Applying SVM to data bypass prediction in multi core last-level caches

Warisa Sritriratanarak[1a)], **Mongkol Ekpanyapong**[2], **and Prabhas Chongstitvatana**[1b)]

[1] *Faculty of Engineering, Chulalongkorn University,*
254 Phayathai Road, Pathumwan, Bangkok 10330, Thailand
[2] *Asian Institute of Technology,*
Km. 42, Paholyothin Highway, Klong Luang, Pathumthani 12120, Thailand
a) *warisa.sr@student.chula.ac.th*
b) *prabhas@chula.ac.th*

Abstract: Bypassing emerged as a performance improvement method for shared Last-Level Caches (LLC) in multicore processors where large data portions are never reused, wasting system resources. This paper proposes an alternative method to predict data bypassing using Support Vector Machine (SVM). Based on access traces obtained from a simulator, SVM is trained to generate bypass models which are integrated into the simulator to quantify LLC performance improvements. Results show that SVM can classify which data to bypass, improving LLC performance, achieving an average 6.72% miss rate decrease across SPLASH2 benchmark combinations.

Keywords: cache bypassing, cache hit rate, last-level cache, SVM
Classification: Integrated circuits

References

[1] M. Kharbutli and Y. Solihin: IEEE Trans. Comput. **57** (2008) 433. DOI:10.1109/TC.2007.70816
[2] L. Li, D. Tong, Z. Xie, J. Lu and X. Cheng: PACT (2012) 315. DOI:10.1145/2370816.2370862
[3] N. Duong, D. Zhao, T. Kim, R. Cammarota, M. Valero and A. V. Veidenbaum: MICRO (2012) 389. DOI:10.1109/MICRO.2012.43
[4] G. S. Tyson, M. K. Farrens, J. Matthews and A. R. Pleszkun: MICRO (1995) 93. DOI:10.1145/225160.225177
[5] A. González, C. Aliagas and M. Valero: ICS (1995) 338. DOI:10.1145/224538.224622
[6] T. L. Johnson, D. A. Connors, M. C. Merten and W. M. W. Hwu: IEEE Trans. Comput. **48** (1999) 1338. DOI:10.1109/12.817393
[7] J. Jalminger and P. Stenstrom: ICPP (2003) 294. DOI:10.1109/ICPP.2003.1240592
[8] J. A. Rivers and E. S. Davidson: ICPP (1996) 154. DOI:10.1109/ICPP.1996.537156
[9] H. Lim, J. Kim and J. W. Chong: IEICE Electron. Express **7** (2010) 850. DOI:10.1587/elex.7.850
[10] L. Xiang, T. Chen, Q. Shi and W. Hu: ICS (2009) 68. DOI:10.1145/1542275.1542290

© IEICE 2015
DOI: 10.1587/elex.12.20150736
Received September 2, 2015
Accepted October 14, 2015
Publicized October 28, 2015
Copyedited November 25, 2015

[11] R. Ubal, B. Jang, P. Mistry, D. Schaa and D. Kaeli: PACT (2012) 335. DOI:10.1145/2370816.2370865
[12] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta: ISCA (1995) 24. DOI:10.1145/223982.223990
[13] P. Vateekul, M. Kubat and K. Sarinnapakorn: IDA **18** (2014) 717. DOI:10.3233/IDA-140665
[14] S. M. Khan, Y. Tian and D. A. Jimenez: MICRO (2010) 175. DOI:10.1109/MICRO.2010.24

1 Introduction

Memory system is today's computer system performance bottleneck. Sophisticated workloads on multi-core processors perform data requests which cannot be easily predicted through traditional algorithms or methods. The memory system, especially shared last-level cache (LLC), is impacted by these accesses; different types of concurrent workloads result in lower hit rate or even thrashing when the working set is larger than the cache size. In order to better manage limited cache space, selectively bypassing data from the LLC has been shown to ameliorate the problem [1, 2, 3].

