Integrated Cache Miss Predictor with Confidence Estimation

Sneha Gorantala, Akshay Sawant
Department of Electrical and Computer Engineering
Georgia Institute of Technology

Abstract—With improved performance and economic energy consumption being the some of the major challenges of technological advancement over the past decades, die-stacked DRAM caches prove to be a decent solution in the area of memory systems as they possess great capacities of hundreds of megabytes to tens of gigabytes. However, one major challenge associated with the use of die-stacked DRAM caches is latency, triggered by the accesses that fail to hit in the cache. This paper demonstrates two methods of cache miss prediction for a system where the last-level cache is composed of die-stacked DRAM, with the notion of confidence. Reduction in the latency of DRAM cache access leads to increase system performance and reduce energy consumption per access.

Index Terms—Miss predictor, confidence estimation, serial lookup, parallel lookup, DRAM cache, history table, hash table.

1 Introduction

THE significant advances in die-stacking technology allow traditional DRAM to be integrated with the processor, providing capacities that are at least an order of magnitude greater than existing on-die SRAM caches. This additional capacity can capture a larger portion of the working set, providing improved performance and reduced energy. Along with the capacity benefit of tens to hundreds of megabytes, architecting DRAM as a cache also maintains software transparency and avoids dependence on operating system vendors.

However, the increased capacity comes at the cost of higher access latency per lookup. The latency associated with DRAM cache lookup is nearly 50% of the time taken to access main memory. Therefore, only the accesses which do not require access of main memory will benefit from the DRAM cache, making cache hit rate a major influencer of performance. Ideally, to gain maximum benefits of the DRAM cache, it should be known whether an access will be a hit or a miss. This work proposes two methods of miss prediction, for a three-level cache hierarchy, which are energy and area efficient with the notion of confidence. By predicting which accesses will be a miss, cache lookup can be done in the lower level at the same time. This parallel lookup results in a significant improvement in performance as well as reduced energy per access.

2 Related Work

Both industry and academia have produced a significant amount of research in prediction for DRAM caches throughout the past few years. One proposed solution, called Caching Hot Pages (CHOP) [1], employs three filter-based caching techniques that are used to cache only frequently used pages in the DRAM cache. By caching only these pages, the memory bandwidth problem can be avoided because bandwidth isn't wasted on pages with low spatial locality. This proposal works on a page level whereas newer research focuses on caching at the block level by attempting to solve the problem with the use of a dual-grain filter [2] which predicts if an access is a hit or miss and if a miss is encountered, the DRAM cache is bypassed and main memory is directly accessed. The filter consists of two filters, the fine-grain filter and coarse-grain filter, which are used to keep track of the cache on a block level and at a page level, respectively. However, this dual-filter was only able to achieve accurate prediction for 85% of the cache accesses and at a significant storage cost. Another work proposed Map Global (Map-G) [3], where a single 3-bit counter is used to monitor previous accesses. Map-G was extended to Instruction-based Map (Map-I) to take advantage of the fact that whether an access is a hit or miss is highly correlated with the address of that access. A table of counters is used to store hit/miss information and the desired counter is accessed by hashing the address of the instruction. These techniques presented respectable results. However, an integrated solution for all levels of cache was not found.

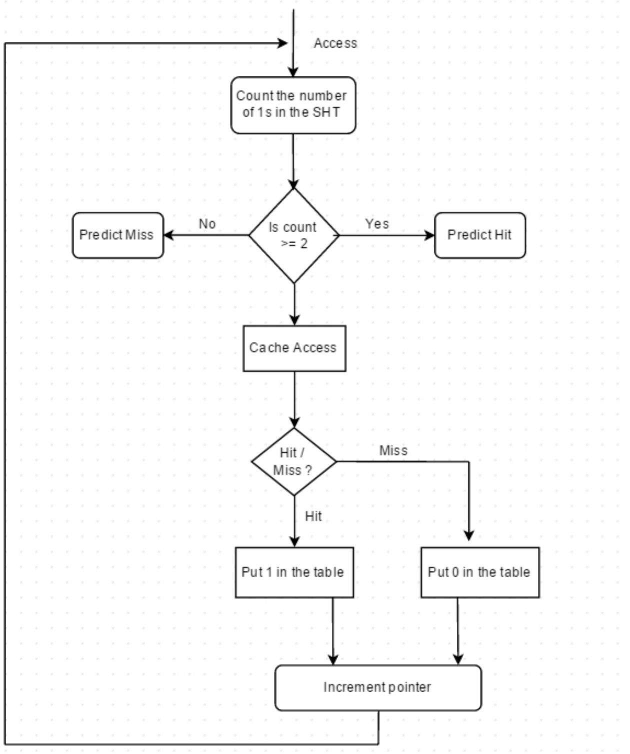
3 Motivation

Parallel lookup of the current level of cache and a lower level of cache for misses results in an ideal situation where all the performance benefits of hits are seen without any of the costs and overheads of misses. This ideal situation can be attained only with accurate prediction of hit/miss information. This work proposes two predictors, Simple History Table (SHT) and Address-based History Table (ABHT).

