CSR:Small: Adaptive Memory Resource Management in a Data Center – A Transfer Learning Approach

Laura E. Brown and Zhenlin Wang, Michigan Technological University

(CSR1422342, Oct. 2014 – Sept. 2017)

Objectives

- A study of transfer learning and its applications to the systems domain; transfer learning have been successfully applied to domains such as document/sentiment classification and WiFi localization, but there have been limited applications to the systems domain.
- An extensive set of applications are profiled and analyzed offline with the program behaviors and hardware configurations as the feature space and the scale of pressure the corresponding label used to create machine and transfer learning models. At runtime, hardware performance counters are inputs to the models to dynamically predict its cache performance with respect to cache allocation and characteristics of the co-run applications.
- Models to predict memory pressure that combines memory demand and duration prediction.
- Development of methods for feature selection, selection of hardware performance counters, that can be used for models related to memory and application performance. The feature selection methods require no *a priori* information and can be used to select counters to monitor for specific applications, or those that would work well in general.

Modeling Cross-Architecture Co-tenancy Performance Interference (CCGrid '15)

Methodology:

1. Profile to collect sensitivity curves and pressures for a set of benchmark programs, $X \in \{ A, B, \ldots, Z \}$, on multiple architectures with different hardware configurations, $\# \in \{ 1, 2, \ldots, N \}$.
2. Create models of the sensitivity curves for each program and architecture, $f_{x\#}$: bubble pressure → perf. degradation

$$y = f_{X\#}(x) = c + \frac{1-c}{\left(1+\frac{x}{a}\right)^b} \quad (logistic3)$$

3. Create cross-architecture sensitivity models, $g_{p,\#1,\#2}$, for each parameter, p = {*a*, *b*, *c*}, describing the relationships between architectures:

$$a_2 = g_{a,1,2}(a_1) = \alpha_a a_1^2 + \beta_a a_1 + \gamma_a$$
$$b_2 = g_{b,1,2}(b_1) = \alpha_b b_1^2 + \beta_b b_1 + \gamma_b$$
$$c_2 = g_{c,1,2}(c_1) = \alpha_c c_1^2 + \beta_c c_1 + \gamma_c$$

4. Create cross-architecture pressure models, $h_{1,2}$, describing the relationship between pressures, *PS*:

$$PS_2 = h_{1,2}(PS_1) = \alpha_{PS} PS_1 + \beta_{PS}$$

5. Given a new program *Y* on HW1, predict sensitivity functions and pressure on a hardware configuration HW2

$$\hat{p}_{Y2} = g_{p,1,2}(p_{Y1})$$
$$\widehat{PS}_{Y2} = h_{1,2}(PS_{Y1})$$

Prediction error (%) on pairwise co-run benchmarks (core2duo to i5)

Program	astar	bzip2	gcc	gobmk	hmmer	sjeng	libq	h264ref	omnetpp	perlbmk	xalanbmk
astar	2.24	0.91	0.05	0.54	1.54	0.12	4.92	0.48	1.04	0.09	0.78
bzip2	2.71	2.14	0.02	1.08	2.38	1.89	10.5	1.48	1.13	0.8	1.24
gcc	3.95	1.57	1.03	0.79	2.22	2.22	7.3	1.17	3.47	0.61	1.64
mcf	1.4	0.26	3.03	0.17	1.26	2.02	13.4	0.34	0.53	0.1	1.23
gobmk	0.57	0.1	0.48	0.08	0.4	0.07	2.93	0.12	0.12	0.17	0.23
libquantum	4.67	1.64	0.48	0.35	2.12	2.87	4.75	0.58	4.7	0.33	3.26
h264ref	0.92	0.67	0.37	0.44	0.86	0.71	2.17	0.47	0.35	0.23	0.31
sjeng	0.21	0.26	0.86	0.06	0.36	0.38	5.3	0.27	1.16	0.14	0.95
perlbench	1.49	2.12	2.56	1.81	1.51	1.65	6.38	1.33	3.49	1.89	2.71
omnetpp	5.08	0.64	2.03	0.65	1.56	2.45	15.9	0.33	0.46	0.35	0.27
xalancbmk	3.74	0.1	4.85	0.59	0.59	2.18	17.9	0.07	3.97	0.26	5.86

Program	bwaves	milc	zeusmp	leslie	namd	povray	sphinx	cactus	calculix	gamess	FDTD
astar	4.47	0.94	1.02	1.11	0.39	0.84	0.33	0.02	0.5	0.23	0.44
bzip2	1.49	1.34	2.29	0.56	1.21	1.16	2.16	1.15	0.35	0.56	1.49
gcc	8.55	1.24	2.46	3.36	1.27	1.44	2.68	1.43	0.25	0.57	0.53
mcf	5.68	1.23	0.67	1.46	1.24	1.16	2.47	0.69	0.43	0.4	3.34
gobmk	1.81	1.12	0.14	0.12	0.18	0.34	0.07	0.5	0.03	0.01	0.59
libquantum	12.1	3.47	3.46	6.71	1.95	1.23	2.76	2.61	0.04	0.13	1.92
h264ref	1.02	0	0.61	0.5	0.35	0.21	0.74	0.46	0.27	0.03	0.12
sjeng	2.1	2.51	0.56	1.74	0.32	0.3	0.38	0.73	0.02	0.15	2.67
perlbench	1.61	3.91	3.47	3.24	0.33	0.11	2.86	2.2	0.75	0.51	4.36
omnetpp	3.5	3.22	1.09	1.1	1.88	1.78	0.9	2.25	0.07	0.32	5.08
xalancbmk	4.11	5.19	0.02	5.19	1.43	1.7	6.39	2.3	0.18	0.28	6.24

Conclusions:

