LLC Dead Block Prediction Considered Not Useful

Priyank Faldu Boris Grot

University of Edinburgh

Abstract

Dead block predictors (DBPs) improve cache efficiency by identifying blocks that have exhausted their useful lifetime in the cache and prioritizing them for eviction regardless of how much reuse these blocks have experienced. In doing so, DBPs go beyond insertion policies that target only those blocks that have no cache reuse. But how much value do DBPs add over insertion policies?

This work examines the opportunity for DBP at the LLC and concludes that its value-add is low over a state-of-the-art insertion policy. Our key result is that LLC evictions are dominated by blocks having no reuse. Even with the optimal replacement policy that maximizes cache reuse through perfect knowledge of the future, an average of 78% of LLC evictions have not experienced any hits. For the remaining evictions that experience LLC reuse, only a fraction is predicted by a state-of-the-art DBP scheme, and the accuracy of these predictions is low. Our first-order limit study shows that a likely coverage ceiling for DBP over the best performing insertion policy is just 6.9% of all LLC evictions. Based on these findings, we argue that the higher complexity of DBP is not justified.

1 Introduction

Existing cache management strategies can be divided into two categories: dead block predictors (DBP) and insertion policies. DBPs try to predict when a block has reached the end of its useful lifetime on chip and turn it into a replacement target before it has reached the end of the LRU chain [1, 2, 3, 4, 5, 6, 7, 8]. Insertion policies seek to identify blocks that do not see any reuse in the cache (i.e., are dead on arrival) and prioritize them for eviction [9, 10, 11].

In principle, both DBPs and insertion policies improve cache performance by rapidly evicting and/or entirely bypassing blocks that will not be useful in the near future in order to allow other blocks to persist and see further hits. The difference, of course, is that insertion policies are limited to just those blocks that have no cache reuse at all, referred to as *0-hit* blocks, whereas DBPs can predict dead both 0-hit and *1+hit* blocks (i.e., those that get one or more hits). Since DBPs can cover a wider range of blocks than insertion policies, they intuitively appear more powerful and, therefore, more useful.

In this work, we characterize reuse behavior at the LLC for SPEC'06 workloads to understand the potential opportunity of DBP over insertion policies. We consider Belady's optimal replacement policy (OPT) [12] to draw an upper bound on block reuse. Even with perfect knowledge of the future, 78% of the blocks evicted under OPT have no reuse and would not need DBP for prediction.

To understand a more realistic distribution, we consider a state-of-the-art DBP – the Sampling Dead Block Predictor (SDBP) – and observe that the vast majority of blocks that it predicts dead are actually 0-hit blocks. Moreover, the majority of the evicted blocks that SDBP does *not* predict dead are also 0-hit blocks. On average, 84% of all blocks evicted under SDBP does not see LLC reuse. As such, they can be easily managed using a simple insertion policy and do not require a DBP.

The opportunity for DBP is thus limited to the blocks that see LLC hits but are eventually evicted. While SDBP predicts dead the majority of these blocks, the accuracy of these predictions is considerably lower than that of predictions for 0-hit blocks. A limit study for the prediction opportunity for 1+hit blocks under best performing insertion scheme shows that just 6.9% of all LLC evictions are a likely coverage ceiling for DBP. Thus, our key conclusion is that the opportunity for DBPs to improve upon insertion policies is small.

Given that blocks that enjoy LLC reuse represent the minority of LLC evictions while being difficult to predict with high accuracy, we argue that simple LLC insertion and/or bypass schemes are superior to DBPs, especially when storage costs and design complexity are taken into account. Thus, rather than focusing on high-complexity DBP schemes, improving the coverage and accuracy of predictions for 0-hit blocks may be more profitable. To prove our point, we show that enhancing a state-of-the-art insertion policy, SHiP [11], with LLC *bypass* improves coverage, accuracy, and performance at virtually no extra hardware cost.

Our contributions are as follows (all numbers are averages):

- 78% of all LLC evictions are of blocks that see no LLC hits for optimal replacement policy. Such blocks can bypass the LLC entirely and do not require a DBP.
- SDBP [4], a state-of-the-art DBP, predicts 85% of all LLC evictions. Of these, only 12% observe LLC hits and require DBP for prediction. However, the prediction accuracy for these blocks is low (67%) as compared to that for 0-hit blocks (78%).
- A simple LLC bypass scheme on top of SHiP [11], a state-of-the-art insertion policy, predicts 75% of all blocks brought on chip with 81% accuracy. By minimizing LLC pollution through bypassing while avoiding premature evictions of blocks that enjoy LLC reuse, our bypass scheme achieves the highest performance gain of 6.6% over LRU, compared to 4.9% for SDBP and 5.6% for baseline SHiP.

