Utilization of Random Profiling for System Modeling and Dynamic Configuration

Jason Hiebel
Michigan Technological University
Houghton, Michigan
jshiebel@mtu.edu

Laura E. Brown
Michigan Technological University
Houghton, Michigan
lebrown@mtu.edu

Zhenlin Wang
Michigan Technological University
Houghton, Michigan
zlwang@mtu.edu

1 MOTIVATION

Dynamically tuning or reconfiguring operating system and architectural resources at runtime can improve performance by adapting system operation to the current workload. We seek to address the problems of both system modeling and dynamic system configuration in the context of sequential decision processes with limited feedback. Random profiling is a technique for sequential decision processes with limited feedback, which can adequately and efficiently capture the relationships between workload behavior, system configuration, and performance. Along with machine learning methods, random profiling can be used for determining descriptive metrics for workload behavior and constructing policies which dynamically reconfigure the operating system and microarchitecture at runtime. We present three such problems: system event selection, dynamic paging mode selection, and dynamic hardware prefetcher configuration.

2 SYSTEM EVENT SELECTION

Given the complex and interconnected nature of modern microarchitectures, the relationship between workload behavior metrics and performance can be non-obvious. Due to hardware limitations, the availability of microarchitecture performance monitoring is limited. The Performance Monitoring Unit (PMU) exposes an interface for specifying and counting microarchitectural events (which number in the hundreds) using a small number of event counters (typically four or eight). Therefore, selecting subset of events which effectively measure workload behavior is challenging, and is influenced by a lack of appropriate domain knowledge, ill-fitting events exposed by the PMU, and the possibility that some events may be inconsistent or incorrectly implemented [1].

Choosing a descriptive set of events which are relevant to workload behavior is a system configuration problem with limited feedback. As only a small subset of events can be measured at any given time, we sample the relationship between event subsets and a performance metric, such as Instruction per Cycle (IPC), at random in order to extract a meaningful subset of events. We utilize Attribute Efficient Regression (AER) [3] to construct a regression model mapping event counts to performance, given a limit on the number of events which can be observed simultaneously. In order to select which events to measure, AER selects events to sample with a probability proportional to the estimated influence an event could have on performance.

AER models were constructed for each SPEC CPU2006 benchmark using the full set of PMU events (on the Sandy Bridge microarchitecture, which provides eight event counters). Events are ranked according to their regression coefficients in order to describe importance. Figure 1 depicts event rankings for a subset of notable events.

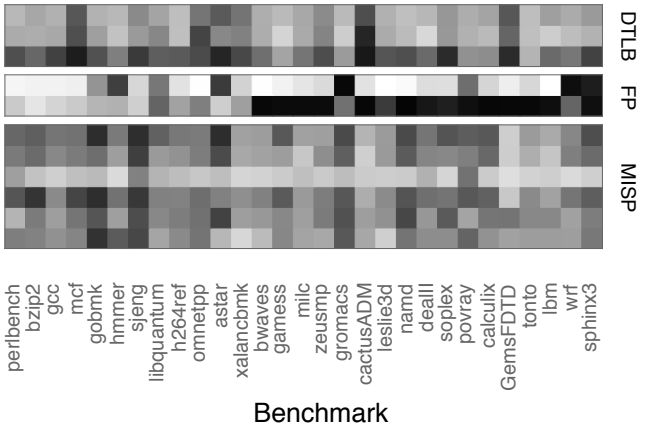

Figure 1: Importance ranking heat map for selected hardware event classes: DTLB, DTLB misses; FP, scalar floating point operations; MISP, branch mispredictions. For each benchmark, higher ranked events are depicted darker.

Qualitatively, these models depict performance relationships which are substantiated by domain knowledge: memory-intensive benchmarks such as `mcf` and `cactusADM` rank data translation lookaside buffer (DTLB) miss events more prominently; the rankings of scalar floating point events (FP) readily identify the floating point benchmarks; and graph/tree search benchmarks such as `astar`, `gobmk`, and `sjeng` have high rates of branch mispredictions (MISP). By noting differences in characteristics between workloads we can inform the selection of meaningful event subsets.

3 DYNAMIC SYSTEM CONFIGURATION

Dynamic system configuration can be modeled as a contextual bandit. The contextual bandit is a well studied method for sequential decision making with limited feedback [7]. At each iteration, the bandit observes some contextual information, and uses that context, as well as existing knowledge about the decision process, to select an action. In response, the bandit receives a reward dependent on both the context and selected action. Policies for action selection can be constructed from logged data, obtained from selecting actions at random and recording both the associated contextual information and resulting reward. The logged data, which is an exploration of the relationship between action and reward, can then be scavenged in order to inform the construction of an action selection policy which attempts to maximize reward [2, 6].

