity every 18 months, went hand-in-hand with doubling of computer processor clock speeds. However in the last ten years clock speeds have increased little. This has not (as yet) limited the exponential growth in Bioinformatics datasets and hence processing demand. Fortunately Moore's Law has continued to apply to the number of transistors per silicon chip. Whilst some of these extra transistors have been used to support more powerful computer instructions, largely they have been and will continue to be used to support parallel computing. In 2005 a typical computer contain one CPU, nowadays quad code (i.e. 4 CPUs) are common place with 6, 8 and 12 cores also being available. This trend will continue.

Modern consumer applications demand high quality and instant response. With user interfaces containing millions of display elements (pixels) and thousands of input sensors, the only practical approach has been parallel processing. Rather than using several CPUs, hardware dedicated to graphical displays typically contains hundreds or even thousands of processing elements. As each each pixel is processed in the same way, the graphics processing units (GPUs) can take short cuts in the hardware. For example, since each of the hundreds of pixel processing programs is doing exactly the same thing, the logic to decode program instructions can be shared. This means the transistors used to decode the program actually drive many streaming processing cores (rather than just one). The research and development of these specialised but highly parallel graphics accelerator cards has been paid for largely by the consumer gaming market. One of the main players in this market is nVidia. They have sold hundreds of millions of their GPUs. (These GPUs are capable of running CUDA, which is nVidia's general purpose framework for programming their GPUs. It is used by BarraCUDA.)

About the time of the end of the serial processor clock speed boom, computer scientists and engineers started to treat GPUs as low cost but highly parallel computers and started using their GPUs for general purpose computing (GPGPUs [2]). This trend continues. Indeed GPGPU has been combined with some enormous volunteer user cloud systems. For example, much of the raw computer power used by the SETI@HOME project is actually derived from GPUs within domestic PCs.

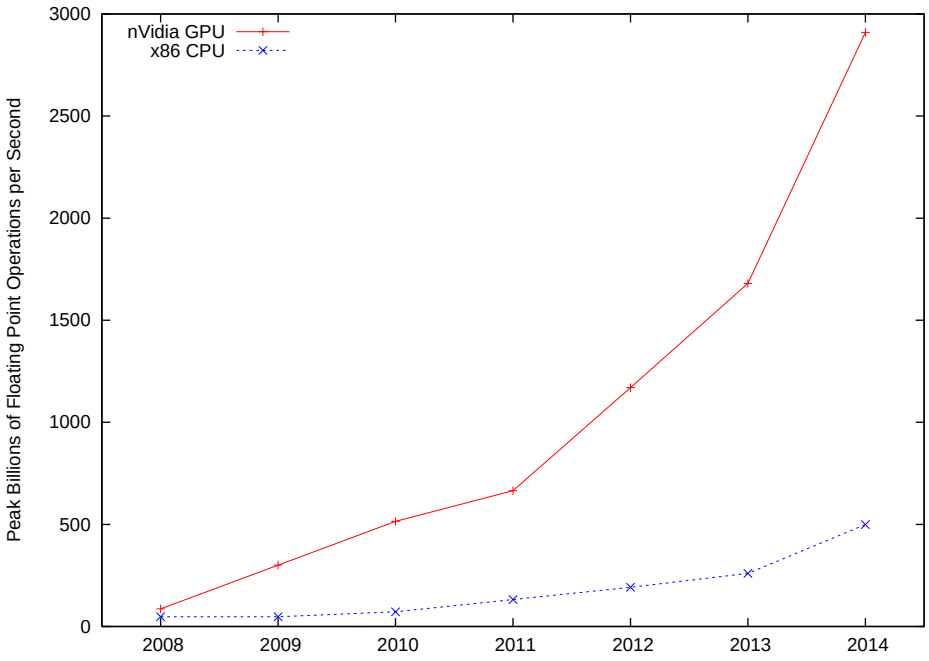

Figure 1: Exponential growth in *peak* processing power. Data from nVidia

Another aspect of GPGPU, has been the introduction by both nVidia and Intel of "screen less" GPUs, where the hardware is dedicate to computing applications rather than computer graphics. Indeed today half of the ten fastest computers on the planet are based on GPUs (http://www.top500.org/ May 2015).

