2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems

Characterizations and Architectural Implications of NVM's External DRAM Cache

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

***Abstract*—To meet the memory capacity requirement of big-data applications, a promising architecture is that NVM (Non-Volatile Memory) is used as main memory and external DRAM (off-chip) is used as cache. Compared to LLC versus DRAM (referred to as L-DRAM), external DRAM cache versus NVM main memory (referred to as E-NVM) are significantly different in terms of capacity and latency, thus traditional design cannot be directly borrowed without reconsideration. In this paper, we explore the characteristics and architectural implications of E-NVM in diverse dimensions including cache line granularity, associativity, replacement policy and prefetching methods. Lots of hints have been found for the optimal design of this emerging architecture.**

***Keywords*-Non-Volatile Memory; DRAM cache; Big-Data Application;**

I. Introduction

As applications are increasingly data intensive, in-memory computing becomes more important, which can store most data sets in memory to boost data access performance. For instance, SAP HANA [1] and Apache Spark [2] store entire data sets into main memory, so they require large memory capacity as much as the size of the processed data. However, the capacity of existing DRAM memory cannot meet this requirement, therefore the performance is severely limited.

Compared to DRAM, NVM is denser, thus provides larger capacity. As a widely used type of NVM, PCM is typically 4x denser than DRAM[3]. However, only using PCM as main memory is not enough to achieve satisfactory performance due to PCM's long access latency. There is an emerging hybrid architecture that combines both the advantages of DRAM and NVM, in which, NVM serves as main memory and DRAM plays the role of cache. Therefore, off-chip data access can be completed with short average access latency. In this paper, the structure "external DRAM cache with NVM" is referred to as **E-NVM**. We call on-chip cache last level cache with DRAM memory **L-DRAM**. Note that SRAM cache and die-stacked DRAM cache are both on-chip.

E-NVM is a promising architecture which has attracted attentions from both academic and industrial community. Earlier research showed that E-NVM improves 3X performance than DRAM memory due to 4X larger memory capacity [3]. Many commodity products have been released. For example, Intel has announced the Optane products which use external DRAM as L4 cache and NVM as main memory. According to Intel's newest reports [19], the average read latency of Optane memory is about 350ns. In which, the access latency of NVMM is about 10x latency of external DRAM cache.

E-NVM is significantly different from L-DRAM in at least two aspects: (1). The capacity of external DRAM cache is much larger than on-chip last level cache. (2). The access latency of external DRAM cache is longer than that of SRAM cache, and the latency of NVM is longer than that of DRAM. The following three aspects require a detailed characterization to bring architectural implications for the optimal design of the promising architecture.

First, a challenge introduced by the large capacity of external DRAM cache is the significant overhead of tags, thus the tags must be stored in external DRAM rather than on-chip. However, it would incur one additional tag access thus increases hit latency. A prior work proposed that direct-map cache can address this problem by placing data and tag adjacently, but it is impractical for set-associative cache [10]. On the contrary, set-associative cache improves hit rate because it reduces conflict misses. Trading off between the improvement on hit rate and the addition of tag latency, the associativity of cache needs a deep exploration.

Second, reducing the compulsory misses, enlarging cache line granularity is effective to improve hit rate. However, the hit latency will be longer along with larger cache line. Considering the trade-off between hit latency and hit rate, existing caches often use 64B as the granularity. However, the hit latency is longer because external DRAM is slower than SRAM thus it affects the selection of cache line granularity. It is uncertain that 64B is the most suitable granularity for larger and slower cache.

Third, the latency of main memory is an important design factor for the design of prefetching because it affects the prefetch lateness. In the design of prefetching, prefetch lateness is an important metric which represents how much prefetch requests are late than its demand. In the case of E-NVM, the late prefetch will be much more than the one

978-1-7281-2058-4/19/$31.00 ©2019 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2019.00064

in L-DRAM due to the longer latency of NVMM. Existing prefetch methods have the risk of losing their effects due to late prefetch thus need to be re-explored.

In this paper, we explore the architectural implications of E-NVM, including cache granularity, associativity, replacement policy and prefetching. We mainly answer the questions as following:

- Does cache granularity need to be reconsidered? Is the commonly used 64B still the most suitable granularity?
- Does set-associative cache still outperform the direct-mapped cache as that in L-NVM? If not, can techniques such as tag cache optimize the set-associative cache?
- Does the sophisticated replacement policies have obvious improvement on hit rate?
- Can the existing prefetching methods improve the performance?

