TDV cache: Organizing Off-Chip DRAM Cache of NVMM from a Fusion Perspective

Tianyue Lu, Yuhang Liu, Haiyang Pan, Mingyu Chen
State Key Laboratory of Computer Architecture, Institute of Computing Technology, CAS
University of Chinese Academy of Sciences
Beijing, China
lutianyue@ict.ac.cn, liuyuhang@ict.ac.cn, panhaiyang@ict.ac.cn, cmy@ict.ac.cn

***Abstract*—Emerging Non-Volatile Memory (NVM) provides both larger memory capacity and higher energy efficiency, but has much longer access latency than traditional DRAM, thus DRAM can be used as an efficient cache to hide the long latency of Non-Volatile Main Memory (NVMM) system. Transparent Off-chip DRAM cache (TOD cache) is a new DRAM cache structure where off-chip DRAM module is used as L4 cache and managed by hardware. The capacity and latency ratio of TOD cache over NVM are both quite different from those of traditional on-chip SRAM or die-stacked DRAM cache over off-chip DRAM memory. All the factors including hit latency, miss latency and hit rate need to be re-considered for TOD cache design. In this study, we first point out that three types of traditional cache schemes cannot be used directly for TOD cache, since set-associative cache suffers from extra tag lookup latency, direct-mapped cache has low hit rate and tag cache is too small to efficiently hold the working sets of tags for DRAM cache. Based on these observations, we propose a novel cache scheme, TDV, that fuses these three different types of cache together to take their advantages. In TDV, a direct-mapped cache is used as the first-level cache to achieve short access latency, a set-associative victim cache is taken as the second-level cache to obtain extra high hit rate, and a SRAM tag cache only serves for the victim cache rather than the whole DRAM cache and thus improves the hit rate of tag cache significantly. The simulation results show that, TDV cache has a performance improvement of 6.3% and 8.3% on average than state-of-the-art direct-mapped (Alloy cache) and set-associative cache (ATCache) with same DRAM and SRAM capacity.**

***Keywords*-Non-Volatile Memory; DRAM cache; Cache associativity; Tag cache; Victim cache;**

I. Introduction

Traditional DRAM based main memory is facing difficulties to overcome the memory wall, especially to meet the growing memory capacity demand of emerging big-data applications [1], [2]. Non-Volatile Memory (NVM) becomes a promising memory technology to provide larger memory capacity and higher energy efficiency than DRAM memory [3]–[7]. However, NVM still has longer access latency especially write latency than conventional DRAM. So DRAM cache is a potentially efficient strategy to achieve short Average Memory Access Time(AMAT) and has been paid attention by industry [8]–[14]. For instances, Intel and Micron have announced their 3D XPoint memory technology for availability in 2018, which uses the DRAM cache together with NVM to combine their advantages [15]. JEDEC is also working on a hybrid memory module standard NVDIMM-P which supports memory-like access to NVM [16].

In academia, many previous studies have researched hardware architectures and scheduling algorithms of DRAM cache [7]–[12], [14]. However, most of them assume on-chip die-stacked DRAM cache, and the main memory of their design is still based on off-chip DRAM module. When main memory becomes NVM, the capacity of main memory is far more larger than that of on-died DRAM. Therefore off-chip DRAM cache is proposed to satisfy the increasing capacity demand. Off-chip DRAM cache can be classified into two categories, software-managed and software-transparent. For software-managed off-chip DRAM cache, data swapping between DRAM and NVM is managed by software (i.e., applications or OS), thus the overhead of cache miss is large. Software-managed cache usually uses coarse-granularity such as 4KB page to reduce this overhead, but it is inflexible for memory extension especially for memory access patterns that have low spatial locality such as randomly memory accesses. In comparison, software-transparent off-chip DRAM cache uses hardware-managed data replacement to support a fine-grained data swapping size of L1 cache line, e.g. 64B, which makes miss latency short and data management efficient. Therefore, transparent off-chip DRAM cache is considered as a promising DRAM cache solution for NVMM in terms of performance. In this paper, we study the new features and design issues of Transparent Off-chip DRAM cache which will be referred to as TOD cache. Compared to die-stacked DRAM cache, TOD cache brings us a new design space that has at least three different aspects to investigate and explore.

The first difference is that, large off-chip DRAM cache brings more significant overhead of tag storage and access time. The amount of cache tags becomes too large to be stored in SRAM buffer on CPU chips. Most of them have to be stored in DRAM together with the cached data. Accessing tags in DRAM brings extra latency overhead. For set-associative

This work is supported by National Key Research and Development Plan of China 2017YFB1001600, NSFC (National Science Foundation of China) No. 61331008 ,61521092 and 61772497, State Key Laboratory of Computer Architecture foundation under grant CARCH2601 and Huawei YBCB2011030. Corresponding author is Yuhang Liu.

1063-6404/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCD.2017.19

cache, memory controller needs to access DRAM tag first to check whether data is cached. At least two memory dependent operations are needed, one for the tag access and matching, and the other for the following data access. This extra tag lookup latency makes traditional efficient set-associative cache inefficient.

The second difference is that, miss latency of TOD cache is high so that system performance is more sensitive to cache hit rate. Alloy cache presents a solution to reduce tag latency overhead that it stores tag and data together and brings them back together each time [17]. But in order to use this mechanism, the DRAM cache has to be organized as direct-mapped. It is well known that direct-mapped cache has lower hit rate than set-associative cache [18]. Since NVM access takes a much longer time than DRAM access, DRAM cache miss latency is much higher than hit latency. Low hit rate results in high amount of slow NVMM accesses and degrading performance.