Bioinformaticians have not been slow in seizing the advantages of GPGPU programming. CUDA versions of several popular applications have been written. However, as with other branches of super computing, it is often not easy to write code to gain the best of parallel machines. For example, often parallel applications are limited not by the processing power available but by the time taken to move data inside the computer to the processing elements.

At the forthcoming GECCO conference [3] we shall present an approach in which a small part of the manually written code has 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 will describe the target system, BarraCUDA [6]. Section 3 gives details of the programs and DNA benchmarks. In particular the standard GCAT Bioinformatics DNA sequence alignment benchmarks [7] and short human paired end next generation DNA sequences taken from The 1000 Genomes Project [8]. This is followed (Section 4) by the overall performance changes genetic improvement [9, 10, 11, 12, 13, 14] gives and comparison with `bwa`. (See particularly Table 3, page 6.)

2 NextGen DNA Sequence Alignment

Since the human genome was sequenced in 2000 [15], increasingly powerful nextGeneration sequencing machines have generated vast volumes of short noisy DNA sequences. Initially sequences where only 30 or so bases long. (The sequences are stored as strings of the four letters A, C, G and T. Each character representing one base.) Where the genetic sequence is variable, simple statistics (i.e., $4^{30} \gg$ length of the genome) that suggest 30 or so bases would be sufficient to identify where the sequence lies in the reference genome. Whilst these data are inevitably noisy, the main difficulty with this approach is that (in particular) the Human genome contains many repeated sequences. Thus sometimes 4^{30} can only identify the repeated pattern not the location itself. This lead to 1) longer sequences but also 2) sequencing both ends of much longer sequences. The second (paired end) approach requires more sophisticated computer algorithms. Each end is matched against the reference genome as before. When an end lies in a repeated sequence, and so gives multiple match points, the matches the other end gives are consulted. As the approximate length of the DNA sequence is known (or can be inferred), in many cases potential matches for the two ends can be ignored as they are simply too far apart. Paired end analysis is now typical. Whilst Barracuda can deal both with single ended and paired end DNA sequences, we shall only benchmark paired end data.

BarraCUDA uses the Burrows-Wheeler algorithm (BWA) [16]. Indeed it source code is derived from a serial implementation of the BWA algorithm, simply called `bwa`. Barracuda gets its speed by using a GPU to processing hundreds of thousands of short DNA sequences in parallel. Typically finding where each DNA sequences matches the reference human genome is the most time consuming part. With paired end (pe) data, Barracuda "aln" matches each end separately and then Barracuda "sampe" combines them. Thus Barracuda sampe does not need a GPU (although it, unlike `bwa` sampe, can exploit multiple CPU cores).

With noise free data and where the DNA sequence matches the reference genome exactly, the Burrows-Wheeler algorithm is relatively straight forward. Before hand, offline, the reference genome is encoded into a compressed format so that all the sequences in the reference genome with the same starting subsequences are given the same location in the compressed file. Since there are four possible bases, extending the prefix sequence by one means this location leads to four subsequences prefix strings which are one base longer. However as the reference genome is finite, the branching factor quickly falls from four to one. If a prefix sequence can be followed by exactly one prefix which is one base longer, this means all the sequences with this particular prefix have the same base in the next position. The index file is arranged to enable rapid sequence look up. Barracuda and `bwa` index files are interchangeable.

Table 1: GPU Hardware. Year each was announced by nVidia in column 2. Price (column 3) is either actual (GT 730) or on line quote (May 2015, which may be lower than original list price). Fourth column is CUDA compute capability level (as can be used with the nvcc compiler's `-arch` parameter). Each GPU chip contains 2, 13 or 15 identical independent multiprocessors (MP, column 5). Each MP contains 48 or 192 stream processors (total given in column 7) whose clock speed is given in column 8. Onboard memory size and bandwidth are given in the right most two columns. ECC enabled.

GPU		compute level		MP	total cores	Clock	L1/L2 caches		Memory	
GT 730	2014	£53.89	2.1	$2 \times 48 =$	96	1.40 GHz	48KB	0.125 MB	4 GB	23 GB/s
Tesla K20	2012	£2,905.20	3.5	$13 \times 192 =$	2496	0.71 GHz	48KB	1.25 MB	5 GB	140 GB/s
Tesla K40	2013	£3,264.83	3.5	$15 \times 192 =$	2880	0.88 GHz	48KB	1.50 MB	11 GB	180 GB/s
Tesla K80[1]	2014	£6,260.65	3.7	$13 \times 192 =$	2496	0.82 GHz	48KB	1.50 MB	11 GB	138 GB/s

K80 is a dual GPU, performance figures given for one half.

