CREST

Benchmarking Genetically Improved BarraCUDA on Epigenetic Methylation data and nVidia GPUs

WB Langdon A Vilella* BYH Lam† J Petke M Harman

University College London.

*Cambridge Epigenetix. †Addenbrooke's Hospital, Cambridge UK

WIKIPEDIA

Genetic Improvement

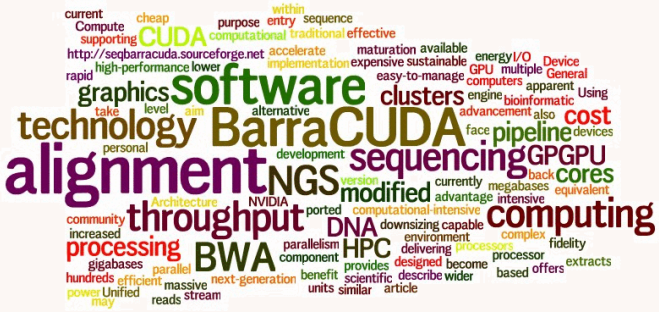
GENETIC IMPROVEMENT 2016

GI 2016 workshop at GECCO 2016
Denver, USA, Wednesday 20th July

2.8.2016

CREST

Genetic Improvement of Programs



- Background
  - What is BarraCUDA
  - Why interest in GPUs
  - DNA sequences & Epigenetic Methylation
  - How BarraCUDA was GIed
- Results
  - 100× speedup. GCAT benchmark (arXiv.org)
  - 1st GI in use.
    - Over 1000 sourceforge downloads in first year.
    - Commercial use by Lab7 (in BioBuilds since Nov2015) and IBM Power8.
    - Microsoft trialing BarraCUDA on Azure GPU cloud

What is BarraCUDA

- BarraCUDA is a Bioinformatics program to do approximate string matching.
- It aligns short noisy DNA sequences against a reference genome. Eg to say which human gene gave the sequence.
- Problems
  - Noise/real mutations: generate data 30x times
  - Genomes repetitive, no unique match: paired ends
  - Data volume (billions of sequences): parallel