The reasoning behind these methods of predictions is the simple observation that previous accesses are a good indication of whether or not the current access will be a hit. To understand this reasoning, first consider a streaming application where each access results in a compulsory miss. If prediction is based on the previous accesses, it is clear that extremely accurate prediction would result in this situation. All accesses would have a miss prediction and the lower level of memory would be checked, resulting in a great saving of latency, especially in the last level of cache (DRAM cache). In general, applications tend to exhibit spatial locality. This property is exploited to build the SHT and ABHT predictors.

4 Methodology

4.1 Simple History Table



The key idea for the Simple History Table (SHT) is to predict if the accesses will be a hit or a miss based on the previous accesses. In the SHT predictor, a table keeps a track of whether the last four accesses were hits or misses. If there is a hit, 1 is entered in the table location corresponding to that access, and if there is a miss, 0 is entered in the correct table location. The table values are updated after every access. A counter is used to count the number of 1s within the table. To make the prediction, the value of the counter is checked. If the counter value is equal to or more than 2, then a hit is predicted. If it is less than 2, a miss is predicted. A single table only has 4 bits of overhead.

After testing the SHT for a single level cache, an integrated solution was implemented for a three level cache in which the last level is a DRAM cache. If a miss is predicted for a particular level, parallel lookup is performed and if a hit is predicted, serial lookup is performed. Separate tables are created for all the three levels of cache, so that the contents of one level don't interfere with other. It was also observed that most of the writes to the lower level memory were writebacks. Hence, the read and write patterns were very disparate and separate read and write tables were employed to preserve this distinction.

SHT provided remarkable accuracy for the DRAM cache (96.48%). However, the overall prediction accuracy for all levels of cache was not satisfactory (82.98%). In order to obtain better performance and better accuracy, Address-based History Table (ABHT) was implemented.

4.2 Address-based History Table

It is a well-known observation that the cache hit/miss information is heavily correlated with the instruction address that caused the cache access. Instead of using a single SHT, an array of SHTs was employed in this scheme. The address of the access is hashed into the ABHT to obtain the desired SHT. All the predictions and updates happen based on this SHT. Each SHT has a counter which counts the number of 1s for that particular SHT. 512 entries are chosen for the ABHT. The storage overhead for this implementation of ABHT is 512 * 4 bits = 256 bytes.
Separate ABHTs are maintained for each level of cache to avoid interference. Also, there are separate ABHTs for read and write accesses to the lower levels of memory (L2 and L3). Thus, the total storage overhead comes out to be 256 * 5 = 1280 bytes which is approximately equal to 1KB. High accuracy can be achieved by using only 0.3425% overhead.

The main purpose of the ABHT predictor is to introduce the notion of confidence in prediction by assigning a level of confidence for different counter values, making predictions more accurate. To predict whether a cache block is present, the address of the instruction is hashed and the value of the corresponding counter is checked. If the counter value is 0, the block is predicted to be very unlikely to be present in the cache. A confidence interval of 0-20% is assigned for that access, meaning that the cache block is 0-20% likely to be present in the cache. As the counter value increases, a higher confidence interval is assigned. Hence, the confidence intervals are 20-40% for a counter value of 1, 40-60% for a counter value of 2, and 60-80% for a counter value of 3. Finally, when the counter value is 4, then the block is predicted to be very likely to be present in the cache. Thus, a confidence interval of 80-100% is assigned for that access.

It is also observed that most of the last level accesses fall in either the 0-20% confidence interval or the 80-100% confidence interval. When the simulator is run for a wide variety of traces, it is observed that around 90% of the accesses fall in the 0-20% or 80-100% confidence intervals. ABHT implementation gives an overall prediction accuracy of 93.94%, proving to be considerably better than the 82.98% accuracy of the SHT implementation.