Table 2: CPUs. 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	Cores	Clock	Memory
Desktop	2	2.66 GHz	4 GB
Darwin	12	2.60 GHz	62 GB
NVK80	24	2.30 GHz	125 GB

On look up, an upper and a lower pointer into the index file are kept. They span all possible matches, and so are initially far apart. As each base in the DNA sequence is processed, data are read from the index file and the position of the two pointers are updated. Actually the distance between them is the number of positions in the reference genome which match the DNA sequence processed so far. If the distance becomes one, then there is a unique match. If the two pointers cross this means the sequence does not exist in the reference genome. In good quality data from the 1000 Genomes Project, about 85% of sequences match uniquely. Where sequences do not match, this may be either due to noise in the data or to real mutations in the patient. To cope with non-exact matches, the algorithm must carefully back up its search and start trying out alternatives. This slows things down considerably.

For the approach to be feasible, Barracuda must load the whole of the index file into the GPU's memory. Thus the GPU must have enough memory to hold it all. For the human reference genome, this means the GPU must have at least four gigabytes of on-board RAM. Also the Burrows-Wheeler algorithm does not allow short cuts. I.e., every base in the sequence must be processed. Thus, even before considering mismatches, Barracuda must make heavy access to the index. Fortunately modern GPUs have 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.

Taking The 1000 Genomes Project as an example, [17, Fig. 4] shows some sequence lengths are much more common than others. In Section 4 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, which is almost the same as one of the benchmarks provided by BioPlanet's Genome Comparison and Analytic Testing (GCAT) platform [7].

3 Programs, DNA sequences and Parallel Operation under Multi-core Unix

3.1 `bwa` *0.7.12*

The current release of `bwa` (May 2015, Version: 0.7.12-r1039), i.e. bwa-0.7.12.tar.gz, was down loaded from GitHub and compiled with default settings (i.e. including support for multi-threading).

3.2 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).

3.3 Barracuda 0.7.107

The current release (May 2015, Version 0.7.0r107), bwa-0.7.12.tar.gz, was down loaded from SourceForge. 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, cf. column 4 in Table 1. (The default is now compute level 3.5 or higher).

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

Although GCAT provides a pointer (http://hgdownload.cse.ucsc.edu/downloads.html#human) to UCSC, the reference human genome was downloaded from the Broad Institute's GATK resource bundle via FTP (approximately 900 megabytes compressed). It was converted into two indexes. Barracuda 0.6.2 converted ucsc.hg19.fasta.gz into an index for itself (4.4 GB). Secondly Barracuda 0.7.0 converted it into an index for itself and for `bwa` (5.1 GB).

3.5 36 base pairs: 1000 Genomes project

The 1000 Genomes Project [8] has made available a vast volume of data via its FTP site. One of its normal (i.e. not color space encoded) paired end data with 36 DNA bases per end was chosen at random (ERR001270) and downloaded in compressed form using wget. ERR001270 consists of two files (one per end of the DNA sequence) each containing 1.1 gigabytes (compressed). ERR001270 contains 14 102 867 36 base DNA sequences. Approximately 5.7% of sequences occur more than once in ERR001270. The files are in "fastq" format and so also contain quality values but these are not used by `bwa` or by either version of Barracuda.

Initially `bwa` objected to the sequence names provided by The 1000 Genome Project. However this was readily resolved so that each pair of sequences had its own unique name.

3.6 100 base pairs: GCAT Benchmark

BioPlanet.com hosts several sequence alignment and variant calling next generation DNA benchmarks on its GCAT web pages [7]. We report results on their 100bp-pe-small-indel alignment benchmark (gcat_set_037). This consists of two files (one per end) each of 3 gigabytes (uncompressed), each containing 5 972 625 100 base sequences. (Less than 0.1% of sequences were repeated.) The files are again in "fastq" format but only contain dummy quality values however again these are not used by `bwa` or by Barracuda.