The third difference is that, the capacity gap between off-chip DRAM and on-chip SRAM becomes larger, making it more difficult for SRAM tag cache to effectively hold the working set of tags of TOD cache. Tag cache is an SRAM buffer on CPU chip used for set-associative DRAM cache. Memory controller queries the tag cache first before looking up into DRAM cache. If accessed data tag matched in tag cache, additional memory access to DRAM for cache tag lookup will be eliminated. So only one DRAM access occurs on a tag cache hit. ATCache presents an effective tag cache scheme which achieves a 10.3% performance improvement compared to baseline set-associative cache [10]. Unfortunately, the hit rate of tag cache is sensitive to its capacity. For high-capacity off-chip DRAM cache, the amount of cache tags is much larger than before (16GB cache data is corresponding to 512MB tags), but tag cache is on-chip and thus its area and capacity is limited to several MB.

These three new differences make that original cache designs cannot be used directly without reconsideration on TOD cache and we make evaluations to analysis them in detail in Section III.

In this paper, we propose Tag-Direct-Victim cache (referred as to TDV cache), which fuses three types of different caches together. TDV cache is able to achieve high cache hit rate and low hit latency simultaneously. It uses direct-mapped scheme for the first-level cache which occupies 75% capacity, set-associative victim scheme for the second-level cache which uses only 25% capacity and for which tag cache only serves. Specifically, compared with direct-mapped cache with the same capacity, part of cache is organized to be set-associative to improve hit rate. Compared to set-associative cache, most cache-hit accesses happen on direct-mapped area which means that average memory access time is short. Meanwhile, the hit rate of tag cache can be improved since the tag working set has been reduced significantly. In TDV cache, a small SRAM tag cache serves only for set-associative victim area. Since this area takes only 1/4 capacity of whole DRAM cache, the cache/data ratio is increased effectively. Test results show that the average hit rate of tag cache is increased from 69.02% to 85.67%.

Our contributions of this work are summarized as two-fold:

On the one hand, we evaluate the characteristics of TOD cache of NVMM which is a new thing for community. We find that the advantage of set-associative cache is counteracted by extra tag latency, especially when TOD cache shows high hit rate. Direct-mapped cache has low hit rate which is much sensitive to high miss latency. We also find that, for high-capacity DRAM cache, small SRAM tag cache shows low hit rate, but the efficiency of tag cache is improved near-linearly when increasing of capacity ratio of tag cache over total cache tags.

On the other hand, based on our observations and analyses, we propose TDV cache which has advantages of direct-mapped, set-associative caches and tag cache. The whole DRAM cache is separated into two areas where one is direct-mapped and the other is set-associative. The direct-mapped area is used as the first-level cache and has low hit latency and the set-associative area is used as victim cache to improve hit rate. Due to the set-associative area only takes 25 percent of whole cache, the capacity ratio of tag cache and the total tags of set-associative cache in DRAM becomes so high that tag cache also reaches high efficiency. The evaluation results show that, our TDV cache has a performance improvement of 6.3% and 8.3% on average than Alloy cache and ATCache respectively which are state-of-the-art direct-mapped and set-associative cache designs.

The rest of this paper is organized as follows. Section II introduces the backgrounds of DRAM cache. Section III analyzes tradeoffs of different cache design policies and shows the motivation of our work. Section IV introduces the proposed TDV cache design and Section V shows evaluation results. Finally in Section VI, we conclude and present future work.

II. Backgrounds: NVMM and DRAM Cache

Non-Volatile Memories (NVM) such as PCM and ReRAM have not only larger capacity, but also larger latency than DRAM. The performance parameters of a memory-level PCM are shown in Table I. To combine the advantages of DRAM and NVM, hybrid memory scheme is more preferable, and the role of traditional DRAM has been often changed from main memory to cache.

TABLE I
Main parameters of DRAM and PCM cell [19]

	Read latency	**Write latency**	**Chip Capacity**
DRAM	5ns	5ns	4GB
MLC PCM	44ns	395ns	32GB-256GB

Researchers are investigating on NVM-aware last level cache (LLC) where the LLC is based on SRAM or eDRAM, so the capacity of LLC is limited due to the low density and high cost [20]. DRAM cache is a promising alternative to SRAM cache. Figure 1 shows different organizations of DRAM cache.

On-chip die-stacked DRAM cache can be in the order of hundreds of megabytes to several gigabytes. Die-stacked

DRAM also delivers several times more bandwidth than off-chip memory due to dense on-chip TSV buses. Many works have studied die-stacked DRAM cache [8]–[11], [17], [21]. However, most of them assume that main memory is based on off-chip DRAM rather than NVMM, as shown in Figure 1A. When replacing off-chip DRAM with NVMM, the hit latency, miss latency and capacity are all quite different, so previous research results need to be re-considered. One of the limitation of die-stacked DRAM is capacity, which is relatively small compared to hundreds GB of NVMM.

