William B. Langdon
Department of Computer Science, University College London (w.langdon@cs.ucl.ac.uk)

Parallel GPQUICK

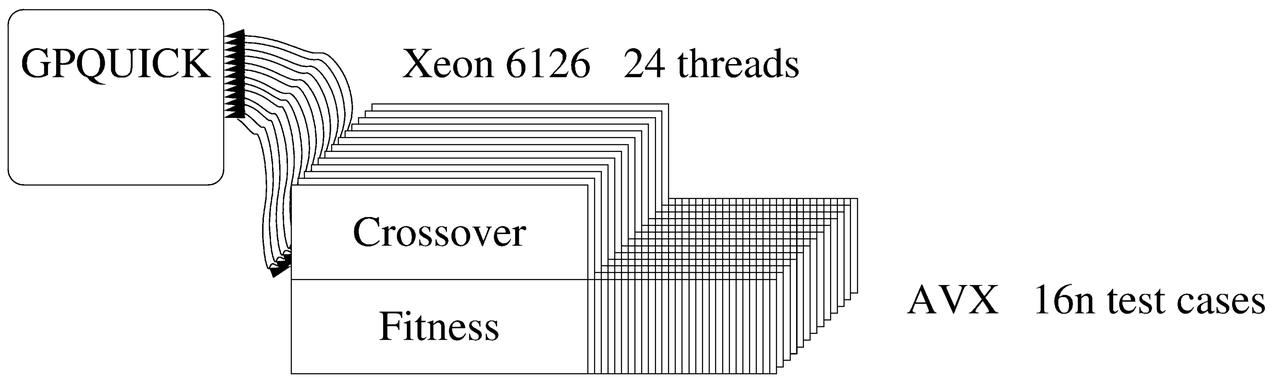


Figure 1: The new version of Andy Singleton's GPQUICK is used in generational mode. It uses both posix threads and Intel AVX 512 which processes 16 floats simultaneously. Crossover parents and subtrees are chosen as normal in the serial code but crossover is delayed until all children have been allocated parents. Each thread waits for a lock and then choses the next free child and releases the lock. Crossover is followed immediately by fitness evaluation of the new child. The thread then looks for another child waiting to be processed. With large populations most cores are almost fully loaded.

1 Why

The future of computing is parallel. Population based approaches, like genetic programming, are embarrassingly parallel. Nonetheless a little effort is required to get the best from single instruction multiple data approaches (like GPUs) and now available in AVX 512 on some Intel servers. The new C++ code combines pthreads with SIMD AVX across test cases to give the fastest GP system at up to 140 billion GP operations per second.

2 AVX Eval

Original GPQUICK recursively passes though whole tree for each fitness case. Fitness is given by comparing tree's evaluation with target value. For AVX process tree once (for large trees good for cache locality) by processing all test cases in parallel and keeping intermediate values on explicit stack.

AVX Eval constant (push onto stack)

Load value (e.g. 0.997) to top of stack 48 times

```
float val = constlist[ ip->op ]
for(i=0;i<ntest;i+=8)
    mm256_store 8 copies of val
```

Top of eval stack

AVX Eval variable

Push whole of test suite onto stack.

x for 16n test cases

memcpy 64n bytes

Top of eval stack

AVX Eval expression

Reclusively evaluation arguments. (Add has two arguments). Push 48 results onto stack

```
Eval;
Eval;
for(i=0;i<ntest;i+=16)
        mm512_load 16 floats from top of stack

sp0 = mm512_load
sp1 = mm512_load
val = sp0+sp1 16 floats
        mm512_store(val) 16 floats
```

Top of eval stack
Next on eval stack

3 Results

GP Primitives Interpreted Per Second. (Default GPU is nVidia GeForce 8800 GTX.) [Fragment of W.B.Langdon. Large scale bioinformatics data mining with parallel genetic programming on graphics processing units. In Massively Parallel Evolutionary Computation on GPGPUs, pp311-347. Springer, 2013. Tab. 3]

