DARTS: Performance-Counter Driven Sampling Using Binary Translators

Rajesh Kumar, Suchita Pati, Kanishka Lahiri
Advanced Micro Devices, Bangalore
{rajesh.kumar, suchita.pati, kanishka.lahiri}@amd.com

***Abstract*—We motivate DARTS, a new technique for workload sampling to drive architectural simulation, that carefully integrates hardware performance counter monitoring with fast, dynamic binary translation. This paper focuses on a key finding that motivates the DARTS methodology. We present our experience applying it to the generation of simulation points for the integer benchmarks of the SPEC CPU2006 suite. We were able to match existing approaches in terms of coverage of dynamic workload behaviours and simulated performance. DARTS achieved this with significant improvements in turn-around time, simulation effort, and storage requirements, with a slight increase in the number of simulation points.**

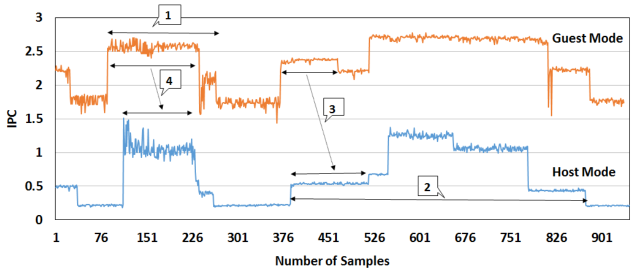
I. Introduction

Detailed cycle-accurate performance models are typically driven by methodologies based on uniform, random [1], or program analysis based sampling [2]. Most prior research in this area has aimed at exploring the trade-off between representativeness and simulation throughput. However, in industrial settings, timely delivery of inputs based on contemporary workloads is a major requirement, often making these approaches too complex to execute.

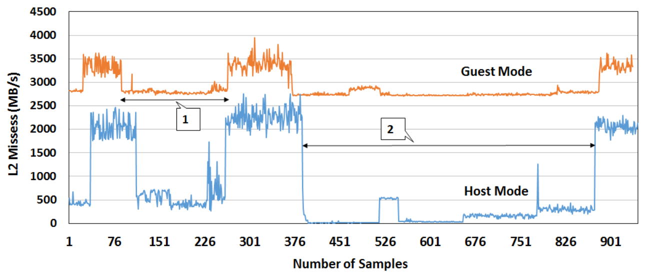
This work aims to dramatically improve the efficiency of workload sampling without compromising accuracy or simulation speed. We do this by monitoring and analyzing Performance Monitoring Counters (PMCs) while the workload of interest runs in a dynamic binary translator on the same machine. Our main contribution is the observation that, for a large class of workloads, *strong similarities* exist between (i) phase behaviours seen when a workload is run natively on a host and (ii) when it is simulated (on the same host) using dynamic binary translation. (In the rest of this paper, we refer to these two modes as *host* mode and *guest* mode, respectively). This finding paves the way for DARTS, a new workload sampling methodology that directly generates a set of simulation points at near-native hardware execution speeds. In the next sections, we present this finding, and illustrate the potential of a sampling flow based on this.

II. Background: Dynamic Binary Translators

Most functional simulators abstract away the details of the micro-architecture: *i.e.* they only simulate *what* a processor does, not *how* it does it, to achieve high simulation speed. Dynamic binary translators refer to a specific category of extremely fast functional simulators that emulate one instruction set by another. A short sequence of code that needs to be simulated (typically of the order of a single basic block), is translated into an instruction sequence that can be natively executed on the host. The resulting sequence is stored in a *code cache* to eliminate the need to repeatedly translate frequently executed code. In this work, we use SimNow^{TM}[3] as our binary translator.



(a) IPC



(b) L2 Misses (MB/s)

Fig. 1: Comparing PMC data for host and guest mode runs of the CPU2006 benchmark `astar`: (a) IPC (b) L2 Misses (MB/s)

III. Guest-Mode vs Host-Mode PMC Study

We ran the integer benchmarks from the SPEC CPU2006 [4] suite (compiled using `gcc` with moderate optimization) first, natively on a host system (Table I) and then on a guest modeled by SimNow^{TM} running on the same host. In each case we collected and analyzed performance counter data from the host. As an example, Figure 1 compares host and guest-mode PMC data for `astar` (one of the CPU2006 benchmarks). To make the plots easier to compare, we compressed the guest-mode time-series data by the slowdown introduced by the binary translator. In addition, we shifted guest-mode statistics vertically by appropriate constants.