3.7 Parallel operation on multi-core CPUs with bash pipes

Barracuda and `bwa` have similar operations and command lines. For paired end data, "aln" is run separately for each fastq sequence file (Sections 3.5 and 3.6 above). For every fastq sequence, "aln" produces zero or more possible alignments in the reference genome (Section 3.4) and saves them in a binary .sai file. (.sai

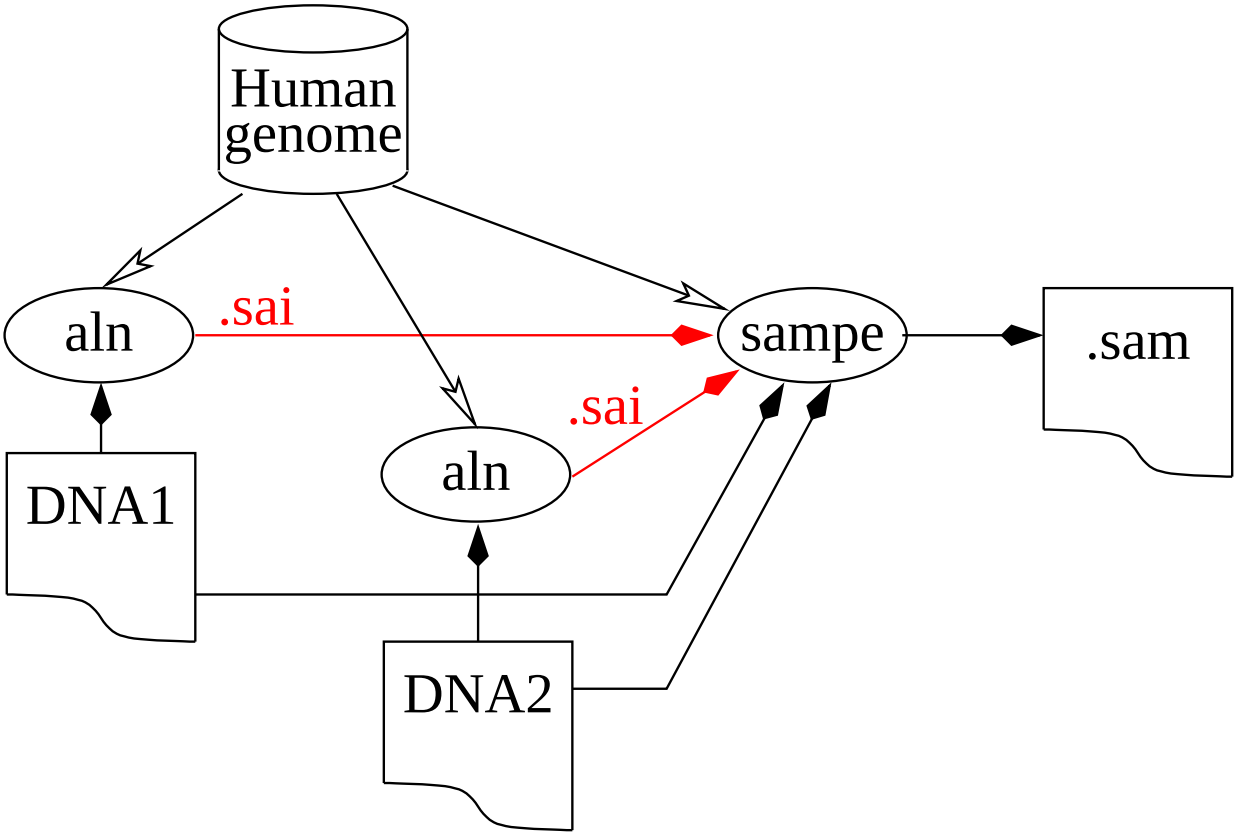

Figure 2: Processing paired end DNA sequences. "aln" is run twice (once per end) and its alignments are piped (red arrows) into "sampe" (sam (pe) paired end). "sampe" also reads the index of the reference human genome and both ends of each DNA sequence in order to give the combined alignment in sam format. 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.

files are generally not compatible between different versions). For each pair of sequences, "sampe" takes the alignments from the .sai files, the sequences themselves and the index file for the reference genome to produce alignments for each end in sam format.

Notice the .sai files are intermediate and can be deleted after the .sam file has been created. However the .sai files are large. (E.g. 830 mega bytes for Barracuda 0.6.2 on the 1000 Genomes Project example and 340 mega bytes for the GCAT example.) Since the .sai files are both written to and read sequentially, under Unix using bash, it is not necessary to explicitly store them. Instead "aln" can write them to a unix pipe and "sampe" can read them from those pipes (see Figures 2 and 3). With large memory multi-core CPUs it is quite feasible to run both "aln" processes and the "sampe" process in parallel (see Table 2).

