A Row Buffer Locality-Aware Caching Policy for Hybrid Memories

HanBin Yoon, Justin Meza, Rachata Ausavarungnirun, Rachael Harding, Onur Mutlu
Carnegie Mellon University

1 Problem

Phase change memory (PCM) is a promising memory technology that can offer higher memory capacity than DRAM. Unfortunately, PCM's access latency and energy are higher than DRAM and its endurance is lower. DRAM-PCM hybrid memory systems [1–4] use DRAM as a cache to PCM to achieve the low access latency and energy, and high endurance of DRAM, while taking advantage of the large PCM capacity.

A key question is what data to cache in DRAM to best exploit the advantages of each technology while avoiding their disadvantages as much as possible.

2 Our Solution: Summary

We propose DynRBLA, a fundamentally new caching policy that improves hybrid memory performance and energy efficiency. Figure 1 illustrates the organization of a DRAM-PCM hybrid memory system. Our observation is that both DRAM and PCM contain row buffers which cache the most recently accessed row. Row buffer hits incur the same latency in DRAM and PCM, whereas row buffer misses incur longer latencies in PCM. To exploit this, we devise a policy which tracks the access and row buffer misses of a subset of recently used rows in PCM, and caches in DRAM only rows which are likely to miss in the row buffer and be reused.

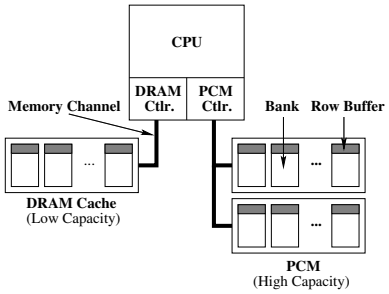


Figure 1: Hybrid memory organization.

3 Row Buffer Locality-Aware Caching

3.1 Motivation and Key Insight

The example in Figure 2 illustrates how row buffer locality-oblivious data placement can result in suboptimal application performance. This figure displays the service timelines of memory requests to DRAM and PCM assuming conventional data mapping and row buffer locality-aware data mapping. For simplicity, we assume that requests to rows A and B always miss in the row buffer (low row buffer locality) and requests to rows C and D always hit in the row buffer (high row buffer locality).

With a *conventional data mapping* scheme which is unaware of row buffer locality, it is possible to map rows A and B to PCM and rows C and D to DRAM (top half of Figure 2). Requests to rows A and B will always miss in the PCM row buffer and access the PCM device at a high latency. Requests to rows C and D hit frequently in the row buffer and thus do not receive much benefit from being placed in DRAM versus being placed in PCM where the row buffer hit latency is the same.

In contrast, a *row buffer locality-aware* caching mechanism would cache rows with high row buffer locality in PCM and low row buffer locality in DRAM (bottom half of Figure 2). The data for accesses which frequently lead to row buffer misses are placed in DRAM, where the row buffer miss latency is less than in PCM. This leads to reduced memory access latency overall.

The crucial observation is that *both DRAM and PCM devices employ row buffers which can be accessed at similar latencies*: placing data which frequently miss in the row buffer, in DRAM, can reduce application stall time, while leaving data which frequently hit in the row buffer, in PCM, will not increase application stall time by much more than if that data were placed in DRAM.

3.2 Mechanism

Based on these insights, we propose our *Row Buffer Locality Aware (RBLA)* caching mechanism.

Measuring Row Buffer Locality and Reuse. To measure row buffer locality and reuse information, we use a small structure called the *statistics store* in the memory controller which monitors the number of row buffer misses and accesses which have occurred for each of a subset of most recently used rows in PCM. The miss and access counters in the statistics store approximate the row buffer locality and reuse of the row they track.

Determining When to Cache Rows. RBLA caches a row when its number of row buffer misses and accesses exceed certain threshold values. Caching rows based on their row buffer locality and reuse attempts to ensure that data is only migrated when it is useful to do so. This affects system performance in several ways. First, placing in DRAM rows which have low row buffer locality improves average memory access latency, due to DRAM's lower row buffer miss latency. Second, by selectively caching data, RBLA reduces data movement while still caching beneficial data, reducing memory bandwidth consumption and allowing more bandwidth to be used to service demand requests. Third, caching data with high reuse helps balance the memory request load across both DRAM and PCM.