As shown in Figure 1B, software-managed hybrid memory of DRAM and NVM is another promising technology for NVMM [22]–[24]. The software scheme has least limitation so large capacity off-chip DRAM and I/O bus attached NVM can be used directly. Off-chip DRAM cache and NVM take separate memory spaces and data migration is managed by software (i.e., applications or OS). Each software managed data migration takes a long latency overhead of several microseconds. So that data are usually swapped by page granularity (4KB or more) which results in that data are over-fetched on a cache miss. As NVM is low-bandwidth memory, coarse-grained data swap becomes the performance bottleneck of page-based DRAM cache. Page-based cache is very inefficient for applications with low spatial locality such as random memory access pattern.

Figure 1C shows the newest Transparent Off-Chip DRAM cache (TOD cache) scheme studied in this work. Off-chip DRAM is managed as cache of NVMM by hardware in a fine-granularity (e.g. 64B) and is transparent to software. TOD cache requires that NVMM can be randomly accessed through a memory bus like DDRx. TOD cache also requires significantly change to CPU memory controller to manage both off-chip DRAM cache and NVMM. Though challenging, TOD Cache is being investigated by industry due to its potential advantages. For the proposed Intel 3D XPoint memory, PCM is used as NVMM together with a Transparent Off-chip DRAM cache.

TOD cache has low access latency, high data efficiency and no software overhead. The capacity of TOD cache can easily reach tens of GB, which is comparable to NVMM which can be hundreds of GB and TDV cache is flexible to scale. TOD cache enables independent design of the processor and memory modules. The memory system can be upgraded or extended after the CPU chip had been released. Such flexibility is impossible for die-stacked DRAM cache.

TOD Cache is quite different with previous die-stacked DRAM cache and has three main factors to be further investigated: 1) The effect of extra DRAM tag access latency on set-associative organization; 2) The effect of long miss latency which is the latency of an NVMM access; 3) The effect of large cache capacity. These factors bring new performance features of TOD cache and make that previous cache design schemes cannot be used directly for TOD cache designs. Detailed analyses will be conducted in Section III.

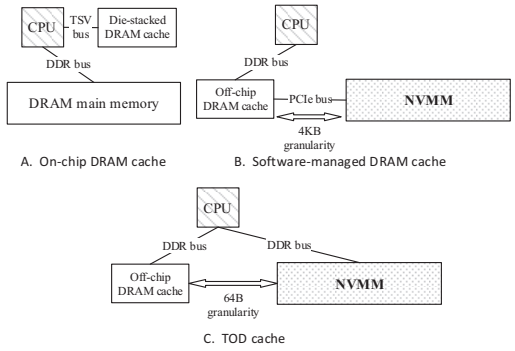

Fig. 1. Different types of DRAM cache

III. DRAM Cache Analysis

In this section, we evaluate the design tradeoff of TOD cache. The used benchmarks are shown in Table II. We mix selected workloads from SPECCPU 2006. Besides SPECCPU 2006, SAP HANA [25] represents database applications and data and query is generated by TPC-DS [26]. We also use pagerank from BigDataBench [27] which represents query algorithms. We use NVMain to simulate the target system [28]. System configuration of which will be shown in Section V-A.

A. Effect of Tag Latency of Set-associative Cache

For all types of cache, cache tag is needed to indicate which data is buffered in cache. Before accessing the cached data, its cache tag is first checked. On cache hit, access to cached data will be issued. On cache miss, an access to the original memory address will be issued, and then the fetched data is put into cache according to certain insertion policy.

In our configuration, TOD cache provides a cache with 32GB capacity, and accordingly the amount of cache tags is also large. Assuming 8-bit tag for each 64B cache line, the total tag size for 32GB is 512MB. In such case, cache tags must be stored in DRAM instead of SRAM buffer. Although cache tags will not occupy many percents of the total DRAM cache capacity, cache tag access does take one full DRAM access latency and thus cannot be ignored.

Direct-mapped cache is a low-hit-latency design that tag and data are stored together in one extended cache block. Alloy cache extends the conventional cache line granularity from 64B to 72B to contain tag and data together [17]. One DRAM burst access reads tag and data together so that hit latency is reduced to the latency of one DRAM read. However, Alloy cache lacks the advantage of high hit rate. According to prior research, direct-mapped cache has a lower hit rate than set-associative cache [18].

We make evaluations to compare the performance of direct-mapped and set-associative TOD Cache. Figure 2 shows the results. Ideal set-associative cache represents the set-associative organization without extra tag latency and has the same hit latency with direct-mapped cache. We find that ideal set-associative cache has a great performance advantage due to

TABLE II
OVERVIEW OF BIG-MEMORY BENCHMARK

Workload	Memory Bandwidth(GB/s)	Footprint(GB)	Description
SPEC-MIX1	17.92 (peek to 34.8)	195.2	401.bzip2 429.mcf 433.milc 464.h264ref 447.dealll 450.soplex 482.sphinx3 462.libquantum
SPEC-MIX2	12.48 (peek to 33.76)	206.8	401.bzip2 403.gcc 433.milc 464.h264ref 447.dealll 450.soplex 482.sphinx3 462.libquantum
SPEC-MIX3	16.4 (peek to 34.64)	187	400.perlbench 401.bzip2 433.milc 464.h264ref 447.dealll 450.soplex 482.sphinx3 462.libquantum
SPEC-MIX4	13.28 (peek to 34.48)	253.4	400.perlbench 401.bzip2 403.gcc 429.mcf 447.dealll 450.soplex 482.sphinx3 462.libquantum
SPEC-MIX5	10.72 (peek to 14.24)	130.6	400.perlbench 401.bzip2 403.gcc 429.mcf 433.milc 464.h264ref 482.sphinx3 462.libquantum
SPEC-MIX6	10.16 (peek to 38.32)	160.5	400.perlbench 401.bzip2 403.gcc 429.mcf 433.milc 464.h264ref 447.dealll 450.soplex
SPEC-MIX7	14.8 (peek to 24.4)	172.3	400.perlbench 401.bzip2 403.gcc 429.mcf 464.h264ref 447.dealll 482.sphinx3 462.libquantum
SPEC-MIX8	22.8 (peek to 30.4)	152.3	400.perlbench 401.bzip2 403.gcc 429.mcf 433.milc 447.dealll 450.soplex 482.sphinx3
SAP HANA	4.16 (peek to 30.88)	111.9	In-Memory Database
Pagerank	9.28 (peek to 34.48)	253.2	Search Engine