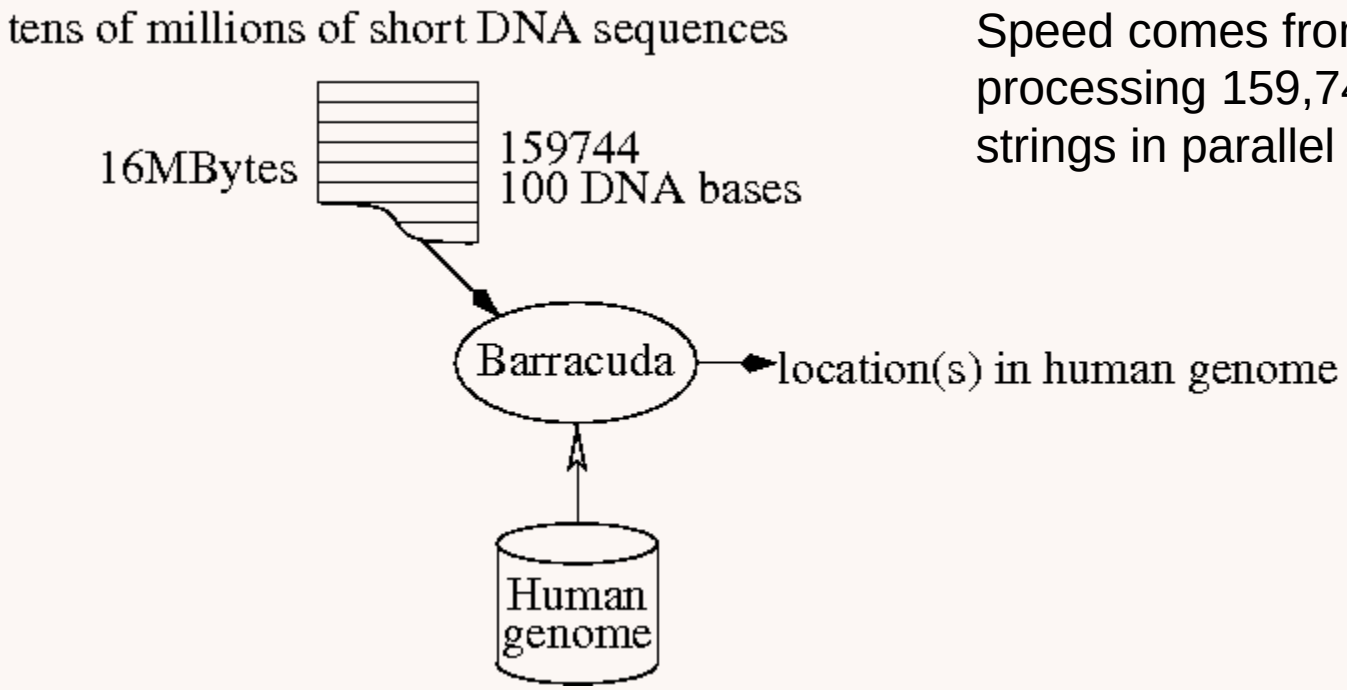
What is BarraCUDA

- BarraCUDA manual port of BWA to run BWA's search algorithm in parallel on nVidia GPUs
- Last year it was genetically improved.
- The GI version has been available for 16months. Down loaded 1,877
