# Utilizing Random Profiling for System Modeling and Dynamic Configuration

Jason Hiebel  Laura E. Brown  Zhenlin Wang
Department of Computer Science, Michigan Technological University

## Motivation

Dynamically tuning or reconfiguring operating system and architectural resources at runtime can improve performance by adapting system resources to the current workload. However, constructing effective policies for dynamic configuration is difficult due to limited feedback. We are limited to capturing workload and performance characteristics for only the current system configuration.

**Goal:** Use sequential decision processes with limited feedback for system performance modeling and dynamic system configuration.

We consider three problems using sequential decision processes as our model:

- system event selection,
- dynamic paging mode selection, and
- dynamic hardware prefetcher configuration,

and use **random sampling** to construct effective efficient decision making policies.

## References

[1] John Langford and Tong Zhang.
The epoch-greedy algorithm for contextual multi-armed bandits.
In *Advances in Neural Information Processing Systems*, 2007.

[2] Alina Beygelzimer and John Langford.
The Offset Tree for learning with partial labels.
In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[3] Nicolò Cesa-Bianchi, Shai Shalev-Shwartz, and Ohad Shamir.
Efficient learning with partially observed attributes.
*Journal of Machine Learning Research*, 2011.

[4] Jason Hiebel, Laura E. Brown, and Zhenlin Wang.
Constructing dynamic policies for paging mode selection.
In *Proceedings of the 47th International Conference on Parallel Processing*, 2018.

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

## Acknowledgements

This research is supported in part by the National Science Foundation under Grant No. CSR1422342 and CSR1618384, the National Science Foundation of China under Grant No. 61232008, 61472008, 61672053 and U1611461, Shenzhen Key Research Project under Grant No. JCYJ20170412150946024, and the 863 Program of China under Grant No. 2015AA015305. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

## System Event Selection

Selecting a descriptive set of events which are relevant to describing workload behavior and system performance.

### Challenges:

- The Performance Monitoring Unit (PMU) exposes hundreds of events; but only a small number of event counters (typically four or eight).
- Events can be ill-fitting to an application, and are sometimes inconsistently or incorrectly implemented.

### Method — Attribute Efficient Regression (AER) [3]:

AER selects a number of events to **sample randomly**, with probability proportional to the estimated influence of the event and produces a linear regression model.

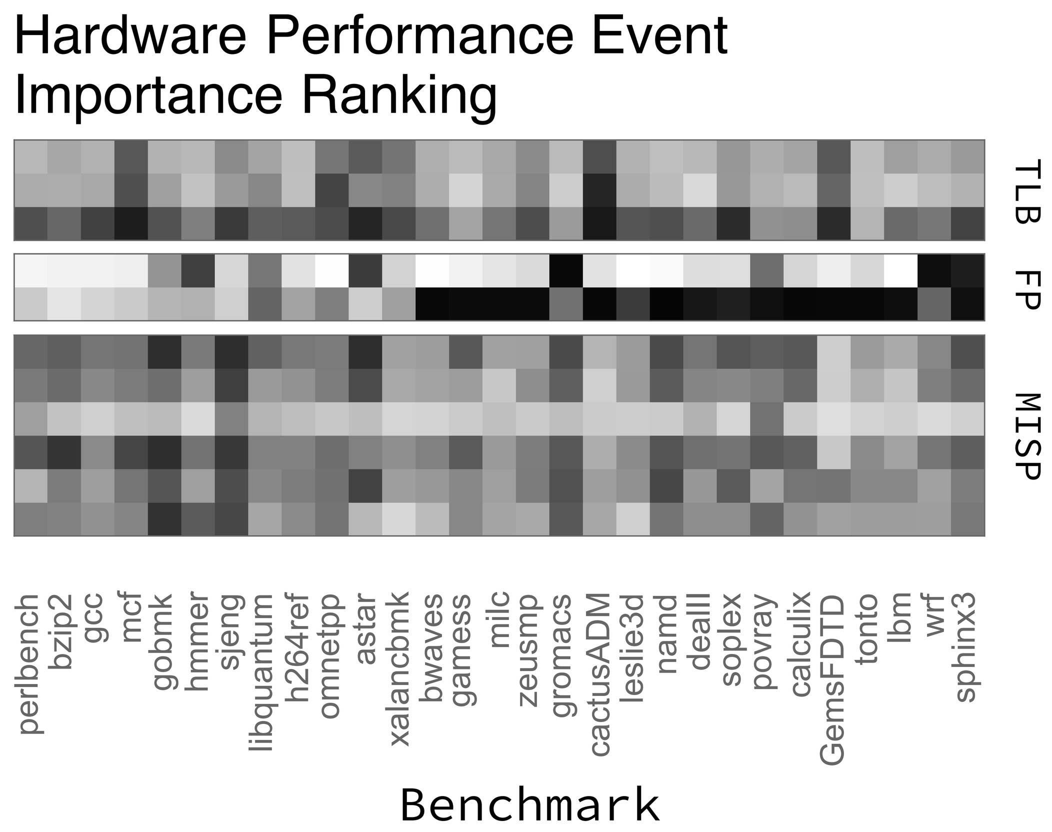


**Figure 1:** Importance ranking for selected event classes: TLB, DTLB miss counts; FP, scalar floating point operation counts; MISP, branch misprediction counts. Higher ranked events are depicted darker.

### Experimental Design:

- Individual AER models are constructed for each SPEC CPU2006 benchmark.

### Conclusions:

Rankings **substantiated by domain knowledge**:

- Memory-intensive benchmarks rank DTLB miss events prominently, e.g., `mcf` `cactusADM`.
- Scalar floating point events readily identify floating-point benchmarks.
- Graph, tree search benchmarks have highly ranked of branch mispredictions, e.g., `astar`, `gobmk`, and `sjeng`.