2 Evaluated Techniques

Our study considers a state-of-the-art dead block predictor, *Sampling Dead Block Predictor (SDBP)*, a state-of-the-art insertion policy, *PC Signature-Based Hit Predictor (SHiP)*, and Belady's *Optimal Replacement Policy (OPT)* [12].

SDBP[4] is a dead block predictor that correlates "last touch" to the block with the PC of the memory instruction making the touch. SDBP is trained by a separate structure called a *sampler* with fewer sets and relatively smaller associativity than the LLC. While the sampler implements LRU replacement, the cache can use any insertion/replacement policy. The blocks that are dead-on-arrival are not inserted into the LLC and, instead, are directly bypassed to the higher level cache. If such blocks map to the sampler sets, they are inserted into the sampler to provide continuous training.

SHiP[11] is a state-of-the-art insertion technique that combines the RRIP [10] replacement policy with history of re-reference behavior of the blocks' previous lifetime on chip. For the latter, SHiP correlates a block's re-reference behavior at the LLC (i.e., whether it saw any hits during its previous on-chip residency) with the PC of the instruction that brought the block on chip. Blocks that are not likely to see LLC hits are inserted with a distant re-reference interval prediction that prioritizes them for eviction over blocks that are likely to see a future hit.

We also consider a bypass-only policy that avoids allocating dead-on-arrival blocks in the LLC altogether. To implement the policy, we combine SHiP with bypassing for blocks predicted to have a distant re-reference interval. Since bypassed blocks do not get an opportunity to see hits (i.e., they cannot get re-trained), we override the bypass decision for a fraction of the blocks and insert them into the cache.

Finally, we study OPT which uses its perfect knowledge of the future to replace the block whose next use is the farther in a given set.

Core Model	OoO: 8-stage 4-wide pipeline 128-entry ROB
L1I Cache	Private 32KB 4-way, 64B-blocks 2-cycle load-to-use
L1D Cache	Private 32KB, 8-way 64B-blocks 2-cycle load-to-use
L2 Cache	Private, Unified 256KB, 8-ways, 64B-blocks 10-cycle hit latency
L3 Cache	Shared, Unified 2MB per core, 16-way, 64B-blocks 30-cycle hit latency
Main Memory	200-cycle access latency

Table 1: System parameters for simulation of SPEC'06 applications

OPT provides a lower bound on misses for a given memory trace and cache configuration. By exploiting its knowledge of future references, OPT also maximizes cache reuse, thus increasing the number of hits that a block may see.

3 Methodology

3.1 Infrastructure and Benchmarks

We study SPEC CPU 2006 benchmarks using a modified version of CMP$im [13] provided with the JILP Cache Replacement Championship [14]. Table 1 summarizes the features of the simulated processor.

We use *SimPoint* [15] to identify up to 6 simpoints of one billion instructions each representing a different phase of a workload. The weights used to calculate the overall performance are generated by the SimPoint tool. Each program is run with the first *ref* input provided by *runspec* command. For each run, the simpoint is used to warm microarchitectural structures for 500 million instructions, then measures and report the result for the subsequent one billion instructions. The result reported for each benchmark is the weighted average of the results for the individual simpoints.

For the sake of understanding the source of improvement for different cache management techniques, we classify all 29 SPEC'06 applications into three categories - *High Gain*, *Mixed Gain*, and *No Gain.* If application's performance improves by more than 10% thanks to either SDBP or SHiP, it is classified into the High Gain category. If the performance difference is within 0.5% for both techniques, it is classified into the No Gain category. The rest of the applications are classified into the Mixed Gain category. Finally, we found that the working set for *gamess* and *povray* fits entirely in the 2MB cache, so no evictions are observed. As a result, we exclude these two benchmarks from our studies. Table 2 shows the classification of each benchmark.

Category	Applications
High Gain	mcf, cactusADM, soplex, astar, sphinx3, xalancbmk
Mixed Gain	perlbench, bzip2, bwaves, milc, zeusmp, gromacs, leslie3d, dealII, calculix, hmmer, sjeng, GemsFDTD, libquantum, h264ref, tonto, omnetpp, wrf
No Gain	gcc, namd, gobmk, lbm, gamess, povray

Table 2: Classification of SPEC'06 applications

3.2 Cache Management Schemes

OPT: We implement Belady's MIN replacement policy [16] with bypassing. If the next reuse of a block that missed on-chip is farther than the next reuse of all the blocks in a set, the block is bypassed to the higher level cache without getting allocated in LLC.

SDBP: We use SDBP implementation available on the JILP Cache Replacement Championship website. SDBP implements a 13-way set associative sampler with a 3-level prediction table of 2-bit confidence counter each. There are 128 sampler sets.