Experiment	Pop size	Prog size	Test cases	Speed 10^6 OP/S	GPU
Mackey-Glass	204 800	11.0	1200	895	
Mackey-Glass	204 800	13.0	1200	1056	
Mackey-Glass	204 800	10.2	1200	1720	
Protein	1 048 576	56.9	200	504	
Laser	18 225	55.4	151 360	656	
Laser	5 000	49.6	376 640	190	
Sextic	100	16	200	.5	XBox 360
Sextic	12 500	70.0	100 000	4 073	
Image processing	2 048	2048	$\approx 10^8$	26 200	28×8200
TMBL	120	300	65 536	191 724	260 GTX
Multiplexor-6	12 500	120.6	64	47	
Multiplexor-11	12 500	156.2	2 048	501	
Multiplexor-20	262 144	428.5	2 048	254 000	295 GTX
Multiplexor-37	262 144	915.6	8 192	665 000	295 GTX
GeneChip	16 384	≤63.0	200	314	
Cancer	5 242 880	≤15.0	128	535	
Cancer	5 242 880	12.9	91	1 352	C2050
Cancer	5 242 880	12.9	91	8 517	C2050
Sextic	**4000, 500, 48**	**4 10^8**	**48**	**138 948**	**GPQuick**

4 Coding Performance false sharing

The hardware transparently allows threads to both read and write data. However for performance do not allow read/write data in the same cache line to be used by different threads. E.g. use 64 byte alignment and padding to ensure each thread has its own read/write data.

5 performance new/free in threads

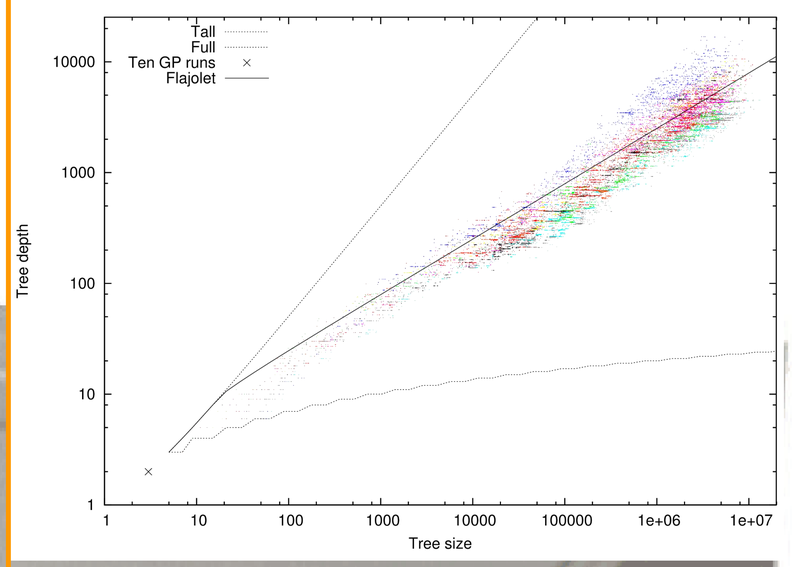
Avoid malloc and delete (new or free) in threaded code. Although library is re-entrant, freeing memory early in a thread is much slower than doing it later in the sequential code.

6 GP parameters

Representation: flattened binary tree.
Terminal set: X, 250 constants -0.995 to 0.997
Function set: MUL ADD DIV SUB
Fitness cases: 48 fixed input -0.97789 to 0.979541 (randomly selected from -1.0 to +1.0
y = xx(x-1)(x-1)(x+1)(x +1)
Selection: tournament 7,fitness=$\Sigma|GP(x_i)-y_i|$
Population: Panmictic, non-elitist, generational
Parameters: Initial population (4000) ramped half and half Koza (1992) depth between 2-6. 100% un-biased subtree crossover. 10 000 gens. Stop if any tree reaches 15 million.

7 Tree depth ≈(2π size)$^{1/2}$

GPquick allocates nthreads (e.g. 24) stacks, each ntest wide (e.g. 48) and maximum tree depth deep. It estimates max evolved tree depth from user specified max tree size using Flajolet's square root approximation.



8 Summary

On a standard Intel Xeon Gold 6126 2.60GHz server, with an unmodified version of Linux, the AVX multi-threaded GPquick is faster than any single computer system except for a few GPU systems on Boolean benchmarks or other specialised problems.

9 Code

Reference: Faster Genetic Programming GPquick via multicore and Advanced Vector Extensions, W. B. Langdon and W. Banzhaf. Technical Report RN/19/01.

Inspired by Dagstuhl Seminar 18052 on Genetic Improvement of Software.

GPQUICK
http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/gp-code/GPavx.tar.gz

Background
Lenovo SD539 Intel Xeon Gold 12C 2.6GHz