To explore the above characteristics, the evaluation of E-NVM need appropriate benchmarks which can support large enough footprint. Unfortunately, previous ones are unsuitable due to their limited footprints. Most existing benchmarks have limited memory footprint no more than 16GB that can easily be covered by DRAM cache. In this paper, we propose a new benchmark suit called "Large Footprint Benchmark" to support large enough footprint for the evaluations. In this paper, we make the following contributions:

First, we build a suite of benchmark named "Large Footprint Benchmark". These workloads are typical big-data applications or commonly used high-performance programs. All of them access more than 100GB memory footprint.

Second, we find several characteristics and architectural implications: (1) As the result of the trade-off between hit rate and hit latency, external DRAM cache needs larger cache line such as 256B or 512B than on-chip cache. (2) Direct-mapped cache is better than set-associative cache with limited tag cache. (3) Existing replacement policies have little effect due to less conflict misses and different temporal locality. (4) Prefetching have large potential improvement due to the good spatial locality of big-data applications. All the findings support valuable hints for the designers of E-NVM.

Third, we analyze the address pattern of large footprint benchmark running over E-NVM. We find that the workloads running on E-NVM have good locality which can explain why larger cache line granularity better. It also explains why prefetching methods have effects. With larger amount of cache sets, reuse distance is quite small so that existing replacement policies have no obvious increase.

This paper is organized as follows. Section 2 presents the background about the detail of E-NVM. Section 3 shows the details about the large footprint benchmarks. Section 4 describes how we setup the evaluation. Section 5 shows our findings from the evaluation results. Section 6 shows the analysis of address pattern and explains the reason of the results. Section 7 reviews related work and Section 8 concludes.

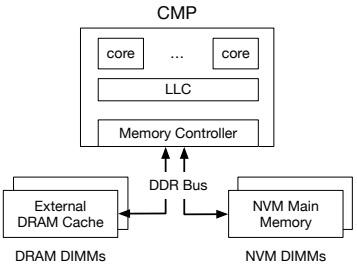

Figure 1. The Structure of E-NVM (External DRAM Cache of Non-Volatile Main Memory).

Table I
THE COMPARISION OF THREE TYPES OF MEMORY HIERACHY USING DIFFERENT CACHE AND MAIN MEMORY.

Type	Size	Latency	Capacity Ratio	Latency Ratio
Last Level Cache in CPU				
SRAM	16MB	20ns	4096	3
External DRAM	64GB	60ns		
Die-Stacked DRAM Cache				
Die-Stacked DRAM	128MB	60ns	512	1
External DRAM	64GB	60ns		
External DRAM Cache				
External DRAM	32GB	60ns	8	10
NVM	256GB	600ns		

II. BACKGROUNDS

As shown in Figure 1, in E-NVM, both NVM and DRAM are accessed via traditional memory buses such as DDR4, a subset of NVMM data are included in the external DRAM and managed by memory controller hardware. The upper-level cache can be either on-chip SRAM cache or die-stacked DRAM cache.

Table I shows the E-NVM's differences with traditional architectures. First, the capacity ratio (CR) between the main memory and cache is significantly different from the one in the traditional structures. As for on-chip cache, CR is 4096 or 512, but for E-NVM, the value turns to be 8. More specifically, the relative capacity of cache enlarges in several magnitudes. Secondly, the latency ratio (LR) changes from 3 or 1 to 10. Therefore, the cache miss penalty gets increase by nearly four or ten times. Thirdly, the cache hierarchy has filter effect that data with short reuse distance have been captured by upper-level caches, so the locality seen by external DRAM cache is different from that of the upper-level cache. Our findings in section V will show that these differences lead to different architectural implications and affect cache behavior.

III. Large Footprint Benchmarks

External DRAM cache mainly serves for big data applications which require large memory capacity. Therefore, the evaluations need to adopt the workloads which can represent big data applications with the large footprint. Many prior evaluations for on-chip cache usually use SPEC CPU2006 [4] and PARSEC [5] as benchmark suites. The footprint of these workloads can only reach gigabytes. Some evaluations for die-stacked DRAM cache use several workloads which have the larger footprints, such as CloudSuite [6]. However, the footprints of these workloads are rarely in the order of hundreds of GB. Therefore, these workloads cannot represent big data applications due to their limited footprints.

Considering large footprint, we select several typical workloads namely "Large Footprint Benchmarks" as shown in Table II. The workloads have large memory footprints. Most workloads have more than 100GB footprint. For the mixture of workloads in SPEC CPU2006 (SPEC-MIX), we change the footprint by adjusting the number of instances. The workloads cover various domains including social network, e-commerce, search engine and high-performance computing (HPC). Besides, we adopt dataset for benchmark seriously. For instance, for SAP HANA, we use TPC-DS to generate data and make requests[7]. The gene sequence data for soapdenovo2 is from open website[15]. Moreover, we adopt the data generated by BigdataBench [14] for Kmeans, Bayes, Terasort, and Pagerank.

