A Hybrid Non-Volatile Cache Design for Solid-State Drives Using Comprehensive I/O Characterization

Mojtaba Tarihi, Hossein Asadi, Alireza Haghdoost, Mohammad Arjomand, and Hamid Sarbazi-Azad

***Abstract*—The emergence of new memory technologies provides us with opportunity to enhance the properties of existing memory architectures. One such technology is *Phase Change Memory* (PCM) which boasts superior scalability, power savings, non-volatility, and a performance competitive to *Dynamic Random Access Memory* (DRAM). In this paper, we propose a write buffer architecture for *Solid-State Drives* (SSDs) which attempts to exploit PCM as a DRAM alternative while alleviating its issues such as long write latency, high write energy, and finite endurance. To this end and based on thorough I/O characterization of desktop and enterprise applications, we propose a hybrid DRAM-PCM SSD cache design with an intelligent data movement scheme. This architecture manages to improve energy efficiency while enhancing performance and endurance. To study the design trade-offs between energy, performance, and endurance, we augmented Microsoft's DiskSim SSD model with a detailed hybrid cache using PCM and DRAM parameters from a rigorous survey of device prototypes. We study the design choices of implementing different PCM and DRAM arrays to achieve the best trade-off between energy and performance. The results display up to 77% power savings compared to a DRAM cache and up to 26% reduction in request response time for a variety of workloads, while greatly improving disk endurance.**

***Keywords*—SSD cache, Hybrid memory, Phase change RAM.**

I. Introduction

OVER the past decade, NAND flash-based *Solid-State Drives* (SSDs) have shown promising ability in addressing the technical issues of legacy HDD based storage systems such as high energy consumption, low reliability, and poor performance on random workloads. However, fully exploiting the performance, energy, and reliability advantages of this technology requires overcoming the challenging issues posed by its underlying technology; the flash technology has finite endurance and its access and erase operations function at mismatched granularities and have very different latencies. As a result, SSD controllers implement the *Flash Translation Layer* (FTL) as a firmware which hides the internal implementation details and presents the drive as a block device to the host computer. As erase operations are slow and energy intensive, FTLs use log-based approaches [1], [2] or other mapping schemes [3], [4], [5] to reduce the number of erase operations and perform wear-leveling which tries to even the wear-out of flash memory cells.

An SSD controller resides between the host interface and flash memories and implements the required functionalities such as FTL. The FTL functionalities can be implemented as software [6] or be synthesized from *Hardware Description Language* (HDL) code as hardware [7]. Most SSDs also come with on-board *Dynamic Random Access Memory* (DRAM) to store cached disk data and FTL data structures, which can significantly reduce the amount of writes committed to flash if data exhibits locality in write operations. A cache can also greatly reduce response time for write operations since acknowledgment can be sent as soon as the data is written to the cache memory, while the slow writes to flash memories will be performed in the background. Due to the long latencies of write and erase operations, cache memory can be highly effective in improving SSD performance. Thus, in order to improve or at least maintain the cache hit ratio with the increased SSD capacity, designers have aggressively increased the cache size in recent prototypes [8], [9], [10].

The empirical evaluations of modern SSDs under desktop applications show that SSDs' front-end logic is idle in most of run time [11]. Hence, the standby power of this logic, leakage and refresh power of DRAM array are the key contributors to the overall power of SSD such that using larger DRAM caches significantly increases overall power consumption due to increased leakage, refresh power, and per-operation energy consumption. The impact of this effect will be especially evident when storage arrays at large-scale installations composed of thousands of disks are considered. Furthermore, larger DRAM caches are increasingly constrained by data integrity issues. Indeed, uncommitted data furthers the dangers of power loss, requiring the storage system or SSDs to include backup batteries or capacitors to commit changes to flash. Worse yet, it has been demonstrated that even some of these enterprise SSDs lose data upon power outage in the presence of battery backup [12].

To address the limitations of DRAM, emerging non-volatile memory technologies such as *Phase Change Memory* (PCM) have been suggested as a candidate to replace DRAM in the main memory [13], [14], [15]. PCM is one of the most promising non-volatile memories, which instead of using electrical charges to store information, stores bit values with the physical state of a chalcogenide material (e.g., $Ge_2Sb_2Te_5$, i.e. GST). Under different current amplitudes and durations, the material can be formed into either an amorphous or crystalline state with completely different levels of resistivity. Due to non-volatility, high density, and fast read access of PCM, the last few years have seen a large body of research attempting to evaluate the potential of PCM as a DRAM alternative for main memory [13], [14], [15]. However, to the best of our knowledge, none of these studies have tried to use PCM in the cache memory of SSDs.

In this work, we introduce and investigate a novel hybrid SSD cache architecture composed of DRAM and PCM memory. The motivation for such an architecture originates from key trade-offs of using these technologies together. While PCM read latency and energy is fairly competitive to DRAM, PCM write operations are more costly in terms of both latency and energy. Also, the durability of PCM cells, while better than NAND flash cells, is

still multiple orders of magnitude lower than that of DRAM. On the other hand, PCM has a very low leakage energy and greatly improves reliability since it does not suffer from data loss upon losing power and occurrence of soft errors initiated from particles strike. Hence, PCM can improve energy efficiency and reliability if its drawbacks are carefully addressed.

The proposed architecture attempts to make use of DRAM for active data to reduce the costly PCM write operations, while storing most of data blocks in PCM, thus, not only improving energy efficiency and reliability, but potentially improving performance as greater capacities can be supported under the same power envelope. A naive idea to combine these two memories is to use a two-level hierarchy, however, our comprehensive characterization and evaluation of *Input/Output* (I/O) workloads demonstrates that such an architecture fails to combine the advantages of these technologies. As an alternative, we propose a flat design where both technologies are placed at the same level, organized with an intelligent data movement scheme which takes the properties of workloads and these memories into account.

Using DiskSim 4.0 [16] with Microsoft's SSD model [5] and a detailed model of PCM and DRAM, we also study the design choices of the proposed architecture such as memory size and algorithm parameters to achieve the best trade-off between energy, performance, and endurance. The experimental results collectively show that utilizing the proposed hybrid cache architecture and corresponding design considerations display up to 77% SSD power savings compared to a DRAM-based cache design with comparable performance, up to 26% improvement in SSD response time and up to 150% better SSD endurance for a variety of benchmarks. The proposed architecture also significantly reduces the chance of data loss as compared to the conventional DRAM-based cache architecture.

The rest of the paper is organized as follows. Background information about SSD internal architecture as well as NAND flash and PCM technologies are given in Sec. II. In Sec. III, we discuss different cache structures and their shortcomings. In Sec. IV, we present a comprehensive analysis of I/O patterns to motivate the proposed architecture. Sec. V presents our SSD cache architecture and discusses its design considerations. Sec. VI reports the simulation results. Sec. VII reviews the main contribution of the prior studies, and concluding remarks are given in Sec. VIII.

II. Background

In this section, we first overview the organization and functionalities of typical SSD designs. Then, technical characteristics of NAND flash and PCM memory technologies are covered.

A. SSD Organization and Functionalities

Fig. 1 illustrates a structural view of an SSD that uses flash memory to retain data. This figure also shows the internals of a flash chip consisting of multiple dies, themselves composed of multiple planes containing memory pages. Each die can run commands independently but the bus is shared which increases operation delay. Much more stringent constraints are in place for inter-plane parallelism. While the minimum granularity of plane read/write operations is a page, erase operates at a block

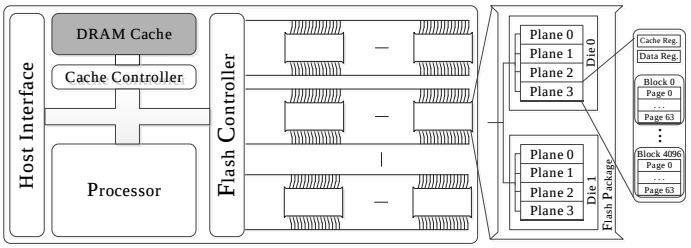


Fig. 1: Structure of an SSD and a NAND flash organization.

granularity which consists of many pages. Erase operations also cause flash cells to wear out, as a result, not only the controller must perform wear-leveling to maximize the SSD life, it must also employ *Error Correction Codes* (ECC) to mask the effects of transient errors while the cells are still usable.

SSD controllers run FTL to emulate a conventional block device. FTL is responsible for a number of key functionalities: (a) address mapping for translating logical addresses to physical addresses, (b) garbage collection for reclaiming invalid pages, (c) wear-leveling to increase the overall life of the SSD, and (d) protecting user data by means of ECC.

An SSD usually consists of the following components: (a) a host interface such as SATA, SAS, or PCIe with the associated buffering, (b) an embedded controller to implement various internal functionalities, (c) an on-board memory for storing FTL data structures and caching user data, (d) high bandwidth buses to distribute access loads across flash memories, (e) flash memory chips for data storage, and (f) battery or capacitors to safely shutdown the SSD on power loss. Table I reports key characteristics of recent SSDs with emphasis on cache size (a central matter in this paper).

B. Phase Change Memory (PCM)

PCM, which is byte-addressable and non-volatile, can be integrated with CMOS fabrication process and is projected to scale well beyond the technology limits of both DRAM and flash, making PCM a potential replacement for both these technologies. Table II presents a comparative analysis of different NAND flash, DRAM, and PCM technologies. Compared to flash memory, PCM has a lower read/write latency while its cell endurance is 3 to 4 orders of magnitude higher. This table also shows that PCM presents comparable read latency. However, PCM write accesses are more costly, preventing it from being a perfect alternative.