5 Experimental Setup

5.1 Simulator

A validated, three-level, parametric cache simulator was built for experimentation. The simulator takes access patterns as inputs and generates results such as the number of read misses, write misses, hits for each level of cache, the number of correct and wrong predictions, the prediction accuracy, the latency for each level of cache, and the overall latency. The simulator can be used in direct mapped, fully associative and set associative mode. It employs the Write

Back Write Allocation (WBWA) policy and the Least Recently Used (LRU) policy for cache replacemen. The cache size, block size and associativity can be varied for experimentation. The cache levels exhibit inclusive property.

5.2 Benchmarks

For experimentation and analysis, four workloads from the SPEC CPU2006 benchmarks were used, namely astar, bzip2, mcf, and perlbench. Additional workloads were generated by mixing the existing traces. Each workload has about 2 million instructions. Simulations are run for all the workloads and results are generated and compared.

6 Results And Analysis

Figure 1 shows the results of SHT in terms of prediction accuracy. It can be seen that SHT provides outstanding accuracy for the last level cache but overall accuracy is not adequate.

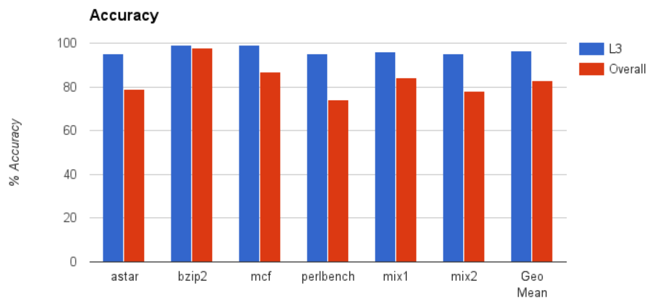
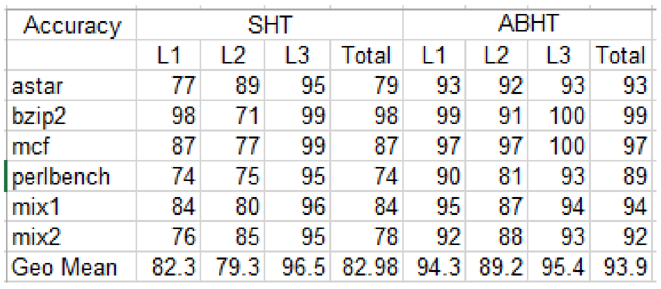


Fig. 1. Accuracy of the DRAM cache compared to the overall cache system

On average, the accuracy for the last level cache for the above workloads is 96.48% wheras the average for the entire cache system is only 82.98%.

Improved accuracy can be seen with the implementation of ABHT.

TABLE 1
Accuracy Comparison of SHT and ABHT



Accuracy	SHT				ABHT			
	L1	L2	L3	Total	L1	L2	L3	Total
astar	77	89	95	79	93	92	93	93
bzip2	98	71	99	98	99	91	100	99
mcf	87	77	99	87	97	97	100	97
perlbench	74	75	95	74	90	81	93	89
mix1	84	80	96	84	95	87	94	94
mix2	76	85	95	78	92	88	93	92
Geo Mean	82.3	79.3	96.5	82.98	94.3	89.2	95.4	93.9

The accuracy of ABHT was significantly better than SHT. It can be seen that for the workloads bzip2 and mcf, the accuracy of prediction for L3 is 100%. This is due to the fact that the working set of the cache was small enough that the last level cache was not required. All the access to the last level were compulsory misses that were accurately predicted.

The notion of confidence increased the overall accuracy of ABHT to 93.94%, compared to the 82.98% accuracy of SHT.

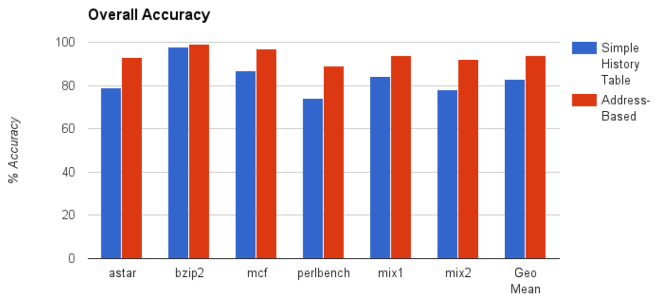
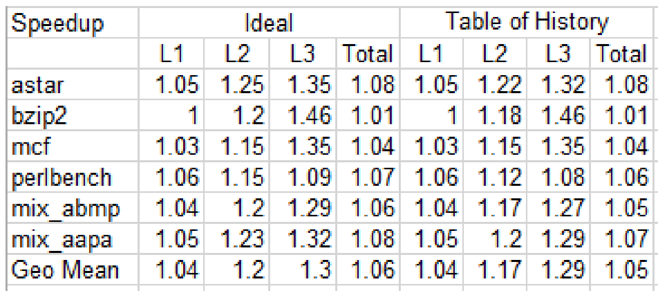


