CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
*Concurrency Computat.: Pract. Exper.* 2016; **00**:1–27
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

# A Survey of Techniques for Architecting SLC/MLC/TLC Hybrid Flash Memory based SSDs

AHMED IZZAT ALSALIBI*, SPARSH MITTAL†, MOHAMMED AZMI AL-BETAR$^{\alpha}$ and PUTRA BIN SUMARI$^{\Phi}$

**Israa university Gaza-Palestine; †Indian Institute of Technology, Hyderabad, India; $^{\alpha}$Al-Balqa Applied University, Jordan; $^{\Phi}$Universiti Sains Malaysia, Malaysia*

SUMMARY

Flash memory based SSDs offer several attractive features and benefits compared to hard disk drive (HDD), such as shock resistance, better performance especially for random data access, etc. Depending on the number of bits in each cell, Flash memory can be designed as single/multi/triple level cell (SLC/MLC/TLC) which have different performance, density, cost and write endurance characteristics. To bring the best of these together, several researchers have proposed designing solid state drive (SSD) using hybrid SLC/MLC/TLC Flash memory. However, these SSDs also present several challenges such as buffer management, placement of hot/cold data in suitable portion, intelligent garbage collection, etc. Several recent techniques aim to address these challenges. In this paper, we present a survey of techniques for managing SSDs designed with SLC/MLC/TLC Flash memory. We classify the works on several axes to bring out their similarities and differences. We aim to synthesize the state-of-art progress in hybrid SSD management and also spark further research in this area. Copyright © 2016 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Hybrid solid state disk, Flash translation layer, NAND Flash memory, garbage collection, wear leveling

## 1. INTRODUCTION

As the amount of digital data continues to grow at an exponential rate and key applications become more data-intensive, efficient storage architectures and management techniques have become more important than ever before. Conventionally, hard disk drive† has been used as a storage device, however, its limitations such as poor performance (especially for random accesses), higher form factor, vulnerability to shocks and magnetic fields, etc. have encouraged researchers to explore its alternatives.

*Correspondence to: Ahmed and Sparsh are co-first authors. Authors' addresses: A. I. Alsalibi (corresponding author), Information Systems College, Israa university, Gaza-Palestine; Sparsh Mittal, IIT Hyderabad, India; P. B. Sumari, School of Computer Sciences, Universiti Sains Malaysia, Penang, 11800, Malaysia; M. A. Al-Betar, Department of Information Technology, Al-Balqa Applied University. Emails: ahmed.salibi@gmail.com, sparsh0mittal@gmail.com, putras@usm.my, betar79@gmail.com. Support for this work was provided by Science and Engineering Research Board (SERB), India, award number ECR/2017/000622.

†We use the following acronyms frequently in this paper: error-correcting code (ECC), flash translation layer (FTL), garbage collection (GC), global positioning system (GPS), least/most recently used (LRU/MRU), least/most significant bit (LSB/MSB), logical page address/number (LPA/LPN), logical sector number (LSN), logical/physical superblock address (LSBA/PSBA), non-volatile RAM (NVRAM), phase change memory (PCM), physical page address/number (PPA/PPN), resistive random-access memory (ReRAM), service-level objective (SLO), spin transfer torque RAM (STT-RAM), storage-class memory (SCM), universal serial bus (USB). We use both SCM and NVRAM to refer to byte-addressable non-volatile memories, viz., PCM, domain wall memory, ReRAM and STT-RAM [1].

Flash memory is a promising technology for designing storage devices due to its several attractive properties, e.g., high performance and density, low power consumption, noise-free operation and immunity to shocks and magnetic fields [2–6]. Also, its cost has been decreasing in recent years and it is expected to provide even better cost efficiency than HDD in near future. Based on the number of bits stored in each cell, Flash can be characterized as SLC (1 bit), MLC[‡] (2 bit) and TLC (3 bits). As shown in Table I, these cell-types provide a spectrum of properties and tradeoffs. Specifically, on going from SLC to MLC to TLC, the performance and write endurance decrease whereas density and cost efficiency improve.