## Dynamic System Configuration

### Dynamic Paging Mode Selection

Dynamically select paging mode, i.e., Shadow Paging (SP) and Hardware-Assisted Paging (HAP), at runtime to improve system performance.

#### Challenges:

- Performance is only measured for the currently selected system configuration (limited feedback).
- System configuration control is typically low-level, which limits online methods or complex models.

#### Method — Contextual Bandit [1]:

A model for sequential decision process with limited feedback. At each iteration,

1. Observe contextual information representing the current state of the world.
2. Select an action using contextual information and existing knowledge about the problem.
3. Receive reward dependent on both the contextual information and selected action.

Action selection policies can be constructed from logged data collected using random action selection [2].

Our DSP-OFFSET [4] is constructed for the contextual bandit using **random profiling data**.

#### Experimental Design:

- Policy constructed using a single, random profiling execution of each integer benchmark from the SPEC CPU2006 benchmark suite.
- Evaluated on full SPEC CPU2006 benchmark suite.

#### Conclusions:

- Per-benchmark performance matches or beats performance of the best static policy.
- DSP-OFFSET paging mode selection matches known program behavior.
- DSP-OFFSET has **equivalent performance** to the state-of-the-art ASP-SVM [5].
- DSP-OFFSET needs **substantially less profiling** than ASP-SVM: 2.5 hours vs. 24 hours.
- Policy generalizes well to workload behavior not seen during training (SPEC FP2006 benchmarks)

### Dynamic Hardware Prefetching

Dynamically enabling or disabling hardware prefetchers according to workload memory and cache behavior to improve performance.

#### Challenges:

- Configuration space is large, with $2^4$ possible prefetcher assignments per core on Intel architectures.
- Prefetchers can cause destructive shared-cache interference (cache pollution) and increased memory bandwidth usage with little improvement to performance.
- Decisions should be made cooperatively across multiple cores to consider co-tenancy resource contention and cache interference.

#### Proposed Methods:

- Naively consider each hardware prefetcher in isolation using the same framework as paging mode selection.
- Developing policies which incorporate the combinatorial structure of multiple hardware prefetchers.
- Comparing independent, per-core configuration policies versus a single, global configuration policy.

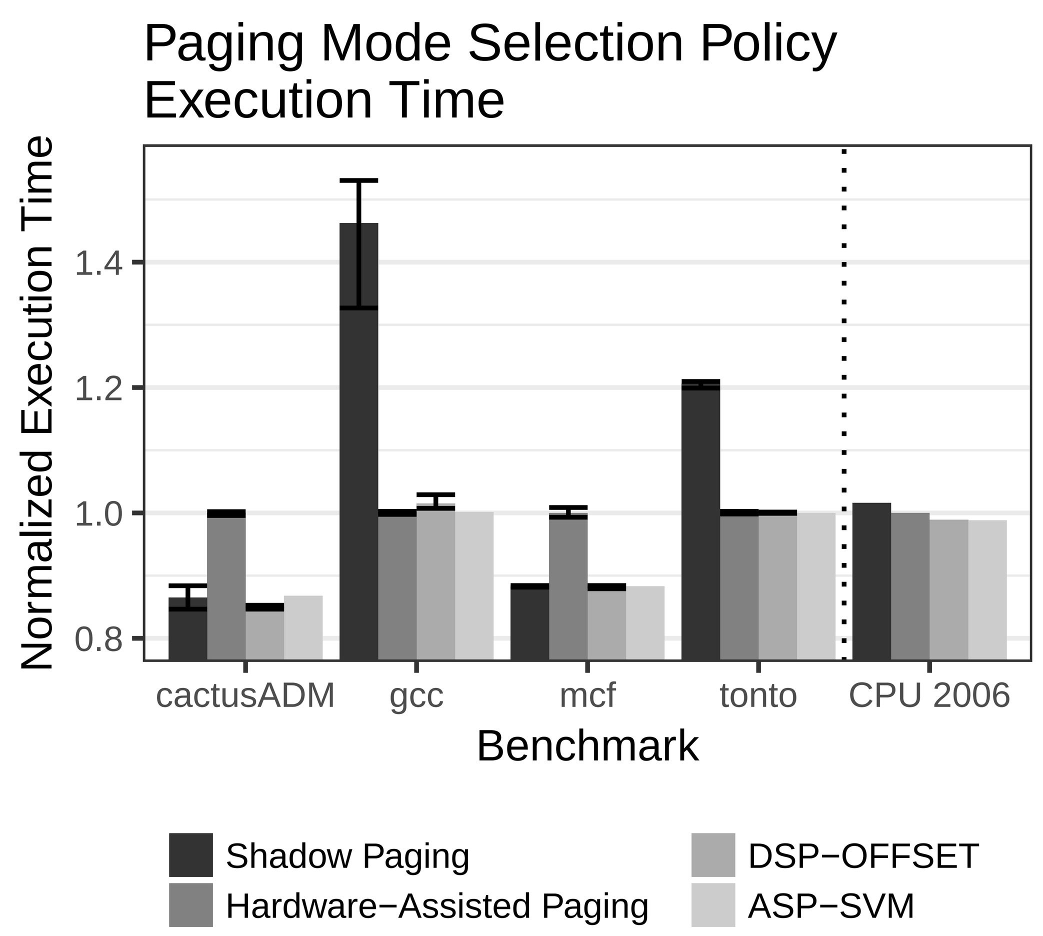


**Figure 2:** Benchmark execution times normalized to HAP for select benchmarks and the overall geometric mean.

Michigan Tech