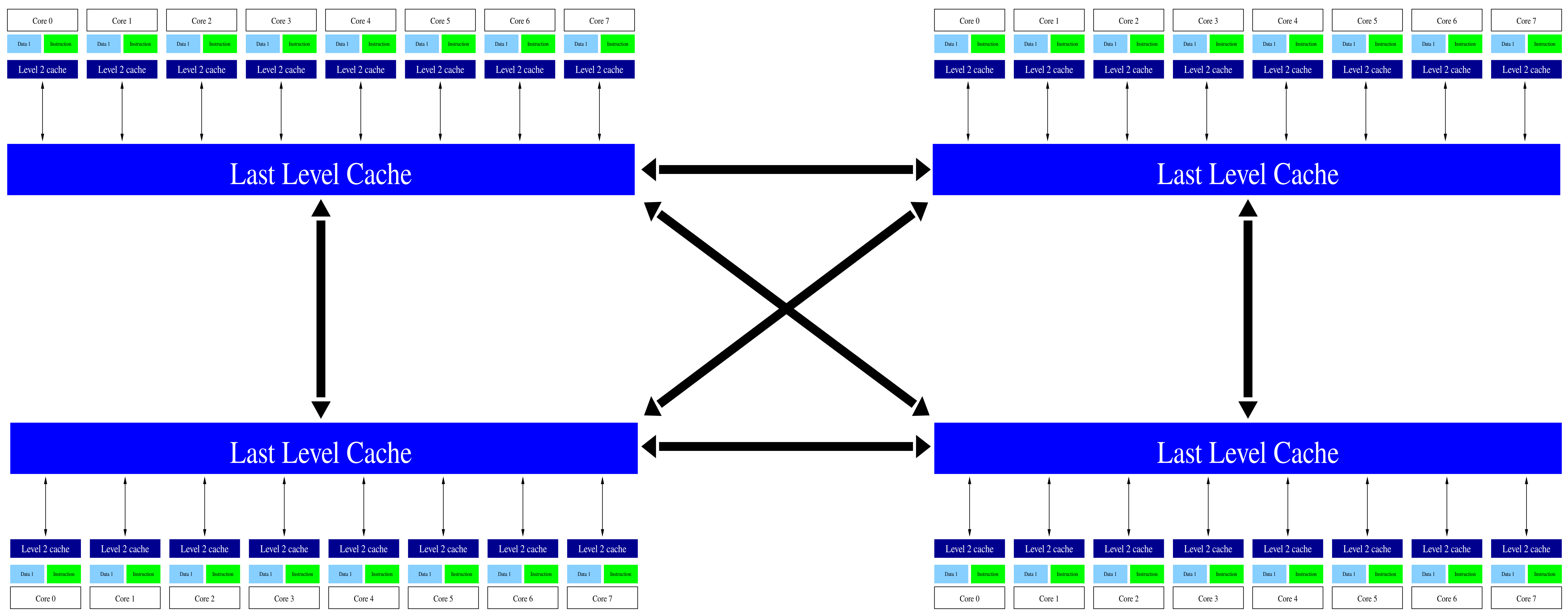
William B. Langdon, David Clark
Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK   Email: w.langdon@cs.ucl.ac.uk

Genetic Improvement of Last Level Cache



1 What is Cache for?

Computation needs data but typically reading data from memory takes hundreds of clock cycles. Originally the cache system tried to keep the CPU busy by keeping a copy of data that is likely to be needed close to the CPU. Nowadays cache is increasingly important to pass data between compute cores.

Data and instruction pass through cache hierarchy. Experiments on Intel I7 three levels of cache. Already many cores are linked via last level cache (LLC), here L3.

2 Cache is automatic

Cache is outside direct programmer's control, i.e., it is totally automatic. It guesses which data will be needed next based on which data the CPU needed recently. E.g. A[1]+B[2] may suggest A[2] and B[2] will be needed soon and the cache may pre-load them or if already in cache, keep them in cache.

The cache algorithm may be secret and the cache will differ between computers.

Data layout and order of code execution can both impact cache effectiveness and so dramatically change performance.