Nearly all PCM cell prototypes store binary data into GST as either high-resistance amorphous (RESET) or low-resistance crystalline (SET) states. This is known as *Single-Level Cell* (SLC) PCM and an example is depicted in Fig. 2. To read an SLC cell, a low current is passed through the material for a short period (about 30ns-40ns) to sense resistivity. In contrast to read, the writing process depends on the previous values as a choice must be made between sending a SET or RESET pulse, which greatly differs in amplitude and duration. More specifically, the low amplitude SET pulse has to be long enough (typically 120ns) to melt a significant portion of the chalcogenide material. On the other hand, RESET pulse has to inject a large current into GST for a short time (typically 90ns) to reach at

TABLE I: Performance Characteristics of Contemporary SSDs

Disk Drive	IOmeter 2MB Sequential[a]		IOmeter 4kB Random		Controller	On-board DRAM[b]	Flash Memory
	RD[MB/s]	WR[MB/s]	RD[IOPS]	WR[IOPS]			
Corsair Neutron GTX 240GB [17]	465	466	6,982	17,195	Link A MD	256MB	32GB Toshiba 24nm
Crucial M500 960GB [18]	482	426	6,567	18,098	Marvell	256MB	64GB Micron MLC 20nm
Intel 520 240GB [19]	486	290	7,298	19,682	Sandforce	No	16GB Intel MLC 25nm
OCZ Vertex 450 256GB [8]	484	480	5,587	17,865	OCZ BF3	512MB	16GB Micron MLC 20nm
Plextor M5S 256GB [9]	491	380	7,208	16,011	Marvell	512MB	16GB Micron MLC 25nm
Samsung 840 Evo 500GB [10]	493	483	10,081	22,772	Samsung	512MB	128GB Samsung TLC 19nm
Seagate 600 240GB [20]	495	442	6,943	17,180	Link A MD	256MB	32GB Toshiba MLC 19nm

[a] For Crucial M500 960GB and Seagate 600 Pro SSDs, the sequential test is performed for 128KB data blocks.
[b] Note that some controllers take advantage of DRAM as a write buffer [21], [22], while some controllers do not [23]. A large amount of DRAM makes it more probable that it is being used as a data cache.

TABLE II: Characteristics of Memory Technologies (DRAM, Flash, and PCM)

Technology	Latency			Energy[pJ/bit]			Endurance
	RD[a]	WR (0/1)	ER	RD	WR (0/1)	ER[b]	
DRAM [15], [24]	10-20.04ns	10-20.04ns	No	29.5	35.2	No	10^{15}
Flash [25], [26], [27], [28], [24]	2.5-200μs	25-1300μs	1.5-3.8ms	2-190	410-1090	5.7-93	3×10^3-1.1×10^6
PCM [15], [14], [29]	12-70ns	10-100ns / 60-400ns	No	2.0-20.9	4.8-64.8 / 2.8-45	No	10^4-10^9

[a] RD: Read, WR: Write, and ER: Erase.
[b] While the per-bit energy of erase is smaller than write, its granularity is a block composed of many pages.

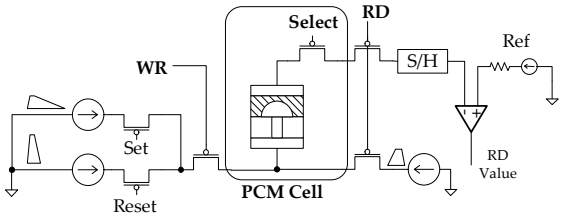


Fig. 2: SLC PCM cell and its read/write access circuits.

least its crystallization point (much lower than the melting point). Compared to the SET operation, RESET is fast, consumes more energy, and significantly contributes to PCM wear-out.

PCM scaling to below 10nm has already been demonstrated [30] and it is projected to well scale to smaller geometries using *Multi-Level Cell* (MLC) cells. In MLC PCM, the huge resistivity window of GST is broken into multiple non-overlapping resistivity ranges, each representing more than a single bit. MLC read and write require accurate mechanisms which are practically achievable using repetitive sensing and programming steps, respectively. Such accuracy comes at the cost of higher access latency and energy and lower cell endurance.

In this study, we did not consider other competitive *Non-Volatile Memory* (NVM) technologies such as STT-RAM and ReRAM due to the following reasons. First, although STT-RAM memory technology presents better write endurance and performance, it needs larger write current and hence higher write power than PCM. Second, our experiments demonstrate that the greatest challenge in integrating NVM in storage cache is managing endurance. This process forces us to direct less writes to NVM and as such, performance is greatly affected; having to manage NVM endurance seriously diminishes the performance gains. We will also demonstrate that write energy is far less relevant than leakage power when it comes to power consumption. Overall, we believe that the choice of PCM is a much better representative of challenges and opportunities presented by hybrid storage caches. Also, available ReRAM arrays are prototyped in small sizes [31], [32], [33], [34], [35], [36] and it is highly possible that its specifications (latency/energy) will show different behavior in large-scaled arrays.

III. Candidate SSD Architectures

In this section, we explore four candidate architectures for SSD cache memory using DRAM and PCM technologies. As shown in Fig. 3, two candidates purely use DRAM or PCM, and the other two structures use a DRAM-PCM hybrid cache in a flat or two-level layout. A pure DRAM-based cache, as can be seen in Table I, is widely used in current SSDs. Unlike PCM, there is no concern for DRAM endurance over device life cycle. Moreover, read and write access latencies of DRAM are much lower than flash memory and can significantly enhance the overall performance of flash-based SSDs. DRAM, however, consumes considerable power and increasing its size increases power consumption further and reduces reliability. While the risk of even more data loss in case of power outage can be alleviated by using backup battery or capacitors, data loss due to power outage and particles strike is still a major concern [12]. This study considers a pure DRAM SSD cache as the reference architecture throughout these evaluations.

Pure PCM completely replaces DRAM (Fig. 3b). While it has better density and lower leakage, this cache architecture suffers from two major shortcomings. First, it will have a lower performance compared to the pure DRAM cache due to costly writes. Second, fast wear-out makes it challenging to guarantee endurance for the majority of workloads. Use of different memory technologies to make appropriate cache design

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2455978, IEEE Transactions on Computers

trade-offs, especially to lower power consumption, has been already investigated in [37], [38], [39], [40], [41], [42]. A two or, multi-level cache design is the first choice that comes to mind for combining different memory technologies. As shown in Fig. 3c, this arrangement places recently accessed data at DRAM which resides at the lower level and as data pages become inactive, they are retired to PCM and eventually, to flash. PCM allows increasing cache capacity and hit ratio while keeping leakage low.

Although the two-level DRAM-PCM cache appears to have appropriate design trade-offs among power, performance, and reliability, a closer look at the dataflow between the two cache levels in response to write operations calls this choice into question. To further elaborate, the idea of using PCM alongside DRAM is centered around using a larger, less active PCM memory as it has lower leakage and higher density, and using DRAM to reduce access time and reduce PCM wear. In other words, DRAM space is scarce and precious and must be used for frequently accessed data. Such a two-level cache design will actively pollute the DRAM space regardless of the eviction policy and will introduce unnecessary traffic into the PCM level. Thus, it can be expected that a two-level cache is not able to properly address the endurance concerns associated with using DRAM and PCM memories together. The I/O characterization study presented in the next section will further validate this point. Also, as two-level cache design is extensively used in hybrid cache and hybrid main memory design, later in the evaluation section, we will compare results of the proposed cache architecture with those of two-level hybrid cache.

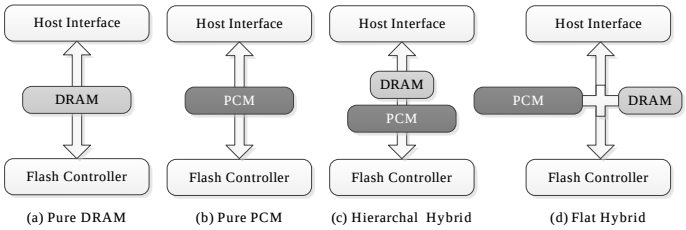


Fig. 3: Candidate SSD cache architectures with separate DRAM and PCM arrays.

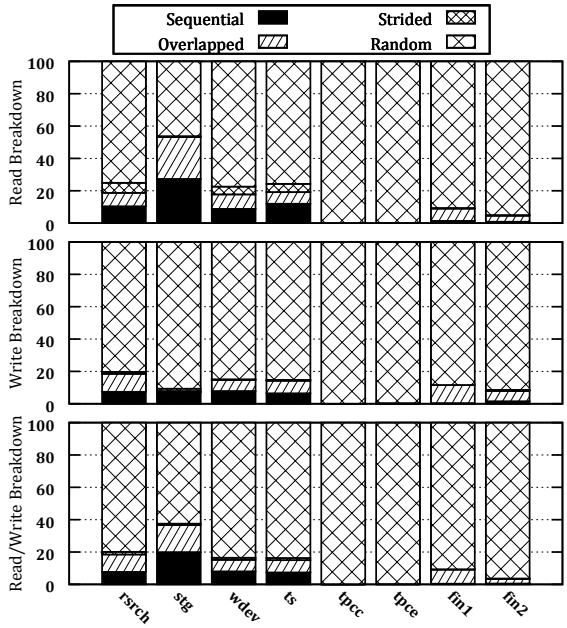


Fig. 4: Breakdown of I/O requests in the MSRC, MSE, and financial workloads for a maximum stride length of 2 and ROD equal to 1.

IV. I/O Characterization

To design an efficient cache architecture for SSDs, it is imperative to investigate I/O workloads behavior. This study, known as I/O characterization, allows designers to optimize the cache architecture with respect to desired parameters such as performance, power, and endurance. A number of studies investigating I/O workloads behavior [43], [44], [45] have been performed. We characterize workloads with great focus on characteristics relevant to cache design such as spatial and temporal locality.

Table III gives detailed characteristics of the workloads under study. The *rsrch*, *stg*, *ts*, and *wdev* traces have been obtained from Microsoft Research Cambridge (MSRC) systems [46], *tpcc* and *tpce* have been obtained from Microsoft systems running the two well-known benchmarks of the same name [47], which will be referred to as the MSE traces. Finally, the Financial traces, *fin1* and *fin2*, were collected at two large financial institutions [48]. In the following subsections, two important characteristics of I/O workloads (i.e., spatial and temporal locality) are examined.

A. Spatial Locality

Spatial locality is the probability of subsequent data accesses being located in the neighborhood of recently accessed data. As previously demonstrated, I/O workloads demonstrate a high amount of spatial locality [45]. Here, we characterize the spatial locality of I/O workloads in two ways. In Fig. 4 through Fig. 6, we categorize every request (bottom) as well as either write (middle) or read (top) request into *overlapped*, *sequential*, *strided*, and *random* based on a given page size. The pages accessed by a *sequential* request immediately follow that of the previous request(s) with no gap. *Strided* requests are defined similarly but the gap between the pages is allowed to be no more than the *stride length*, and *overlapped* requests have at least one page in common with the previous request(s). Finally, if a request does not fall under any of these three categories, it is classified as *random*.

