The Case for Associative DRAM Caches

Paul Tschirhart
University of Maryland,
College Park
pkt3c@umd.edu

Jim Stevens
University of Maryland,
College Park
jims@cs.umd.edu

Zeshan Chishti
Intel Labs
Hillsboro, Oregon, USA
zeshan.a.chishti@intel.com

Bruce Jacob
University of Maryland,
College Park
blj@umd.edu

ABSTRACT

In-package DRAM caches are a promising new development that may enable the continued scaling of main memory by facilitating the creation of multi-level memory systems that can effectively utilize dense non-volatile memory technologies. However, determining an appropriate storage scheme for the large amount of meta-data needed by these new caches has proven to be difficult. As a result, prior work has suggested that associativity, with its additional meta-data requirements, may not be well suited for use in large in-package DRAM caches. This work makes the case that despite these problems, associativity is still a desirable feature for DRAM caches by demonstrating the benefits of associativity for a wide range of cache configurations and workloads.

CCS Concepts

•Computer systems organization → Architectures; Heterogeneous (hybrid) systems; •Hardware → Emerging architectures; Memory and dense storage;

Keywords

Hybrid Memory, DRAM Cache, Multi-Level Main Memory, Associativity

1. INTRODUCTION

The current memory system has been pushed to its limits by the increasing demands of modern workloads. To counteract this, multi-level main memory architectures have been suggested by researchers as a way to enable the continued improvement of main memory performance [24, 15, 22, 7, 17]. These architectures use the existing DRAM system as a cache to maintain or improve the performance of the system while simultaneously increasing its overall memory capacity by using a larger, relatively slow backing store. The recent development of in-package DRAM caches has helped to enable these architectures by allowing for a higher bandwidth DRAM cache implementation [2, 3]. In addition, the announcement of a new potential non-volatile main memory technology has renewed the possibility that multi-level memory architectures will be needed to hide the slower access latencies of non-volatile backing stores [1, 20].

One of the main challenges facing the design of these DRAM cache systems is the storage of the megabytes of tags that are needed to identify entries in the cache. Ideally, this sort of meta-data would be kept in an on-chip SRAM store that would provide rapid access to the tags during a cache access. However, the potential sizes of DRAM caches result in quantities of tags that are impractical to store on-chip in SRAM (20MB for a 256MB cache). So, the meta-data for the DRAM cache must be stored in some other way that does not negatively impact hit latency. Recent work in the area of associative DRAM caches have proposed several different ways to address this problem [17, 9, 11, 12, 16, 8, 14, 13]. However, all of these designs still introduce significant complexity to the system and still require non-trivial amounts of on-chip SRAM. In addition, the time it takes to access the tags, even when they are stored in SRAM, adds to the hit latency of the cache which can reduce the overall performance of the system.

These meta-data storage and delay issues have led prior work to conclude that associativity may not be a desirable trait for this new level of the memory hierarchy. This view is supported by two effects that have been observed in these systems. First, the miss rate reduction that can be achieved by introducing associativity at this level is just around 5-6% on average for workloads such as those from SPEC CPU2006 or PARSEC. And second, thus far, most implementations of DRAM caches have utilized DRAM technology for both the backing store and the cache, so their miss penalty is not significantly larger than the hit latency. The combination of these effects significantly reduces the potential performance improvement that can be achieved by introducing associativity. As a result, the incorporation of associativity in these systems often results in performance degradation as the negative impact of their increased hit-latency tends to outweigh the benefits of miss reduction. However, this is only the case for a narrow range of system configurations and does not take into account the full potential of multi-level memory systems.

In this work, we make the case for associative DRAM

MEMSYS 2016 October 3–6, 2016, Washington, DC, USA

DOI: http://dx.doi.org/10.1145/2989081.2989120

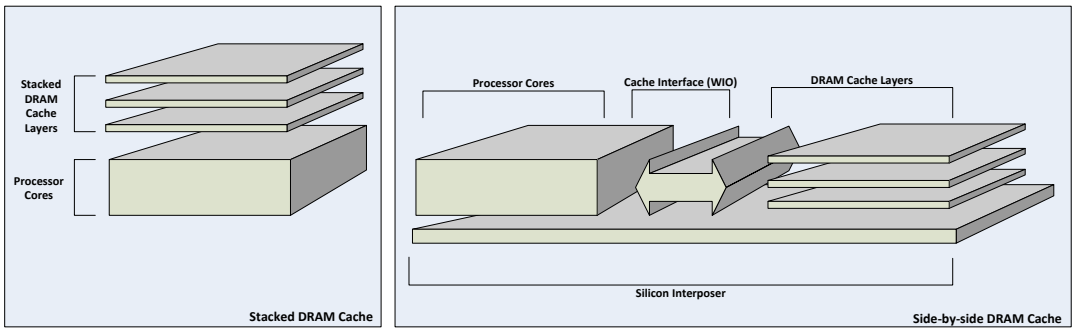

Figure 1: Some examples of recently proposed in package DRAM caches.

caches by demonstrating that associativity can provide considerable benefits for systems that differ significantly from the existing memory system in terms of organization or technology choice.