TABLE I: Experimental Setup

Benchmarks	SPEC CPU2006 INT Benchmark Suite
Desktop System	AMD A10-7850K Radeon R7 APU 3.5 GHz (fixed frequency) 16 GB DDR3-1866 MHz RAM Ubuntu v15.04
Binary Translator	SimNow^{TM}
Microarchitectural Statistics for PMC Profiling	IPC, L1 IC Misses PKI, L1 DC Misses PKI, L2 Accesses PKI, L2 Misses PKI, Branches PKI, Branches Taken PKI, Branches Mispredicted PKI, SSE Instructions PKI, x87 Instructions PKI, DRAM Bandwidth (MB/s)

The most important observation is that the translator's overhead notwithstanding, *the guest-mode PMC profile retains most of the dynamic phase behaviours* (e.g., *Instructions Per Cycle or IPC) as seen in the host-mode run*. We observed this to be true for all the workloads in the CPU2006 suite and for a variety of micro-architectural statistics that we captured. Note that the absolute values of statistics in guest and host modes can vary significantly, which can be attributed to the overhead of the simulator. Also, the fraction of time spent in a certain phase in host mode may not be the same in guest mode, as shown in the regions marked "3" and "4" in Figure 1(a). This indicates that the scaling of phase lengths could depend on dynamic conditions, such as frequency of code cache invalidations. However, for our purposes, absolute values of statistics, or the duration and stability of phases in guest-mode are not relevant. The key need is to identify phase *boundaries*, based on the relative changes in statistics. From these studies, we are convinced that if properly identified and exploited, guest-mode PMC behaviour can be the driver of a new sampling methodology. One may ask, why use a guest at all. This is to ensure that representative simulation points be reproduced deterministically (based on instruction counts), something that cannot be guaranteed in host mode.

These plots also show that using only one or two statistics from the guest-mode PMC profile increases the risk of missing workload phases that fail to manifest due to the influence of the translator. In the regions marked "1" and "2", we observe relative stability of the guest-mode L2 Miss profile of `astar`, whereas host-mode shows a certain amount of dynamic behaviour (Figure 1(b)). However, taken together with guest-mode IPC profile (marked "1" and "2" in Figure 1(a)), the variation in workload behaviour is apparent. Hence, in our work, we use a vector of statistics, in most cases expressed as Per Kilo Instructions (PKI), to profile guest-mode workload behaviour (Table I).

IV. Experimental Evaluation

Based on the above findings, we have developed a prototype methodology called DARTS, that applies machine learning techniques to guest-mode PMC data and generates a minimal set of simulation points that are then fed to a cycle-accurate performance model of a target architecture.

To illustrate how well the selected sample points cover the different phases of each workload, we reduced each sample from the original host-mode hardware PMC profile to 2 dimensions using Principal Components Analysis [5], the original dimensionality being 11 (Table I). Next, we did the same for each simulation point, using statistics obtained from simulating it on a cycle-accurate model of the same machine. The resulting points were plotted in 2-dimensional space. Figure 2 shows the results of this exercise for `astar`. We observe that we have at least one simulation point for all the major phases (dense clusters) in the benchmark.

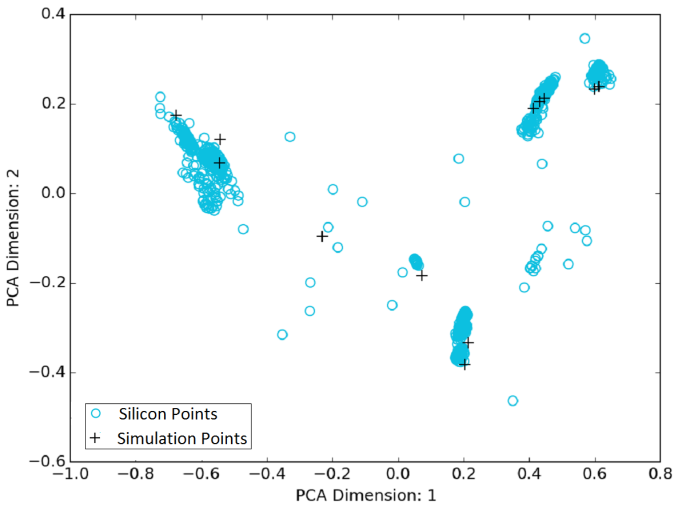


Fig. 2: 2D PCA plot comparing host-mode PMC profile and simulated sample points for `astar`

Similar studies on other CPU2006 workloads yielded good results as well, and we were able to generate and validate a complete set of simulation points for the CPU2006 suite. Cycle-accurate simulations of these points yielded IPC errors (2%) that were comparable to a mature in-house methodology, with a 15% increase in the number of points simulated. However, we estimate significant savings in (i) the time taken to deliver these points (from a month to less than a week), (ii) disk space requirements (13X) and (iii) simulation effort (16X). We believe with additional tuning of our methodology, we can outperform traditional approaches on the first two metrics as well. This methodology is now deployed and is being used for ongoing workload sampling efforts.

References

[1] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe, "SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling," in *Proc. Int. Symposium on Computer Architecture (ISCA)*, pp. 84–97, 2003.

[2] T. Sherwood, E. Perelman, G. Hamerly, S. Sair, and B. Calder, "Discovering and Exploiting Program Phases," *IEEE Micro*, vol. 23, pp. 84–93, Nov. 2003.

[3] "AMD SimNow." http://developer.amd.com/tools-and-sdks/cpu-development/simnow-simulator/.

[4] "SPEC CPU2006." https://www.spec.org/cpu2006/.

[5] "Principal Component Analysis." https://en.wikipedia.org/wiki/Principal_component_analysis.