It is hard to optimise software to get the best of the cache system.

3 What is Genetic Improvement?

Genetic Improvement uses evolution to modify existing software. Typically GI is applied to human written source code but it can be applied to anything. Eg C, Java, Java byte code assembler, LLVM IR and machine code. Non-program software could include comments, documentation, specifications. Improvements can mean faster, less energy, less network bandwidth, cheaper, fewer bugs, fairer, more secure, better use of hardware, e.g. GPU.

4 Why? Future is hybrid, parallel. LLC joins compute cores

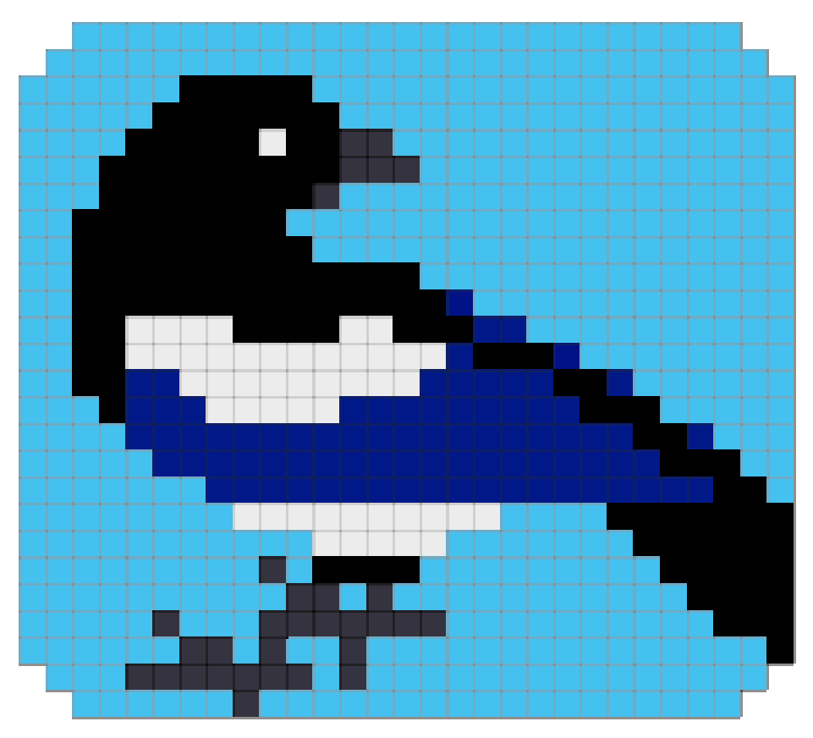
The future is hybrid:
  DSP, FPGAs, multicore, GPU.
  In four years 2018-2022, core count 4x
  If continues by 2028: 131072 cpu per laptop.

Compute is cheap, data is expensive.

5 Cache will be vital

Cache is the bottleneck.
It needs to be automatically programmed.
Evolution can program cache.

6 If you can measure it, you can evolve it



github.com/bloa/magpie

Linux perf reports cache statistics.
Fitness = LLC cache load plus stores.
Evolve using Magpie.

7 VIPS benchmark

VIPS 90,000 line C++ part of PARSEC. Profile with perf and GDB reduced to 37 files (7000 LOC) used in thumbnail creation.

3264 x 2448

128 x 96 thumbnail

8 Optimise VIPS parameters

Five command line options.

9 Optimise GCC and Linker

22 compiler and linker command line options.