The sam files are plain text and also large, nearly 6 GB for ERR001270 and well over 4 GB for the GCAT benchmark. GCAT uses the compressed binary format bam, even so gcat_set_037.bam is almost a gigabyte. gcat_set_037.sam was converted to gcat_set_037.bam by samtools. The time for this post processing (a few minutes) is not included in Table 3.

3.8 Problems and Work Arounds

A single GeForce GT 730 was available. It was mounted in a desktop linux PC with 4 GB of RAM. This was enough to run "aln" but not "sampe". (The .sai files could be transferred to a much larger linux server to run "sampe".) Typically in Barracuda "sampe" takes little time and on a typical large multi-core server can be run in parallel with the two "aln" processes with little impact on total wall clock time (see Figures 2 and 3). For ease of comparison the data in Table 3 are an estimate for two GT 730's mounted in a large multi-core CPU. They are calculated from the wall clock time of the slowest of the two .sai files.

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

Figure 3: Example bash command line using process substitution, pipes and input-output redirection to run two "aln" processes in parallel with "sampe", thus avoiding use of intermediate disk files. `$exe1`, `$hg19`, `$seq1`, `$seq2` and `$sam` are the names of bash environment variables. `$exe1` is the program, `$hg19` the location of the reference genome index, `$seq1` and `$seq2` are the files holding the pairs of DNA sequences and `$sam` is the output. See also Figure 2.

Table 3: Mean number of paired end sequences processed per second. In (brackets) speed relative to `bwa` 0.7.12. $\pm$ gives standard deviation estimated from five runs. There was almost no variation in accuracy reported by GCAT.

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

[a] 2.60GHz, see "Darwin" in Table 2
[b] Estimated for two GT 730 GPUs. See Section 3.8

Barracuda version 0.6.2 beta "sampe" failed on ERR001270 if run with multiple threads. The only work around was to used `-t 1` to prevent use of multiple threads. (Bug fixed before version 0.7.107.)

There is a small HTML web page which lists a few issues (some open but several now fixed) and common misunderstandings about Barracuda at http://www.cs.ucl.ac.uk/staff/W.Langdon/barracuda/.

4 Results

`bwa` and the original and the GI improved version of Barracuda were each run five times on both the fourteen million real world paired end DNA sequences from The 1000 Genomes Project (Section 3.5) and the almost six million paired end DNA sequences provided by GCAT as a benchmark (Section 3.6). `bwa` was run on 12 core 2.60 GHz CPUs (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.

Apart from the low end GT 730, Barracuda is typically between two and ten times faster than the current release of `bwa` on a 12 core CPU (see figures within round brackets in the upper part of Table 3). The lower part of Table 3 presents the Barracuda data in the upper part as ratios between the previous release of Barracuda (0.6.2) with the current genetically improved [3] version (0.7.107). Table 3 shows the newer version is up to three times faster on the real world DNA sequences (36bp) and typically about 10% faster on the longer benchmark strings (100bp).

5 Discussion

`bwa` "sampe" does not support multiple host threads. As shorter sequences are expected to give rise to more duplicate matches and so give "sampe" more work to do, this may explain why `bwa` performs relatively badly on the short 36 bp 1000 Genomes Project data (see row `bwa` 36bp in Table 3). On the 1000 Genomes Project example (36bp), even a lowly GT 730 running Barracuda 0.7.107 can beat `bwa` on at 12 core super computer node. On the longer GCAT benchmark (100bp), *one* GT 730 is about quarter of the speed of the 12 core computer. However both the pair of K20s and the K80 are faster than `bwa` on both real world and GCAT examples.

On the GCAT benchmark, the new version of Barracuda is more accurate than 0.6.2 and approaches the accuracy of `bwa` 0.7.12.

The variability of run time on the super computer (Table 3 columns 4, 10 and 14) is typical of use on shared disk systems. In contrast, typically elapse times of CUDA kernels are very consistent (cf. Table 3 column 6). We anticipate slightly higher and but more stable performance could be achieved on the super computer by placing files on local or ram disks. (Perhaps a couple of percent improvement may be obtained by using a ramdisk.) However ignoring the time to transfer data files within the super computer might give unrepresentative results.