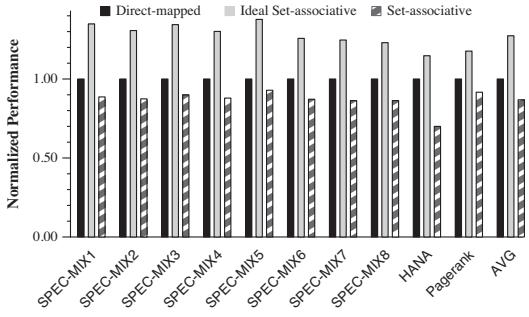

Fig. 2. System performance of direct-mapped and set-associative cache

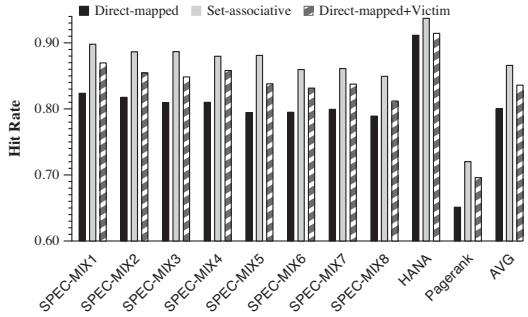

Fig. 3. Hit rate increasing of victim cache

its high hit rate. But when considering the tag latency, set-associative cache has lower overall performance even than direct-mapped cache. Therefore, the problem of tag latency is so critical that prevents TOD cache from using set-associative directly.

B. Effect of Associativity on Hit Rate

Although set-associative suffers from tag latency, it provides higher hit rate than direct-mapped one, because set-associative cache can effectively reduce cache conflict misses. Lower miss rate implies fewer NVMM accesses which are much slower. Specifically, for TOD cache, the cache miss penalty could be up to 10 times of cache hit latency. Therefore, TOD cache designers must consider high hit rate as an important target to achieve final high performance.

Victim cache is a second-level cache in which data evicted from first-level cache will be inserted. We find that, configuring set-associative victim cache together with direct-mapped cache is an efficient way to reduce conflict cache misses. As shown in Figure 3, we make an evaluation to compare the hit rate of different cache organizations. In our evaluation, we take 1/4 of DRAM capacity to be victim cache which is organized as set-associative. Although the first-level direct-mapped cache loses 1/4 capacity, the overall cache hit rate is increased. The hit rate of Direct-mapped+victim cache is close to the hit rate of ideal set-associative cache. Victim cache decreases cache conflict misses of direct-mapped cache effectively.

C. Effect of Tag Cache Capacity on Tag Cache Hit Rate

One way to mitigate the problem of tag latency is put all tags in on-chip SRAM. Page-based DRAM caches, such as Unison cache and Footprint cache [9], [29], manage data in page size, e.g. 4KB. This would reduce the amount of tags significantly. For the above case, 32GB TOD cache with 4KB granularity needs only 8MB tags which is acceptable for an on-chip SRAM buffer. But when replacement occurred in DRAM cache, it has to take a long time and lots of bandwidth to swap a page completely. If the miss rate is high, the performance of memory system would still be extremely low. Besides, capacity of SRAM buffer on chip is still limited and cannot match the pace of DRAM cache whose capacity is increasing. As a result, Tag-in-SRAM is not a scalable design.

Tag cache is alternative way to reduce the extra tag latency of fine-grained TOD cache. A tag cache is an on-chip SRAM cache only for storing cache tags. Each time a memory access will first check whether the tag is cached in tag cache. If matched, another DRAM access to the cached data will be issued immediately. Only when tag is not found in tag

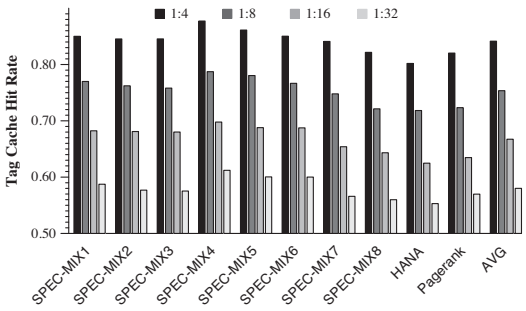

Fig. 4. Tag cache hit rates in different tag cache capacities

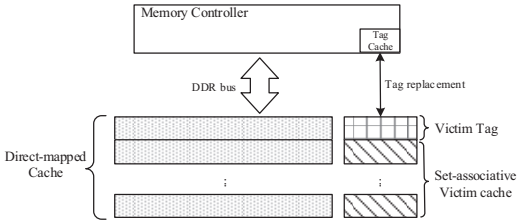

Fig. 5. Structure of TDV cache

cache should an access to DRAM cache tag be issued. A tag cache with high hit rate will reduce most unnecessary DRAM accesses to tag space.