Fig. 2. Accuracy of Simple History Table compared to Address-based History Table for the overall cache system

Performance is another important metric to provide insight into the efficacy of the predictors. Table 2 shows the speedup of ABHT from the baseline of no prediction at all. The table compares the speedup of ABHT to the ideal situation where 100% of the misses are predicted in advance.

TABLE 2
Speedup of ABHT compared to Ideal Speedup



Speedup	Ideal				Table of History			
	L1	L2	L3	Total	L1	L2	L3	Total
astar	1.05	1.25	1.35	1.08	1.05	1.22	1.32	1.08
bzip2	1	1.2	1.46	1.01	1	1.18	1.46	1.01
mcf	1.03	1.15	1.35	1.04	1.03	1.15	1.35	1.04
perlbench	1.06	1.15	1.09	1.07	1.06	1.12	1.08	1.06
mix_abmp	1.04	1.2	1.29	1.06	1.04	1.17	1.27	1.05
mix_aapa	1.05	1.23	1.32	1.08	1.05	1.2	1.29	1.07
Geo Mean	1.04	1.2	1.3	1.06	1.04	1.17	1.29	1.05

The speedup in the last level of cache compared to ideal and baseline speedup is shown in the figure below.

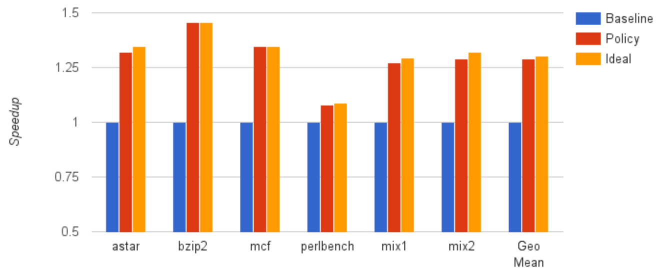


Fig. 3. Speedup of the DRAM cache compared to the baseline and ideal speedup

7 Conclusion

Our work shows that significant speedup can be achieved for DRAM cache from accurate prediction. For DRAM cache, ABHT policy achieves speedup of 1.2892 which is almost equal to ideal speedup of 1.3030. The overall speedup is 1.05013 which is also almost equivalent to the ideal speedup of 1.0553. Thus, it can be concluded that with accurate prediction and the notion of confidence, DRAM cache latency can be masked to an extent. Observations show that for applications where having low latency and low power consumption is crucial (over accuracy), our results can have a great impact. If a low confidence interval (0-20%) is predicted, then DRAM lookup can be bypassed and main memory can be directly accessed since the block is unlikely to be present in the DRAM cache. This would save DRAM cache lookup latency for all the predicted misses, giving significant performance improvement. For general applications, if a confidence interval of 0-80% is predicted, parallel lookup to the DRAM cache and main memory can be performed. If a high confidence interval (80-100%) is predicted, then serial lookup can be performed as the block is highly likely to be present in the DRAM cache. Experimentation shows that almost 50% of the accesses lie in this interval. Therefore, latency is saved on more than 50% of the accesses.

The overall saving in latency helps bridge the gap between performance of processors and memory. Also, the drawbacks of using die-stacked DRAM as a cache are diminished, allowing for increased adoption of DRAM caches.

ACKNOWLEDGMENT

The authors wish to thank Dr. Moin Qureshi for providing an opportunity to perform research under him. The support, feedback and guidance were tremendous.

REFERENCES

[1] X. Jiang, et al., CHOP: Integrating DRAM Caches for CMP Server Platforms, In IEEE Micro Top Picks, 2011.

[2] M. El-Nacouzi, I. Atta, M. Papadopoulou, J. Zebchuk, N. E. Jerger, and A. Moshovos. A Dual Grain Hit-miss Detector for Large Die-stacked DRAM Caches. In Proc. of the Conf. on Design, Automation and Test in Europe, pages 89–92, 2013

[3] M.K. Qureshi and G.H. Loh. Fundamental Latency Trade-offs in Architecting DRAM Caches. In Proc. of 45th Int'l Symp. on Microarchitecture, 2012.