Specifically, this work makes the following contributions:

- We show that the effects of associativity in DRAM caches are workload dependent and that averaging across workloads obfuscates the significance of associativity
- We quantify the impact of cache size and page size on the effect of associativity and show that while associativity provides some benefit for all DRAM cache organizations, it provides a greater benefit for smaller caches and caches that utilize large page sizes
- We demonstrate that the miss penalty of the overall system plays an important role in determining the importance of associativity as large miss penalties amplify the importance of otherwise minor miss rate reductions

2. THE ROLE AND LIMITATIONS OF ASSOCIATIVITY

Associativity has long been a feature of L2 and L3 caches because it can help to improve cache performance by reducing the number of conflict misses. Conflict misses occur because more than one piece of frequently used data maps to the same location in the cache. Associativity helps reduce this type of miss by providing multiple places for the data to reside, thereby reducing contention over individual cache locations. Associativity can also help with some amount of capacity misses by allowing critical data to reside continually in a line while other less important data is constantly swapped in and out of other lines.

However, the benefits of associativity are limited in certain situations. For instance, associativity cannot provide a significant miss rate improvement for some workloads because those workloads only exhibit a small percentage of conflict misses. In addition, if the workload is significantly larger than the size of the cache, then even large amounts of associativity will provide only moderate benefits because there is simply too much set contention. Similarly, if the workload is significantly smaller than the size of the cache, then associativity will not provide any noticeable benefit because the majority of misses will be compulsory. Therefore, the benefits of associativity will vary significantly between different workloads and for different cache configurations. This often leads to a somewhat low average performance improvement across large groups of workloads, as we will see later. But associativity is still included in many cache designs because its benefits in specific cases make up for its otherwise low average improvement.

3. DRAM CACHE DESIGN OVERVIEW

The decision of whether or not to include associativity in DRAM caches is a difficult one because there is currently no ideal way to store the required meta-data (such as tags and valid bits). The simplest approach is to store the meta-data in SRAM, either on chip or off. By storing the tags in SRAM we can ensure that the tag lookup time will be relatively negligible compared to the data access time in the DRAM. However, the amount of SRAM required to store all of the tags is prohibitively large. For instance in a 48-bit address space, the tags for a 128MB cache with 64B blocks would need roughly 6MB of SRAM for storage. This represents a significant portion of the typical on-chip SRAM capacity. More importantly, though, the size of the SRAM tag store would need to scale with the size of the DRAM cache. So, a 1GB DRAM cache would need roughly 8x the SRAM to store its tags or 48MB, which is larger than the total SRAM on most chips today.

The primary alternative to an SRAM tag store is to store the tags in DRAM along with the data. The problem with storing the meta-data in DRAM, though, is that it negatively impacts the latency of the cache. This is because two DRAM accesses are required for each cache hit access, one to fetch the tags and another to fetch the data. Figure 2 provides a comparison of the DRAM access process to the fast SRAM tag store access process.

To address this problem there have been many proposed DRAM cache designs that attempt to efficiently store and access the necessary meta-data. These designs tend to fall into two categories: block based designs that utilize the standard 64B DRAM access as their line size, and page based designs that utilize a large line size.

3.1 Block Based Designs

One of the first DRAM cache architectures was proposed by Loh et al. and stored the meta-data for a set in the same row as the data for the set [17]. This leveraged row buffer locality to reduce the access latency of the subsequent tag and data accesses after the initial access had opened the row. However, this design used an entire DRAM row for each set and so the row buffer locality was only exploited for a sin-

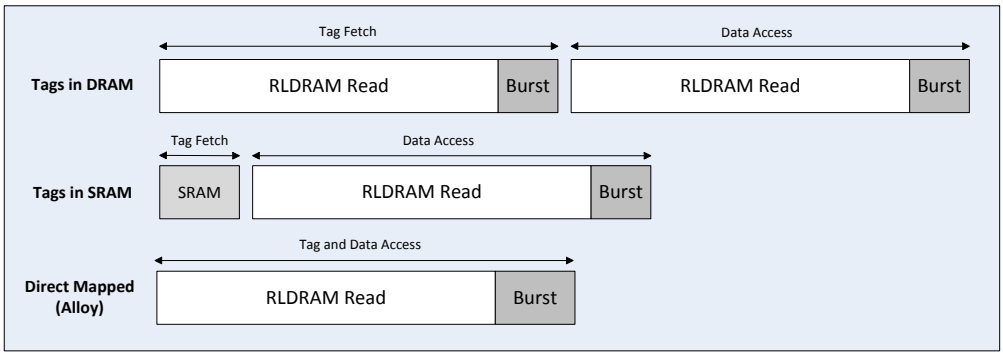

Figure 2: A comparison of the different tag access schemes for DRAM caches.

gle set access. The LAMOST design improved upon Loh's design by reducing the degree of associativity from 28-way to 7-way, which allowed for 4 sets to be placed in a row instead of just one [9]. As a result, row buffer locality could be leveraged for multiple set accesses thereby improving performance. However, both of these designs still require multiple accesses to the DRAM for each cache hit and this ultimately becomes a limiting factor in the performance of the system.