Figure 4 shows the comparison results of tag cache hit rates for different tag cache capacities. In this evaluation, we add tag cache prefetching mechanism to improve hit rate as in [10]. We can see that, with the tag capacities increasing, the tag cache miss rate drops down dramatically. If a 2MB tag cache is used for 64MB DRAM tag space, as 1:32 in Figure 4, the average hit rate will drop from 84.1% to below 58%. This implies that for large amount of DRAM cache tags, a small tag cache fails to hold the working set of tags and thus cannot help much for performance.

According to our above evaluations, we conclude the main issues of TOD cache design as: 1) Extra tag latency prevents set-associative organization; 2) High miss latency of NVMM demands high hit rate; 3) Low efficiency of small-capacity tag cache. Motivated by these observations, we propose TDV cache scheme, which fuses direct-mapped, set-associative and tag cache together to solve these three problems simultaneously.

IV. Tag-Direct-Victim cache

In this section, we will propose Tag-Direct-Victim cache (TDV cache) architecture. According to the evaluation results in Section III, small tag cache is not effective for large capacity DRAM cache, and set-associative cache can improve hit rate but suffers from long tag access latency, and direct-mapped cache has short hit latency but high conflict miss rate. The main purpose of TDV cache is to take a good utilization of SRAM tag cache, DRAM cache and DRAM tags to explore as many as possible the advantages of direct-mapped, set-associative cache and tag cache to achieve low average cache hit latency and high cache hit rate simultaneously.

Figure 5 shows the structure of TDV cache. As our first leverage, victim cache is used to mitigate the cache conflict miss rate of direct-mapped cache. We use direct-mapped scheme for first-level DRAM cache. Most of memory accesses will be served by direct-mapped cache according to the evaluation. Since NVMM access after miss needs about a 10-fold longer latency, even a small amount of cache misses access to NVMM would bring relevant performance degradation. A set-associative victim cache is integrated with direct-mapped cache to improve the total hit rate. For the direct-mapped cache, cache misses caused by cache block conflicts occur frequently because many memory addresses are mapped to the same set which has only one cache block space. By adding set-associative victim cache, cache blocks evicted from direct-mapped area could be stored in victim cache and reused later. As such, these cache blocks could still exploit the advantage of set-associative cache to avoid being evicted prematurely from DRAM cache.

However, a set-associative victim cache hit access still needs 2-3 longer latency than direct-mapped cache, which counteracts the hit rate improvement. As our second leverage, tag cache is used to reduce extra tag latency. Specifically, the key point of our design is that ONLY tags for set-associative victim cache are cached in a SRAM tag cache to utilize the limited space as well as possible. Since set-associative victim cache occupies only 1/4 of total DRAM cache, the total tags that need to be cached in DRAM are reduced dramatically. In such manner, we improve tag cache hit rate by increasing the ratio of tag cache capacity over the total tag capacity.

Figure 6 shows the flow of one memory access in TDV cache. Each memory access first checks tag cache. A tag cache hit represents that the corresponding tags for current access cache set are in SRAM tag cache. So whether the to-be-accessed data is in victim cache set can be determined immediately. If tag cache is missed, the direct-mapped cache is checked first. If direct-mapped cache misses, the tag of set-associative victim cache is loaded into tag cache to check whether the target data is in victim cache. If yes, access goes to victim cache. If no, a cache replacement process will start.

Figure 7 shows the DRAM cache organization. The whole DRAM cache is divided into three areas, direct-mapped area, victim cache area and victim-tag area. Direct-mapped area takes the most capacity and victim cache area is relatively small. Victim-tag area stores all the tags of set-associative victim cache. In our study, direct-mapped area takes 75 percent of total space and victim cache and tags takes the rest. Direct-mapped area works like normal direct-mapped cache and victim cache contains data which is evicted from direct-mapped cache. Besides, victim cache is set-associative and each set is associated with a fixed number of direct-mapped area blocks. In Figure 7, we can see that 24 direct-mapped cache blocks are associated to one victim cache set which contains 8 cache blocks. This mapping relation indicates where

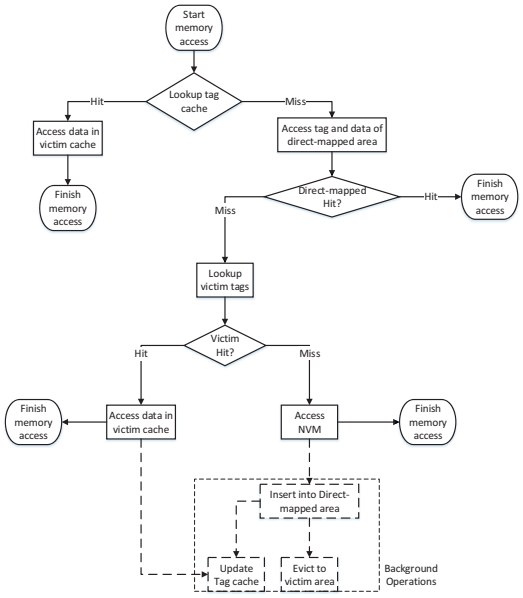

Fig. 6. Memory access flow on TDV cache

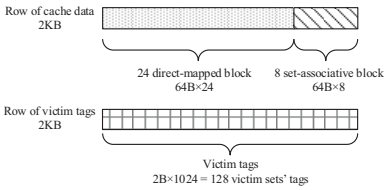

Fig. 7. DRAM row organization of TDV cache

a cache block to be inserted when it is evicted from direct-mapped area. Each cache block is 64 bytes for a minimum DRAM burst, so the total 32 cache blocks occupy a full DDR4 DRAM 2KB row.