- Tuned for short next generation DNA sequences.
- Here used for epigenetics data

What is BarraCUDA

DNA analysis program

- 8000 lines C code, SourceForge.
- Rewrite of BWA for nVidia CUDA



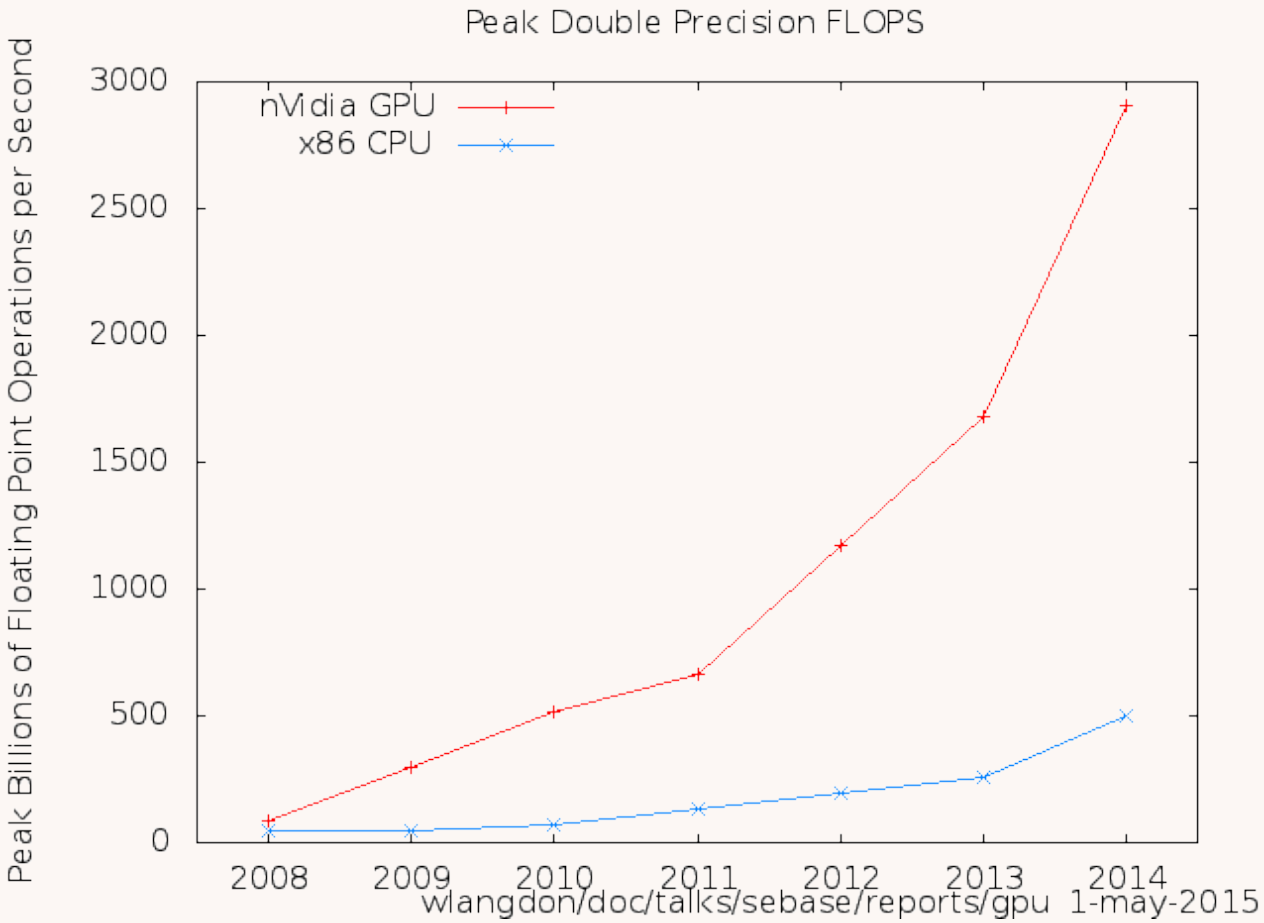
Speed comes from processing 159,744 strings in parallel on GPU