To address this problem of multiple DRAM accesses to fetch tags, Qureshi et al. proposed Alloy cache which stores the tag and data together and fetches them with a single 72 byte DRAM access [21]. This approach provides good performance for many workloads at the expense of higher miss rates due to the lack of associativity.

Another approach is to store a subset of tags in SRAM while the majority of tags are stored in DRAM. In effect these designs, such as [11], create an additional structure to cache the tags from the DRAM cache. In [11], this tag cache is structured and operates in much the same way as any other cache. It attempts to retain tags that have been accessed and replaces ones that aren't used. This approach achieves some considerable speedups over tags in DRAM caches but requires 46KB of SRAM for the tag cache. Another similar design which also used an SRAM tag cache was proposed in [18].

Still another way to reduce the overhead of storing metadata is to compress it. This is explored in the Tag Tables paper which proposes a novel compression scheme for DRAM cache tags [8]. The compression scheme achieves a significant reduction in the amount of space needed to store the tags for the DRAM cache. However, despite its notable compression ratio, this design still often requires 1-2MB of tag storage.

3.2 Page Based Designs

All of the previous approaches that we have discussed have utilized standard 64B block sized accesses in their designs. However, another way to approach building a DRAM cache is to use large pages. This is the approach that is explored by the Unison and Footprint cache designs [13, 14]. Utilizing large page sized accesses results in higher hit rates and reduced tag overhead for the cache. However, additional mechanisms are required to manage the contents of the pages in the cache to prevent wasting bandwidth to fetch data that won't be used.

Another interesting page based DRAM cache design was proposed in [16]. This design utilizes the existing TLB infrastructure to keep track of the contents of the DRAM cache, essentially indexing the cache with virtual addresses.

Table 1: Baseline Simulator Configuration

Processor	
Number of cores	8-core
Issue Width	4
Frequency	3.2GHz
On Chip Caches	
L1I (private)	128 KB, 8-way, 64 B block size
L1D (private)	128 KB, 8-way, 64 B block size
L2 (private)	256 KB, 8-way, 64 B block size
L3 (shared)	32 MB, 20-way, 64 B block size

4. METHODOLOGY

The results that are featured in this work were generated using a full-system simulation driven detailed model of a multi-level main memory system. A modified version of HybridSim [23] was utilized to model the DRAM cache with a typical LRU replacement policy. The input to the HybridSim model was generated using the MARSSx86 [19] full system simulator with the baseline configuration in table 1.

4.1 Benchmarks

For the studies presented in this paper we use a selection of multi-threaded workloads from the SPEC CPU2006 [10], NPB [4], and PARSEC [5] benchmark suites. The SPEC workloads were run in rate mode with 8 copies while the NPB and PARSEC benchmarks were run with 8 threads. Table 2 presents the relevant characteristics of the chosen benchmarks. The baseline miss ratio values included in this table are for a 64MB direct mapped DRAM cache with a 64B block size.

5. RESULTS AND DISCUSSION

In order to demonstrate the potential benefits of associativity for DRAM caches we perform a series of experiments that illustrate the effects that different system configuration choices have on the impact of associativity. In particular, we look at the consequences of different cache capacities, cache line sizes, and miss penalties with regard to the improvement that can be achieved by introducing different degrees of associativity. These experiments show that the benefits