Table II
Overview of Workloads

Workload	Memory Bandwidth (GB/s)	Footprint (GB)	Trace Size (GB)	Description
K-Means	1.18	95.2	425	Cluster
Bayes	0.30	206.8	96	Classification
specjbb	1.08	87	458	JVM Benchmark
pagerank	0.36	253.4	553	Search Engine
soapdenovo2	0.14	160.5	143	Oligonucleotide Analysis
SAP HANA	0.27	111.9	362	In-Memory Database
Terasort	0.61	253.2	235	Sort
Quicksort	0.64	247.9	195	Sort
HPCG	0.16	145.8	248	High performance computing
HPL	0.31	173.7	213	High performance computing
SPEC-MIX	1.50	52.3	1100	SPECCPU 2006

IV. Configuration

The default configurations are shown in Table III. For collecting traces, we run benchmarks on an Intel Xeon E5 Server with 256GB memory. The server has eight interleaved memory channels, and we collect trace using a hardware memory tracing tool named HMTT [8] to monitor the bus on one of the channels.

For evaluations of direct-mapped external DRAM cache, we organize the cache as Alloy cache [10]. In this case, tag and data are unified. When a hit occurs, data is accessed in a single burst without an additional access to tag. For set-associative cases, external DRAM cache is organized as Loh-Hill cache [9], in which tags of the same set are stored in the same row along with data for the same set.

Table III
Default configuration of evaluation

System Generating Traces Configurations	
Processor	Intel Xeon E5@2.10GHz 24-core
Operating System	CentOS 7, SLES 12.1(SAP HANA)
Memory Size	256GB
External DRAM cache configuration	
DRAM Cache	Size:1/8 Footprint 64-byte blocks 1 Chanel, 8 Banks, 4KB Set Size 64-bit bus width, 1333MHz RCD=9, RP=9, AL-BURST-RTP-CCD=0-4-3-2
NVMM configuration	
NVMM	Size: 256GB 64-byte blocks, 256-byte blocks 64-bit bus width, 800MHz RCD=90, RP=270, AL-BURST-RTP-CCD=0-4-3-2 (10 times slower than DRAM cache)

V. Findings

As described in section I, we mainly explore the architectural Implications of E-NVM in four aspects including cache line granularity, associativity, replacement policy and prefetching methods.

A. Granularity

In this section, we explore the effect of cache line granularity. Results are shown in Figure 2. Each sub-figure shows the result for one workload and each series represents the hit rate along with different cache granularities of cache with some fixed capacity.

Larger cache line can increase hit rate, but the increase will be less along with enlarging cache line. Figure 2 shows that hit rate benefits from the increase of cache line granularity in three aspects: (1). The tendency of increasing for hit rate is prominent at the beginning since larger cache line can significantly reduce compulsory misses. (2) When we continue to enlarge the cache line, the trend becomes smooth, and a number of workloads, such as SAP HANA and HPL, even begin to drop down because larger cache incurs conflict misses. (3). The inflection points represent the value of most suitable granularities and they are 256B or 512B. For SPECJbb, Pagerank and Terasort, the most suitable granularity is 512B and the one of SAP HANA, Soapdenovo2, and SPEC-MIX is 256B. For other workloads, such as Bayes and Kmeans, their hit rate stops increasing when the granularity is larger than 1KB.

Enlarging cache line may introduce additional latency although larger cache line can increase hit rate. Assuming

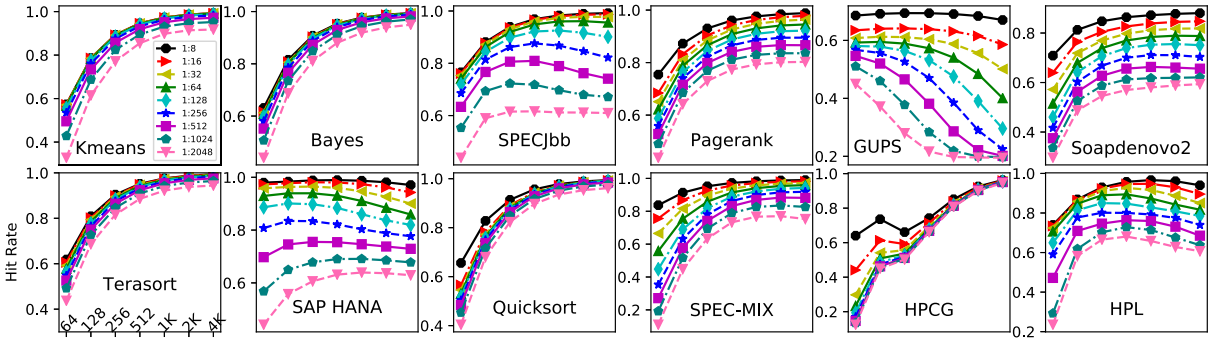

Figure 2. The hit rate of DRAM cache with different granularities and capacities.