Due to the presence of multiple streams of requests in a typical trace, it is beneficial to compare a request against multiple previous requests [49], [45]. We call the distance between line numbers of requests as *Request Order Difference* (ROD), e.g., lines R_i and R_{i+n} have an ROD of n. The maximum allowed ROD for comparing requests against each other is 1 in Fig. 4 and 64 in Fig. 5 and Fig. 6.

Fig. 4 reports I/O breakdown with maximum stride length and ROD being equal to two and one, respectively. In this figure, randomness seems to be a dominant pattern in all workloads. More accurately, *random* requests constitute 83% and 90% of

TABLE III: Characteristics of the Evaluated Workloads

Workload	Description	I/O Requests		I/O Traffic		WR Traffic		RD Traffic		Unique
		Total	WR[%]	Total[GB]	Size[KB]	Total[GB]	Size[KB]	Total[GB]	Size[KB]	Locations[GB]
rsrch	Research projects, vol. 1	1,433,655	90.68	13.1	9.14	11.6	8.93	1.5	11.18	0.414
stg	Web staging, vol. 1	2,196,861	36.25	91.8	41.79	6.4	8.07	85.4	60.97	85.778
ts	Terminal server, vol. 0	1,801,734	82.42	16.6	9.22	12.2	8.20	4.4	14.01	1.050
wdev	Test web server, vol. 0	1,143,261	79.92	10.6	9.30	7.7	8.40	3.0	12.87	0.588
tpcc	TPCC, vol. 2	484,649	36.40	4.1	8.42	1.6	8.84	2.5	8.19	4.331
tpce	TPCE, vol. 8	1,345,380	5.74	11.1	8.28	0.8	9.77	10.4	8.19	13.951
fin1	Financial 1	5,334,987	76.84	18.5	3.47	15.6	3.82	2.8	2.30	0.569
fin2	Financial 2	3,699,195	17.65	9.1	2.45	2.0	2.99	7.1	2.33	0.481

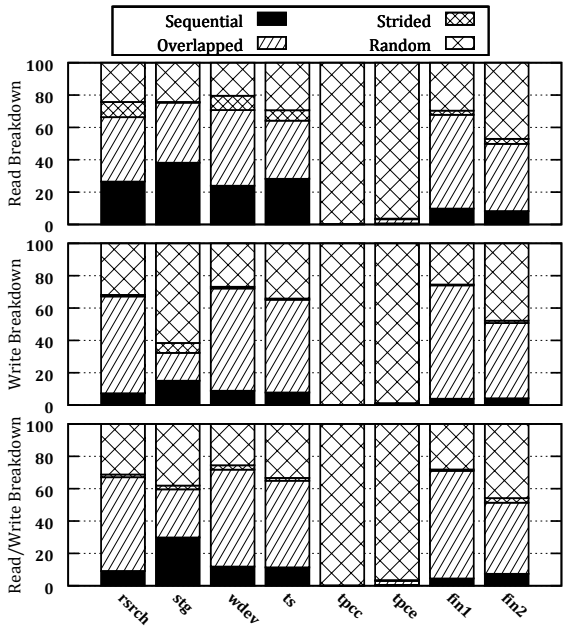


Fig. 5: Breakdown of I/O requests in the MSRC, MSE, and financial workloads for a maximum stride length of 2 and ROD equal to 64.

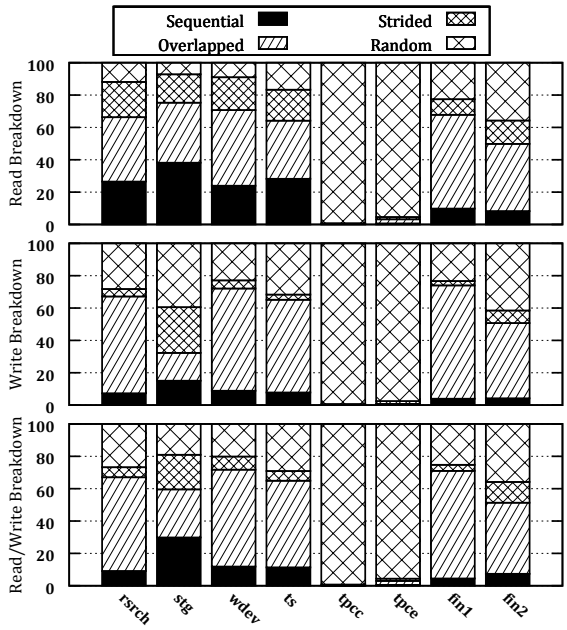


Fig. 6: Breakdown of I/O requests in the MSRC, MSE, and financial workloads for a maximum stride length of 16 and ROD equal to 64.

read and write requests (on average), respectively. However, when we characterize these workloads with larger maximum stride lengths and RODs, the percentage of *random* requests in all workloads (except *tpcc* and *tpce*) drops significantly. Fig. 5 reports I/O breakdown for maximum stride length of 2 and ROD being set to 64. In this figure, the average percentage of *random* requests for read and write accesses is less than 46% and 53%, respectively. Finally, Fig. 6 gives I/O breakdown with maximum stride length and ROD being 16 and 64, respectively. As shown in this figure, the average percentage of *random* requests for read and write requests has fallen to less than 37% and 48%, respectively. Excluding the highly random *tpcc* and *tpce* traces, we can observe that the percentage of *random* requests for read and write accesses is less than 17% and 31%, respectively.

As shown in Fig. 6, all workloads except *tpcc* and *tpce* demonstrate considerable amount of *overlapped* writes, highlighting an opportunity to enhance performance using cache memory. As mentioned previously, *tpcc* and *tpce* have very *random* write accesses, which can significantly increase the number of evictions, resulting in frequent garbage collection and reduced performance.

While the breakdown does present a good overview of requests behavior, it gives no information about the distribution of requests which is of high value when observing spatial locality. To visualize the request distribution over time, we split each trace into multiple phases and take a fixed size interval at the end of each phase. We then plot the access frequency of each interval over the address space at a logarithmic scale. Fig. 7 is an example of such visualization using 10K intervals from the *wdev* workload. In this figure, the X-axis shows the address range and Y-axis is the number of I/O requests in logarithmic scale. Over the course of this paper, we refer to this data representation as *Temporal Working Set Distribution* (TWSD) which demonstrates the locality of I/O requests over an address range and can be used as a representative of the working set. TWSD results are presented in three ways: (a) all I/O requests including both read and write requests (left sub-figures), (b) read requests (middle sub-figures), and (c) write requests (right sub-figures) where each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2455978, IEEE Transactions on Computers

row belongs to the same interval.

In Fig. 7, we have provided TWSD results for the *wdev* workload as a workload demonstrating high spacial locality[1]. Since each set of figures cover the TWSD results for 10K consecutive requests, we have divided each trace to five equal phases and provided TWSD results for only three samples, due to space limitations. Note that we did TWSD extraction and analysis for the last 1K, 10K, 100K, and 1M requests, but here we present the results for the 10K intervals only as they can better represent the working set of a typical cache[2].

A key observation derived from TWSD results is that the number of lines for each graph is a good representation of spatial locality. For instance, *wdev* requests have much more locality than *tpcc* and *tpce* which have poor locality and are spread throughout the disk space. This observation was also confirmed by workload breakdown results in Fig. 4 through Fig. 6. Aside from these two workloads, the rest have significant spatial locality. The TWSD results also reinforce the previous observations about self-similarity across different workloads as different snapshots are often highly similar. A cache policy can take advantage of this property to focus on speeding up accesses to the active areas. In the next section, we investigate the temporal locality of the selected workloads.

B. Temporal Locality

Another important aspect of I/O workloads which is vital for cache effectiveness is temporal locality, which is the likelihood of a recently accessed data block being referenced again in the near future. Our previous observations on access distribution also hints at a high temporal locality for most workloads. This might not necessarily be the case as frequent sequential accesses can also generate such a pattern. As a result, we also decide to use *stack distance* analysis, which tracks the frequency of different re-reference distances. In Fig. 8, we have reported the accumulative stack distance of the studied workloads and similar to previous evaluations, we have used a page size of 8KB. Approaching the accumulative frequency of 1 at a lower distance means that the workload has a high amount of temporal locality. For example, *stg*, *tpcc*, and *tpce* have a much lower temporal locality than the rest which initially experience a very steep rise and then level off afterwards. Accurately identifying the locations that cause this steep rise and focusing on speeding them up can greatly boost the cache performance.

Note that the accumulative stack distance can also be used to roughly estimate the appropriate cache size. Indeed, if a *Least Recently Used* (LRU) cache has a certain amount of entries, it will have a hit ratio equal to the accumulative frequency associated with that distance [50]. For example, if the cache has 10,000 pages, based on the stack distance chart in Fig. 8, it is expected to have a hit ratio near or above 80% for *rsrch*, *wdev*, *ts*, *fin1*, and *fin2*.

For stack distance calculation, we use Algorithm 1. In this algorithm, $Addr_{list}$ structure keeps the addresses of the most recently accessed pages with the most recent access being added to the front, and SD_d counts the number of page accesses that have a stack distance of d. On receiving a request with address *Addr*, the algorithm compares its page with the addresses in $Addr_{list}$. If *Addr* is found at position d, the counter SD_d is incremented by one; otherwise, *Addr* is simply added to top of $Addr_{list}$. Finally, the *Cumulative Distribution Function* (CDF) is generated based on the values of SD_d.

Algorithm 1: Stack Distance Computation using requests

```
Request_list: List of trace requests.
Addr_list: List of previously requested pages.
SD_d: Number of requested pages having a stack distance of d.
foreach Request ∈ Request_list do
    foreach Page ∈ Request do
        begin
            if Page.address found in Addr_list then
                distance ← 0
                while Addr ∈ Addr_list and
                Page.address ≠ Addr do
                    distance ← distance + 1
                Move Page.address in front of Addr_list
                SD_distance ← SD_distance + 1
            else
                Insert Page.address in front of Addr_list
        end
```

V. DRAM-PCM Flat Cache for SSDs

In this section, we begin by analyzing the characterization results presented in the previous section and describe the reasoning behind the design decisions of the proposed hybrid cache. We then describe the structure and the policy of the proposed cache architecture. Finally, we use some simple methods to measure the effectiveness of the adopted cache policy.