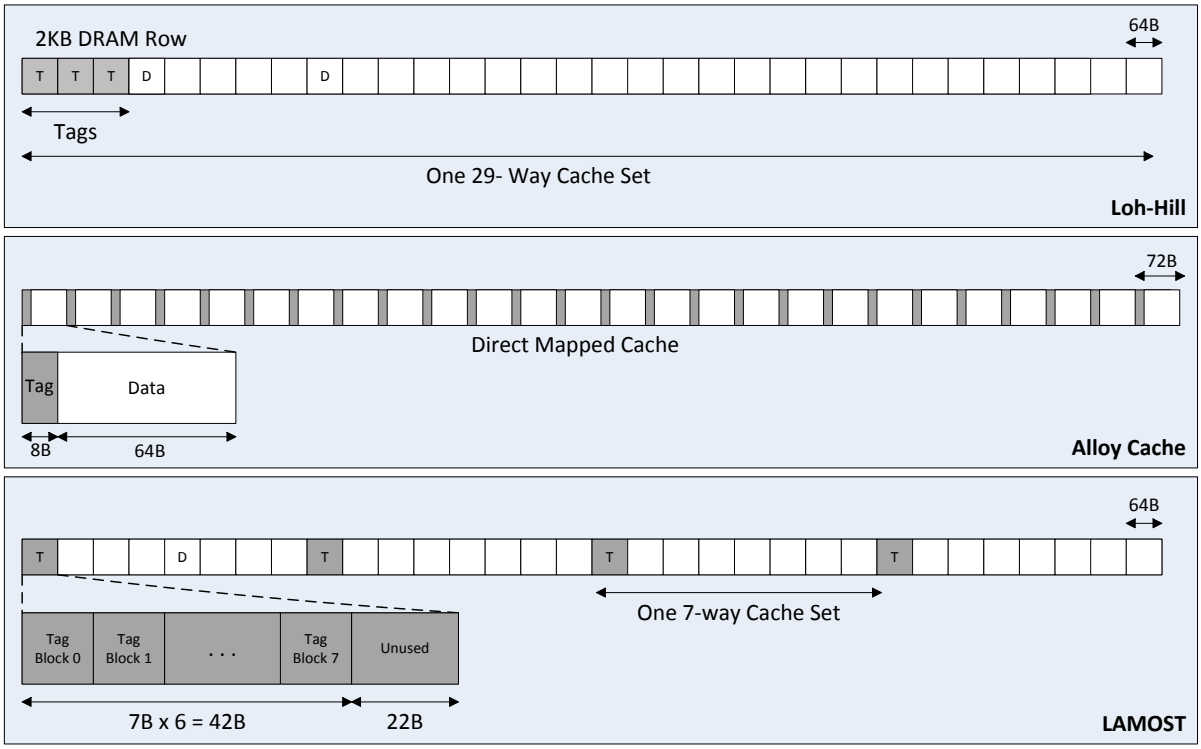

Figure 3: A comparison of some of the different possible row layouts for DRAM caches.

Table 2: Benchmark Characteristics

	Observed Footprint	L3 MPKI	DRAM Cache Miss Ratio
ft	1287.27 MB	7.116	0.62
is	264.87 MB	10.424	0.83
mg	430.87 MB	13.597	0.63
blackscholes	269.53 MB	0.64	0.9
bodytrack	534.28 MB	0.5	0.66
canneal	497.60 MB	6.5	0.57
freqmine	894.04 MB	1.318	0.57
bzip2	2559.93 MB	61.007	0.61
gcc	441.31 MB	6.1	0.68
leslie3d	601.15 MB	18.725	0.71
milc	1909.66 MB	22.47	0.7

of associativity become significantly more noticeable in systems that utilize larger line sizes and that have longer miss penalties than the systems that have been primarily studied in prior work. In addition, we also include individual results for each of the benchmarks that are evaluated to show their distinct reactions to associativity.

5.1 The Role of Cache Capacity

The first experiment that we present in this paper varies the capacity of the cache in order to show how the impact of associativity is tied to the ratio of cache capacity to workload size. In this experiment we vary the capacity of the cache from 64MB (a cache that is significantly smaller than all of the workloads) to 512MB (a cache that is roughly the size of more than half of the tested workloads). To isolate the effects of associativity, each series of these results represents the percent improvement over the miss rate of a direct mapped cache with the same size. The results of this experiment are presented in Figure 4.

Comparing the graphs in Figure 4, we can see that the effects of cache size and associativity vary greatly from workload to workload. For instance, canneal is sensitive to associativity at all three cache sizes and experiences significant benefits from increasing the degree of associativity. GCC also consistently benefits from increased associativity but only when the cache capacity is small enough to keep the percentage of conflict misses high. Freqmine, however, reacts very differently to associativity. For the two middle cache sizes (128MB and 256MB), freqmine shows almost no effect from increasing associativity. When the size of the cache is increased to 512MB, freqmine shows an increase in miss rate as a result of increasing associativity. Similar trends can also be noted for several other workloads and cache capacities. To check this somewhat odd result we

reran the experiment using the Dinero cache simulator and got similar trends [6]. We believe that this behavior is due to the additional set contention that can sometimes result from increasing the associativity. Adjusting the replacement algorithm from LRU to RRIP or something similar may help to alleviate this effect.

Averaging the effects of size and associativity across all benchmarks appears to indicate that associativity has an insignificant effect for all but the smallest caches. However, this obfuscates the important fact that while associativity is not always helpful on average, it is sometimes very helpful in specific cases as can be seen in these results.

5.2 The Effect of Line Size

Having established that associativity can provide noteworthy benefits to some workloads provided that the percentage of conflict misses is significant, we next look at the effect of different cache organizations on the impact of associativity. Several proposed DRAM cache designs utilize line sizes that are considerably larger than the standard 64B DRAM cache access. To investigate the effect of associativity on these larger line size DRAM caches we conduct an experiment where we increase the size of the cache line from 64B to 4KB. The cache in this experiment is 128MB because this capacity provided the most consistent results in the previous experiment. As was the case in the previous experiment, each series of these results is the percent improvement over the miss rate of a direct mapped cache with the same line size in order to isolate the effects of associativity. Figure 5 contains the results from this experiment.