Why interest in GPUs

- From 1960 to 2005 CPU clock doubled in speed every 18 months

  (10^9 increase in a life time)
- If trend had continued, laptop 400Ghz.
- It has not happened. It will not happen.
- But Moore's law has continued. Doubling of transistors per chip has continued.
- Extra transistors have gone into parallel operations.
- Future is parallel

Why interest in GPUs



Data from nVidia

Graphics Cards

£53.85

Next Generation DNA Sequences

- NGS sequencing machines use four fluorescent dyes (one per DNA base) to read sequences of DNA bases.
- Very fast, billion sequences per day
- Noise. Four colours mix. Typically worse at end of sequence.
- Paired end to cope with short repeating sequences in human genome. Long (1500bp) molecule but only sequence ends (eg 100bp)

Epigenetic Methylation of DNA

- In Nature smallest base C can have addition methyl group CH_3 attached C*. Common in human. Still active research.
- Methylated DNA continues to “work” but CH_3 may disable some gene expression
- Chemistry to differentiate C/C* gives DNA sequences readable by NGS machines
- But software changes

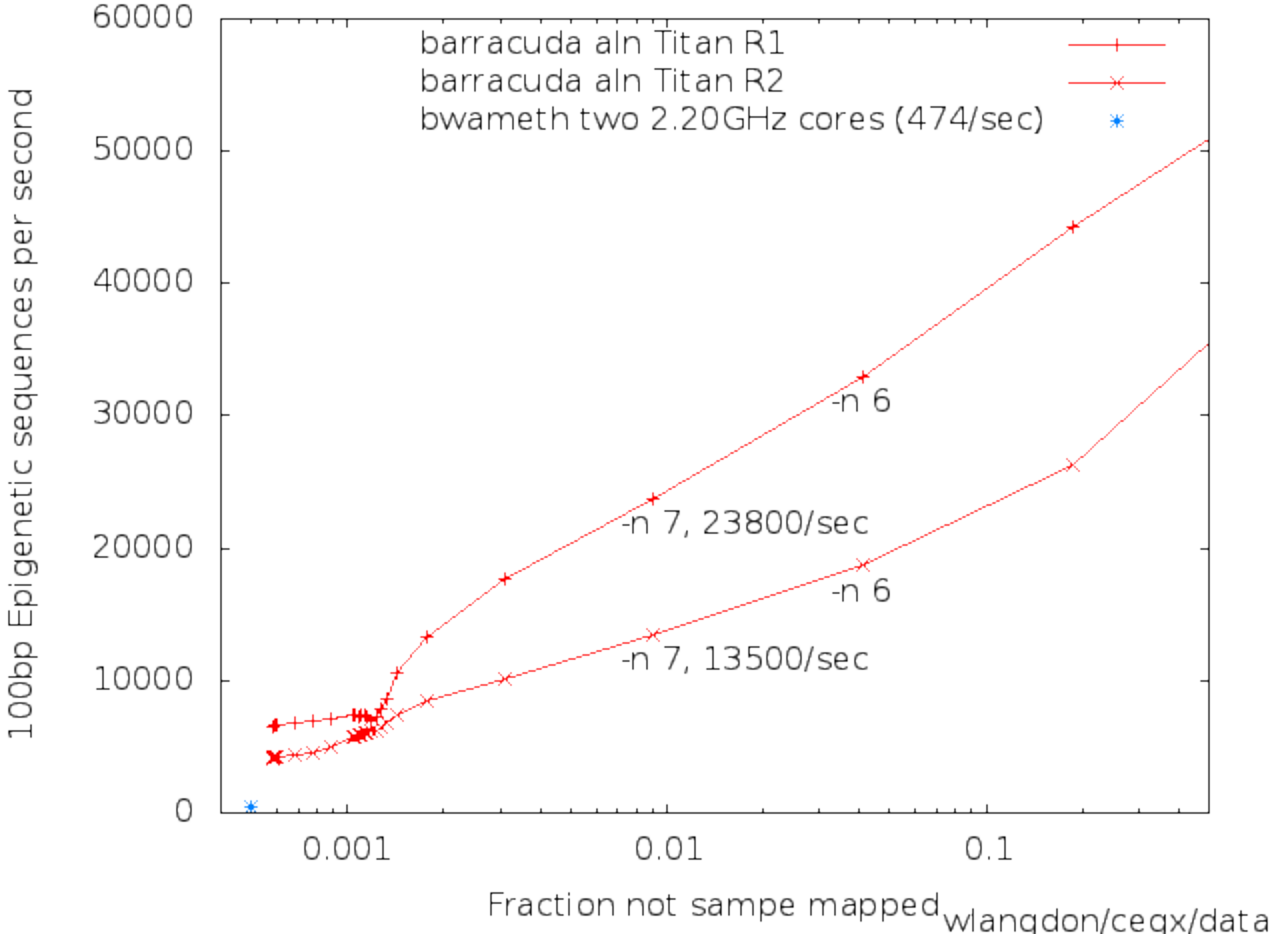
BarraCUDA for Epigenetic DNA

- Human genome 3.2 billion DNA bases. BWA/BarraCUDA compress reference genome into <4GBytes.
- Until recently largest GPUs had 4GB.So ok
- Epigenetics reference genome approx twice as big. Effectively storing twice, once for C and once for C*. (Could be better?)
- GPU needs >6GB. So epigenetics on Titan and K40/K80 (etc.)

BarraCUDA for Epigenetic DNA

- Released version of BarrCUDA
- Epigenetic even noisier. “R2” strand noisier than “R1”, so slowing BarraCUDA.