Table I. Comparison of SLC/MLC/TLC Flash memories.[10–15] (out-of-band size is not shown for clarity)

| Features | SLC | MLC | TLC |
|---|---|---|---|
| Bits/cell | 1 | 2 | 3 |
| Cost | High | Medium | Low |
| Page size | 2,048 bytes | 16,384 bytes | 16,384 bytes |
| Pages per block | 64 | 1024 | 1536 |
| Block size | 128K bytes | 16,384K bytes | 24,576K bytes |
| Page read | 25 $\mu$s | 50 $\mu$s | 75 $\mu$s |
| Program time | 200-300 $\mu$s | 600-900 $\mu$s | 900-1350 $\mu$s |
| Block erase | 1.5-2 ms | 3 ms | 4.5-10 ms |
| P/E cycles | 100,000 | 3,000-10,000 | 300-3,000 |
| Application | USB, SSD, digital camera, mobile handset & networking | USB, SSD, media player, digital camera, mobile handset & GPS | USB, SSD, media player & mobile GPS |

To achieve the best of these three cell-types, hybrid SSD designs have been proposed which use Flash memories of multiple cell-types to improve performance, energy efficiency and reliability [16]. However, these hybrid SSD designs also bring challenges, such as selection of relative proportion of SLC/MLC/TLC, efficient mapping/moving of hot/cold data to them, accounting for their disparate write endurance and density values, ensuring efficient GC, etc. Clearly, management of hybrid SSDs brings challenges of its own and hence traditional techniques for managing homogeneous (e.g., MLC-only) SSD may not work well for hybrid SSDs. Recently, several techniques have been proposed to address these challenges.

**Contributions:** In this paper, we present a survey of techniques for designing and managing hybrid SSD devices. Figure 1 shows the overall organization of this paper. We first present a brief background on Flash memory architecture, operations and SSD management approaches (Section 2). We then provide classification of research works from multiple perspectives to offer insights (Section 3). Then, we review hybrid SSD management techniques in terms of their partitioning techniques (Section 4), buffer design (Section 5) and optimization objective and solution schemes (Section 6). In these sections, we discuss each research work in one group only, although many of the works fall under multiple groups. We conclude this paper with a mention of future research directions in this field (Section 7).

**Scope:** For sake of a concise presentation, we limit the scope of this paper as follows. We focus on software-level management techniques for hybrid SSDs and not their circuit-level design issues. We include techniques which use at least two types of Flash and not those that merely use an SCM with a Flash cell-type. We focus on the key ideas of each work and include only selected quantitative results, since different works use disparate evaluation platforms and workloads. We hope that this paper will be useful for computer architects, SSD designers and researchers in the area of storage architectures.

[‡]Note that the term MLC is also sometimes used to refer to Flash cell with multiple (two or more) levels per cell which includes TLC [7–9], however, in this paper, we use MLC to refer to Flash with 2 bits per cell only.



DOI: 10.1002/cpe

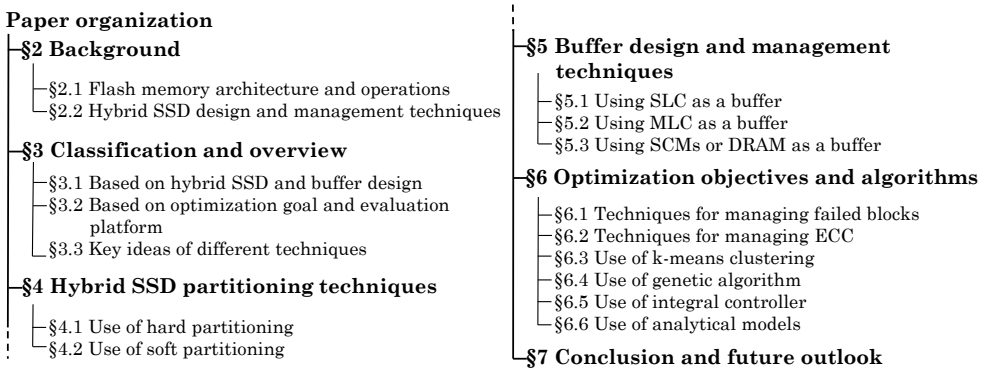



Figure 1. Paper organization

## 2. BACKGROUND

### *2.1. Flash memory architecture and operations*

We now briefly review the organization of SSD and Flash memory and refer the reader to prior works [17, 18] for more details.

