A method for automatically generating analogue benchmark suites using micro architectural event counters

Simon McIntosh-Smith
Microelectronics Research Group
University of Bristol
Woodland Road, Bristol,
BS8 1UB, UK
simonm@cs.bris.ac.uk

Owen Thomas
Red Oak Consulting
2 Lymington Rise
Four Marks, Alton,
Hampshire, GU34 5BA, UK
owen@redoakconsulting.co.uk

ABSTRACT

Real workloads on large-scale, multi-user high performance computing systems can be complex in multiple dimensions: the range of applications run on the hardware system may be diverse and could number in the hundreds or even thousands, and the application mix may vary over time. Constructing and then maintaining representative benchmark suites for such situations can be extremely challenging and can be further complicated when the systems are running codes of a confidential nature.

In this short paper we present a new method which can automatically characterise any workload on a large-scale, multi-user and multi-application system. Our approach uses microarchitecture-level performance metrics, such as the number of branch mispredictions or cache misses. These low-level metrics can be gathered using standard tools on live systems running production codes with very little performance overhead and with no change to any of the codes being analysed. Our method uses microarchitecture-level metrics to construct a statistical model of the real workload. In a second step, a set of 'analogue' benchmarks are also profiled using the same set of microarchitecture performance metrics. In the final step, an automated process constructs a benchmark workload from the set of simple analogue benchmarks. This 'analogue workload' closely approximates the real workload in terms of its statistical behaviour on the hardware and can be used for subsequent relative performance benchmarking.

Categories and Subject Descriptors

B.8.2 [**Performance Analysis and Design Aids**]; C.1.2 [**Multiple Data Stream Architectures**]

Keywords

Performance Model, Multi-core CPU, Hardware counters

1. INTRODUCTION

The challenges of benchmarking complex systems consisting of large-scale heterogeneous hardware running large numbers of different user applications are well documented [1, 2]. These challenges are compounded when the applications may be confidential. The long-term maintenance of a benchmark suite can also be a problem: changes in the mix of user applications and in the hardware systems can impact the accuracy and relevance of a benchmark suite.

To address these problems, we have developed a technique that can automatically assemble an non-confidential benchmark suite that closely approximates the behaviour of a set of target codes [3]. Our approach uses microarchitecture-level hardware counters which record information on a per core and per socket basis. We selected a set of eight common hardware counters which gave good coverage across the most important microarchitecture-level features which influence performance. These hardware counters were: the average number of instructions per cycle, the number of cache misses at each level of the cache (L1 instruction, L1 data, L2, last level), the number of branch instructions and branch mispredictions, the number of DRAM accesses and the number of data TLB misses. Based on these low-level hardware counters, we construct a representative 'analogue' benchmark suite in using the following approach.

In step one, the target codes are *characterised*: each code is run in isolation and the hardware counters are recorded during execution. Each target is characterised by running n copies of the target code in parallel, one on each of n cores of an n-core CPU. Hardware counters are recorded using the standard Linux tools Oprofile and System Tap. These analyses allow us to produce a run-time performance characteristic for each target code, as perceived by the microarchitecture-level hardware counters. A variant of this step involves characterising a live system running a production mix of real codes over substantial period of time, such as 24 hours. This final variant results in an *aggregate characterisation*.

In step two, we select a set of readily-available potential benchmark codes, which we call *analogues*. We select analogue codes using a number of criteria. First, an analogue code should be open source and actively maintained. This addresses the problem with benchmark codes rapidly becoming out of date. Second, an analogue should ideally display an extreme behaviour in one or more of the eight microarchitecture-level hardware counters we are measuring. We construct the set of analogues to give us good coverage across the range of hardware counters. The analogues are characterised using a similar approach to that described in step one, but in addition we characterise each analogue multiple times, each with a different set of input parameters. Analogues which support command-line parameters that significantly alter their run-time behaviour with respect to our chosen hardware counters are favoured. We term the

PMBS'11, November 13, 2011, Seattle, Washington, USA.
ACM 978-1-4503-1102-1/11/11.

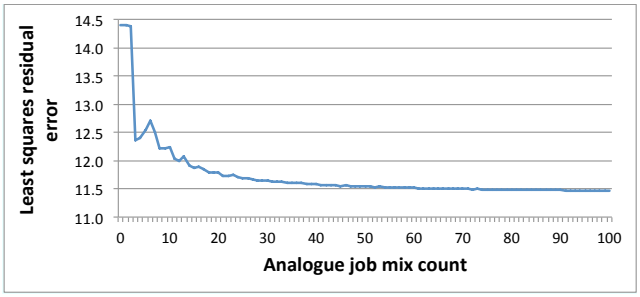

Figure 1: Least squares difference between the analogue and target job mixes.