A. Characterization Analysis

As can be seen from Sec. IV, there is a significant temporal and spatial locality in most of the selected workloads, which is highly beneficial for a cache policy to take advantage of these savings. TWSD results show that certain addresses are accessed much more frequently, showing an uneven use of memory pages. In other words, in order to make effective use of the reduced DRAM capacity, we must try to only store the active data in this memory. The breakdown results reported in Fig. 4 through Fig. 6 demonstrate that even after increasing the maximum stride length and ROD, a significant portion of the requests are still random, while the rest exhibit high temporal and spatial locality. This reinforces the importance of differentiating between different memory addresses.

Finally, the initial steep climb of the graph for most traces shows a very small stack distance for high percentage of requests. In other words, an intelligent cache policy should discard pages that are to be referenced far in the future and only store the most frequently accessed data in the cache. Observations in Sec. IV-A demonstrated uneven and self-similar access pattern. Hence, one can expect that active data for a certain period of time to remain

[1]The TWSD results for other workloads are provided as supplementary to our submission.

[2]With a page size of 8KB, 10K non-overlapping requests for TWSD analysis take up at least 80MB of storage.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2455978, IEEE Transactions on Computers

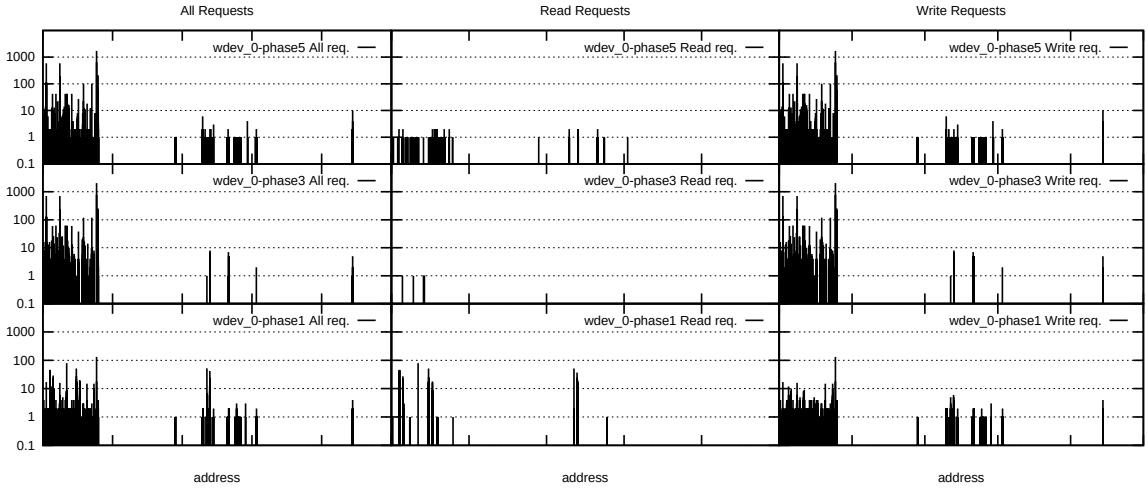


Fig. 7: TWSD for the 10K most recent accesses for *wdev* workload from the MSRC traces.

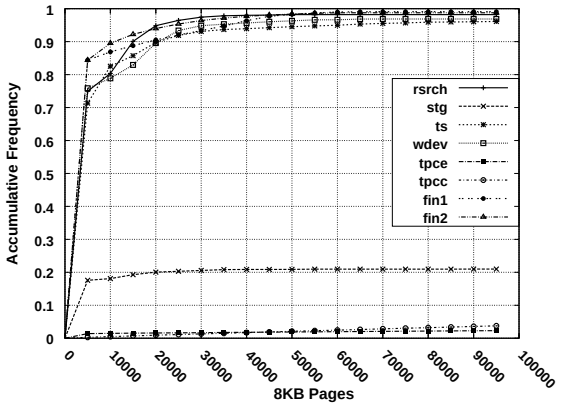


Fig. 8: Results of stack distance analysis.

there and we should take access history into account for placing active data into DRAM, which will not only improve hit ratio and performance, but also absorb writes that will otherwise cause PCM and flash to wear out.

In the next subsection, we discuss how the above findings will be used to efficiently design a hybrid cache and then optimize it. Indeed, we will show that (1) how our design uses temporal locality to store active data in DRAM, (2) how our design enhances PCM lifetime by storing inactive data in SSD, and (3) how it adapts to changes in workload behavior.

B. Cache Structure and Policy

In this section, we will present an architecture which is aimed to satisfy the requirements listed above. The proposed architecture, shown in Fig. 3d, uses a flat cache configuration instead of a multi-level design. While a cold block will inevitably traverse both DRAM and PCM in a two-level cache architecture, the flat cache tracks the block usage and will only move hot blocks to DRAM.

The cache policy can be seen in Fig. 9. While our characterization study as well as the study presented in [45] demonstrate high self-similarity for both read and write requests, technology limitations must be considered when designing cache policies. Specifically, flash memory is much more read friendly as its read operations are very fast and cause no wear-out. In contrast, write requests lead to erase as well as write operations, both of which are much slower than read operations. Write requests also reduce the lifetime of flash memories and require complex out-of-place write operations which can lead to even more wear-out. Another important fact is the existence of strong read caching in most applications where not only more memory is at disposal, but large amounts of non-dirty data can be cached in DRAM without risking data loss. Thus, it can be clearly seen that cache space is far better spent at servicing write requests than read requests, as they are far more critical for performance, endurance, and energy consumption. As a result, for a read request (Fig. 9a), (1) on cache hit, data is read from PCM or DRAM partitions which reduces both latency and power consumption, and (2) on cache miss, the controller simply reads the requested blocks from flash memory with no entry allocation in either of cache partitions. Our cache operates on a per-page basis. Although it may appear that this slows down multi-page requests, our analysis shows that SSD reads and writes are so much slower than PCM and DRAM that overall such delays dwarf that of the cache latency.

Our study on trace behavior has demonstrated a generally high amount of temporal locality for write operations both in the short and long term, with the stack distance showing a very steep rise for most traces, indicating that even small amounts of cache can have a high hit ratio. The excellent write performance of DRAM and its virtually infinite endurance makes it an ideal storage media for highly active data, which also leads to reduced PCM wear-out. Two write policies evaluated in this paper can be seen in Fig. 9b and Fig. 9c where we try to store and move hot data to DRAM while using PCM and SSD for cold data.

To recognize whether a memory line is hot or cold, we keep a saturation counter per cache line in both cache partitions. On write hit, the associated counter is incremented. If a counter associated with a cache line in the PCM array reaches a determined threshold, the page is considered as hot and is scheduled to be moved to the DRAM array. As a result, we define *Hotness Threshold* as a design parameter for differentiating

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2455978, IEEE Transactions on Computers

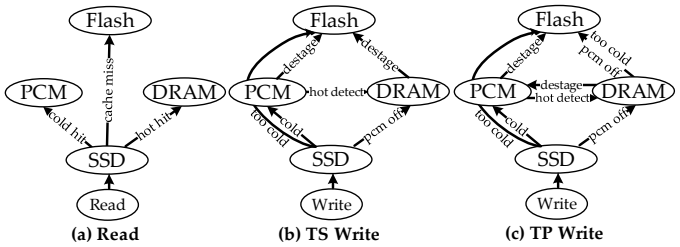


Fig. 9: Page state machines for read and write operations.

hot and cold pages. On the other hand, the proposed cache architecture is associated with a *Coolness Threshold* as another design parameter to determine whether an incoming memory page should be stored in the PCM or flash. If the counter is below the *Coolness Threshold*, the write operation is directed to flash; otherwise, the page is stored in the PCM array. When *Coolness Threshold* equals to *Hotness Threshold*, this situation means that incoming disk patterns are very random and if we continue saving data on PCM, it may hurt the PCM lifetime. Indeed, very random workloads not only fail to benefit from the hybrid architecture, they may cause fast wear-out of the PCM portion of the cache. *Coolness Threshold* in our architecture prevents certain blocks or every block (should the workload be too random) from being written to PCM.

Finally, when either PCM or DRAM cache are full, destaging is triggered which selects and evicts a victim page. On destage from the PCM cache, the victim block is written back to the flash substrate. However, for destaging a page from DRAM, two different write schemes are possible. The first scheme is named as To SSD Write or *TS Write* (Fig. 9b) in which the victim page from DRAM is written back to the flash, the same as the case for PCM. The second scheme is named as To PCM Write or *TP Write* (Fig. 9c) which takes different ways for a victim destage from DRAM depending on the value of *Coolness Threshold*. If page access counter is lower than *Coolness Threshold* or the PCM array is turned off for writes (*Coolness Threshold* = *Hotness Threshold*), the victim page is moved to flash, as TS Write scheme does; otherwise, a cache line is allocated in the PCM array for storing the destaged page from DRAM. Note that new allocation in the PCM array may require a page from PCM to be destaged to flash. For simplicity, we assume that both cache arrays make use of LRU replacement policy with independent counters. Both cache structures are fully-associative, using SRAM tag arrays for fast lookups during hotness prediction search.

C. Determining Design Thresholds

In the proposed cache design, there are two thresholds that should be carefully determined to maximize SSD performance and endurance. *Hotness Threshold* ensures that performance and endurance are maximized by keeping as many hot blocks as possible in DRAM memory. This is determined by detecting unique accesses and setting the threshold to a value that prevents hot blocks from being destaged. *Coolness Threshold* determines whether the block should be stored in PCM or SSD. This threshold controls PCM writes versus SSD writes so that they would last evenly. These thresholds should be updated periodically to keep track of the variations in disk accesses and hence SSD performance and endurance will be enhanced based on application behaviors. Next, we will describe our methodology for determining these thresholds.