**Flash SSD architecture:** An SSD has multiple packages, each consisting of one or more chip dies. Each die consists of multiple planes which has several blocks. Further, each block has multiple pages, e.g., the size of a page and a block may be 4KB and 256KB, respectively [19]. Each page is logically divided into a large 'user area' which stores the user data and a small 'out-of-band' area which stores mapping information, metadata (e.g., erase counter and page state) and ECC [20, 21].

**Flash memory operations:** Flash memory allows program (write), read, and erase operations, which are managed by the FTL [22]. Reads/writes happen at page granularity whereas erase happens at block granularity. A write operation can only change the stored-bits from 1 to 0. Hence, the only way to change a bit in a page from 0 to 1 is to erase the block containing the page which sets all bits in the block to 1. The pages in Flash are classified as free (available for storing new data), invalid (storing dead data) and valid.

**Garbage collection and wear-leveling:** Since erase operations are much slower than write operations, FTL seeks to hide erase latency by performing 'out-of-place' writes whereby a new write is performed to a free page and the previous location of the page is invalidated [2]. To release invalid pages, FTL periodically performs garbage collection whereby the valid pages of a block are copied elsewhere; the block is erased and all its pages are marked as free. By performing erase operations in background, FTL hides the erase operations and exposes only read/write operations to the user. Since Flash write endurance is small, the wear-leveling module seeks to distribute the number of program/erase cycles evenly among all the blocks to increase overall device lifetime [23].

**Address translation:** The address translation software records the mapping between logical addresses in the file system to physical addresses in Flash memory. Since writes to Flash are performed 'out-of-place', the address mapping between file system and Flash memory is continuously updated.

### *2.2. Hybrid SSD design and management techniques*

*2.2.1. Hybrid SSD design:* A hybrid SSD (with hard-partitioning) is designed by connecting SLC/MLC/TLC chips over different channels. Figure 2 shows a generic hybrid SSD design.

*2.2.2. Hybrid SSD partitioning techniques* Hybrid SSDs may use either hard or soft partitioning. We discuss them with example of an SLC/MLC hybrid SSD.

**Hard partitioning:** In hard partitioning, SLC and MLC chips are physically separated. A particular chip continues to work as SLC or MLC during entire the execution time and thus, there is no mode switching or change in their capacity. For example, in Figure 3(a), 3 SLC and 9 MLC chips



DOI: 10.1002/cpe

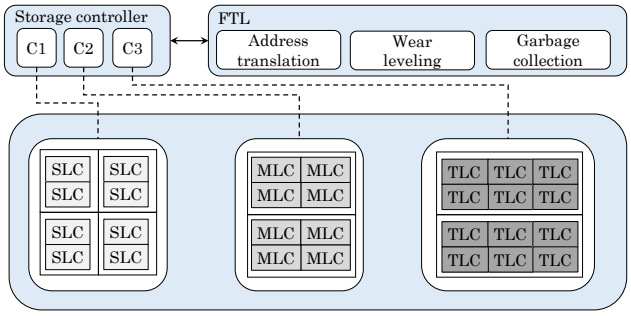



Figure 2. A generic hybrid SSD architecture. C1/C2/C3 refers to channels 1, 2 and 3, respectively [18, 24]

are used. Assuming that the SLC and MLC portions are used as the buffer and data, respectively, the buffer write traffic (e.g., random writes) is mapped entirely to the SLC chips and the data write traffic (e.g., sequential writes) is mapped entirely to the MLC chips. However, due to this, SLC chip may reach the end of its lifetime very soon (as shown by the dotted red line) even though MLC chip can sustain more writes.