For dynamic system configuration, the contextual information would consist of relevant workload characteristic measurements

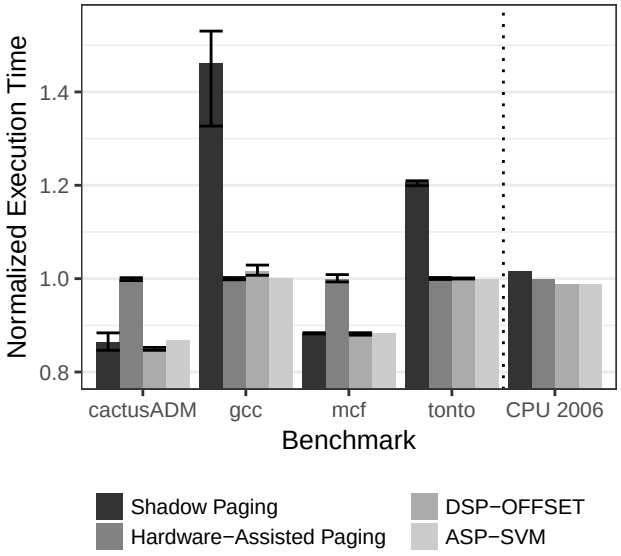

Figure 2: Summary of dynamic paging mode selection benchmark execution time, normalized to Hardware-Assisted Paging, using the static Shadow Paging and Hardware-Assisted Paging, and dynamic DSP-OFFSET and ASP-SVM. The geometric mean normalized time is also given for the full SPEC CPU2006 benchmark suite.

obtained from the microarchitecture and operating system, such as cache or memory events. Each action corresponds to the specification of a system configuration. The reward would reflect the desired metric to optimize, such as maximizing workload performance. Logged data is obtained, at regular intervals, by observing workload characteristics, choosing a system configuration randomly, and measuring the resulting performance. Dynamic configuration policies can be constructed from this random profiling data using machine learning methods, e.g., using Binary-Offset [2].

Paging Mode Selection: There are two common paging modes used by virtual memory managers, Shadow Paging and Hardware-Assisted Paging, each of which work well for different types of memory access characteristics. The best performing paging mode can vary both between programs and between phases within a single program. Dynamic paging mode selection can improve performance by adapting the system, at runtime, to use the paging mode which favors the current workload.

We utilized the contextual bandit and random profiling to construct a dynamic paging mode selection policy (DSP-OFFSET) for the Xen virtual memory manager [4]. We first generated a training dataset by selecting paging modes randomly at 1 s intervals for the integer workloads of the SPEC CPU2006 benchmark suite, observing the memory access characteristics (DTLB misses and page faults) and performance (IPC) of the current workload over the given interval. Using Binary-Offset [2], we constructed a policy for paging mode selection which effectively chooses paging modes for a broad set of workload behaviors.

We evaluated the performance of DSP-OFFSET compared to the static paging mode selections (Shadow Paging, Hardware-Assisted Paging) and the state-of-the-art dynamic policy ASP-SVM [5] on the full SPEC CPU2006 suite. Figure 2 illustrates the performance for a subset of benchmarks and the average performance across the full benchmark suite. Overall, DSP-OFFSET was competitive in performance with ASP-SVM while requiring substantially less profiling time (2.5 hours, compared to over 24 hours).

Hardware Prefetcher Configuration: Our dynamic system configuration framework is applicable to other system configuration problems such as hardware prefetching. Modern Intel systems are equipped with four hardware prefetchers (per core), which can be enabled or disabled at runtime. While the use of hardware prefetchers is generally beneficial, the increased memory traffic and cache contention can have a destructive effect on performance, especially when considering parallel workloads in a multi-core, co-tenancy setting. Dynamically enabling or disabling hardware prefetchers according to a workload's memory and cache behavior can improve performance.

The previous problem of paging mode selection is a small and well understood instance of dynamic system configuration. In contrast, hardware prefetching presents added, interesting challenges. The configuration space is considerably larger with 2^4 possible prefetcher assignments per core and decisions must be made cooperatively across multiple cores to account for possible resource contention and destructive cache interference from multiple, independent workloads. We propose, first, to naively consider each hardware prefetcher in isolation so that the configuration space remains binary and Binary-Offset [2] is directly applicable. Additional explorations include comparing independent, per-core configuration policies versus a single, global configuration policy, and developing policies which incorporate the combinatorial structure of multiple hardware prefetchers.

REFERENCES

[1] Reza Azimi, Michael Stumm, and Robert W. Wisniewski. 2005. Online Performance Analysis by Statistical Sampling of Microprocessor Performance Counters. In *Proceedings of the 19th Annual International Conference on Supercomputing (ICS '05)*. 101–110.

[2] Alina Beygelzimer and John Langford. 2009. The Offset Tree for Learning with Partial Labels. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. 129–138.

[3] Nicolò Cesa-Bianchi, Shai Shalev-Shwartz, and Ohad Shamir. 2011. Efficient Learning with Partially Observed Attributes. *Journal of Machine Learning Research* 12 (2011), 2857–2878.

[4] Jason Hiebel, Laura E. Brown, and Zhenlin Wang. 2018. Constructing Dynamic Policies for Paging Mode Selection. In *47th International Conference on Parallel Processing (ICPP '18)*. forthcoming.

[5] Wei Kuang, Laura E. Brown, and Zhenlin Wang. 2015. Selective switching mechanism in virtual machines via support vector machines and transfer learning. *Machine Learning* 101, 1 (2015), 137–161.

[6] John Langford, Alexander Strehl, and Jennifer Wortman. 2008. Exploration Scavenging. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. 528–535.

[7] John Langford and Tong Zhang. 2007. The Epoch-Greedy Algorithm for Contextual Multi-armed Bandits. In *Advances in Neural Information Processing Systems 20 (NIPS)*. 817–824.