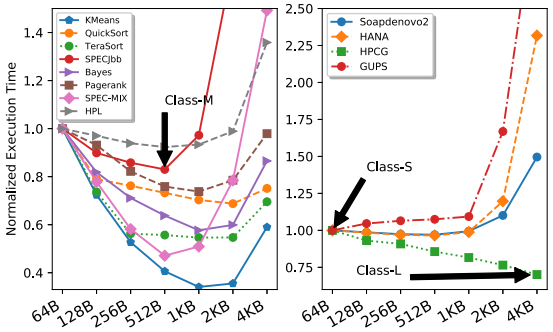

Figure 3. The normalized execution time for different cache lines.

fixed width of the data bus, hit latency is proportional to the transferred data size. For instance, using DDR3 memory bus with 64 bit, the memory controller needs four bursts to gain 256 bytes from DRAM. As this latency is equal to the latency of four memory reads, a trade-off exists between hit rate and hit latency and the trade-off will be the best cache line granularity. To explore this trade-off in external DRAM cache, we get performance using a simulator named NVMain. The results are shown in Figure 3. To show the result clearly, the lines are divided into two sub-figures. According to the effects of enlarging cache line granularity, the workloads can be split into three classes including "Class-S", "Class-M" and "Class-L".

The optimal granularity of "Class-M" is in middle. A phenomenon exists in all Class-M workloads: with smaller cache line granularity, enlarging cache line increases performance since the hit rate rise rapidly. However, as cache line size increases to a specific point, the hit rate rises slowly (even drops down), and the latency introduced by larger granularity become dominant. For SAP HANA, HPL, GUPS and Soapdenovo2, the benefits on increasing hit rate cannot make up the latency introduced by larger cache line. Therefore, the performance is worse as long as enlarging cache line. The optimal granularity is smaller, so they are called "Class-S". On the contrary, the third class is HPCG. Its benefit from enlarging cache line is continue so its performance is better as long as enlarging cache line. Its optimal granularity is larger so it is called "Class-L".

B. *Associativity*

In this subsection, we explore the impact of associativity by changing the associativity and capacity. To isolate the effect of associativity, we use optimal replacement policy in this section. Each sub-figure shows the result of the same workloads. Each series represents the percent improvement over the hit rate of a direct-mapped cache with the same cache capacity. The results are shown in Figure 4.

When the cache capacity is smaller, moderate associativity can significantly increase hit rate for most workloads. Only three workloads are not sensitive to associativity including HPCG, HPL and SAP HANA. Their percent improvement on hit rate are about 6%-8%. Apart from these three workloads, the percent improvement of other workloads are about 20%-30%. Set-associative cache has effect for smaller cache due to the larger number of conflict misses.

However, along with enlarging cache capacity, the effect of associativity is smaller. With less conflict misses in a smaller cache, set-associative cache has no effect on increase hit rate. As shown in Figure 4, even for the kind of workloads which are sensitive to cache associativity, these percent improvements on hit rate reduce to less than 5%.

Apart from the advantages on reducing conflict misses, set-associative cache has disadvantage due to the additional tag access. As described in [10], direct-mapped cache can avoid the access to cache tag by placing data and tag adjacently but this off-chip access is inevitable for set-associative cache. Therefore, the hit latency of set-associative cache is higher that direct-map cache.

To decrease the hit latency of set-associative cache, tag cache [11] is an efficient method. As described in [11], one tag cache hit can estimate one DRAM tag access by caching part of tags in SRAM. To explore the effect of tag cache, we re-evaluate the method in ATCache [11] and obtain the