- In general, predicted performance degradation is less than 2%.
- A few benchmarks are problematic with higher error, e.g., *libquantum*

Selective Switching Mechanism in VMs via Support Vector Machines and Transfer Learning (ML '15)

Goal:

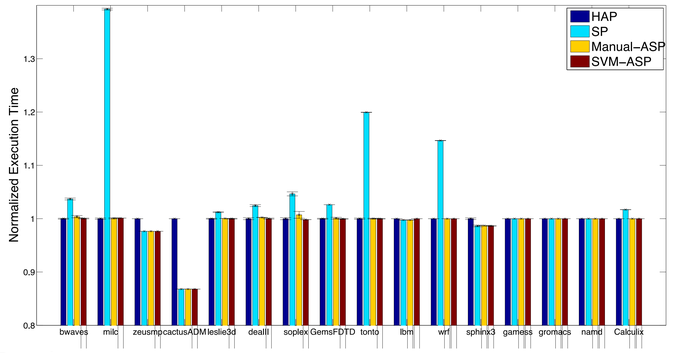
- Learn a decision model to dynamic select paging mode between shadow pages (SP) and hardware assistant paging (HAP)
- Transfer model to new hardware with limited profiling required

Methodology:

1. Collect training data from running SPEC CPU2006 INT benchmarks in HAP and SP mode; multiple training samples are collected running in one phase alone or switching modes for each phase of a benchmark
   a. Collected features are TLB misses counts and page fault counts
   b. Time-gain-ratio labeling indicates if switching gives at least a 5% performance gain
2. A linear SVM model was used as the decision function for the dynamic switching mechanism:

$$2.11TLB - 29.95PF - 24.70 = 0$$

   This linear model was encoded in the XEN hypervisor to select switching modes
3. A transfer learning algorithm TrAdaBoost is used to learn a model for a new architecture (AMD) using only a limited amount of new profiled data combined with the original data from an Intel architecture.



Results:

- Running times of benchmarks were compared using HAP only, SP only, a hand-tuned switching mode - Manual-ASP, and the new model SVM-ASP
- The model trained on the SPEC CPU2006 INT benchmarks is tested on the SPEC CPU2006 FP benchmarks (figure above)
- Use of knowledge from the source domain (Intel data samples) may improve learning in the target domain (AMD)

Train Set	Accuracy (%)	Train Set	Accuracy (%)
Intel-All	52		
AMD4	90 ± 10.0	Intel + AMD4	94 ± 3.9
AMD6	90 ± 11.8	Intel + AMD6	96 ± 5.6
AMD8	96 ± 2.2	Intel + AMD8	97 ± 1.6
AMD10	96 ± 1.8	Intel + AMD10	97 ± 2.2

Intelligent Sampling of Hardware Performance Counters for Performance Prediction

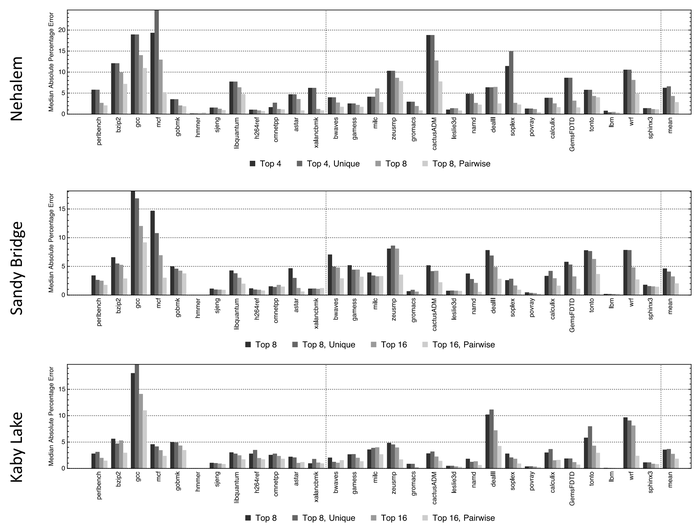
Overview:

- Present a method for fine-grained performance modeling through strategic sampling of available architecture events, Attribute Efficient Regression (AER)
  - Constrained by number of performance monitor counters available; limits number of events-hardware performance counters (HPCs) to observe
  - Assumes no *a priori* knowledge of event statistics
- AER's regression coefficients, ***w***, are used to rank and select events for individual benchmarks or a common set across benchmarks
- With the set of selected events a model to predict performance is learned via least squares

Architecture	Nehalem Intel Core i5	Sandy Bridge Intel Core i5	Kaby Lake Intel Core i7
Processor (GHz)	2.8	3.3	3.6
Cache (MB)	8	6	8
Memory (GB)	4	4	32
HPCs, *k*	4	8	8
Events	213	176	117

Results:

- Across three architectures, the rankings of events is mostly consistent (spearman rank correlation coefficient > 0.65)
- Prediction performance error decreases with the number of events sampled (Top *k* vs. Top 2*k*)
- In general, error decreases when selected events are restricted to a single element of each class (Top *k*, Unique); exceptions include Nehalem: *mcf* and *soplex*
- Inclusion of higher-order relationships (pair-wise relationships – Top *k*, Pairwise), has the lowest error across architectures and benchmarks



References

Kuang, W.; Brown, L.E.; and Wang, Z. **Modeling Cross-Architecture Co-Tenancy Performance Interference.** In *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 231-240, 2015.

Kuang, W.; Brown, L.E.; and Wang, Z. **Selective Switching Mechanism in Virtual Machines with Support Vector Machines and Transfer Learning.** *Machine Learning*, 101(1-3): 137-161, 2015

Collaborators

- **Michigan Tech:** Wei Kuang, Jason Hiebel
- **Western Michigan University:** Steven Carr

Michigan Technological University 1885