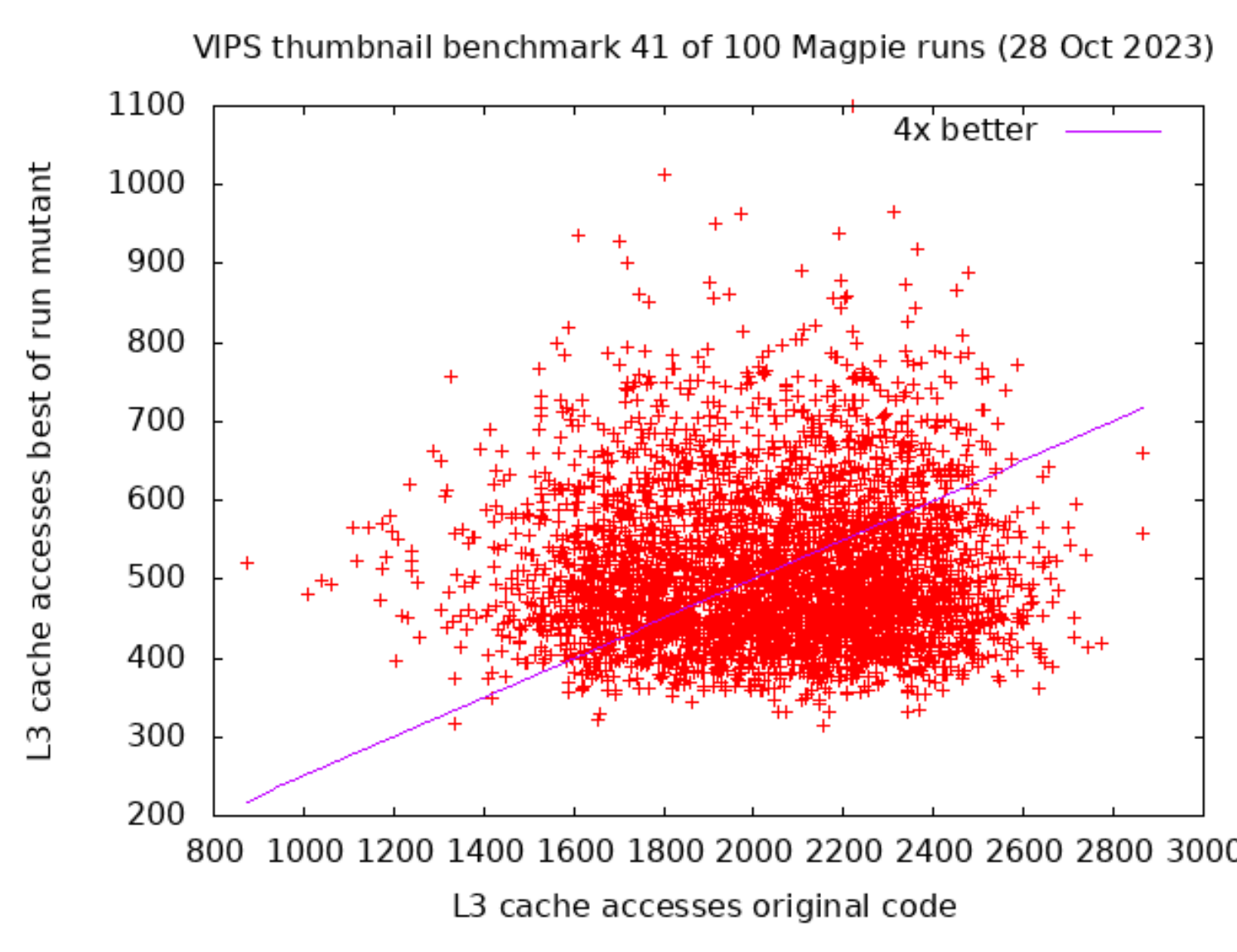
10 C++ as XML

Magpie represents source code as 13,810 lines of XML created by srcml. The mutations were: literal numbers, StmtReplacement, StmtInsertion, StmtDeletion, ComparisonOperatorSetting, ArithmeticOperatorSetting, NumericSetting, RelativeNumericSetting.
100 local searches each with 100 steps.

11 Fitness Function

- Does the mutated code compile?
- Does the mutant run with new cmd line?
- Is output the same as unmutated code's?
- Fitness is difference L3 load + stores

12 Results: In 41 runs LLC reduced by factor of four



Second experiment shows improvement due to Magpie optimising VIPS vips-fatstrip-height

13 Conclusion

Multiple Magpie local search runs simultaneously optimised parameters and code. Substantial improvement due to tuning single command line parameter despite noise.

EuroGP 2024, Aberystwyth. 3-5 April 2024.
Springer