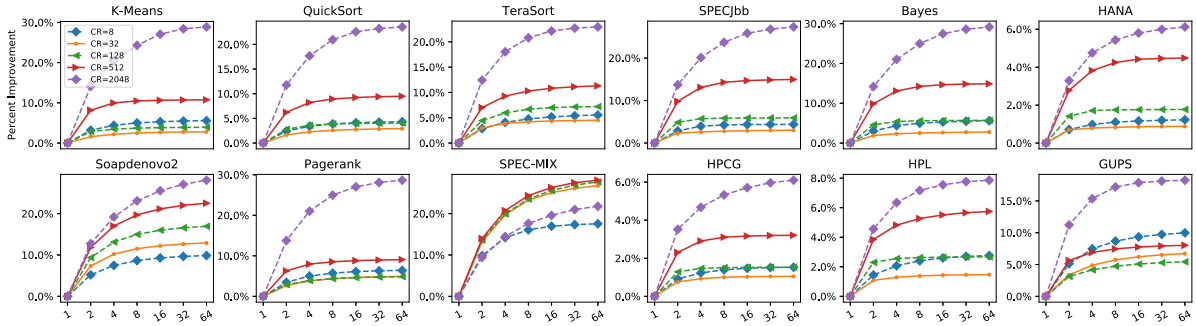

Figure 4. The hit rate in different associativity over different CRs. Each sub-figure shows the result of some workload. Each line in the same sub-figure represents the hit rate along with different associativity in some fixed capacity. CR is the ratio between the size of footprint and cache capacity.

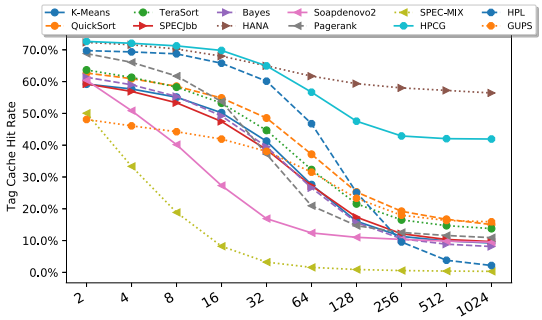

Figure 5. Tag cache hit rates along with the ratio between the capacity of cache and the one of tag cache. For instance, 64 represents that the capacity of cache is 64x tag cache.

tag cache hit rate. In this evaluation, we use 32-way set-associative cache with 64B cache line granularities. Figure 5 shows the results.

As illustrated in Figure 5, tag cache can reduce hit latency efficiently, and this decreasing depends on the size of tag cache. For HPCG and SAP HANA, the tag cache hit rates have little increase even enlarging the capacity of tag cache. For most workloads, the capacity of tag cache need to be increased to at least 1/8 of capacity of DRAM cache, the tag cache has obvious effect on reduce the number of tag accesses.

However, tag cache is commonly equipped with insufficient on chip SRAM space. In our configuration, the capacity of external DRAM cache is 32GB, and the associativity is 32. If cache line granularity is 64B, the smallest tag storage is 448MB. According to the results in Figure 5, 56MB SRAM can estimate 40% tag access and 448KB SRAM can eliminate 19% tag access. For the commercial systems, the typical total capacity of L3 Cache reaches 20MB, thus little capacity remains for tag cache. Tag cache has fairly effect given enough tag storage in external DRAM cache, more specifically, it will be limited by the capacity of SRAM. Therefore, if we want tag cache play a role, some optimal technologies should be used to improving the utilization of tag cache.

C. Replacement Policies

To explore the effect of replacement policies, we evaluate sophisticated replacement policies including Random, FIFO, LRU and SRRIP. Besides, Belady's MIN algorithm has been proved optimal replacement policy in reducing cache miss. We call it OPT to represent the best replacement policy. We config the cache with 8 ways associativity. The results are shown in Figure 6. Each sub-figure shows the result of the same workloads. Each series represents the percent improvement over the hit rate of Random with the same cache capacity.

The workloads are divided into two categories. The one is friendly to replacement policies including K-Means, QuickSort, TeraSort, SPECJbb, Bayes, GUPS, HPCG, HPL, SAP HANA. Especially when CR value is 2048, the sophisticated replacement policies can obviously increase hit rate comparing to Random. The other one is not friendly to replacement policies including Soapdenovo2, Pagerank, SPEC-MIX. Besides, OPT can increase hit rate for all workloads.

However, when CR values is smaller which means cache is larger, the replacement policies have little effect on increasing hit rate, even for the replacement policies friendly workloads. Replacement policy mainly serves for reducing conflict misses. As a larger cache has more cache sets, lots of conflict misses are estimated. Therefore, we conclude that, in external DRAM cache, existing sophisticated replacement policies cannot have so much effect as that in on-chip cache.

D. Prefetching

Data prefetching is a useful optimization technique by issuing memory requests and loading data into the cache before demands. To explore the effects of prefetching, we evaluate four typical existing prefetch algorithms including NextLine, Stream, AMPM and SPP. The baseline is non-prefetching.