SHiP: We use the RRIP code provided on the JILP Cache Replacement Championship website and extend it to implement SHiP. Our implementation uses a 2-bit RRIP replacement policy with 3-bit confidence counters in the prediction table. We find that SHiP benefits from sampling to filter out the noise. Empirically, we determine that 128 sets work best for our simulated processor.

SHiP+Bypass: We augment the baseline SHiP policy as described above with LLC bypassing for PCs predicted to have a distant rereference interval. To ensure continuous training, a fraction of bypassable blocks is inserted into the LLC using the normal SHiP policy; i.e., with distant re-reference interval, making them quickly eligible for eviction if they don't see a hit. We empirically determine that overriding the bypass decision for 20% of the predicted blocks maximizes performance.

For both SDBP and SHiP-based schemes, we use PC-indexed prediction tables. As we are interested in understanding the full potential of each policy, we do not limit the number of PCs that can be tracked, effectively eliminating table conflicts as a source of performance loss. In practice, we found that tables with 16K entries provides a nice balance between performance and storage cost, corroborating prior work [11].

We also evaluated a Counting dead block predictor (CDBP) [3]. However, we found it to be vastly inferior to SDBP in coverage, accuracy, and performance, corroborating prior work [4]. Thus, we do not include results for CDBP in our analysis, but note that the trends and conclusions for CDBP parallel those for SDBP.

4 Results

Effectiveness of a cache management technique can be measured by its prediction *coverage* and *accuracy*. Coverage is defined as the ratio of predictions to evictions. A high coverage (trending toward 1.0) indicates that the majority of LLC evictions were predicted dead. Accuracy is the fraction of predictions that is *correct*.

To provide insight into the behavior of the different cache management schemes, we classify all evictions into four categories - *Predicted 0-hit*, *Not-predicted 0-hit*, *Predicted 1+hit* and *Not-predicted 1+hit*. If DBP predicts a block dead, the block is classified into a Predicted category. If it does not find a dead block, an LRU block is evicted and classified as Not-Predicted. For SHiP, the block predicted to have a distant re-reference interval (i.e., not expected to hit in LLC) is classified into a Predicted category. Because SHiP only predicts 0-hit blocks, it does not have a Predicted 1+hit category.

To measure the accuracy of the prediction, we maintain a separate *Shadow Victim Cache* (SVC) with the same number of sets and associativity as the original cache. When a block is evicted from the cache, it is inserted into the SVC with a bit indicating whether the block was predicted dead. Upon an LLC miss, the SVC is queried to check if the block was recently evicted. If the block hits in SVC and was predicted dead in LLC, the block is marked as a wrong prediction and is removed from the SVC. If, on the other hand, the predicted block is evicted from the SVC without a hit, the prediction is considered correct.

4.1 0-hit Blocks Dominate Evictions

Figure 1 quantifies the dominance of 0-hit blocks as a fraction of all the evicted blocks (including bypass) under SDBP, SHiP & OPT. The three groups of bars in each figure correspond to High Gain, Mixed Gain, and No Gain application categories, in that order.

OPT, knowing the future, allows maximum number of blocks to incur hits. Still, on an average, 78% of the evicted blocks do not see any reuse. The fraction is higher for both SDBP and SHiP, with 84% and 89%, respectively, of all blocks evicted from the LLC having 0 hits.

The reason why the fraction of 0-hit LLC evictions is higher for practical schemes is twofold. First, they do not have the perfect knowledge of the future, which fundamentally limits their ability to anticipate reuse for blocks (or PCs) lacking a regular behavior. Secondly, blocks that are predicted 0-hit are evicted early, depriving them of the opportunity to see future hits. For instance, on *calculix*, SDBP predicts