BarraCUDA for Epigenetic DNA



-n switch used to overcome noise.
Plot –n from 0 to 150
-n 7 recommended

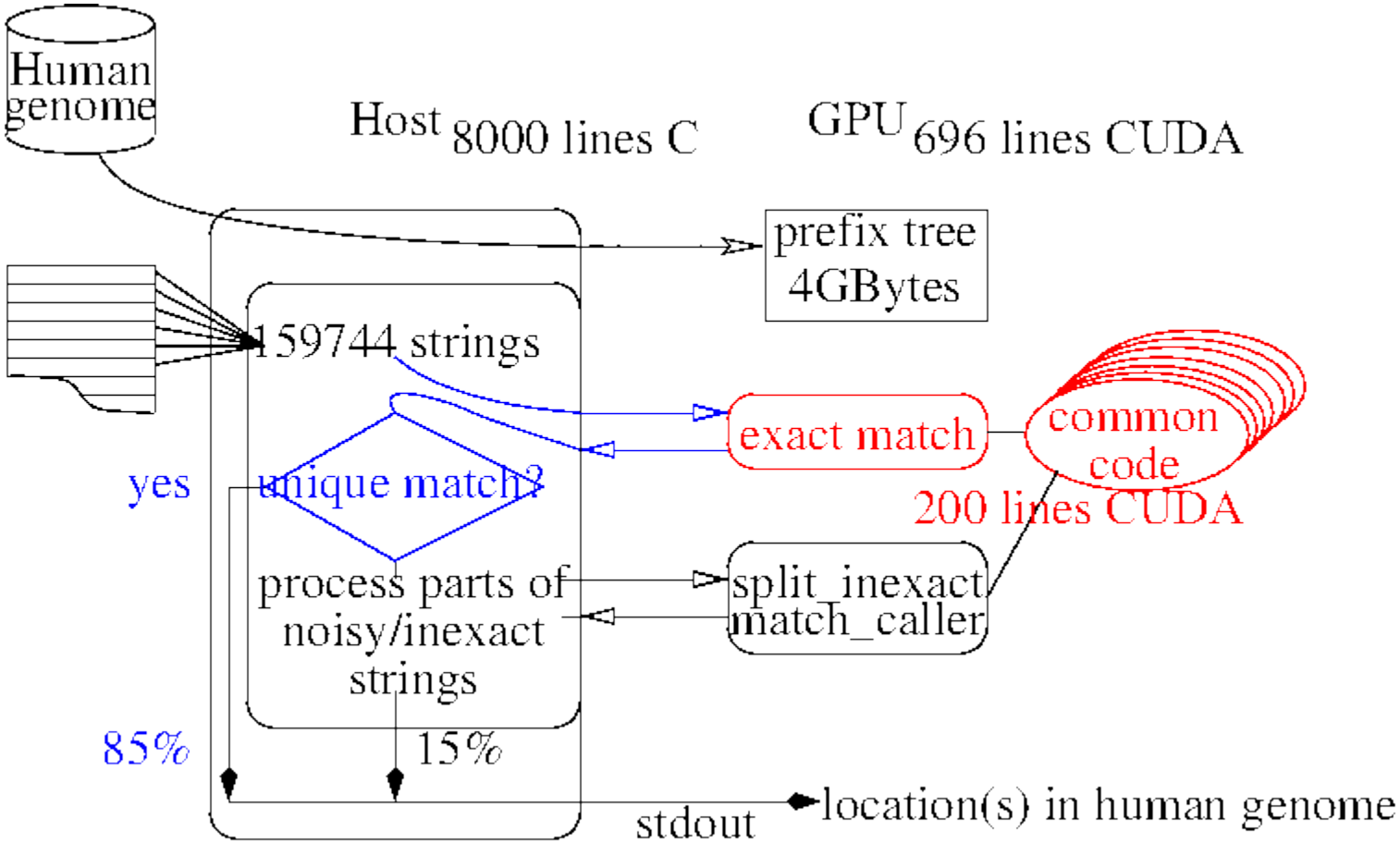
Both *aln* and *sampe* can be run in parallel using CPU and two GPUs

How BarraCUDA was GI ed

BarraCUDA 0.7.107

Manual host changes to call exact_match kernel

GI parameter and code changes on GPU



Why 1000 Genomes Project ?

- Data typical of modern large scale DNA mapping projects.
- Flagship bioinformatics project
  - Project mapped all human mutations.
- 604 billion short human DNA sequences.
- Download raw data via FTP