The tags for direct-mapped cache lines are put in the ECC bits and thus need not extra storage as in previous works [30]. Each victim cache set needs 8 extra tags, and each tag takes 9 bits. We choose 2-byte tag for each victim cache block with other bits for the valid and dirty flags. For an 8-way set-associative victim cache, each set only needs 2Bx8=16B for tagging while a cache block is 64B, thus it is not economic to put tag together with victim cache data in the same DRAM row. On the contrary, we choose to use a centralized victim cache tag storage on DRAM. Such design is convenient for tag cache prefetching. Every DRAM burst read will prefetch tags of 4 sets that cover 4 DRAM rows and 32 cache lines.

TDV cache keeps the advantage of direct-mapped cache. In our evaluation in Section V, we find that most cache hits occur in direct-mapped area. The ratio of cache hit in direct-mapped area and victim cache is 6.17 on average. Additionally, TDV cache improves tag cache efficiency. In Section III, we have found that, tag cache hit rate grows with the increasing of tag cache capacity. For TDV cache, tag cache is only associated to the victim cache area. This means, compared to normal set-associative cache, tag cache only corresponds to 1/4 capacity of DRAM cache. Our evaluation also shows that, tag cache hit rate increases from 69.06% to 85.67% relative to full-set-associative cache using the same tag cache capacity.

TABLE III
SYSTEM CONFIGURATIONS

Cache Capacity	1/8 of each programs' footprint
Memory	2 64-bit Channels, 2 Ranks/Channel, 8 devices/Rank, x8-width (1 device)
Device	DDR3-1333MHz, 8 banks, 2KB Row Buffer, 32768 Rows/bank, Time/power parameters from Micron SDRAM [32]
Tag Cache	16MB
Alloy cache	
Access latency	One DRAM read for cache tag and data
ATCache	
Associativity	32-way, Random replacement policy
Access latency	One DRAM read for cache tag and one read for data if tag cache miss
TDV Cache	
Associativity	Victim cache area: 8-way, Random replacement policy
Access latency	One DRAM read for direct-mapped area Two DRAM reads for victim area if tag cache miss

V. EVALUATIONS

A. Experimental Methodology

Benchmarks: We evaluate 8 mixed SPECCPU 2006 benchmarks and two other big-memory benchmarks which are described in Section III.

Simulation Framework: We use NVMain simulator to evaluate our TDV cache [28]. The simulator configuration is shown in Table III. Our traces are collected from benchmarks running on an Intel E5 2620 v2 CPU platform equipped with eight 32GB DDR3 LRDIMM (256GB memory capacity) [31]. For performance analysis, we use sampled trace segments and NVMain integrated with checkpoint support. For each benchmark, one thousandth of the original full traces are sampled to be simulated in NVMain. The total number of running traces of each benchmark in NVMain is 1 billion. We compare TDV cache with Alloy cache and ATCache that have the same capacity of whole DRAM cache. According to the work in [10], we add tag cache prefetching for TDV cache. Note that ATCache uses the same hit prefetching mechanism.

B. Results and Analysis

First we evaluate the hit rates of different DRAM cache organizations. Because the hit rate calculation does not need cycle-accurate simulation, the evaluation is very fast and can be conducted using complete program traces. The results are shown in Figure 8. On average, TDV cache improves the hit rates by 4.5% over Alloy cache. TDV cache's hit rate is also close to ATCache up to 96.5%. Due to high cache hit rate by using large-capacity DRAM cache, the miss rate is reduced by 18.0% on average compared to Alloy cache. The result implies that small set-associative victim cache in TDV cache improves the hit rate effectively.