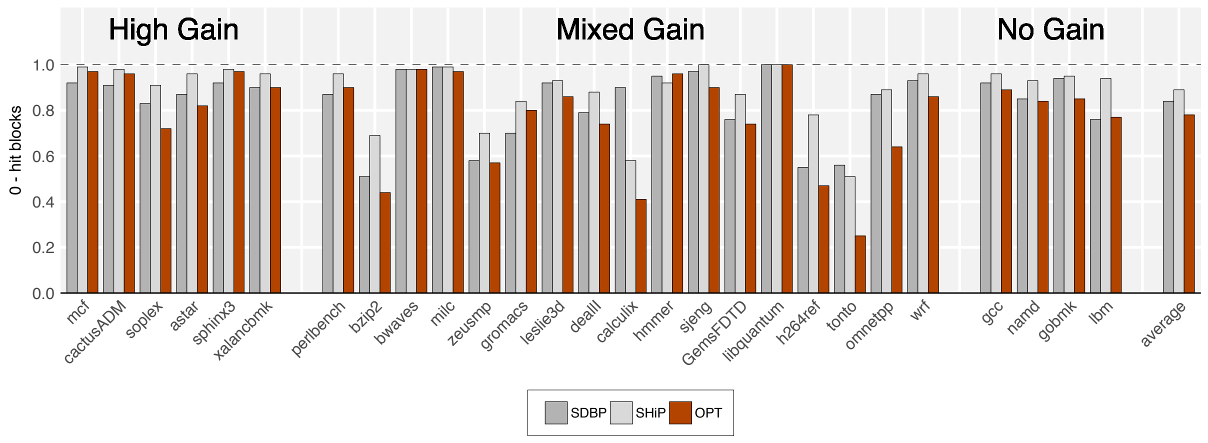
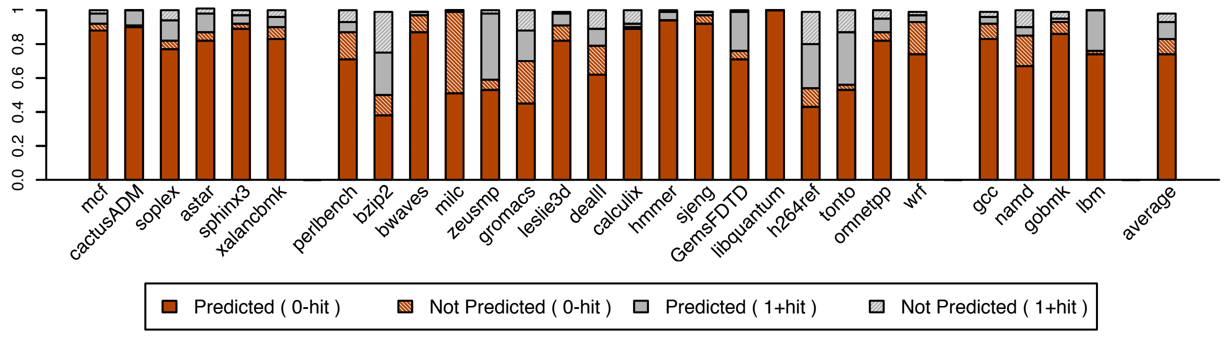
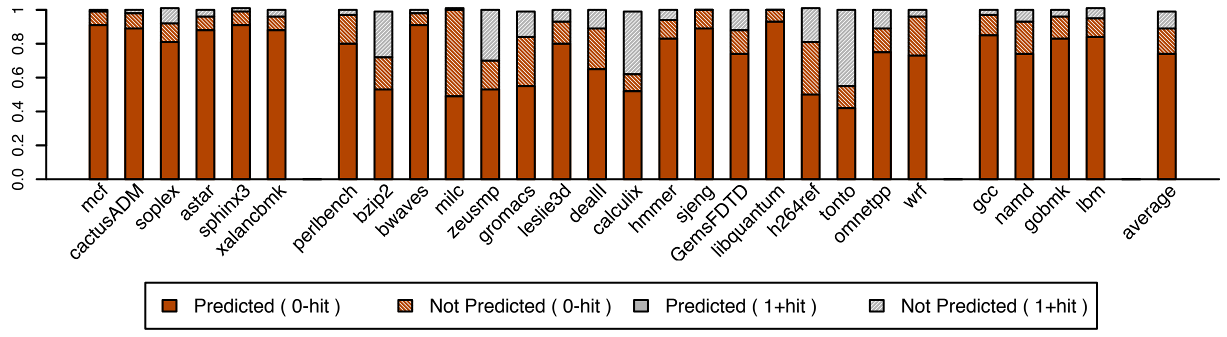


Figure 1: Fraction of 0-hit blocks



(a) SDBP



(b) SHiP

Figure 2: Break up of blocks evicted from the LLC

90% of the blocks as 0-hits; however, the accuracy of these predictions is poor at just 12%. In contrast, OPT observes that only 41% of all LLC evictions have 0-hits.

4.2 Dissecting Existing Techniques

In this section, we analyze the effectiveness of SDBP and SHiP by dividing the predicted evictions into 0-hit and 1+hit categories. We similarly divide the not-predicted evictions to understand the further potential of these schemes.