Algorithm 2: Hotness Threshold Calculation

$Page$: Requested page.
$PageCounter$: Number of pages so far.
$BlockTrackers$: Array of eight registers tracking blocks.
$AccessCounter$: Page access count, reported by cache or Bloom filters.
$AddressBloomfilter$: A Bloom filter storing block addresses.
$Period$: Threshold update interval length.
$Interpolate$: Linear interpolation function.
if $Page.address$ *found in* $AddressBloomFilter$ **then**
  $BlockTrackers[AccessCounter-1] \leftarrow$
  $BlockTrackers[AccessCounter-1]-1$
else
  Add $Page.address$ to $AddressBloomFilter$
$BlockTrackers[AccessCounter] \leftarrow$
$BlockTrackers[AccessCounter]+1$
if $PageCounter \mod period = 0$ **then**
  $Index \leftarrow Length(BlockTrackers)$
  $Sum \leftarrow 0$
  while $Sum < DRAM_{capacity}$ *and* $Index > 0$ **do**
    $Sum \leftarrow Sum + BlockTrackers[Index]$
    $HotThreshold \leftarrow Index$
    $Index \leftarrow Index - 1$
  Empty $Bucket_{list}$ and $AddressBloomFilter$

Our methodology to set *Hotness Threshold* tracks page access counter (stored in the SRAM memory) of the recently accessed blocks. This policy attempts to use a *Hotness Threshold* that would allow as many hot blocks as possible that would fit in the cache. Hence, we can track the changes in disk access pattern. To accomplish this without adding extra bits to the cache, we follow Algorithm 2 that uses (1) a Bloom filter [51] to detect new blocks and (2) *bucket* registers to restore the counter value of old pages (pages that were previously in cache). We use 8 registers to cover the range of 1 to 8. By conducting experiments, we chose a period of 512, so these counters only require 9 bytes of storage. This algorithm uses a Bloom filter to check whether the page has been accessed before and if this is the case, the bucket range will restore the correct value for the associated counter. This scheme allows tracking access counter of each individual page as it progresses to the next bucket.

To set the *Coolness Threshold*, the ratio of writes to the PCM and SSD should be considered adaptive to make sure that the wear-out of PCM and SSD happens at the same pace. To this end, we define the ratio of PCM to SSD endurance (*Endurance Ratio*) as

$$Endurance_{ratio} = \frac{Size_{PCM} * Endurance_{PCM}}{Size_{SSD} * Endurance_{SSD}}$$

Based on this definition, an ideal lifetime for the proposed hybrid SSD is achievable when

$$SSD_{writes} = period \times \frac{1}{Endurance_{ratio} + 1}$$

Algorithm 3: Calculating Coolness Threshold

```
Page: Newly Accessed Page.
PageCounter: Arrival order of the page.
AccessCounter: Access counter from cache.
AccessCounterFrequency: Four counters where ith index is
occurance of i as page access counter.
SSD_writes: Required Number of SSD writes.
WriteDisparity: Total disparity of SSD writes from previous
periods.
if AccessCounter ≤ 4 then
    AccessCounterFrequency[AccessCounter] ←
    AccessCounterFrequency[AccessCounter] + 1
if PageCounter mod period = 0 then
    SSDRequestCount ← WriteDisparity
    CoolnessThreshold ←1
    while SSDRequestCount < SSD_writes do
        SSDRequestCount ← SSDRequestCount +
        AccessCounterFrequency[CoolnessThreshold]
        CoolnessThreshold ← CoolnessThreshold + 1
    if SSDRequestCount < SSD_writes then
        CoolnessThreshold ← HotnessThreshold
```

where $period$ refers to the duration of an examined period and is fixed in our methodology. We try to regulate writes onto PCM and SSD such that writes onto SSD be in the range of SSD_{writes} in this equation. In contrast to *Hotness Threshold* which is determined by tracking the access counter of each unique page, *Coolness Threshold* should be based on the frequency of access counters values; this is because each write consumes PCM life. For implementation, we use page access counters and access counter frequency. The page access counter is obtained from the saturating counter of cache memory. Frequency counters store the number of times each access count has happened, i.e., the ith counter counts the number of times that an access count of i has happened. Algorithm 3 uses these counters to set *Coolness Threshold*. This algorithm tries to select the lowest *Coolness Threshold* which would cause the ratio of PCM writes to SSD writes follow the above formulation. After being assigned, *Coolness Threshold* is enforced by the Bloom filters in Algorithm 3.

A key point of Algorithm 3 is the use of $WriteDisparity$ which is calculated by subtracting the real number of SSD writes from the expected number of SSD writes:

$$WriteDisparity = ActualSSD_{writes} - ExpectedSSD_{writes}$$

This implementation is necessary because access patterns change over time and there is no warranty that the *Hotness Threshold* or the *Coolness Threshold* result in very similar wear-out in PCM and SSD. This mechanism can also act completely independent of the Bloom filters used for controlling accesses by simply disabling PCM fill operations.

Algorithm 4 controls page destination based on *Coolness Threshold*. Theoretically, pages with an access counter below *Coolness Threshold* should not be written to PCM. However, for the pages that are not allowed to be in the cache before, we have no access history. Thus, we use Counting Bloom filters in Algorithm 4 to count the number of times a block has been accessed. We use an array of four 1KB Bloom filters (requiring 4KB storage in total) to count accesses to blocks not yet allowed in cache. These counters are used to calculate the *Coolness Threshold* and to determine whether a block should be allowed in cache based on this threshold. These Bloom filters as well as the Bloom filters used for *Hotness Threshold* are reset upon every period.

Algorithm 4: Enforcing Coolness Threshold

```
AccessBlooms_list: An array of four Bloom filters.
Page: Newly Accessed Page.
AccessCounter: Access count based on Bloom filters.
if Page not found in Cache then
    AccessCounter ← 1
    while Page.address found in
    AccessBlooms_list[AccessCounter] do
        AccessCounter ← AccessCounter + 1
    if AccessCounter ≤ 8 and
    AccessCounter < CoolnessThreshold then
        Store Page.address in
        AccessBlooms_list[AccessCounter]
    if AccessCounter >= CoolnessThreshold then
        Store in Cache
    else
        Store in SSD
```

VI. Experimental Results

A. Experimental Setup

Environment. We perform trace-driven simulation of an SSD using DiskSim version 4.0 augmented with the Microsoft SSD Model. Although this model precisely implements most of the SSD functionalities, it does not provide a reasonable implementation of the SSD cache model. To be consistent with state-of-the-art SSDs, we have extended the SSD model source code to support a cache memory. We have also modified the parameters with new timings and structure information which boosted the original 32GB capacity to 512GB.

SSD system. We model a 512GB SSD as detailed in Table IV which consists of 8 chips. For PCM arrays, we used NVSim [52], a fully-parametrized latency/power/area simulator for non-volatile arrays, with built-in parameter "PCRAM_ISSCC_2010_14_7.cell" [29]. As our experiments determined that leakage is a main contributor to overall power of large cache arrays, we set the "forcebank" option to minimize leakage power. For DRAM array models, on the other hand, we used CACTI 6.5 [53] structured as a *Non-Uniform Cache Architecture* (NUCA) cache (because of its large size) and single read/write port array. Both DRAM and PCM have a width of 64 bytes, needing 128 operations for an 8KB access. CACTI 5.3 [54] was used to create an SRAM-based fully associative lookup table in the 45nm node. Although the SRAM dynamic power and delay are negligible, we have included the static power consumption in our calculations. Note that in our evaluation, we assume a flash write endurance of 3000 and evaluate cache behavior under PCM per-cell endurances of 10^7 and 10^8.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2455978, IEEE Transactions on Computers

TABLE IV: Characteristics of the simulated SSD model.

Bus-based SSD
8 chips, each independently connected to the controller, Reservation ratio = %15, Sector Transfer Time = 0.238 us per 512 bytes, Garbage collection threshold = %5, Greedy Cleaning Policy, Per-cell Endurance = 3000
SSD cache
Fully-associative, LRU replacement, Write-back, Detailed SSD parameters in Table V
NAND flash (Micron [27])
Page Size = 8KB, Metadata Size = 448B, Block Size = 256 pages, Planes per Die = 2, Dies per Chip = 8[a], Flash Chip Capacity = 32GB, Read Latency = 75μs, Program Latency = 1300μs, Erase Latency = 3.8ms

[a] The organization in the 512Gbit Micron chip has 4 "targets", each containing 2 dies composed of 2 planes. Because of this, the SSD model was configured with 4 "Parallel Units" with 4 planes each.

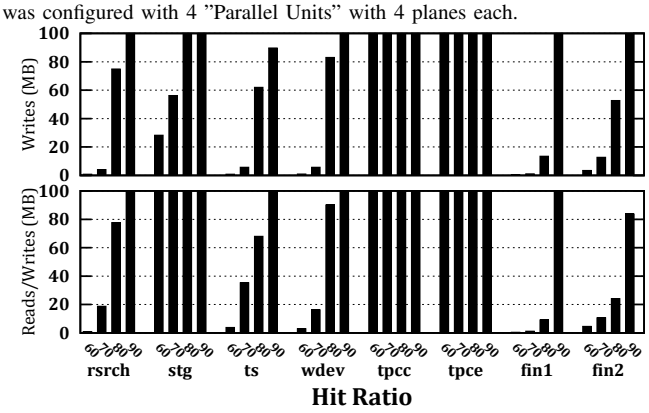


Fig. 10: Minimum SSD cache size required to guarantee a desired hit ratio for write (top) and read/write operations (bottom).

B. Cache Size

While the controller can be configured for different *Hotness Threshold* and *Coolness Threshold* values, the size of DRAM and PCM arrays are constant and has an important impact on performance, power consumption, reliability, and endurance of the proposed cache. Using the results of stack distance analysis in Sec. IV, Fig. 10 shows the minimum required cache size for different workloads for specific hit ratios (60%, 70%, 80% and 90%) for either all requests or only write requests. As can be seen, a cache size between 40MB and 80MB will have a hit ratio of over 70% for most workloads and the hit ratio improvement greatly diminishes after that. As such, we target a minimum hit ratio of 70% and evaluate the proposed cache architecture using the configurations listed in Table V with respect to different latency and energy parameters.

C. Simulation results

Performance, energy consumption, reliability, and endurance of PCM cache are the characteristics that should be considered for the proposed cache design. Regarding performance, higher hit ratio will result in lower response time as the SSD cache subsystem is able to respond both read and write operations quickly. In our experiments, we use the whole SSD response time as a representative of cache setup performance. The per-operation power consumption and latencies for the evaluated configurations can be seen in Table V. It can be seen that while PCM has a much lower leakage, its write operations are not only much slower than DRAM, they also consume much more energy. We initially focus on comparing the energy consumption of the proposed configurations while running the traces. Aside from variations in capacity, we evaluate the scenarios where per-cell PCM endurance is 10^7 or 10^8.