$120million 180Terra Bytes

Preparing for Evolution

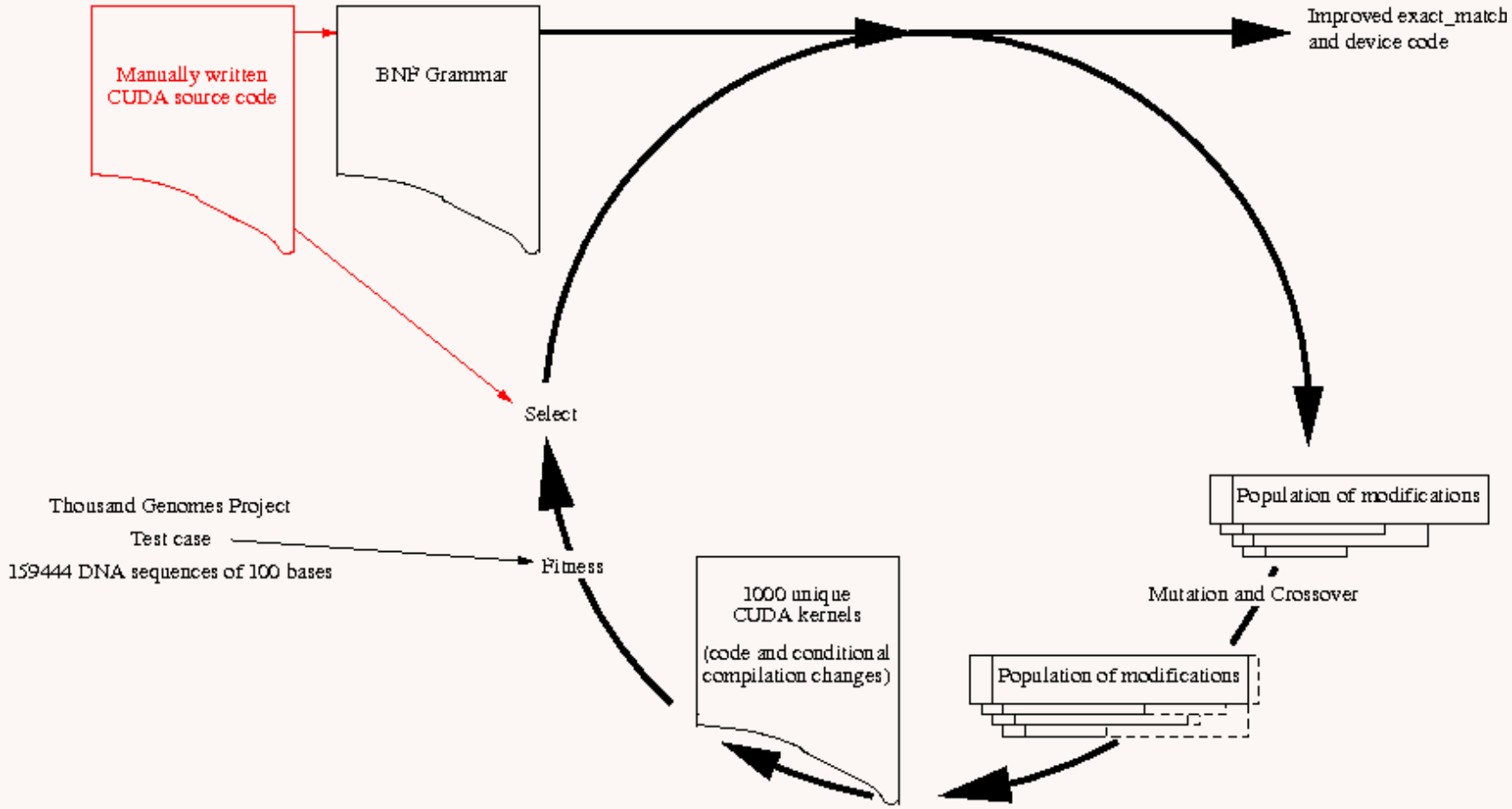
- Re-enable exact matches code
- Support 15 options(conditional compilation)
- GP fitness testing framework
  - Generate and compile 1000 unique mutants
    - Whole population in one source file
    - Remove mutants who fail to compile and then re-run compiler to compile the others
  - Run and measure speed of 1000 kernels
    - Reset GPU following run time errors
  - For each kernel check 159444 answers

Parameter		default	Lines of code affected
BLOCK_W	int	64	all
cache_threads	“” int	“”	44
kl_par	binary	off	19
occ_par	binary	off	76
many_blocks	binary	off	2
direct_sequence	binary	on	63
direct_index	binary	on	6
sequence_global	binary	on	16
sequence_shift81	binary	on	30
sequence_stride	binary	on	14
mycache4	binary	on	12
mycache2	binary	off	11
direct_global_bwt	binary	off	2
cache_global_bwt	binary	on	65
scache_global_bwt	binary	off	35

Evolving BarraCUDA kernel

- Convert manual CUDA code into grammar
- Grammar used to control code modification
- GP manipulates patches and fixed params
  - Small movement/deletion of existing code
  - New program source is syntactically correct
  - Automatic scoping rules ensure almost all mutants compile
  - Force loop termination
- GP continues despite compilation and runtime errors

Evolving BarraCUDA



51 gens in 11 hours

BNF Grammar

Configuration parameter

```
if (*lastpos!=pos_shifted)
{
#ifndef sequence_global
  *data = tmp = tex1Dfetch(sequences_array, pos_shifted);
#else
  *data = tmp = Global_sequences(global_sequences,pos_shifted);
#endif /*sequence_global*/
  *lastpos=pos_shifted;
}
```

CUDA lines 119-127

```
<119>   ::= " if" <IF_119> " \n"
<IF_119>::= "(*lastpos!=pos_shifted)"
<120>   ::= "{\n"
<121>   ::= "#ifndef sequence_global\n"
<122>   ::= "" <_122> "\n"
<_122>  ::= "*data = tmp = tex1Dfetch(sequences_array, pos_shifted);"
<123>   ::= "#else\n"
<124>   ::= "" <_124> "\n"
<_124>  ::= "*data = tmp = Global_sequences(global_sequences,pos_shifted);"
<125>   ::= "#endif\n"
<126>   ::= "" <_126> "\n"
<_126>  ::= "*lastpos=pos_shifted;"
<127>   ::= "}\n"
```