Figures 2a and 2b show the breakdown for

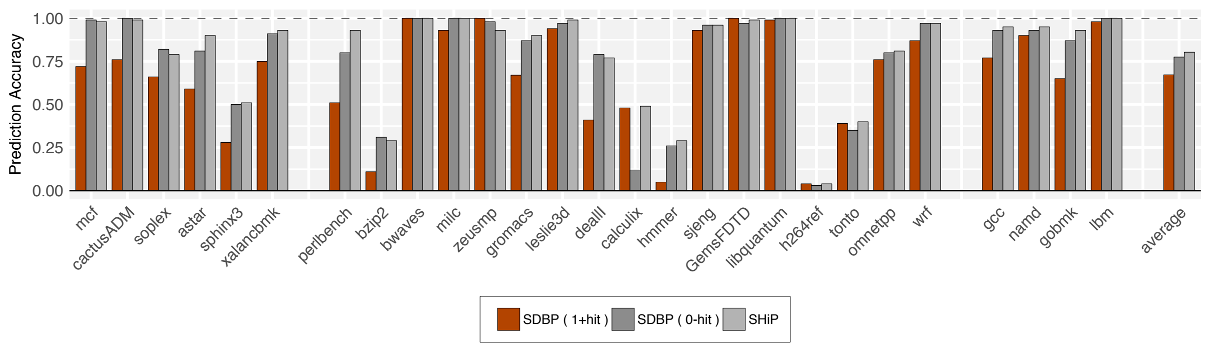


Figure 3: Prediction Accuracy for SDBP and SHiP

SDBP and SHiP, respectively. On average, SDBP has higher coverage than SHiP (85% vs 74%). This is expected as SHiP is limited to predicting only 0-hit blocks, whereas SDBP can predict both 0-hit and 1+hit blocks. However, only 11% of SDBP's evictions are predictions to 1+hit blocks (solid grey slices in the figure); the vast majority (74%) of evictions are predictions to 0-hit blocks (solid red slices in the figure) that are also covered by SHiP. If we only consider the High Gain applications (left-most group of bars), SDBP's coverage improves to 93%, of which 1+hit comprise just 8% of total evictions. This indicates that DBP offers little additional coverage over an insertion scheme, particularly for workloads that stand to benefit the most from cache management.

We further analyze the blocks that are evicted without prediction (Not-Predicted 0-hit and Not-Predicted 1+hit – shown with angled stripes in the figure) to understand the remaining opportunity for DBP. Together, these two categories account for 15% of LLC evictions under SDBP; of these, just a third are due to Not-Predicted 1+hit (grey angled stripes), meaning that in the limit, a DBP could predict only an extra 5% of LLC evictions. For High Gain workloads, that fraction shrinks to just 3%.

Figure 3 shows prediction accuracy of SDBP and SHiP. For SDBP, we show the accuracy of 0-hit and 1+hit predictions separately to gain insight into predictability for each kind of blocks. For 0-hit blocks, SDBP achieves a 78% prediction accuracy, closely tracking SHiP's accuracy of 81%. However, prediction accuracy of 1+hit blocks (67%) is significantly lower than that of 0-hit blocks. This gap is even wider if we look at the High Gain applications where SDBP's average prediction accuracy for 0-hit blocks is 84% versus 63% for 1+hit blocks.

Summary: Our results show that DBP's opportunity is limited not only by the low fraction of 1+hit LLC evictions, but also by the challenge of accurately predicting these blocks. In contrast, 0-hit blocks represent the majority of the evictions *and* present an easier prediction target in terms of accuracy. Furthermore, these 0-hit blocks do not require DBP and, instead, can be successfully predicted through a simple insertion scheme.

4.3 SHiP+Bypass

While SHiP provides much of the coverage of SDBP, a small gap remains due to SDBP's predictions for 1+hit blocks. In this section, we investigate whether a simple enhancement to SHiP can partly close the coverage gap.

We observe that SHiP allocates *all* blocks in the LLC, including those that are predicted to not see any hits. Since such blocks represent the majority of all allocations, we investigate whether we can improve cache efficiency by bypassing these blocks, thereby avoiding an unnecessary eviction.

Our results show that SHiP+Bypass yields a small coverage improvement to 75%, versus 74% for SHiP. For High Gain applications, SHiP+Bypass improves the coverage from 88% to 91%. The accuracy of SHiP+Bypass remains