Temporal locality in the LLC is inverted from inner-level caches [2]; several LLC blocks are never reused before eviction, deteriorating its performance. Recent works proposed methods to improve LLC hit-rate by avoiding allocating every datum, selectively bypassing not-reused data. The most common method is to predict when the data will be reused by collecting data access profiles into table(s) and use counter to selectively cache the data. Most works require a large space to store table(s) and involve updates on access to maintain accuracy, wasting bandwidth and cache space when the patterns become more complex than the counter can predict.

In this work, we convert the prediction into a binary classification problem, deciding whether each datum should be bypassed or not. We propose bypass prediction using Support Vector Machine (SVM) to learn data access patterns on LLC. The output model (Bypass classifier) automatically determines whether to bypass the data. SVM is used because of its great performance on binary classification. Our method proves that it is possible for SVM to learn access patterns not easily identifiable through traditional methods. With proper training data and parameters, SVM can provide a bypass model which improves LLC performance. The main contribution of this work is a proof of concept of SVM as a method for determining data bypass. SVM-generated bypass classifier's integration in hardware is outside the scope of this paper, reserved for future work.

2 Related work

Determining cache bypassing relies on analyzing which data will be reused and their reuse distance. A number of works focus on predicting which data will be reused using predictors and information collected based on either the program counter (PC) [2, 4, 5] or the address [6, 7, 8].

© IEICE 2015
DOI: 10.1587/elex.12.20150736
Received September 2, 2015
Accepted October 14, 2015
Publicized October 28, 2015
Copyedited November 25, 2015

Tyson et al. seminal work [4] used the PC to block some instructions from allocating data on cache. Instructions which create a lot of misses are not allowed to allocate, identified by a two-bit predictor; later used in [7] to predict which data should be bypassed based on addresses instead of PC. Similarly, Johnson et al. [6] store counters in Memory Access Table, looked up to decide bypassing.

Since prior works bypass the lines from all caches, additional hardware is required to keep the cache inclusion property. In [1], Kharbutli et al. relaxed cache inclusion and bypassed data from L2 (LLC) only, requiring less hardware overhead. Similarly focused on LLC, [9] creates new replacement policy on last-level shared cache (L2) by adding two tables and counter. Later, Xiang et al. [10] suggested that bypassing only never-reused lines is not enough; lines which are least reused should also be bypassed. Apart from bypassing, PDP [3] uses reuse prediction to create a new replacement policy and protect blocks from replacement; if all are protected, then the incoming block is bypassed. OBM [2] involves predicting whether the incoming or the victim block will be reused first and keeping the closest reused one.

To the best of our knowledge, all previous work in cache data bypassing has relied on ad-hoc prediction methods; our work is the first presenting the use of machine learning applied to memory system performance.

3 Materials and methods

We used Multi2Sim simulator [11] to model a quad-core processor with X86 ISA, non-inclusive, write-through LRU cache; parameters depicted in Table I.

Table I. Parameters of the simulated cache

L1 Instruction Cache	8 KB, 32B-line size, 2-way
L1 Data Cache	8 KB, 32B-line size, 2-way
L2 Shared Last-level Cache	64 KB, 32B-line size, 4-way

Cache size is small in relation to benchmark memory usage (working set) in order to force contention. Since the purpose is to prove the SVM's ability to selectively bypass the data, two-level cache is sufficient to demonstrate the feasibility, exploiting the relationship between each core's private cache and the shared LLC. Fig. 1 illustrates the system structure.

Benchmark combinations are run, and cache access traces (Instruction and Data) from each processor core are collected. Subsets of these traces are used for SVM training, in order to output a bypassing model. This model is then integrated on a modified Multi2Sim cache, performing bypassing, and LLC hit rates are compared (only for the trace subsets not used for SVM training, with and without bypassing, i.e., different training and testing data).