6 Conclusions

The new version of Barracuda, particularly on large real world examples, is a substantial improvement. Details of the genetic improvement process used to both tune its parameters and code will shortly be presented at the GECCO conference [3]. Both the new version and the genetic improvement process are freely available. (The genetically improved version of BarraCUDA has been in use via SourceForge since 20 March 2015. The GI code may be down loaded from the author's FTP site from file gp-code/barracuda_gp.tar.gz.)

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

Acknowledgements

I am grateful for the assistance of Gareth Highnam of bioplanet.com, Filippo Spiga, Stuart Rankin, Timothy Lanfear, Neil Daeche, Dave Twisleton, John Andrews and Tristan Clark.

K20 and K40 tesla donated by nVidia. K80 runs used the Darwin Supercomputer of the University of Cambridge High Performance Computing Service.

References

[1] Moore, G.E.: Cramming more components onto integrated circuits. Electronics **38**(8) (1965) 114–117

[2] Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C.: GPU computing. Proceedings of the IEEE **96**(5) (2008) 879–899 Invited paper.

[3] Langdon, W.B., Lam, B.Y.H., Petke, J., Harman, M.: Improving CUDA DNA analysis software with genetic programming. In: GECCO '15: Proceeding of the Seventeenth annual conference on genetic and evolutionary computation conference, Madrid, ACM (2015) Forthcoming.

[4] Koza, J.R.: Genetic Programming: On the Programming of Computers by Natural Selection. MIT press (1992)

[5] Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk` (2008) (With contributions by J. R. Koza).

[6] Klus, P., Lam, S., Lyberg, D., Cheung, M.S., Pullan, G., McFarlane, I., Yeo, G.S.H., Lam, B.Y.H.: BarraCUDA - a fast short read sequence aligner using graphics processing units. BMC Research Notes **5**(27) (2012)

[7] Highnam, G., Wang, J.J., Kusler, D., Zook, J., Vijayan, V., Leibovich, N., Mittelman, D.: An analytical framework for optimizing variant discovery from personal genomes. Nature Communication **6**(6275) (2015)

[8] Durbin, R.M., et al.: A map of human genome variation from population-scale sequencing. Nature **467**(7319) (2010) 1061–1073

[9] Langdon, W.B., Harman, M.: Optimising existing software with genetic programming. IEEE Transactions on Evolutionary Computation **19**(1) (2015) 118–135

[10] Petke, J., Harman, M., Langdon, W.B., Weimer, W.: Using genetic improvement and code transplants to specialise a C++ program to a problem class. In Nicolau, M., Krawiec, K., Heywood, M.I., Castelli, M., Garcia-Sanchez, P., Merelo, J.J., Rivas Santos, V.M., Sim, K., eds.: 17th European Conference on Genetic Programming. Volume 8599 of LNCS., Granada, Spain, Springer (2014) 137–149

[11] Langdon, W.B.: Genetic improvement of programs. In Matousek, R., ed.: 18th International Conference on Soft Computing, MENDEL 2012, Brno, Czech Republic, Brno University of Technology (2012) Invited keynote.

[12] Jia, Y., Harman, M., Langdon, W.B., Marginean, A.: Grow and serve: Growing Django citation services using SBSE. In Yoo, S., Minku, L., eds.: SSBSE 2015 Challenge Track, Bergamo, Italy (2015) Forthcoming.

[13] Langdon, W.B., Petke, J., White, D.R.: Genetic improvement 2015 chairs' welcome. In Langdon, W.B., Petke, J., White, D.R., eds.: GECCO'15: 2015 Genetic and Evolutionary Computation Conference Companion Proceedings, Madrid, ACM (2015)

[14] Langdon, W.B.: Genetically improved software. In Gandomi, A.H., Alavi, A.H., Ryan, C., eds.: Handbook of Genetic Programming Applications. (Springer) Forthcoming.

[15] International Human Genome Sequencing Consortium: Initial sequencing and analysis of the human genome. Nature **409**(6822) (2001) 860–921

[16] Li, H., Durbin, R.: Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics **26**(5) (2010) 589–595

[17] Langdon, W.B.: Mycoplasma contamination in the 1000 genomes project. BioData Mining **7**(3) (2014).