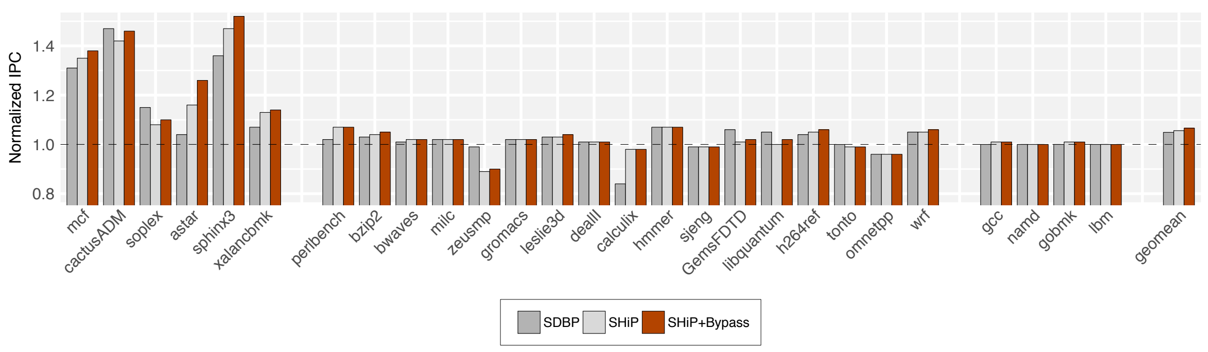


Figure 4: Normalized IPC for SPEC'06 applications over LRU

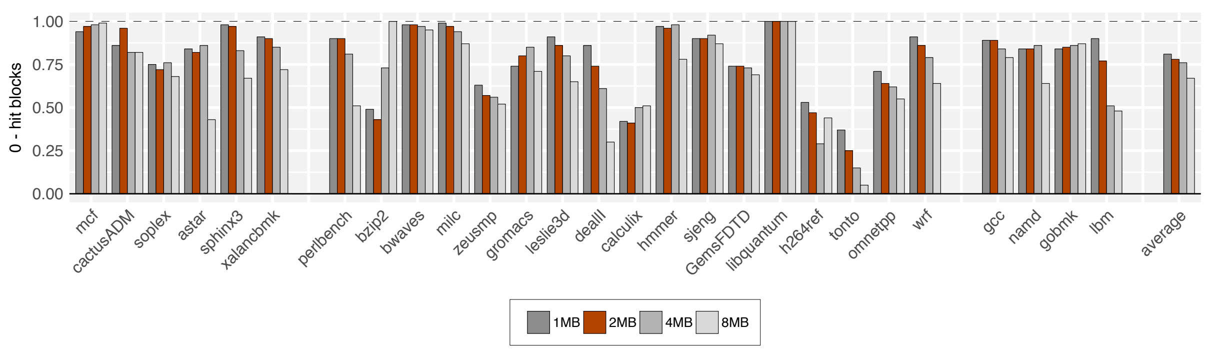


Figure 5: Fraction of 0-hit blocks under OPT

the same as that of baseline SHiP – 81%.

4.4 Performance

Figure 4 shows overall performance improvement for all evaluated cache management schemes over LRU. Our results corroborate prior work: in general, SHiP matches or outperforms SDBP. *Zeusmp*, *cactusADM*, *soplex*, *GemsFDTD* and *libquantum* are the only applications for which SDBP outperforms SHiP through accurate predictions of 1+hit blocks (Figure 3).

SHiP+Bypass consistently outperforms SHiP and also narrows the gap with SDBP for applications listed above. SHiP+Bypass improves the average performance of all SPEC'06 applications by 6.6%, versus 5.6% for SHiP and 4.9% for SDBP. For High Gain applications, the performance improvement is 29.9% for SHiP+Bypass, 25.7% for SHiP, and 22.5% for SDBP.

4.5 Sensitivity Analysis

In this section, we perform a sensitivity analysis of 0-hit blocks for different LLC sizes using OPT. Figure 5 shows 0-hit blocks as a fraction of all evictions for LLC sizes of 1, 2, 4 and 8 MB. As cache size increases, blocks stay longer in the cache, which in turn leads to fewer evictions for 0-hit as well as 1+hit blocks. While the average fraction of 0-hit blocks reduces for larger cache sizes, it still significantly dominates all the evictions. On an average, under OPT, 67% of the evicted blocks have no reuse for 8MB LLC as compared to 76%, 78% and 81% for 4MB, 2MB and 1MB LLC respectively.

It is important to point that for many applications, their working set size largely fits in an 8MB cache, and the distribution of 0-hit and 1+hit blocks stops being meaningful due to the fact that there are very few evictions. Such workloads are *perlbench, bzip2, gromacs, cactusADM, dealII, hmmer, sjeng, h264ref, tonto & sphinx3.* In this case, any cache management technique is obviously not useful. On the other hand, when the working set size does not fit in the cache – which is generally the case for smaller cache sizes – 0-hit blocks vastly dominate the evictions and can easily be targeted by simple insertion/bypass optimizations.

5 Discussion

Limits of dead block prediction. Our results indicate that the vast majority of evicted blocks do not see hits during their LLC residency. This is good news from a cache management perspective, because blocks that consistently do not see hits are easy to identify and filter from the cache via insertion or bypass policies. The remaining blocks (i.e., those that see 1+hits), averaging 11% & 16% in SHiP & SDBP, serve as the upper bound for the further opportunity of DBP. In the rest of this section, we use qualitative reasoning and quantitative analysis to understand which fraction of these 1+hit blocks is within practical reach of DBP.

First, we focus on 1+hit blocks that are predicted by SDBP. Such blocks represent 11% of all LLC evictions (Figures 2a). As noted in Section 4.2, the accuracy for these predictions is relatively low, averaging just 67% and indicating that these blocks are difficult to predict.