In almost all cases shown in Figure 5, caches with longer line sizes benefit from increased associativity. The only workload for which this trend did not hold is canneal, which showed noticeable benefits only for the smaller 64B and 256B line sizes. This is because canneal tends to have a high degree of set contention when the number of sets is small, resulting in poor hit rates even when associativity is increased. The same effect can be seen in the previous experiment where canneal's benefit from associativity improved noticeably when the cache capacity was increased resulting in more sets. Interestingly, the 256B line size benefited more from associativity than the 64B line size suggesting that the 256B line resulted in a more optimal set assignment for this workload.

In general, these results suggest that both block based and page base DRAM caches should implement some form of associativity in order to minimize their miss rate. They also suggest that the limited utility of associativity observed in prior work may be an effect that is largely limited to the 64B cache configurations that were being investigated in those studies.

5.3 The Impact of Miss Penalty

We conclude our experiments by investigating the effect that miss penalty has on the impact of associativity in these systems. One of the potential uses for in-package DRAM caches would be to help hide the longer access latencies of backing stores constructed with a dense non-volatile memory technology. In these systems, the miss penalty would be considerably longer than it is in the DRAM backed, DRAM cached systems that have been focus of prior work. To evaluate the influence that this technology choice could have on the effect of associativity we perform a study where we increased the miss penalty relative to the hit latency from 2x to 8x. The series in these results represent the percent improvement over the average access latency of a direct mapped cache with the same miss penalty. This was done to expose the effects of associativity. The capacity of the cache was 128MB for this study and the line size was 64B.

Figure 6 displays the effects of backing store speed on the impact of associativity. From these results it is clear that, as the ratio of miss penalty to hit latency increases, the role of associativity in preventing misses becomes more and more important. However, even at a ratio of 8x, little benefit is seen from increasing associativity beyond 4-way. The average results also show that 2-way associativity provides the greatest difference in performance. Interestingly, the performance of the 8x ratio system actually degrades after peaking at 4-way associativity. It is also worth noting that bzip2, leslie3d and milc show significantly less performance improvements for the 8x ratio than they did for the 4x ratio. This is because the miss latency became the dominant aspect of the system at 8x for these workloads and contributed much more to the average latency than was saved by introducing associativity. Overall though, these results suggest that implementing associativity should be a high priority for future multi-level memory systems that employ a relatively slow non-volatile backing store.

6. CONCLUSION

In-package DRAM caches are an exciting area of memory architecture research that could help to provide a path towards larger, faster, more energy efficient main memory systems. However, implementing associative DRAM caches has been difficult due to the large amounts of meta-data that they require and the additional access latencies that they introduce. Prior work has suggested that these additional requirements make associativity a less than desirable feature for DRAM caches.

This work makes the case for associativity in DRAM caches in three ways. First, we show that when a wide range of configurations is considered, the benefits of associativity become more significant. Second, we show that when the effects of associativity on individual workloads are considered, its true impact becomes more apparent. And finally, we establish the need for associativity in multi-level main memory systems that utilize slower non-volatile memory as their backing store technology and, as a result, have a longer miss penalty. Looking forward, this work emphasizes the importance of the development of associative DRAM cache designs that utilize small amounts of on-chip storage and provide low latency accesses as a critical step toward the realization of the full potential of multi-level main memory systems.

7. REFERENCES

[1] 3D XPoint Technology. http://www.micron.com/about/innovations/3d-xpoint-technology.

[2] High Bandwidth Memory. http://www.amd.com/en-us/innovations/software-technologies/hbm.

[3] Hybrid Memory Cube Consortium. http://hybridmemorycube.org.

[4] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga.

The nas parallel benchmarks. *International Journal of High Performance Computing Applications*, 5(3):63–73, 1991.

[5] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, pages 72–81, New York, NY, USA, 2008. ACM.

[6] J. Edler and M. D. Hill. Dinero iv trace-driven uniprocessor cache simulator, 1998.

[7] A. Ferreira, B. Childers, R. Melhem, D. Mosse, and M. Yousif. Using pcm in next-generation embedded space applications. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, pages 153–162, April 2010.

[8] S. Franey and M. Lipasti. Tag tables. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, pages 514–525, Feb 2015.

[9] F. Hameed, L. Bauer, and J. Henkel. Simultaneously optimizing dram cache hit latency and miss rate via novel set mapping policies. In *Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2013 International Conference on*, pages 1–10, Sept 2013.

[10] J. L. Henning. Spec cpu2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17, Sept. 2006.

[11] C.-C. Huang and V. Nagarajan. Atcache: Reducing dram cache latency via a small sram tag cache. In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, PACT '14, pages 51–60, New York, NY, USA, 2014. ACM.