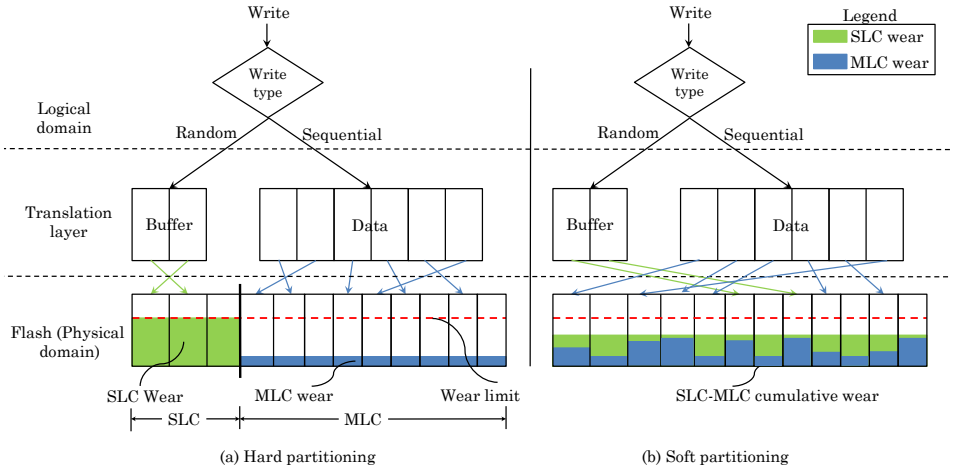



Figure 3. (a) Hard partitioning (SLC and MLC chips are physically separated. Mapping of random writes to SLC can lead to its early wear-out) (b) Soft partitioning (Only MLC blocks are used, some of which can be selectively programmed as SLC to improve performance and achieve wear-leveling) [25]

**Soft partitioning:** Soft partitioning works on the observation that for writing MLC, first LSB and then MSB needs to be programmed and writing MSB takes 4-5× higher latency than writing LSB. However, if only the LSB of MLC is written, the write performance becomes close to that of SLC at the cost of lower capacity [25]. Thus, MLC blocks can be selectively written as SLC blocks which keeps performance close to that of SLC [26]. For example, in Figure 3(b), 12 MLC chips are used. At any time, some of these chips may be programmed in SLC mode and thus, they exhibit SLC properties. At different times, different chips may be used as SLC, but the total number of SLC chips at anytime is fixed. The buffer write traffic is now more uniformly spread and thus, the device lifetime is increased due to wear-leveling. The SLC mode switching is also termed as dynamic write acceleration [27].

Table II compares the properties of these techniques. It is clear that soft partitioning scheme provides higher flexibility and device lifetime. Due to this, soft partitioning based techniques are increasingly being used in real systems [27, 28] compared to the hard partitioning techniques.

**Buffer design techniques:** Flash memory is generally divided into a small log-buffer portion and a large data-portion. Buffer portion handles hot/random writes, which are written-back to data portion at large granularity to reduce the number of write operations. Since hot writes constitute a



DOI: 10.1002/cpe

Table II. A comparative evaluation of hard and soft partitioning techniques

| | |
|---|---|
| Hard | (+) Improves performance and lifetime. Allows separate addressing and management of SLC and MLC chips.<br>(-) Despite higher endurance of SLC, use of it as a buffer may lead to its early wear-out, degrading device lifetime. To avoid this, SLC partition ratio needs to be kept high (e.g., ~10%-30% [7]) which increases SSD cost and requires maintaining huge address translation table. |
| Soft | (+) Allows higher buffer bandwidth since multiple channels and even multiple interleaved chips can be concurrently accessed to write to the buffer. Allows flexible tradeoff between buffer capacity, write performance and device lifetime (since SLC writes are less damaging to the Flash than MLC writes). Faster writes allow the device to spend higher time in a low-power state. Allows handling evictions from buffer on-chip without requiring off-chip migration.<br>(-) Due to different capacity of SLC and MLC, a mode-switch complicates addressing mechanism. Converting large fraction of blocks to SLC reduces device capacity drastically and hence, after a point, the device has to switch back to MLC which reduces speed and bandwidth. Also, MLC-turned SLC region may not be as optimized as an SLC chip. Further, program/erase operations of different modes complicate the design and operation of program-erase controller. |