Dynamic Threshold Adaptation. To improve the re-

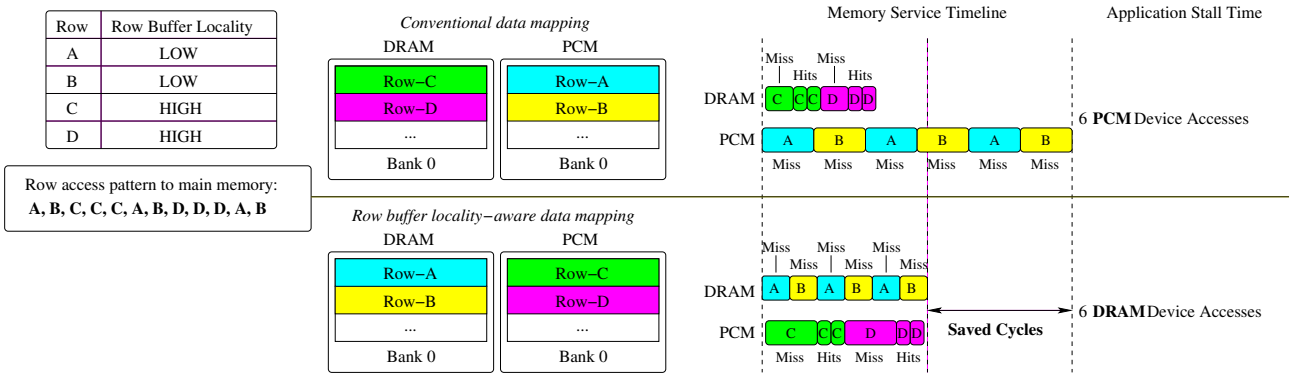


Figure 2: Conceptual example showing the importance of row buffer locality-awareness in hybrid memory data placement decisions.

siliency of RBLA to workload and system variations, we propose a technique which dynamically adjusts the caching thresholds at runtime, DynRBLA. The key observation behind our technique is that the number of cycles saved by caching rows in DRAM should outweigh the bandwidth cost of migrating that data to DRAM. Our mechanism monitors and estimates the first order costs and benefits of employing a given set of threshold values, over intervals. We quantify the cost as the number of migrations generated times the latency per migration, and the benefit as the aggregate number of cycles saved due to accessing data in DRAM at a lower latency than PCM. Our DynRBLA technique uses a simple hill-climbing algorithm, evaluated at the end of each interval, to adjust the caching thresholds in the direction of increasing net benefit.

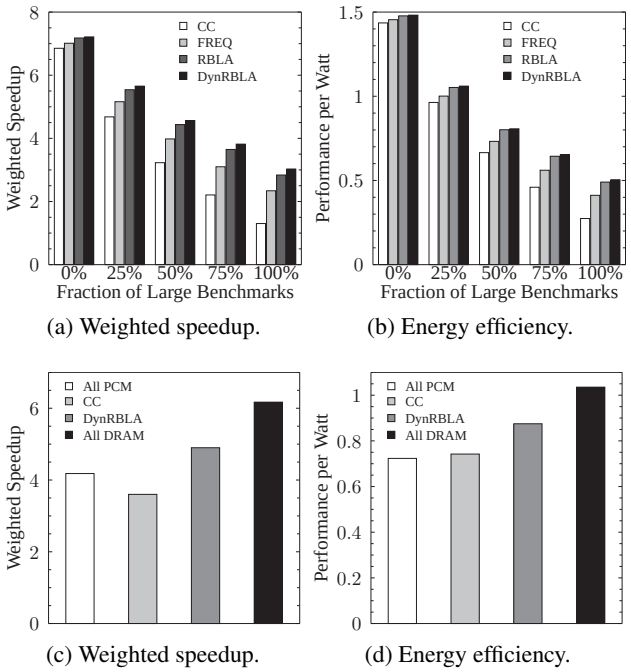
3.3 Implementation

Row Migration. The DRAM is mapped as a region of the virtual address space and the OS updates the virtual addresses of 2 KB rows migrated to and from the DRAM. The memory controller raises an interrupt to the OS to trigger a migration of the row. The OS manages the DRAM capacity in a 16-way set associative cache organization, and invalidates TLB entries for migrated rows.

Statistics Store. We find that a 16-way 32-set statistics store (2.3 KB) with LRU replacement achieves performance within 4% of an unlimited-sized stats store.

4 Results

Figure 3 shows our 16-core evaluations using multiprogrammed workloads consisting of SPEC CPU2006 benchmarks. We model a 256 MB DRAM cache to an 8 GB PCM main memory. Compared to a cache management technique that only takes into account the frequency of accesses to data (denoted as FREQ in Figure 3), our row buffer locality-aware scheme improves performance by 15% and energy efficiency by 10%. DynRBLA also improves system performance by 17% over an all-PCM memory, and comes to within 21% of the performance of an unlimited-size all-DRAM memory system.



(a) Weighted speedup. (b) Energy efficiency.

(c) Weighted speedup. (d) Energy efficiency.

Figure 3: 16-core system evaluations comparing the various caching techniques (a, b) and against systems with all PCM and all DRAM (c, d). CC = Cache data on every PCM access.

References

[1] G. Dhiman et al. PDRAM: a hybrid PRAM and DRAM main memory system. DAC. ACM, 2009.

[2] M. K. Qureshi et al. Scalable high performance main memory system using phase-change memory technology. ISCA'09.

[3] L. E. Ramos et al. Page placement in hybrid memory systems. ICS '11, pages 85–95.

[4] W. Zhang et al. Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures. PACT. IEEE Computer Society, 2009.