Policy-wise, we present results of destage to SSD (*TS*), destage to PCM (*TP*), and two-level (*2L*). Fig. 11 presents the energy consumption results for our architecture. Among the studied workloads, *MSE* traces behave differently; not only they are highly random, they happen over an extremely short period of time, spanning over only 361 and 609 seconds. In contrast, MSRC and Financial workloads have millions of requests and span over a week and a few hours, respectively. Despite all the differences, power consumption results reported in Fig. 11 demonstrate surprisingly similar patterns of power consumption. In this figure, power consumption is dominated by static leakage of DRAM and PCM, even in the very dense MSE traces. As a result, as long as we are able to keep the static leakage low, we can replace DRAM with an even larger PCM memory. In Fig. 11, it can be seen that except for `12-72`, which increases power by 7%, other configurations result in power savings as high as 77%, 66%, and 64% for `1-20`, `2-40`, and `3-18` configurations, respectively.

Another property mainly affected by the cache configuration is reliability. As all data blocks in a write buffer are dirty, any corruption or data loss will damage the integrity of SSD data. An unmitigated power loss will result in complete loss of DRAM contents and as previously pointed out, less DRAM power consumption requires less expensive backup capacitors and batteries. For example, the `1-20` configuration reduces DRAM power by 86% compared to the pure DRAM configuration. DRAM is also susceptible to soft errors while PCM is highly resilient to particle strikes, which increases the SSD reliability.

Two other main concerns for our architecture are performance and endurance. As PCM may have a lower *Endurance Ratio* (Sec. V-C) than SSD, lifetime management of cache is necessary. Thus, (1) the cache must give an acceptable response time and (2) overall device lifetime should not be bound by the cache subsystem. As stated previously, we use two mechanisms to optimize performance while improving endurance. We set the *Hotness Threshold* to reduce unnecessary evictions from DRAM. We also select *Coolness Threshold* based on the *Endurance Ratio* and a feedback mechanism to ensure that PCM and SSD wear evenly.

Since we try to balance PCM and SSD wear, larger DRAM or higher per-cell PCM endurance can greatly improve endurance as more writes can be directed to the cache. Figure 12a shows the overall longevity of the three evaluated cache policies under different traces when per-cell endurance is 10^7 while Figure 12b displays it for a per-cell endurance of 10^8. The *2L* configuration uses PCM too aggressively that endurance greatly suffers as reported in Figure 12a. If per-cell endurance is an order of magnitude higher, *TS* and *2L* have similar endurance. With low endurance PCM, *TP* and *TS* closely match PCM and SSD endurance (within 2% difference) in MSRC and Financial workloads and ensure that PCM outlasts SSD in the MSE workloads. When per-cell endurance is increased to 10^8, *TP* and *TS* often

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2455978, IEEE Transactions on Computers

TABLE V: Different configurations used in evaluation of the proposed architecture. Latency and energy values assume 8KB pages.

Config.	Size	DRAM					PCM					
		RD Lat.	RD Eng.	WR Lat.	WR Eng.	Leakage	Size	RD Lat.	RD Eng.	WR Lat.	WR Eng.	Leakage
	[MB]	[μS]	(μJ)	[μS]	(μJ)	[mw]	[MB]	[μS]	[μJ]	[μS]	(μJ)	[mw]
42	42	4.352	6.478	4.352	5.899	115.882	No	No	No	No	No	No
12-72	12	3.855	2.973	3.855	2.435	74.239	72	5.454	0.042	48.983	11.049	38.325
6-36	6	3.749	1.699	3.749	1.073	35.175	36	2.37	0.029	42.262	11.036	22.09
4-80	4	3.749	1.699	3.749	1.073	25.315	80	5.454	0.042	48.983	11.049	41.653
3-18	3	3.518	1.593	3.518	1.072	25.275	18	2.775	0.033	44.26	11.041	9.652
2-40	2	3.679	1.03	3.679	0.361	8.419	40	2.37	0.024	42.262	10.442	25.014
1-20	1	3.679	1.03	3.679	0.361	8.419	20	2.775	0.033	44.26	11.041	10.484

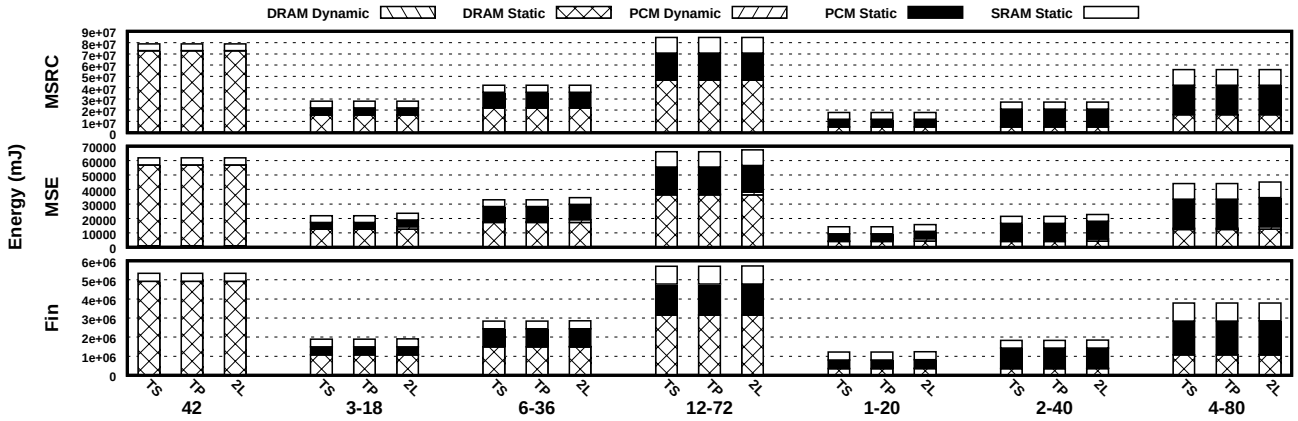


Fig. 11: Energy consumption of the proposed architecture with Destage to SSD (*TS*), Destage to PCM (*TP*), and Two Level (*2L*) policies for different DRAM-PCM configurations.

match SSD to PCM wear while ensuring PCM never limits cache lifetime.

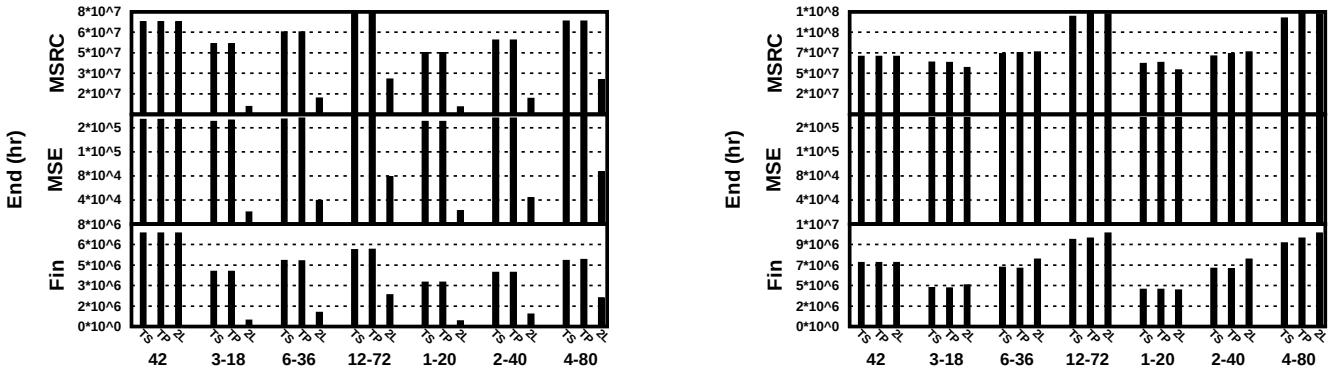
We use the response time of the SSD to compare performance of different design choices. Not only response time is a measure of how effective DRAM and PCM have been at absorbing requests but it is also the most visible identifier of performance for a storage device. Figure 13a and Figure 13b compare the whole drive response time of the three policies being evaluated. Performance-wise, 4-80 and 12-72 are much better than the rest, but they greatly diminish energy-efficiency gains and in case of 12-72 actually increase power consumption. 12-72 has very close performance to 42 in low endurance PCM and has much better performance (up to 26%) in the high endurance configuration. 4-80 has up to 21% penalty on low endurance and increases performance up to 26% when endurance is 10^8. 2-40 and 6-36 have too much of a gap in performance on 10^7 per-cell endurance but have a worst-case performance penalty of less than 8% when endurance is higher. *2L* usually has better performance across different configurations and especially with low endurance PCM memory. However, this comes with a massive hit in endurance with the low endurance memory. With high endurance PCM, both endurance and performance are close. It must be noted that although hybrid configurations are slower than a pure-DRAM solution, performance gains are made possible by increased cache hit ratio which reduces the number of accesses to the SSD.

Some key observations can be made with respect to these results, (1) our hybrid cache can consistently preserve device lifetime even if PCM per-cell endurance is low, (2) in low PCM endurance situations, only 12-72 and 4-80 can compete with the 42, but with high PCM endurance, 2-40 and 6-36 often have similar endurance and performance competitive to this configuration, (3) 4-80 has better energy efficiency, competitive performance and endurance in all scenarios compared to 42, (4) Our methodology for balancing wear of PCM and SSD has made it possible for *TP* and *TS* to balance SSD and PCM usage and obtain much better endurance than *2L*, (5) *2L* universally achieves better performance but usually at cost of lower endurance compared to *TP* and *TS*.

Finally, to compare our methodology against using fixed *Hotness Thresholds*, we did a comprehensive run over a wide range of *Hotness Thresholds* (from 1 to 8) for both *TP* and *TS* policies with *Coolness Threshold* disabled (every page is assumed to be above the *Coolness Threshold*) and compared it against *TP* and *TS* with *Hotness Threshold* active and wear control deactivated. For comparison with our dynamic methodology, for each combination of configuration (we omitted 42 as it turns our cache into a simple LRU cache) and workload, we picked the best *Hotness Threshold*. Our automatic threshold control for *TS* and *TP* has a response time penalty of less than 1% compared to manually picking the threshold with the best performance. When *TS* and *TP* wear control is off, they only have a performance penalty of 2% and 1% over *2L*, respectively.