Fragment of Grammar (Total 773 rules)

9 Types of grammar rule

- Type indicated by rule name
- Replace rule only by another of same type
- 650 fixed, 115 variable.
- 43 statement (e.g. assignment, **Not** declaration)
- 24 IF
- <_392> ::= " if" <IF_392> " {\n"
- <IF_392> ::= " (par==0)"
- Seven for loops (for1, for2, for3)
- <_630> ::= <okdeclaration_> <pragma_630> "for(" <for1_630> ";" "OK()&&" <for2_630> ";" <for3_630> ") \n"
- 2 ELSE
- 29 CUDA specials

Representation

- 15 fixed parameters; variable length list of grammar patches.
  - no size limit, so search space is infinite
- tree like 2pt crossover.
- mutation flips one bit/int or adds one randomly chosen grammar change
- 3 possible grammar changes:
  - Delete line of source code (or replace by “”, 0)
  - Replace with line of GPU code (same type)
  - Insert a copy of another line of kernel code

Example Mutating Grammar

```
<_947> ::= "*k0 = k;"
<_929> ::= "((int*)l0)[1] =
__shfl(((int*)&l)[1],threads_per_sequence/2,threads_per_sequence);
"
```

2 lines from grammar

```
<_947>+<_929>
```

Fragment of list of mutations

Says insert copy of line 929 before line 947

Copy of line 929

New code

```
((int*)l0)[1] =
__shfl(((int*)&l)[1],threads_per_sequence/2,threads_per_sequence);
*k0 = k;
```

Line 947

Recap

- Representation
  - 15 fixed genes (mix of Boolean and integer)
  - List of changes (delete, replace, insert). New rule must be of same type.
    - no size limit, so search space is infinite
- Mutation
  - 1 bit flip or small/large change to int
  - append one random change to code
- Crossover
  - Uniform GA crossover
  - GP tree like 2pt crossover

CREST

Best K20 GPU Patch in gen 50

		new
scache_global_bwt	off	on
cache_threads	off	2
BLOCK_W	64	128

Store bwt cache in registers

Use 2 threads to load bwt cache

Double number of threads

line	Original Code	New Code
635		#pragma unroll
578	if(k == bwt_cuda.seq_len)	if(0)
947	`*k0 = k;`	`((int*)l0)[1] = __shfl(((int*)&l)[1],threads_per_sequence/2,threads_per_sequence);*k0 = k;`
126	`*lastpos=pos_shifted;`	

Line 578 `if` was never true

`l0` is overwritten later regardless

Change 126 disables small sequence cache 3% faster

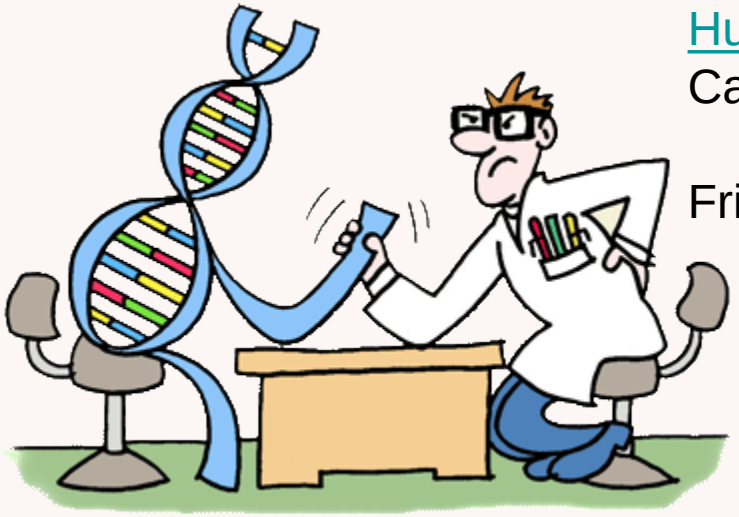
Results

- Ten randomly chosen 100 base pair datasets from 1000 genomes project:
  - K20 1 840 000 DNA sequences/second (original 15000)
  - K40 2 330 000 DNA sequences/second (original 16 000)
- 100% identical
- manually incorporated into sourceForge
- 1219 downloads (11 months)