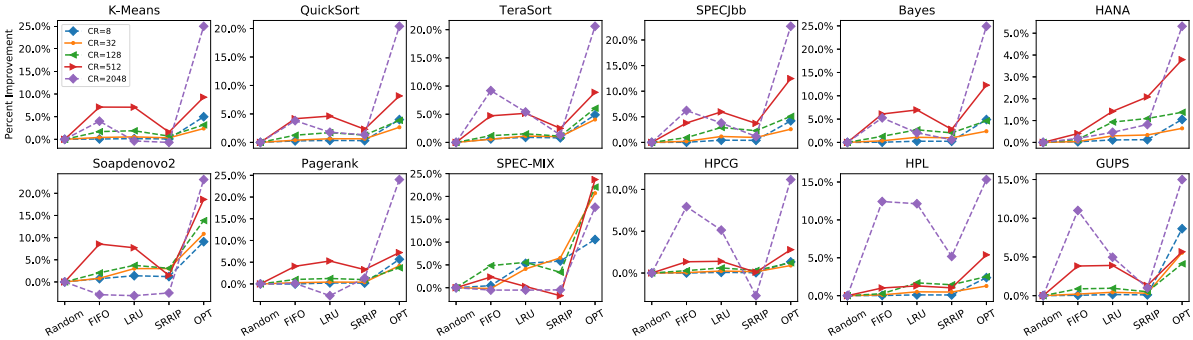

Figure 6. The hit rate of cache using different replacement policies over different CRs. Each sub-figure shows the result of some workload. Each line in the same sub-figure represents the hit rate along with different replacement policies in some fixed capacity. CR is the ratio between the size of footprint and cache capacity.

NextLine is the simplest algorithm. Once an access arrives, the prefetcher will issue the prefetch request for next address. Stream Buffer is also a classical and commonly used prefetcher. It tracks the access of each memory region and detects the stream within them. If some stream exists in some region, the prefetch will be triggered. AMPM uses a bitmap to track the access in some fixed area. When some pattern of area occurs again, it will issue prefetch requests according to the pattern learned from the history. SPP is the typical look-ahead prefetcher and the state-of-the- art prefetcher according to the results of DPC2. It tracks the signature for each memory page and accounts the count for each signature. When the count of some signature is over some threshold, the prefetch will be triggered.

Figure 7 shows the results of prefetching. Apart from NextLine, the rest prefetcher can significantly improve whole performance. On average, SPP has highest speedup and it can improve the performance by 16.3%. Stream prefetcher improves performance by 16%, while AMPM bring in 12.8% improvement. Most workloads are prefethingfriendly. For instance, SPP can improve HPCG by 69% and improve Pagerank by 33.3%. Other workloads including SPECJbb, SAP HANA, Soapdenovo2 and GUPS are prefetching-unfriendly. Besides, some prefetching-friendly workloads are sensitive to prefetching method such as HPCG and HPL. For instance, AMPM only has 13.3% improvement for HPCG and SPP has 69% improvement. For these workloads, the prefetching algorithm need be selected carefully.

VI. Analysis

We analyze the memory access pattern in terms of the following two aspects: temporal and spatial locality.

A. Temporal Locality

For the temporal locality, we explore the reuse distance, which is a commonly used analytical method for cache studies. We calculate the reuse distance using an 8-way associated cache with different capacities. For L-DRAM, the

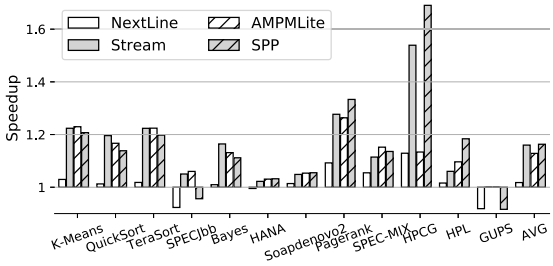

Figure 7. The result of overall performance of prefetching.

CR value is 2048 and for E-NVM, the CR value is 8. Figure 8 shows the distribution of reuse distance with different CR values. As illustrated in this comparison, two aspects should be noted.

First, all reuse distances are decreased due to the larger amount of cache set. It means that external DRAM cache has less access pressure. For example, a series of access is [A, B, C, A, B, D, A, C]. If the number of cache set is 1, the reuse distance distribution will be [rd_3=3, rd_4=1]. If we enlarge the cache capacity, an additional cache set will be added and the access will be split into two sets: [A, B, A, B, A] and [C, D, C]. Then the distribution will turn to be [rd_2=4]. For instance, the 6.4% HPL accesses' reuse distance is 1 when the cache is small. But after enlarging cache capacity, 98.9% accesses' reuse distance is 1.

Another aspect is that when CR is 8, mostly reuse distance value is 1, which means that these memory addresses are accessed continually after filtered by last level cache. Our experiments about replacement policy shows that many enhanced replacement policy have no effects for such accesses whose reuse distance is 1.