Table III. A comparative evaluation of different buffer design approaches

| | |
|---|---|
| SLC | (+) Improves performance and lifetime<br>(-) Increases SSD cost and area due to low density of SLC, can degrade SLC lifetime (refer Figure 3). Once SLC wears, future writes are redirected to MLC which harms performance |
| MLC | (+) Improves density and lowers cost<br>(-) Lowers lifetime due to small write endurance of MLC |
| DRAM or SCM | (+) Improves performance and lifetime; bridges speed-gap between memory and storage [29]. Allows performing reads/writes in near-constant time regardless of the location of data inside memory. SCMs do not require erase operations and have much higher endurance than Flash memory [30, 31]. The latency of SCMs is in the range of DRAM latency, whereas their leakage power consumption is near-zero (since only peripheral circuit consumes leakage power).<br>(-) Increases cost; no crash consistency on using DRAM; SCMs are not very mature and not available in large capacity |

large fraction of writes, most writes are directed to buffer partition and hence, intelligent choice of memory technology for designing the buffer is important in hybrid SSDs. Table III summarizes the advantages/disadvantages of different buffer design approaches.

**Mapping techniques:** The hybrid SSD management techniques can also be classified based on the granularity of mapping. *Page-level mapping* allows direct mapping of any logical page to any physical page in the Flash memory. *Block-level mapping* approach stores the logical to physical address translation information at the granularity of each block [18, 32]. Further, multiple *hybrid-level mapping* schemes have been proposed which use some combination of page and block level mapping schemes. For example, when Flash memory is divided into a large data portion and a small log-buffer portion [8, 33, 34], hybrid-level mapping may use block-level mapping for data portion and page-level mapping for log-buffer portion. Random/hot writes to log-buffer benefit from the fine-grained page-level mapping and sequential/cold writes benefit from the coarse-grain block-level



DOI: 10.1002/cpe

Table IV. A comparative evaluation of different mapping schemes

| | |
|---|---|
| Page-level | (+) This provides high performance and lifetime and also facilitates flexible separation of hot and cold data [35, 36].<br>(-) Requires large address translation table. Since this table is stored in SRAM which has poor density, the overall area overhead is increased. GC needs to be performed in background which leads to variable write latencies. |
| Block-level | (+) Reduces the size of translation table<br>(-) Does not allow fine-grain separation of hot/cold data [37]. Also, due to the requirement of maintaining equal offset of logical and physical block, on a write to a page in the block, all valid pages of this block need to be copied to a free block. |
| Hybrid-level | (+) Mitigates 'erase-before-write' issue with lower memory requirement than the page-mapping table [8, 38]<br>(-) Requires maintaining both block and page-level mapping tables, leading to higher storage overhead. Block-merge operation leads to multiple erase and write operations. |
| Page-level and block-level | Its advantages/disadvantages are similar to those of page and block-level schemes discussed above. |

mapping. Finally, some techniques use *page-level and block-level mapping* schemes for SLC and MLC portions, respectively. In case where SLC and MLC are used as log-buffer and data partitions, respectively, this mapping becomes same as the hybrid-level mapping described above. Table IV compares the properties of these techniques. Note that the page-level mapping is the most widely used mapping strategy in the industry, since in practice, other strategies provide small improvement and/or incur large overheads.

## 3. CLASSIFICATION AND OVERVIEW

In this section, we classify the research works based on several parameters.

### *3.1. Based on hybrid SSD and buffer design*

Table V classifies the works based on the Flash cells used in hybrid SSD design and memory/cell used for designing the buffer. Note that the techniques proposed for SLC+MLC hybrid SSD may also be applicable to SLC+TLC SSD, however, in Table V, we mention a technique in the category for which it was originally evaluated. While most techniques use SLC or MLC as buffer, some techniques use other memories such as SCM or DRAM as a buffer.

### *3.2. Based on optimization goal and evaluation platform*

Table VI classifies the works based on optimization metric and it is clear that the techniques proposed are guided by multiple optimization goals which need to be carefully balanced.