© IEICE 2015
DOI: 10.1587/elex.12.20150736
Received September 2, 2015
Accepted October 14, 2015
Publicized October 28, 2015
Copyedited November 25, 2015

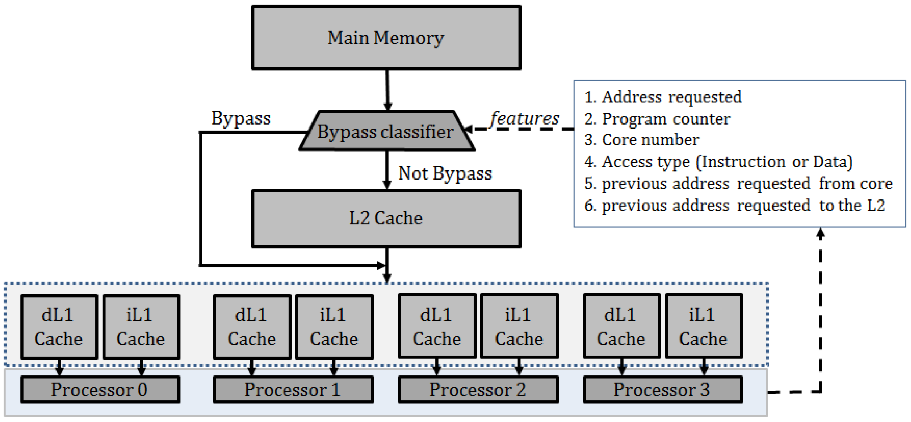

Fig. 1. System structure. The dotted arrows represent the information required by the classifier. The solid arrows represent data flows when cores request data from main memory (L2 missed).

3.1 Benchmarks

To demonstrate the effectiveness of SVM prediction, simulator runs multithreaded SPLASH2 [12] benchmarks. We created 7 benchmark combinations by randomly selecting 4 out of 11 SPLASH2 benchmarks to run simultaneously, 4 threads per benchmark. Each combination is listed in Table II and simulated for 200 million committed instructions after fast forwarding the first 100 million instructions.

Table II. Benchmarks combination

com 1	Raytrace, Radiosity, Water-nsquared, Water-spatial
com 2	Radiosity, Lu, Ocean, FFT
com 3	Cholesky, Fmm, Water-nsquared, Radix
com 4	Barnes, Fmm, FFT, Radix
com 5	Ocean, Lu, Barnes, Water-spatial
com 6	Fmm, Cholesky, Lu, Raytrace
com 7	FFT, Barnes, Radiosity, Water-nsquared

3.2 Support vector machine

HR-SVM [13] is an SVM-based technique specifically tailored for hierarchical multi-label classification, suitable for binary classification with the imbalanced class issue. To provide data for the HR-SVM, we generate the following attributes each time the cores requested data from cache: address requested, program counter, core number which requested data, access type (Instruction or Data), previous address requested from the core, previous address requested to the LLC. This set of data is referred to as *features*, scaled appropriately for SVM training. Accesses to the LLC (misses from L1 caches) are extracted from the trace and separated into two parts: the first one million accesses are training data and the rest is reserved for testing. SVM training is supervised, i.e., the training data must be marked as bypass or not. Marking the training data uses our knowledge of future behavior, similar to the optimal lookahead in [2]; a look-ahead window of size *n* is used to check for

© IEICE 2015
DOI: 10.1587/elex.12.20150736
Received September 2, 2015
Accepted October 14, 2015
Publicized October 28, 2015
Copyedited November 25, 2015

reuse. E.g., when address A is accessed, LLC trace is checked for n addresses, determining whether or not address A is reused. If so, then A will be marked as *to be reused* and will not be bypassed. If not, then A is marked as *to be bypassed* and will not be allocated in LLC. Experimental results suggest the most efficient window size for our experiment is $n = 5000$; varying window size between 1000 and 10000, we noticed no significant improvement for n greater than 5000. Benchmark combination 6 is an exception; further detail will be explained in the last section. For feasible training, sampling was performed to obtain adequate training data. Previous work has shown that a small fraction of data could represent the access pattern of the entire trace [14]. Empirical results showed that sampling every fifth address for a total of 100,000 data yielded adequate SVM training results.