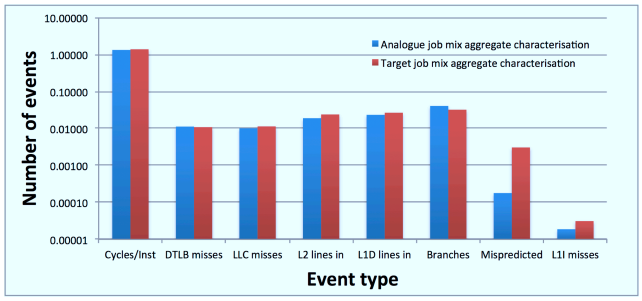

Figure 2: Comparison of aggregate metrics for the analogue and target job mixes.

repeated execution of an analogue with different command line parameter values a *parameter sweep*.

In step three, we automatically assemble a combination of one or more analogues in order to approximate the behaviour of one or more target codes. In the simplest variant of this stage, we find the single analogue whose microarchitecture-level counter profile most closely matches that of a single target code. In a more complex variant, we can combine a number of different analogue codes in order to more closely approximate the microarchitecture-level counter profile of a single target. We do this by assembling a *job mix* – a set of m analogues $A_m = \{a_1 \dots a_m\}$, each of which may be executed a different number of times. It is the aggregate characterisation of such an analogue job mix which is optimised to model the behaviour of the target code. Blending multiple analogues in this way gives us greater flexibility to reproduce target behaviour. In the most complex variant of this step, we can construct an analogue job mix that will model the behaviour of a target job mix. That is, having obtained the aggregate characterisation of a job mix of target codes, possibly from characterising a live production system, we can construct an analogue job mix with a very similar aggregate characterisation.

The algorithm for constructing an analogue job mix to closely match a target job mix is as follows. We employ a *residual refinement* approach, which takes an aggregate view of the target job mix, using the hardware metrics averaged across all of the target benchmarks and also across all of the different hardware platforms that we used for our experiments. We then take the analogue sweep dataset previously constructed by running each analogue benchmark with a range of different input parameters across all the hardware platforms, resulting in a large number of microarchitecture characterisation data points from which to choose. We then construct an analogue job mix by creating by incrementally selecting individual analogue job instances which reduce the residual least squares error. This method iterates until either the residual error drops below 0.01 or the number of iterations exceeds some upper threshold.

2. RESULTS

From a twelve hour run of a target job mix consisting of a random selection of six benchmark codes, we used the residual refinement method described above to construct an analogue job mix from a set of three analogue codes which were characterised across a number of hardware platforms and for a set of different input parameters, 141 characterisation data points in total. This method converged rapidly, getting close to the optimal with just 100 jobs in the analogue job mix with a least squares residual error of 11.4; with 5,000 jobs in the analogue job mix the residual error had only improved by 0.1. The graph of residual error as we add jobs from our pool of 141 analogue candidate instances to our analogue job mix is shown in Figure 1. The hardware counter characterisation for the input target job mix is shown alongside the corresponding profile of the auto generated residual refinement analogue job mix in Figure 2, with the metrics ordered with the most important on the left and least important on the right. The results were generated on a system with dual socket quad-core Intel Core i7 ('Nehalem') E5520 2.27GHz CPUs, each with 8MB of L3 cache per CPU, and 24GB of 1066MHz DDR3 memory. As one can see, the most important metrics have been matched very closely, with less important metrics showing larger errors. The performance penalty for measuring the microarchitecure-level hardware counters during characterisation was consistently between 1.0 and 1.5%.

3. CONCLUSIONS

We have demonstrated the use of microarchitecture-level metrics to automatically construct analogue benchmarks that closely match the behaviour of a set of target benchmarks, even though the analogue job mix is considerably simpler than the target workload that it approximates. The analogue workload is non-proprietary and non-confidential by construction, and can thus be shared openly with partners, for example for procurement purposes. The full version of this paper [3] includes results for a range of different x86 processor architectures, and investigates the use of the constructed analogue job mix for relative performance benchmarking.

4. REFERENCES

[1] Armstrong, B., Bae, H., Eigenmann, R., Saied, F., Sayeed, M., and Zheng, Y. HPC benchmarking and performance evaluation with realistic applications. In *SPEC Benchmark Workshop* (January 2006).

[2] Carlton, A. Lies, damn lies, and benchmarks. http://www.spec.org/osg/news/articles/news9412/lies.html, December 1994.

[3] McIntosh-Smith, S., and O., T. A method for automatically generating analogue benchmark suites using micro architectural event counters. *SIGMETRICS Performance Evaluation Review 40*, 2 (2012).