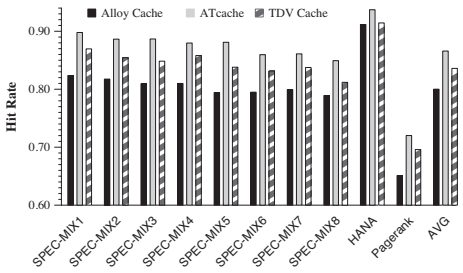

Fig. 8. Hit rates of DRAM cache using different cache organizations

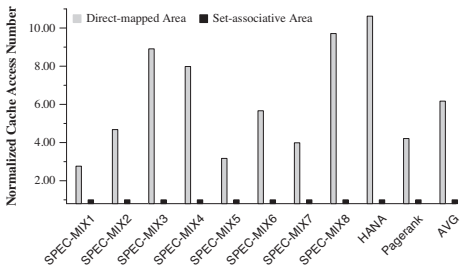

Fig. 9. Ratio of cache hit accesses on direct-mapped area and victim cache

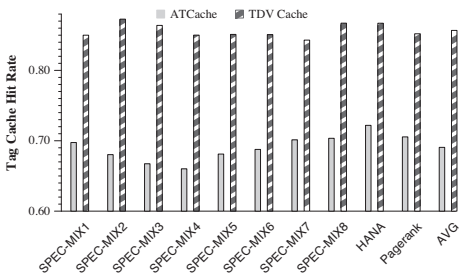

Fig. 10. Tag cache hit rates of TDV cache and ATCache

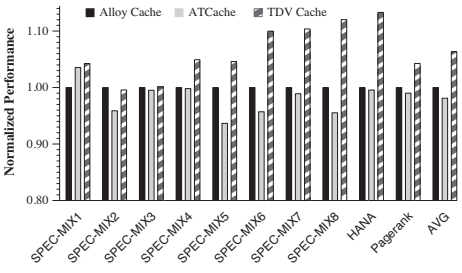

Fig. 11. System performance of different cache configurations

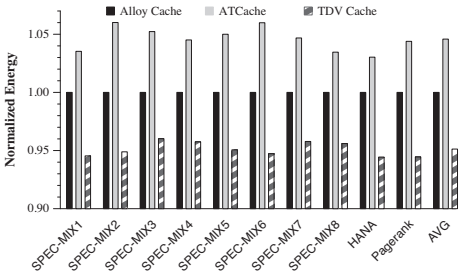

Fig. 12. Energy consumption of different cache configurations

Figure 9 shows the ratio of the numbers of hits in direct-mapped area over that in victim cache area. On average, the ratio is 6.17, which implies that most DRAM cache hits happen in direct-mapped area and thus low average hit access latency has been attained.

Secondly, we compare the hit rate of tag cache with ATCache. In experiments, we add tag cache prefetching to read tags of continuous 4 sets to tag cache once tag cache replacement occurs. This hit prefetching policy is the same as that in ATCache. Results are shown in Figure 10. TDV's tag cache shows a higher hit rate than ATCache's tag cache by 24.05% on average. As victim cache only takes 25% of whole DRAM cache capacity, the capacity ratio of tag cache over victim cache tag space becomes 1:8 while the ratio is 1:32 in ATCache. The increasing of capacity ratio eases the cache competition and reduces the tag cache misses caused by conflicts.

With the improved DRAM cache hit rate and tag cache hit rate, we also evaluate the accurate performances of different DRAM cache configurations using NVMain on sampled trace segments. Figure 11 shows the simulated running time comparison of different cache structures using the same trace inputs. The values are normalized to those of Alloy cache. TDV cache has the best performance among all the cache configurations. Compared to Alloy cache, the maximum performance improvement of TDV cache is more than 10% for HANA and SPEC-MIX8. Compared to Alloy cache and ATCache, TDV cache achieves 6.3% and 8.3% performance improvement respectively. This validates our analysis in Section IV, and verifies that TDV cache achieves short hit latency and high hit rate simultaneously.

Besides performance results, we also evaluate the energy consumptions. Figure 12 shows the energy results that are normalized to the values of Alloy cache. The energy consumption of TDV cache is less than that of Alloy cache by 4.9% on average (excluding the energy consumption of tag cache).

We also evaluate the effect of different capacity ratio of direct-mapped area over victim area. The results are shown in Figure 13. Smaller set-associative victim cache can improve tag cache hit rate further, but the DRAM cache hit rate becomes lower. On the contrary, larger set-associative victim cache increases DRAM cache hit rate but decreases the hit rate of tag cache. As shown in Figure 13, the ratio of 3:1 achieves the optimal performance for most programs. Compared to the ratio of 1:1 and 7:1, the performance improvements of 3:1 are 1.01% and 4.93% on average, respectively.

VI. Conclusions

Providing an efficient DRAM cache design for emerging NVMM is necessary but challenging. High capacity ratio of TOD cache over NVMM brings a new design space for cache architecture design. Tags of TOD cache must be stored off-chip due to its large size. Off-chip tag access latency makes

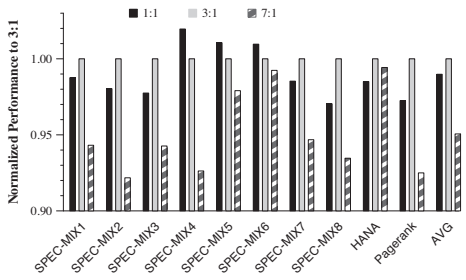

Fig. 13. System performance of different capacity ratios of direct-mapped area over victim area

it difficult to directly adopt the complex set-associative cache design. Our analysis shows that direct-mapped cache achieves low hit rate but set-associative cache suffers from extra tag latency even with a small SRAM tag cache. We propose Tag-Direct-Victim cache (TDV cache) scheme to combine all advantages from direct-mapped, set-associative and tag cache. By associating SRAM tag cache only with a set-associative victim cache which only takes 1/4 capacity of whole DRAM cache, TDV cache achieves low cache hit latency and high cache hit rate simultaneously. Evaluations show performance improvement of 6.3% and 8.3% in average than state-of-the-art direct-mapped (Alloy cache) and set-associative cache (ATCache). TDV cache presents an effective tradeoff in the huge design space of TOD cache of NVMM.

REFERENCES

[1] Y. H. Liu and X. H. Sun, "Lpm: Concurrency-driven layered performance matching," in *International Conference on Parallel Processing*, pp. 879–888, 2015.

[2] Y. Liu and X.-H. Sun, "Reevaluating data stall time with the consideration of data access concurrency," *Journal of Computer Science and Technology*, vol. 30, no. 2, pp. 227–245, 2015.

[3] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *ACM SIGARCH Computer Architecture News*, vol. 37, pp. 2–13, ACM, 2009.