[12] C.-C. Huang and V. Nagarajan. Atcache: Reducing dram cache latency via a small sram tag cache. In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, PACT '14, pages 51–60, New York, NY, USA, 2014. ACM.

[13] D. Jevdjic, G. Loh, C. Kaynak, and B. Falsafi. Unison cache: A scalable and effective die-stacked dram cache. In *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, pages 25–37, Dec 2014.

[14] D. Jevdjic, S. Volos, and B. Falsafi. Die-stacked dram caches for servers: Hit ratio, latency, or bandwidth? have it all with footprint cache. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, pages 404–415, New York, NY, USA, 2013. ACM.

[15] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. Architecting Phase Change Memory as a Scalable Dram Alternative. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 2–13, New York, NY, USA, 2009. ACM.

[16] Y. Lee, J. Kim, H. Jang, H. Yang, J. Kim, J. Jeong, and J. W. Lee. A fully associative, tagless dram cache. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 211–222, New York, NY, USA, 2015. ACM.

[17] G. H. Loh and M. D. Hill. Efficiently enabling conventional block sizes for very large die-stacked dram caches. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, pages 454–464, New York, NY, USA, 2011. ACM.

[18] J. Meza, J. Chang, H. Yoon, O. Mutlu, and P. Ranganathan. Enabling efficient and scalable hybrid memories using fine-granularity dram cache management. *Computer Architecture Letters*, 11(2):61–64, July 2012.

[19] A. Patel, F. Afram, S. Chen, and K. Ghose. MARSSx86: A Full System Simulator for x86 CPUs. In *Design Automation Conference 2011 (DAC'11)*, 2011.

[20] J. T. Pawlowski. Vision of Processor-Memory Systems. Keynote Presentation.

[21] M. K. Qureshi and G. H. Loh. Fundamental latency trade-off in architecting dram caches: Outperforming impractical sram-tags with a simple and practical design. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-45, pages 235–246, Washington, DC, USA, 2012. IEEE Computer Society.

[22] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable High Performance Main Memory System Using Phase-Change Memory Technology. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 24–33, New York, NY, USA, 2009. ACM.

[23] J. Stevens, P. Tschirhart, M.-T. Chang, I. Bhati, P. Enns, J. Greensky, Z. Chisti, S. Lu, and B. Jacob. An integrated simulation infrastructure for the entire memory hierarchy: Cache, dram, nonvolatile memory, and disk. *Intel Technology Journal*, 17(1):184–200, 2013.

[24] M. Wu and W. Zwaenepoel. envy: A non-volatile, main memory storage system. In *ASPLOS*, pages 86–97, 1994.