CREST

WIKIPEDIA

Genetic Improvement



Humies: Human-Competitive
Cash prizes

Friday July 22 14:00-15:40

CREST

Conclusions

- Genetic programming can automatically engineer small programs
  - hash algorithms
  - random numbers which take less power, etc.
- Fix bugs (>10^6 lines of code, 16 programs)
  - auto-port (gzip to GPU). Merge programs (miniSAT Humie)
  - new code to extend application (gggp babel pidgin)
  - code transplant
  - speed up 50000 lines of code
- On real data speed up can be >3×
- use £50 to $325 million
- Software is not fragile
  - break it, bend it, Evolve it

CREST

END

http://www.cs.ucl.ac.uk/staff/W.Langdon/

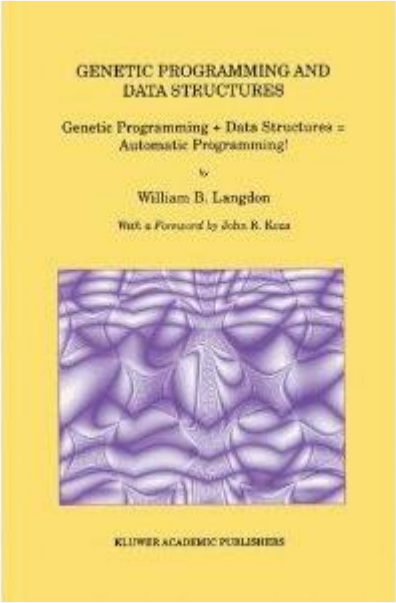
http://www.epsrc.ac.uk/ EPSRC

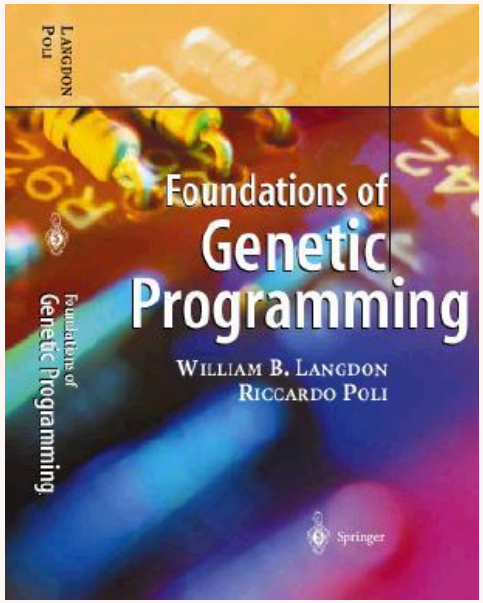
Genetic Improvement

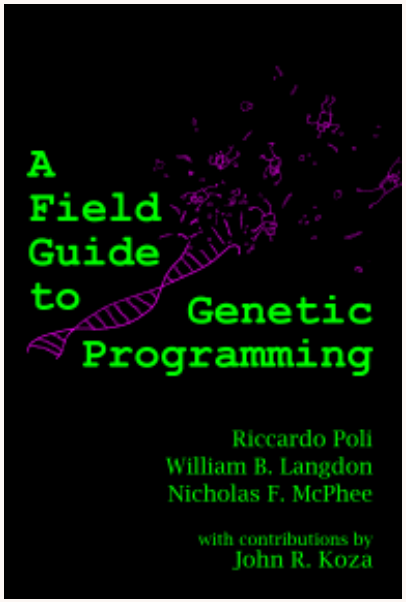
W. B. Langdon

CREST

Department of Computer Science







CREST

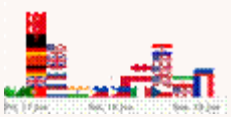
The Genetic Programming Bibliography

http://www.cs.bham.ac.uk/~wbl/biblio/

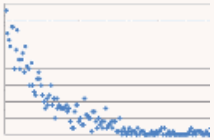
11021 references

RSS Support available through the Collection of CS Bibliographies. XML RSS

A web form for adding your entries. Co-authorship community. Downloads



A personalised list of every author's GP publications.



blog

Google scholar citations

Search the GP Bibliography at
http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html