[4] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," *ACM SIGARCH Computer Architecture News*, pp. 24–33, 2009.

[5] A. P. Ferreira, B. Childers, R. Melhem, D. Mossé, and M. Yousif, "Using PCM in next-generation embedded space applications," in *2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 153–162.

[6] G. H. Loh and M. D. Hill, "Efficiently enabling conventional block sizes for very large die-stacked dram caches," in *MICRO-44*.

[7] M. Wu and W. Zwaenepoel, "eNVy: a non-volatile, main memory storage system," in *ACM SIGOPS Operating Systems Review*, 1994.

[8] L. Zhao, R. Iyer, R. Illikkal, and D. Newell, "Exploring dram cache architectures for cmp server platforms," pp. 55–62, 2007.

[9] D. Jevdjic, G. H. Loh, C. Kaynak, and B. Falsafi, "Unison cache: A scalable and effective die-stacked dram cache," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 25–37, 2014.

[10] C. C. Huang and V. Nagarajan, "Atcache: reducing dram cache latency via a small sram tag cache," in *International Conference on Parallel Architectures and Compilation*, pp. 51–60, 2014.

[11] N. Gulur, M. Mehendale, R. Manikantan, and R. Govindarajan, "Bi-modal dram cache: A scalable and effective die-stacked dram cache," in *Ieee/acm International Symposium on Microarchitecture*, pp. 38–50, 2014.

[12] Y. H. Liu and X.-H. Sun, "C^2-bound: a capacity and concurrency driven analytical model for many-core design," in *the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–11, 2015.

[13] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, *et al.*, "Die stacking (3d) microarchitecture," in *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, pp. 469–479, IEEE, 2006.

[14] T. Jiang, P. Su, P. Kim, C. Bassett, K. Sichak, J. Gandhi, J. Li, J. Im, R. Huang, and P. S. Ho, "Investigation of thermo-mechanical stresses and reliability of 3d die-stack structures by synchrotron x-ray micro-diffraction," in *ECTC*, pp. 1718–1724, IEEE, 2015.

[15] Micron, "Introducing Micron QuantX Technology, Based on 3D XPoint Memory." https://www.micron.com/about/our-innovation/3d-xpoint-technology, 2016.

[16] Micron, "NVDIMM." http://www.micron.com/products/dram-modules/nvdimm.

[17] 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 *MICRO-45*.

[18] W. A. Wulf and S. A. McKee, "Hitting the memory wall: implications of the obvious," *ACM SIGARCH computer architecture news*, vol. 23, no. 1, pp. 20–24, 1995.

[19] H. G. Lee, S. Baek, and J. Kim, "A compression-based hybrid mlc/slc management technique for phase-change memory systems," in *VLSI (ISVLSI), 2012 IEEE Computer Society Annual Symposium on*.

[20] 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*, pp. 454–464, ACM, 2011.

[21] Y. Lee, J. Kim, H. Jang, and H. Yang, "A fully associative, tagless dram cache," in *ACM/IEEE International Symposium on Computer Architecture*, pp. 211–222, 2015.

[22] O. Mutlu, R. Ausavarungnirun, R. A. Harding, J. Meza, and H. Yoon, "Row buffer locality aware caching policies for hybrid memories," in *Proceedings of International Conference on Computer Design*, pp. 337–344, 2012.

[23] A. P. Ferreira, M. Zhou, S. Bock, B. Childers, and M. D. Melhem, R, "Increasing pcm main memory lifetime," in *DATE*, pp. 914–919, 2010.

[24] T. J. Ham, B. K. Chelepalli, N. Xue, and B. C. Lee, "Disintegrated control for energy-efficient and heterogeneous memory systems," in *International Symposium on High Performance Computer Architecture*, pp. 424–435, 2013.

[25] SAP, "SAP S/4HANA Cloud." http://www.sap.com/index.html.

[26] D. Chen, X. Ye, and J. Wang, "A multidimensional data model for tpc-ds benchmarking," in *Proceedings of the 5th Asia-Pacific Symposium on Internetware (Internetware'13), Article No. 2*.

[27] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zhang, G. Lu, K. Zhan, X. Li, and B. Qiu, "Bigdatabench: A big data benchmark suite from internet services," in *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on (HPCA 2014)*.

[28] M. Poremba, T. Zhang, and Y. Xie, "NVMain 2.0: a user-friendly memory simulator to model (non-)volatile memory systems," in *IEEE Computer Architecture Letters (Volume: 14, Issue: 2, July-Dec. 1 2015)*.

[29] D. Jevdjic, S. Volos, and B. Falsafi, "Die-stacked DRAM caches for servers: hit ratio, latency, or bandwidth? have it all with footprint cache," *ACM SIGARCH Computer Architecture News*, pp. 404–415, 2013.

[30] D. S. McGinnis, C. S. Huddleston, R. Agarwal, and M. R. Chinthamani, "Mechanisms and techniques for providing cache tags in dynamic random access memory," Dec. 17 2013. US Patent 8,612,832.

[31] Y. Bao, M. Chen, Y. Ruan, L. Liu, J. Fan, Q. Yuan, B. Song, and J. Xu, "HMTT: a platform independent full-system memory trace monitoring system," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 1, pp. 229–240, 2008.

[32] "DDR3 SDRAM." http://download.micron.com/pdf/datasheets/dram/ddr3/2Gb_DDR3_SDRAM.pdf, 2006. Micron Technology, Inc.