B. Spatial Locality

For the spatial locality, we detect the local stride. We define the local stride is the difference between consecutive

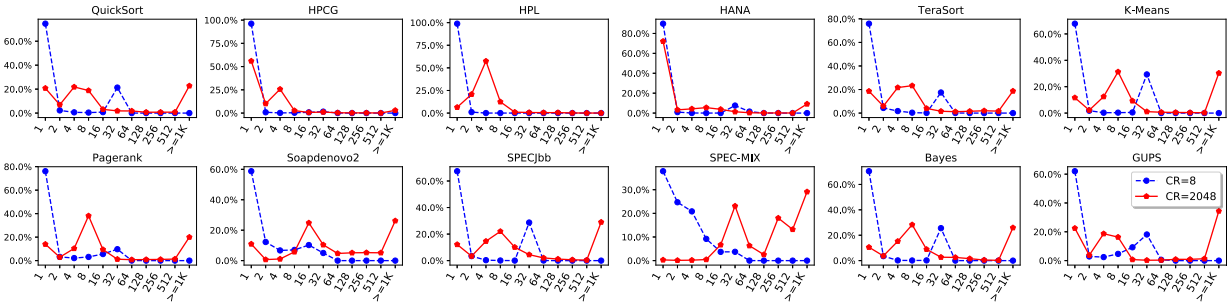

Figure 8. The distribution of reuse distance with different capacity ratios(CR).

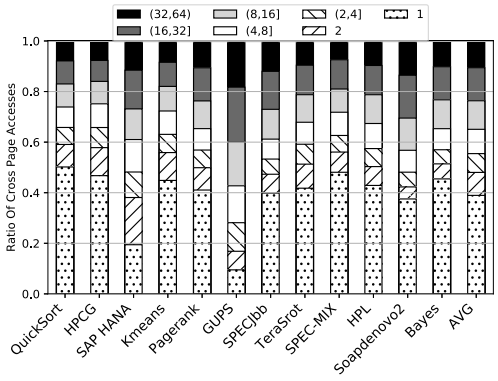

Figure 9. The distribution of stride inner-page. 1 represents that the difference between consecutive effective memory addresses is one block and the size of block is 64B.

effective memory addresses localized per memory page. Figure 9 shows the ratio distribution of strides.

First, about 40% stride value is 1 which means 40% access address of the same page are continuous. It represents these workloads has quite good spatial locality. About 65% stride value is less than 8. The distribution of strides can explain the results in Figure 3 and Figure 2. Enlarging the cache line can fully utilize the good spatial locality.

VII. Related Work

Most existing work focus on DRAM cache that is die-stacked [12], [11], [10], [13], [18]. Both capacity ratio and latency ratio between main memory and cache are significantly different from prior ones. These differences are the fundamentals of our evaluation method, findings.

For the evaluation methodology, prior work mainly uses workloads from SPEC CPU2006, PARSEC whose footprints are relatively small for the evaluations of external DRAM cache. Other works try to evaluate using large-scale workloads. For instance, the evaluation in Unison cache [12] uses CloudSuite, and the largest footprint is 3.7GB. Footprint cache uses workloads with reduced data sets (5-20GB). However, considering the diversities of application fields we need select more representative big data applications from various application fields. Therefore, compared with the above benchmark suits, ours is more suitable for the evaluations of external DRAM cache.

For the cache line granularity, prior works have explored the impact of various cache line granularities. For instance, Amoeba cache [13] shows that for SRAM last level cache, 64B is the optimal granularity. Unison cache [12] indicates that for die-stacked DRAM cache, workloads still exhibit significant spatial locality, so it chooses 4KB as its granularity. However, these works assume the last level cache is die-stacked DRAM cache and we have shown that capacity ratio and latency ratio between main memory and cache are significantly different when last level cache is off-chip and main memory is NVMM. Therefore, the choice about cache line granularity need to be reconsidered.

For the set-associativity, prior work have compared direct-mapped cache and set-associative cache. For instance, the work in MemSys [18] shows that the effectiveness of associativity still exists for die-stacked DRAM cache. And the Table 6 in work of Alloy cache [10] shows that set-associative cache can increase hit rate and the improvement on hit rate will become less along with enlarging cache capacity. Our results resemble the results in Alloy cache. For tag cache, ATCache [11] saves over 60% of DRAM cache tag access on average by consuming 0.4% of overall tag size. But in our paper, the DRAM cache we use is off-chip and has larger tag size. So for the same size of tag cache, our results represent that effect of tag cache has less effect because of the limited size of SRAM.