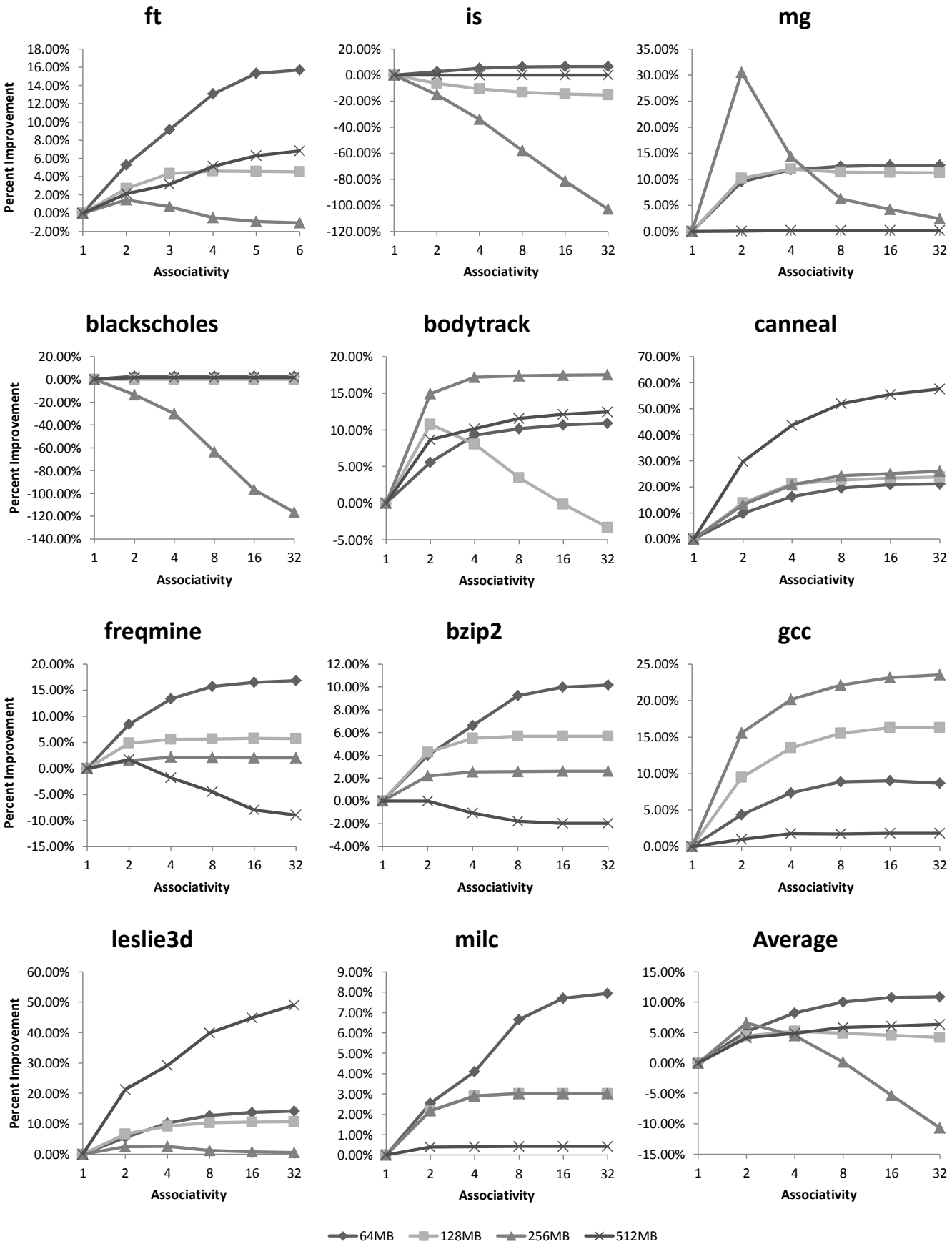

Figure 4: The effect of different cache capacities and different degrees of associativity on the miss rate of the cache. Each series is the percent improvement over the miss rate of a direct mapped cache with the corresponding size. This was done to isolate the effects of associativity.