3.3 Features and kernel functions

Four different SVM kernels with various parameters are tested to find the most suitable model: Linear, Polynomial, Radial basis function, and Sigmoid. Fig. 2 shows the percentage of miss rate decrease of each combination compared to the baseline achieved through bypassing for each SVM kernel.

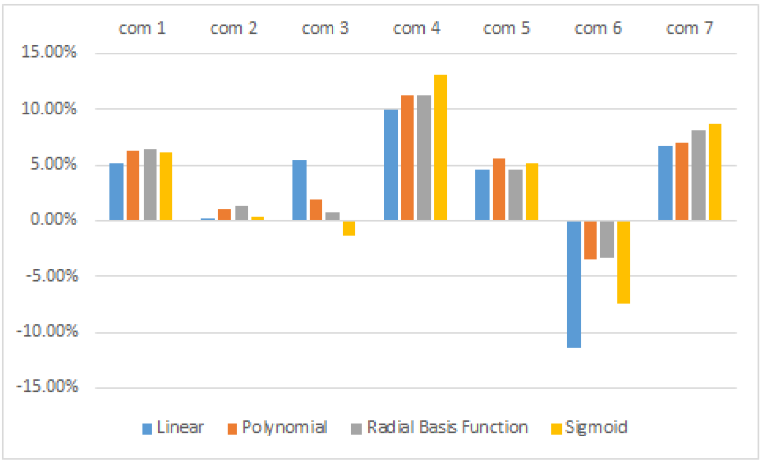

Fig. 2. Percentage of L2 miss rate achieved for each combination. The positive results mean miss rate decrease and the negative results mean miss rate increase.

4 Results and discussion

Fig. 2 displays the results for our combinations. We present the percentage of the cache miss improvement compare to the baseline, as predicted by four SVM kernels per combination. Evaluating results yields several conclusions: (1) cache miss rate is improved across combinations (except for combination 6). This shows SVM feasibility as a tool for bypass prediction; to the best of our knowledge, this is the first proof-of-concept of applying machine learning techniques to caches or memory systems. (2) Radial basis function is, across most configurations, the kernel function which yields the best gains. Experiments suggest that Radial basis function is the optimal kernel for memory systems behavior, paving the way for

© IEICE 2015
DOI: 10.1587/elex.12.20150736
Received September 2, 2015
Accepted October 14, 2015
Publicized October 28, 2015
Copyedited November 25, 2015

further studies. (3) Perhaps the most interesting result is the observation that combination 6 miss rate does not improve, regardless of the kernel. Experiments on this combination showed hit rate can be improved, using 7 features by adding the time of access as a 7th feature. The result is that the miss rate of combination 6 reduced from 29.52% to 28.83% improvement which could possibly translate as combination 6 access is time-sensitive. This analysis will provide clues into classifying SVM kernels and number of features across particular memory access behaviors in the future.

Across positive results for 6 features, SVM-predicted bypass yields an average miss rate decrease of 6.72%, which compared to related work such as [9] which achieves a 6.01% average, shows that SVM-predicted bypass can provide cache utilization comparable to ad hoc replacement policy mechanisms.

5 Conclusion

We presented a novel method to determine cache bypassing in LLC by using SVM-based prediction. The proposed method provides an alternative to traditional ad-hoc bypass methods, outperforming related work.

We showed the suitability of SVM for accurately predicting cache bypassing for shared LLC and characterized SVM kernel functions, features and training data look-ahead window-size for best prediction. Results show that a window-size of 5000, coupled with Radial basis function kernel and 6 features, is the most efficient training parameters combination for data-bypassing prediction across our work-loads.

Future work will focus on refining SVM kernel analysis for application suitability; especially determining why combination 6 differs from others, requiring a different set of training parameters for accurate prediction. Future work will also focus on efficient cache hardware implementation of bypass classifier.

Acknowledgments

This work was partially supported by Graduate School Thesis Grant, Chulalongkorn University and CP Chulalongkorn Graduate Scholarship.

© IEICE 2015
DOI: 10.1587/elex.12.20150736
Received September 2, 2015
Accepted October 14, 2015
Publicized October 28, 2015
Copyedited November 25, 2015