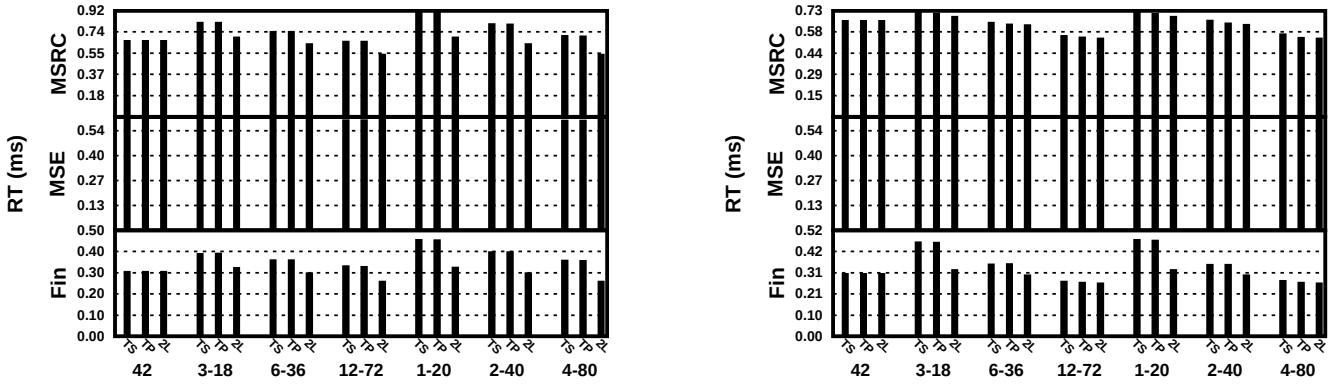
To conclude, the choice of configurations and their poli-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2455978, IEEE Transactions on Computers



(a) Per-cell endurance = 10^7 (b) Per-cell endurance = 10^8

Fig. 12: Lifetime (hours) of the proposed architecture with Destage to SSD (*TS*), Destage to PCM (*TP*), and Two Level (*2L*) policies for different DRAM-PCM configurations and workload categorizes.



(a) Per-cell endurance = 10^7 (b) Per-cell endurance = 10^8

Fig. 13: Response time of the proposed architecture with Destage to SSD (*TS*), Destage to PCM (*TP*), and Two Level (*2L*) policies for different DRAM-PCM configurations and workload categorizes.

cies are user preferences. If the goal is to minimize cache power consumption, small-sized configurations excel at that albeit with performance and endurance penalties. Medium-sized configurations can nearly match the performance while greatly reducing power consumption. On the other hand, large sized configurations are energy-efficient with regards to their capacity, and improve performance and endurance. The key benefit of our proposed architecture, aside from energy-efficiency gained from combining hybrid technologies, is the ability of our flat hybrid design to cope with low endurance NVM cache. Endurance is most likely to worsen by technology scaling, a good example of that being flash scaling which has worsened endurance by orders of magnitude. We believe that our cache design is able to optimize lifetime in varying scenarios, especially real-world designs where endurance is a challenging issue.

VII. Related Work

A. Flash-Aware Write Buffer Management Schemes

Jo *et al.* proposed *Flash-Aware Buffer* (FAB) [55] which is similar to the traditional cache structures but sorts page clusters by their size. The largest cluster has been chosen as a victim. This method aims to increase destage size and reduce FTL merge operations. Also, Kim *et al.* proposed *Block Padding LRU* (BPLRU) buffer management to enhance the performance of random writes to flash storage [56]. Page padding changes the fragmented write patterns to sequential ones using spatial locality in order to reduce the cost of buffer flush. However, this scheme can expedite the wear-out of the flash media.

Kang *et al.* proposed *Cold and Large Cluster* (CLC) as another write buffer replacement algorithm for NAND flash memory [57]. CLC uses non-volatile memory as write buffer and partitions page cluster lists into two groups. Size-independent clusters (similar to FAB) are associated to high locality page clusters and are ordered using LRU. Size-dependent clusters are

associated to low locality clusters, ordered by the size. Wu *et al.* proposed *Block-Page Adaptive Cache* (BPAC) management scheme for buffering SSD write requests exploiting temporal and spatial localities [58] and compared it against CLC. This method estimates spatial locality of the workload, and selects cache elements with low spatial locality as candidates for replacements.

B. Flash-Aware Cache Management Schemes

Park *et al.* proposed *Clean-First LRU* (CFLRU) as a flash-aware replacement algorithm for the read/write in-memory I/O cache [59]. CFLRU resides in the operating system and addresses the asymmetric speed of flash by allowing dirty pages to stay in a buffer longer than clean pages. Jung *et al.* introduced *Write Sequence Reordering LRU* (LRU-WSR) [60] which changes the ordering of pages in LRU queue to avoid eviction of hot pages. This reduces the total number of destages to flash memory. More recently, Li *et al.* have addressed shortcomings of CFLRU and introduced *Cold Clean First LRU* (CCF-LRU) [61]. This scheme maintains a separate LRU for cold clean pages and tries to find a victim page from the list.

Finally, Huang *et al.* proposed a method aimed at providing a balanced read/write performance for flash SSDs [62]. This method attempts to flush pages from the cache based on the LRU or LFU policy at idle times. However, the results indicated very minor difference between these two policies.

There are also several algorithms which have been introduced to utilize non-volatile write buffer for HDDs. These algorithms, however, are not suited for NAND flash-based SSDs due to the limitations of the flash memory and FTL.

C. SRAM/STT-RAM Hybrid Caches

Owing to real concerns of SRAM power in nanometer era, resistive memories such as STT-RAM are presented to offer a highly-scalable low-leakage alternative for large cache arrays. Compared to competitive non-volatile memories (such as ReRAM, PCM, and FeRAM), STT-RAM benefits from best attributes of fast nanosecond access time, partial CMOS process compatibility, high density, and better write endurance. This way, Dong et al. give a detailed circuit-level comparison between SRAM cache and STT-RAM cache in a single-core microprocessor [63]. Based on the findings given in this study, Sun et al. extend the application of STT-RAM to NUCA cache substrate in *Chip Multi-Processors* (CMPs) and study the impact of the costly write operation in STT-RAM on power and performance [64]. To address the slow write speed and high write energy of STT-RAM, one approach is SRAM/STT-RAM hybrid cache hierarchies and some enhancements, such as write buffering [65], data migration [65], [66], [67], and data encoding [68]. For instance, it is shown that few SRAM lines can considerably reduce write frequency over a large number of STT-RAM lines in hybrid cache architectures [65]. The same as our proposal in this paper, these techniques require some procedures to detect write-intensive data blocks in each set for data migration. Due to fast and high power features of large SRAM/STT-RAM caches, this hybrid memory is only meaningful when applying to on-chip cache hierarchy. Besides, in contrast to SSD cache design, a designer should reduce the complexity of management techniques for on-chip hybrid caches to keep the performance overhead low. However, SSD cache has much larger slack times and it is meaningful to integrate more complex design to have best efficiency (as we did by implementing methodologies for identification and tracking of hot data and balancing wear between PCM and SSD).

D. Storage Optimization Using Bloom Filter

Basu et al. tried using spectral Bloom filters to identify the most missed data and to give them higher priority [69]. Strunk et al. tested Spectral Bloom filters to detect hot data and migrate them to SSD (from HDD), but their design suffers from lack of getting maximum efficiency. Yun et al. used counting Bloom filter to control hot and cold thresholds which are then swapped data blocks for wear-leveling in PCM memories [70]. Park et al. and Hsieh et al. used Bloom filter-based solutions to track block access frequency and identify hot blocks based on a fixed threshold. In contrast to prior studies, our design uses the concept of Bloom filters to control data transactions within the proposed cache design/state machine. In fact, for Hot blocks, we only use a single Bloom filter to detect recently-accessed blocks within a period, and for Cold blocks, we use Counting Bloom filters and shifting for decay.

VIII. Conclusion

The PCM emerging technology has shown great potential for use in SSD cache subsystems if its shortcomings in endurance, performance, and energy consumption are efficiently addressed. In this paper, we showed that using pure PCM or two-level hierarchical DRAM-PCM SSD caches face serious lifetime problems. Then, by detailed characterization of the I/O workloads, we demonstrated that the most promising design candidate is the arrangement of the two partitions in a flat design. Such a design requires more than a naive replacement of cache arrays and must actively migrate data (especially those that are write-intensive or hot) across PCM and DRAM cache structures to maximize the benefits. Our experiments demonstrated up to 77% power savings in the cache and up to 20% reduction in request response time. Using low endurance PCM, our hybrid design can improve the lifetime of a flash-based SSD by 23% (150% when per-cell endurance is 10^8). This work showed that emerging PCM technology can be efficiently integrated with DRAM in SSD cache.

We should note that although this paper uses and evaluates PCM technology in the proposed hybrid SSD cache, the same technique can be easily applied when alternative memory technologies are employed instead of PCM and DRAM. We leave this study as a future work.

References

[1] S. Lee *et al.*, "A log buffer-based flash translation layer using fully-associative sector translation," *ACM Trans. Embedded Computing Systems*, vol. 6, no. 3, 2007.

[2] S. Lee *et al.*, "LAST: locality-aware sector translation for NAND flash memory-based storage systems," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 6, pp. 36–42, 2008.

[3] A. Gupta, Y. Kim, and B. Urgaonkar, "DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings," in *ASPLOS*, pp. 229–240, 2009.

[4] J.-U. Kang *et al.*, "A superblock-based flash translation layer for nand flash memory," in *EMSOFT*, pp. 161–170, 2006.

[5] N. Agrawal *et al.*, "Design tradeoffs for SSD performance," in *USENIX ATC*, pp. 57–70, 2008.

[6] "The openssd project." http://www.openssd-project.org/wiki/The_OpenSSD_Project.

[7] J. Y. Shin *et al.*, "FTL design exploration in reconfigurable high-performance SSD for server applications," in *ICS*, pp. 338–349, 2009.

[8] J. Shaman, "OCZ Vertex 450 SSD review." http://www.storagereview.com/ocz_vertex_450_ssd_review, May 2013.

[9] K. OBrien, "Plextor PX-M5S SSD review." http://www.storagereview.com/plextor_pxm5s_ssd_review, July 2012.

[10] J. Shaman, "Samsung 840 EVO SSD review." http://www.storagereview.com/samsung_840_evo_ssd_review, July 2013.