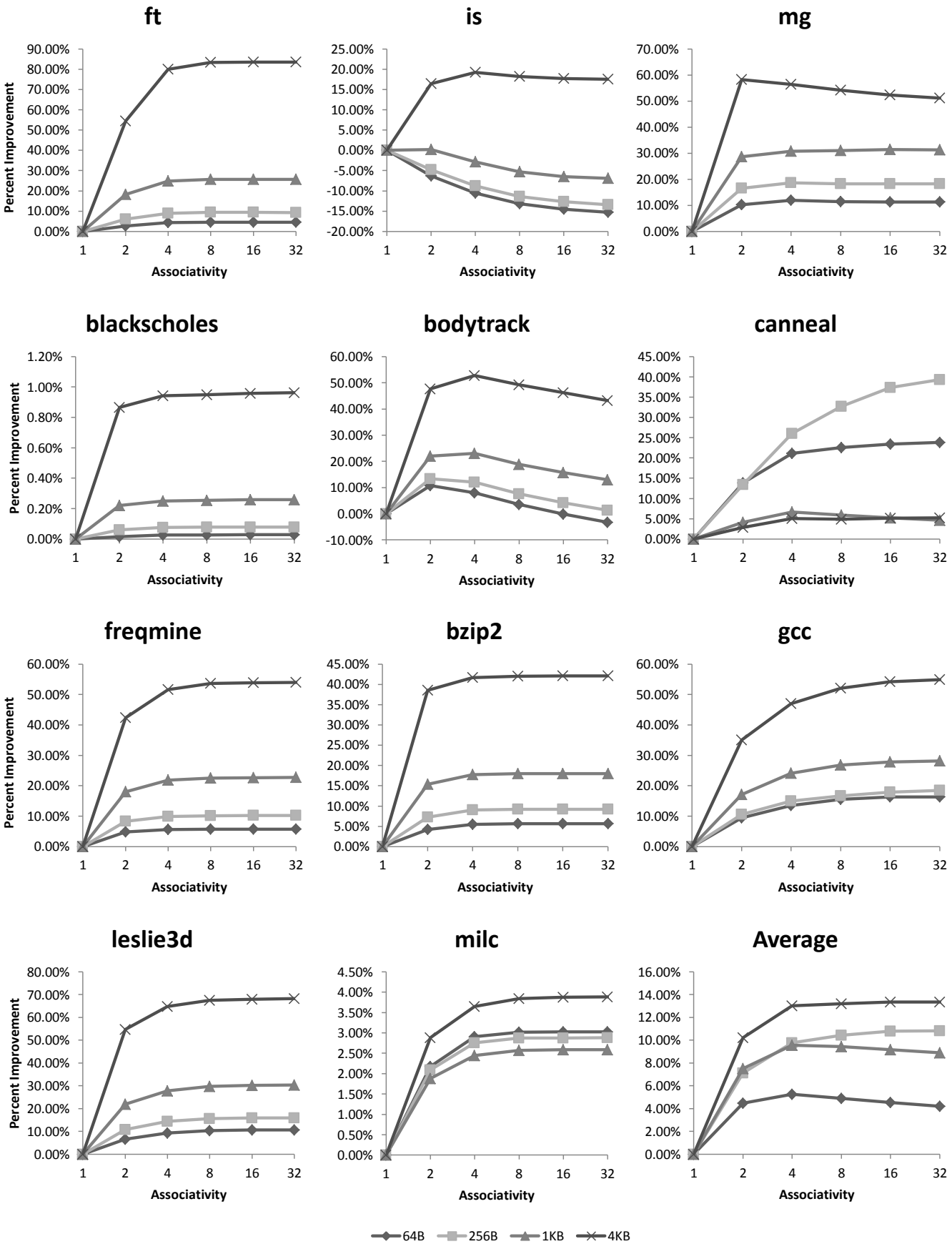

Figure 5: The effect of different cache line sizes and different degrees of associativity on the miss rate of the cache. Each series is the percent improvement over the miss rate of a direct mapped cache with the same line size. This was done to isolate the effects of associativity.

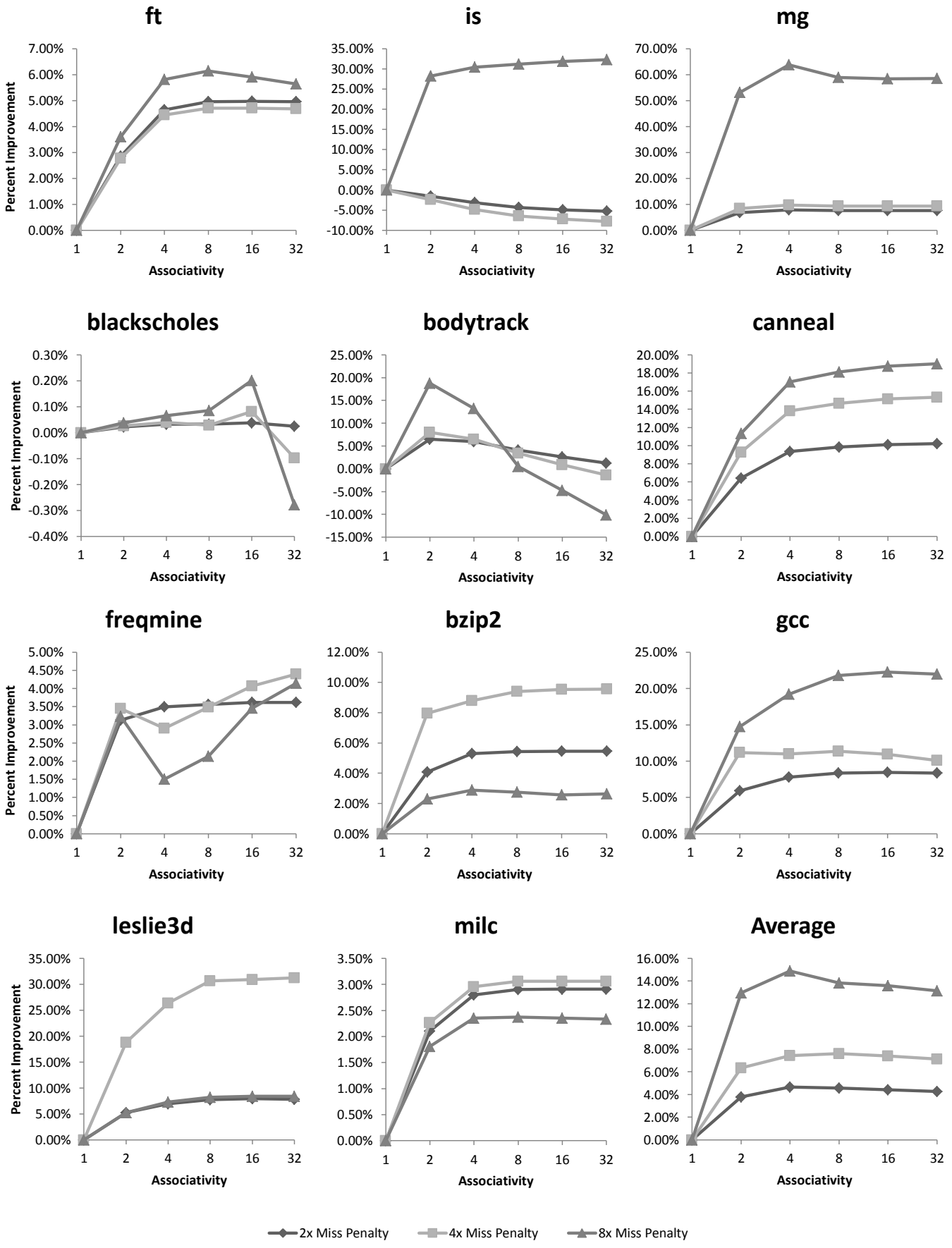

Figure 6: The effect of different miss penalties relative to hit latency and different degrees of associativity on the average access latency of the cache. Each series is the percent improvement over the average access latency of a direct mapped cache with the corresponding miss penalty. This was done to isolate the effects of associativity.