We hypothesize that many such blocks that experience reuse in the LLC represent an inherent challenge for DBPs due to noise in their access streams. This noise includes accesses on a misspeculated control flow path (i.e., hits that would not have occurred in the absence of speculation), accesses on a data-dependent control flow path (making the occurrence of these accesses difficult to predict), and conflicts in the higher-level cache that force additional accesses to the LLC.

Multi-core processors present two further sources of uncertainty for DBP. The first is LLC and directory pressure exerted by the multitude of threads on chip, which causes early eviction of cache blocks and makes effective learning of reuse behavior difficult. The second is inter-thread data sharing that confounds dead block prediction due to thread scheduling uncertainties and false sharing.

Because of the above, we believe that dramatic improvements in DBP accuracy for blocks with LLC reuse are unlikely unless coverage is sacrificed. Naturally, reducing coverage limits the potential for improvement of DBP versus a simple insertion or bypass scheme.

We next try to estimate the upper opportunity bound for DBP by analyzing the re-use distance of 1+hit blocks under SHiP+Bypass, the best performing cache management scheme. SHiP+Bypass minimizes LLC pressure through high-accuracy bypass, and averages 11% of 1+hit LLC evictions (including bypasses). To estimate an upper bound of DBP's improvement over SHiP+Bypass, we analyze what fraction of these evictions could have been dead-block predicted. To perform this analysis, we track whether the 1+hit blocks hit in the SVC following their eviction from the LLC under SHiP+Bypass. If a block does not observe an SVC hit, we assume that its next use is distant and that it should have been dead block predicted after its last LLC hit.

Figure 6 shows the results of the study. Each stacked bar corresponds to the fraction of 1+hit LLC evictions (including bypasses) and is composed of two segments – evictions that should have been predicted (labeled Opportunity) because they did not hit in the SVC, and those that should not be predicted (labeled Non-Opportunity) because they hit in the SVC, which indicates that their re-use distance is possibly within the reach of a well-managed LLC.

On average, the opportunity for DBP is 62% of all 1+hit blocks, representing 6.9% of all LLC evictions under SHiP+Bypass. Although we acknowledge that our SVC-based analysis is not a perfect indication of predictability, we believe it represents a reasonable estimate of the ceiling for DBP.

One interesting outlier in the figure is *zeusmp*, whose opportunity is 100%. This benchmark has few PCs that frequently miss in the LLC and, following each allocation, see exactly one hit, making it trivially predictable by a DBP. While it represents a pathology for insertion and bypass schemes, this behavior supports our as-

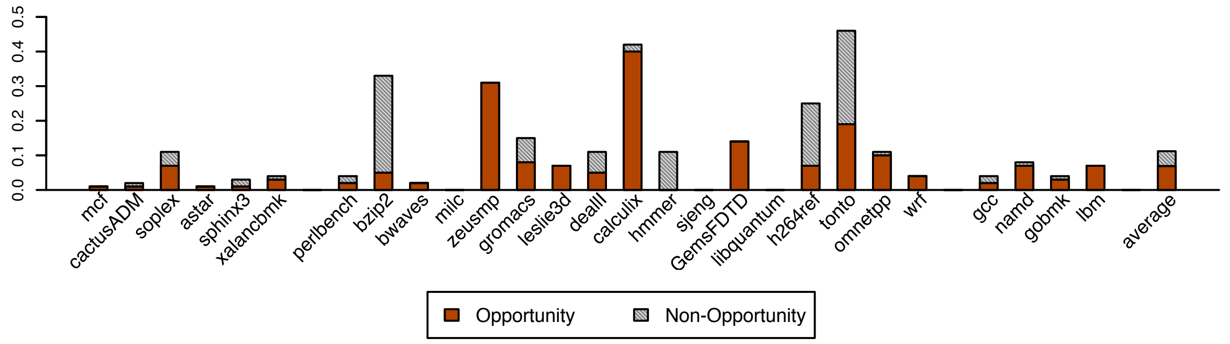


Figure 6: Opportunity for DBP for 'Not-Predicted 1+hit' blocks under SHiP+Bypass as a fraction of total evictions

sertion that blocks with the highest opportunity are trivially predictable.

What should future cache management research focus on? We believe that existing cache management strategies may be approaching the limits of predictability. While our analysis shows that some further gain is possible, the question is whether it is worth the trouble. The performance results in Section 4.4 indicate that any improvements in coverage should not come at the cost of accuracy.

We argue that improving the accuracy and coverage of blocks with short LLC lifetimes may offer the highest potential for improvement, especially when hardware costs are taken into account. The longer the lifetime of a block, the more challenging it is to predict due to noise in its access stream. Accounting for hardware costs, insertion schemes seem to offer the best performance per transistor as they need to maintain the least amount of state. Our study with SHiP+Bypass indicates that additional gains in cache efficiency are possible with low-cost and low-complexity hardware.