For the replacement policy, at present, multiple choices are available. A large number of algorithms have been used for a long time, for instance, LRU, FIFO, and MRU have been commonly used in real system. Besides, a prior research [17] propose RRIP and improves hit rate by about 10% versus LRU. Our results show that these sophisticated replacement policies do not have so much effectiveness in external DRAM cache.

For the evaluation of prefetching for big-data applications, a prior research [20] discussed the effect of prefetching

for cloud workloads. It discovered that prefething for cloud workloads with a large cache could improve performance. Our results of prefetching resemble this results and we analyze the reason is the good spatial locality.

VIII. Conclusions

In this paper, to evaluate E-NVM, we build a benchmark suite each of which has a large footprint. Using the benchmark, we find some architectural implications of E-NVM. First, we find that the suitable cache granularity needs to be re-considered due to larger capacity of external DRAM cache and good spatial locality of big-data applications. Second, we find that direct-mapped cache is better than set-associative cache even tag cache with limited capacity is used. Third, we find that existing replacement policy have little effect due to the different temporal locality. Forth, we find that prefetching has large potential improvement due to the good spatial locality. All these finding can be used as valuable hints for efficiently architecting and utilizing NVM related structures.

Acknowledgment

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

References

[1] Färber, Franz, et al. “SAP HANA database: data management for modern business applications.” ACM Sigmod Record 40.4 (2012): 45-51.

[2] Zaharia, Matei, et al. “Apache spark: a unified engine for big data processing.” Communications of the ACM 59.11 (2016): 56-65.

[3] Qureshi, Moinuddin K., Vijayalakshmi Srinivasan, and Jude A. Rivers. “Scalable high performance main memory system using phase-change memory technology.” ACM SIGARCH Computer Architecture News. Vol. 37. No. 3. ACM, 2009. https://soundcloud.com/intelchipchat/intel-optane-technology

[4] Henning, John L. “SPEC CPU2006 benchmark descriptions.” ACM SIGARCH Computer Architecture News 34.4 (2006): 1-17.

[5] Bienia, Christian, et al. “The PARSEC benchmark suite: Characterization and architectural implications.” Proceedings of the 17th international conference on Parallel architectures and compilation techniques. ACM, 2008.

[6] Ferdman, Michael, et al. “Clearing the clouds: a study of emerging scale-out workloads on modern hardware.” ACM SIGPLAN Notices. Vol. 47. No. 4. ACM, 2012.

[7] Nambiar, Raghunath Othayoth, and Meikel Poess. “The making of TPC-DS.” Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, 2006.

[8] Bao Y, Chen M, Ruan Y, et al. HMTT: a platform independent full-system memory trace monitoring system[J]. ACM SIGMETRICS Performance Evaluation Review, 2008, 36(1): 229-240.

[9] 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. ACM, 2011, pp. 454–464.

[10] 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. IEEE Computer Society, 2012, pp. 235–246.

[11] Huang, Cheng-Chieh, and Vijay Nagarajan. “ATCache: reducing DRAM cache latency via a small SRAM tag cache.” Proceedings of the 23rd international conference on Parallel architectures and compilation. ACM, 2014.

[12] Jevdjic, Djordje, et al. “Unison cache: A scalable and effective die-stacked DRAM cache.” Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, 2014.

[13] Kumar, Snehasish, et al.“Amoeba-cache: Adaptive blocks for eliminating waste in the memory hierarchy.” Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, 2012.

[14] Wang, Lei, et al. “Bigdatabench: A big data benchmark suite from internet services.” 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2014.

[15] Gene Data. http://www.bio8.cs.hku.hk/dataset.

[16] M. Poremba, T. Zhang, and Y. Xie, “Nvmain 2.0: A user-friendly memory simulator to model (non-) volatile memory systems,” IEEE Computer Architecture Letters, vol. 14, no. 2, pp. 140–143, 2015.

[17] A. Jaleel, K. B. Theobald, S. C. Steely Jr, and J. Emer, “High performance cache replacement using re-reference interval prediction (rrip),” in ACM SIGARCH Computer Architecture News, vol. 38, no. 3. ACM, 2010, pp. 60–71.

[18] P. Tschirhart, J. Stevens, Z. Chishti, and B. Jacob, “The case for associative dram caches,” in Proceedings of the Second International Symposium on Memory Systems. ACM, 2016, pp. 211–219.

[19] “Intel’s Optane DC Persistent Memory DIMMs Push Latency Closer to DRAM”, 2018 https://www.pcper.com/news/Storage/Intels-Optane-DC-Persistent-Memory-DIMMs-Push-Latency-Closer-DRAM

[20] Wang, Jiajun, Reena Panda, and Lizy Kurian John. “Prefetching for cloud workloads: An analysis based on address patterns.” 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 2017.