[11] A. Riska and E. Riedel, "Disk drive level workload characterization," in *USENIX ATEC*, pp. 9–9, 2006.

[12] M. Zheng *et al.*, "Understanding the Robustness of SSDs under Power Fault," in *USENIX FAST*, pp. 271–284, 2013.

[13] M. K. Qureshi *et al.*, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in *MICRO*, pp. 14–23, 2009.

[14] B. C. Lee *et al.*, "Architecting phase change memory as a scalable DRAM alternative," in *ISCA*, pp. 2–13, 2009.

[15] P. Zhou *et al.*, "A durable and energy efficient main memory using phase change memory technology," in *ISCA*, pp. 14–23, 2009.

[16] J. S. Bucy *et al.*, "The DiskSim Simulation Environment Version 4.0 Reference Manual," Tech. Rep. CMU-PDL-08-101, Parallel Data Laboratory, Carnegie Mellon University, May 2008.

[17] K. OBrien, "Corsair Neutron GTX SSD review." http://www.storagereview.com/corsair_neutron_gtx_ssd_review, August 2012.

[18] J. Shaman, "Crucial M500 SSD review." http://www.storagereview.com/crucial_m500_ssd_review, April 2013.

[19] K. OBrien, "Intel SSD 520 review." http://www.storagereview.com/intel_ssd_520_review, February 2012.

[20] J. Shaman, "Seagate 600 SSD review." http://www.storagereview.com/seagate_600_ssd_review, May 2013.

[21] "SSD overview." http://www.samsung.com/global/business/semiconductor/product/flash-ssd/overview.

[22] D. Rollins, "An Overview of SSD Write Caching," Tech. Rep. 05/12 EN.L M:11841, Micron Technology, Inc., 2012.

[23] A. Shimpi, "The intel SSD 320 review: 25nm G3 is finally here." http://www.anandtech.com/show/4244/intel-ssd-320-review, 2011.

[24] J. H. Yoon, C. H. Hunter, and G. A. Tressler, "Flash & DRAM Si scaling challenges, emerging non-volatile memory technology enablement - implications to enterpris storage and server compute systems," in *Flash Memory Summit*, 2013.

[25] Toshiba Corporation, "August 2012 Semicondutor General Catalog, Memories and Storage Devices," 2012.

[26] J. Cooke, "The Inconvenient Truths of NAND Flash Memory," in *Flash Memory Summit*, 2007.

[27] Micron NAND Flash Memory, "64Gb, 128Gb, 256Gb, 512Gb Asynchronous/Synchronous NAND," 2009.

[28] L. M. Grupp *et al.*, "Characterizing flash memory: anomalies, observations, and applications," in *MICRO*, pp. 24–33, 2009.

[29] G. De Sandre *et al.*, "A 90nm 4Mb embedded phase-change memory with 1.2V 12ns read access time and 1MB/s write throughput," in *ISSCC*, pp. 268–269, 2010.

[30] R. Jeyasingh *et al.*, "Phase change memory: Scaling and applications," in *IEEE CICC*, pp. 1–7, 2012.

[31] Z. Wei, Y. Kanzawa, K. Arita, Y. Katoh, K. Kawai, S. Muraoka, S. Mitani, S. Fujii, K. Katayama, M. Iijima, T. Mikawa, T. Ninomiya, R. Miyanaga, Y. Kawashima, K. Tsuji, A. Himeno, T. Okada, R. Azuma, K. Shimakawa, H. Sugaya, T. Takagi, R. Yasuhara, K. Horiba, H. Kumigashira, and M. Oshima, "Highly reliable TaOx ReRAM and direct evidence of redox reaction mechanism," in *IEDM*, pp. 1–4, 2008.

[32] K.-H. Kim, S. H. Jo, S. Gaba, and W. Lu, "Nanoscale resistive memory with intrinsic diode characteristics and long endurance," *Applied Physics Letters*, vol. 96, pp. 2237–2251, May 2010.

[33] H. Lee, Y. S. Chen, P. Chen, P. Gu, Y. Hsu, S. Wang, W. Liu, C. Tsai, S. Sheu, P.-C. Chiang, W. Lin, C. H. Lin, W.-S. Chen, F. Chen, C. Lien, and M. Tsai, "Evidence and solution of over-RESET problem for HfOX based resistive memory with sub-ns switching speed and high endurance," in *IEDM*, pp. 19.7.1–19.7.4, 2010.

[34] Y.-B. Kim, S. R. Lee, D. Lee, C. B. Lee, M. Chang, J. H. Hur, M.-J. Lee, G.-S. Park, C.-J. Kim, U.-i. Chung, I.-K. Yoo, and K. Kim, "Bi-layered RRAM with unlimited endurance and extremely uniform switching," in *VLSIT*, pp. 52–53, 2011.

[35] M.-J. Lee, C. B. Lee, D. Lee, S. R. Lee, M. Chang, J. H. Hur, Y.-B. Kim, C.-J. Kim, D. H. Seo, S. Seo, U.-I. Chung, I.-K. Yoo, and K. Kim, "A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta2O5-x/TaO2-x bilayer structures," *Nature Mater*, vol. 10, pp. 625–630, Aug. 2011.

[36] K. Aratani, K. Ohba, T. Mizuguchi, S. Yasuda, T. Shiimoto, T. Tsushima, T. Sone, K. Endo, A. Kouchiyama, S. Sasaki, A. Maesaka, N. Yamada, and H. Narisawa, "A novel resistance memory with high scalability and nanosecond switching," in *IEDM*, pp. 783–786, 2007.

[37] X. Wu *et al.*, "Power and performance of read-write aware hybrid caches with non-volatile memories," in *DATE*, pp. 737–742, April 2009.

[38] G. Sun *et al.*, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *HPCA*, pp. 239–249, 2009.

[39] X. Wu *et al.*, "Hybrid cache architecture with disparate memory technologies," in *ISCA*, pp. 34–45, 2009.

[40] Y.-T. Chen *et al.*, "Dynamically reconfigurable hybrid cache: An energy-efficient last-level cache design," in *DATE*, pp. 45–50, 2012.

[41] T. Kgil, D. Roberts, and T. Mudge, "Improving NAND flash based disk caches," in *ISCA*, pp. 327–338, 2008.

[42] L. Shi *et al.*, "Hybrid nonvolatile disk cache for energy-efficient and high-performance systems," *ACM Trans. Design Automation of Electronic Systems*, vol. 18, no. 1, pp. 1–23, 2013.

[43] C. Ruemmler and J. Wilkes, "HP laboratories UNIX disk access patterns."

[44] D. Roselli, J. R. Lorch, and T. E. Anderson, "A comparison of file system workloads," in *USENIX ATC*, pp. 4–4, 2000.

[45] S. Kavalanekar *et al.*, "Characterization of storage workload traces from production Windows Servers," in *IISWC*, pp. 119–128, 2008.

[46] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: Practical power management for enterprise storage," *ACM Trans. Storage*, vol. 4, no. 3, pp. 1–23, 2008.

[47] "Microsoft enterprise traces, storage networking industry association IOTTA repository." http://iotta.snia.org/traces/130.

[48] "UMassTraceRepository." http://traces.cs.umass.edu/index.php/Storage/Storage.

[49] I. Ahmad, "Easy and Efficient Disk I/O Workload Characterization in VMware ESX Server," in *IISWC*, pp. 149–158, 2007.

[50] V. Almeida *et al.*, "Characterizing reference locality in the WWW," in *DIS*, pp. 92–107, 1996.

[51] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[52] X. Dong *et al.*, "NVSim: a circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.

[53] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing

NUCA organizations and wiring alternatives for large caches with CACTI 6.0," *MICRO*, pp. 3–14, 2007.

[54] S. Thoziyoor *et al.*, "CACTI 5.1," *HP Laboratories, Palo Alto, Tech. Rep*, vol. 20, 2008.

[55] H. Jo *et al.*, "FAB: Flash-aware buffer management policy for portable media players," *IEEE Trans. Consumer Electronics*, vol. 52, no. 2, pp. 485–493, 2006.

[56] H. Kim and S. Ahn, "BPLRU: a buffer management scheme for improving random writes in flash storage," in *USENIX FAST*, 2008.

[57] S. Kang *et al.*, "Performance trade-offs in using NVRAM write buffer for flash memory-based storage devices," *IEEE Trans. Computers*, vol. 58, no. 6, pp. 744–758, 2009.

[58] B. E. G. Wu and X. He, "BPAC: An adaptive write buffer management scheme for flash-based solid state drives," in *MSST*, pp. 1–6, 2010.

[59] S. Park *et al.*, "CFLRU: a replacement algorithm for flash memory," in *CASES*, pp. 234–241, 2006.

[60] H. Jung *et al.*, "LRU-WSR: integration of lru and writes sequence reordering for flash memory," *IEEE Trans. Consumer Electronics*, vol. 54, no. 3, pp. 1215–1223, 2008.

[61] Z. Li *et al.*, "CCF-LRU: A new buffer replacement algorithm for flash memory," *IEEE Trans. Consumer Electronics*, vol. 55, no. 3, pp. 1351–1359, 2008.

[62] M. Huang *et al.*, "Efficient cache design for solid-state drives," in *CF*, pp. 41–50, 2010.

[63] X. Dong *et al.*, "Circuit and microarchitecture evaluation of 3D stacking magnetic ram (MRAM) as a universal memory replacement," in *DAC*, pp. 554–559, June 2008.

[64] S. Microsystems, "ULTRASPARC T2 supplement to the ULTRASPARC architecture 2007," tech. rep., 2007.

[65] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *HPCA*, February 2009.

[66] A. Jadidi, M. Arjomand, and H. Sarbazi-Azad, "High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement," in *ISLPED*, 2011.

[67] X. Wu *et al.*, "Power and performance of read-write aware hybrid caches with non-volatile memories," in *DATE*, pp. 737–742, March 2009.

[68] M. Arjomand, A. Jadidi, and H. Sarbazi-Azad, "Relaxing writes in non-volatile processor cache using frequent value locality," in *DAC*, June 2012.

[69] A. Basu *et al.*, "Scavenger: A new last level cache architecture with global block priority," in *MICRO*, pp. 421–432, December 2007.

[70] J. D. Strunk, "Hybrid aggregates: Combining ssds and hdds in a single storage pool," *SIGOPS Oper. Syst. Rev.*, vol. 46, pp. 50–56, Dec. 2012.