We conclude by urging the authors of future DBP papers to breakdown their coverage and accuracy into 0-hit and 1+hit blocks, and to isolate the contribution of each to performance to demonstrate where the benefits of their proposal are coming from and whether the choice of DBP is justified.

6 Conclusion

This work questions the value addition of dead block prediction over simple insertion based optimization for LLC management. Our insight is that the majority of LLC evictions are 0-hit blocks that can be predicted without a DBP scheme. Moreover, we show that blocks that experience LLC hits and are dead block predicted often suffer from low prediction accuracy, which we attribute to noise in the access stream. SHiP, a state-of-the-art insertion scheme, already outperforms all the existing DBPs. Using a limit study under SHiP+Bypass, we established that just 6.9% of all LLC evictions can further benefit from DBP. When hardware cost and design complexity are taken into account, we conclude that simple insertion and/or bypass schemes appear superior to dead block predictors.

Acknowledgement

We thank Daniel Jiménez for providing us with simpoint traces of SPEC'06 applications and Artemy Margaritov for his feedback on the work. We also thank the anonymous reviewers for their invaluable feedback.

References

[1] A.-C. Lai and B. Falsafi, "Selective, Accurate, and Timely Self-invalidation Using Last-touch Prediction," in *Proceedings of*

the International Symposium on Computer Architecture, pp. 139–148, 2000.

[2] A.-C. Lai, C. Fide, and B. Falsafi, "Dead-block Prediction & Dead-block Correlating Prefetchers," in *Proceedings of the International Symposium on Computer Architecture*, pp. 144–154, 2001.

[3] M. Kharbutli and Y. Solihin, "Counter-Based Cache Replacement and Bypassing Algorithms," *IEEE Trans. Comput.*, vol. 57, pp. 433–447, Apr. 2008.

[4] S. M. Khan, Y. Tian, and D. A. Jimenez, "Sampling Dead Block Prediction for Last-Level Caches," in *Proceedings of the International Symposium on Microarchitecture*, pp. 175–186, 2010.

[5] H. Liu, M. Ferdman, J. Huh, and D. Burger, "Cache Bursts: A New Approach for Eliminating Dead Blocks and Increasing Cache Efficiency," in *Proceedings of the International Symposium on Microarchitecture*, pp. 222–233, 2008.

[6] S. M. Khan, D. A. Jiménez, D. Burger, and B. Falsafi, "Using Dead Blocks As a Virtual Victim Cache," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, pp. 489–500, 2010.

[7] N. Duong, D. Zhao, T. Kim, R. Cammarota, M. Valero, and A. V. Veidenbaum, "Improving Cache Management Policies Using Dynamic Reuse Distances," in *Proceedings of the International Symposium on Microarchitecture*, pp. 389–400, 2012.

[8] Z. Hu, S. Kaxiras, and M. Martonosi, "Timekeeping in the Memory System: Predicting and Optimizing Memory Behavior," in *Proceedings of the International Symposium on Computer Architecture*, pp. 209–220, 2002.

[9] A. Jaleel, W. Hasenplaugh, M. Qureshi, J. Sebot, S. Steely, Jr., and J. Emer, "Adaptive Insertion Policies for Managing Shared Caches," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, pp. 208–219, 2008.

[10] A. Jaleel, K. B. Theobald, S. C. Steely, Jr., and J. Emer, "High Performance Cache Replacement Using Re-reference Interval Prediction (RRIP)," in *Proceedings of the International Symposium on Computer Architecture*, pp. 60–71, 2010.

[11] C.-J. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely, Jr., and J. Emer, "SHiP: Signature-based Hit Predictor for High Performance Caching," in *Proceedings of the International Symposium on Microarchitecture*, pp. 430–441, 2011.

[12] L. A. Belady, "A Study of Replacement Algorithms for a Virtual-storage Computer," *IBM Syst. J.*, vol. 5, pp. 78–101, June 1966.

[13] A. Jaleel, R. S. Cohn, C.-K. Luk, and B. Jacob, "CMP$im: A Pin-Based On-The-Fly Multi-Core Cache Simulator," in *Proceedings of the Workshop on Modeling, Benchmarking and Simulation*, 2008.

[14] A. R. Alameldeen, A. Jaleel, M. Qureshi, and J. Emer, "JILP Workshop on Computer Architecture Competitions: Cache Replacement Championship," 2010.

[15] E. Perelman, G. Hamerly, M. Van Biesbrouck, T. Sherwood, and B. Calder, "Using SimPoint for Accurate and Efficient Simulation," in *Proceedings of the International Conference on Measurement and Modeling of Computer Systems*, pp. 318–319, 2003.

[16] K. Beyls and E. H. D'Hollander, "Reuse distance as a metric for cache behavior," in *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 617–662, 2001.