The choice of the platform for evaluating hybrid SSD techniques is crucially important. While simulators provide high flexibility to test even those designs which may be currently infeasible to implement, they may be too slow to allow full exploration of the design-space. By comparison, real-systems allow more accurate evaluation and due to their fast speed, they allow executing larger number of instructions. Clearly, both these platforms are vital for evaluating the proposed techniques and provide complementary insights. For this reason, Table VI also classifies the works based on their evaluation platform.




DOI: 10.1002/cpe

Table V. Classification based on hybrid SSD architecture and mapping technique

| Category | References |
|---|---|
| Hybrid SSD architecture | |
| SLC+TLC | [10, 39] |
| MLC+TLC | [24, 40] |
| TLC + (SLC or MLC) | [36] |
| SLC+MLC | Nearly all other works discussed in this survey |
| Mapping technique | |
| Page-level | [35, 36] |
| Block-level | [18, 32] |
| Page-level and block level | [10, 38, 41–44] |
| Hybrid-level | [7, 8, 25, 33, 34] |

Table VI. Classification based on optimization metric and evaluation platform

| Category | References |
|---|---|
| Optimization metric | |
| Performance | [9, 10, 15, 18, 24, 34–36, 38–48] |
| Energy consumption | [7, 18, 40, 42, 47] |
| Lifetime | [7–10, 25, 32–36, 41–43, 45, 48–50] |
| Evaluation platform | |
| Both simulator and real system | [25, 48] |
| Real-system/prototype | [44] |
| Simulator | nearly all others |

### *3.3. Key ideas of different techniques*

We now discuss some key ideas which are used in different techniques discussed in the survey.

1. Most works classify the data or write operations as hot or cold depending on length of write requests or random/sequential nature [8, 10, 15, 18, 24, 34, 35, 38, 41, 45, 47, 51] or access frequency [8, 45] to store them to different memory cells. Also, pages belonging to metadata can be directly marked as hot [15]. Other works classify the data as user data or ECC data [32] and write or read-intensive [39, 46]. Also, data may be stored in SLC or MLC based on performance/deadline [9] and instantaneous wear [18] considerations.

2. Several works discuss strategies to migrate data from SLC to MLC [24, 27, 35, 38, 43, 45, 47, 48], SLC to TLC [28], MLC to TLC [24, 40] or PCM to Flash [15, 41, 42] to utilize the space efficiently and improve performance.

3. Performing data-migration [15, 27, 28, 36, 45] or GC [9] at idle time helps in mitigating its latency. In case of bursty traffic or low free space, data migration [35] or GC [9, 44, 48] may have to be performed on-demand.

4. The data-item not accessed for a period is a candidate for migration [35, 36, 40, 43].

5. Based on the number of valid pages in a block, the migration candidates may be selected [38] or the decision about performing migration within MLC or from MLC to TLC [24] may be made.



DOI: 10.1002/cpe

6. To improve lifetime, some techniques seek to regulate amount of migrated data from SLC to MLC [43, 45] or postpone flushing of data by giving extra chances to cold data to stay in SLC [18, 35, 38].
7. The techniques using SCMs or DRAM as a buffer may flush the data from the buffer to SLC or MLC based on the number of pages to be flushed [42, 51].
8. Some works discuss strategies to deal with block merge operations [8, 34, 38], whereas other works avoid merge operations by using intelligent mapping techniques [35]. Also, some works merge the data in DRAM buffer before flushing it to Flash memory [41, 42].
9. Some works use different mapping schemes in SLC and MLC [8, 18] or for hot and cold data [10].
10. Some techniques schedule different operations on different channels [32, 34, 51] to boost performance.
11. Some works leverage partial programmability of SLC to reduce write overhead [32, 50].

## 4. HYBRID SSD PARTITIONING TECHNIQUES

We now discuss the techniques in terms of their partitioning approach, viz. hard (Section 4.1) and soft (Section 4.2) partitioning. Table VII classifies the works based on their partitioning techniques.

Table VII. Classification based on the partition strategy

| Category | References |
|---|---|
| Hard partitioning | [8, 15, 18, 24, 32, 34, 36, 38, 40–43, 45–47, 49, 51] |
| Soft partitioning | [7, 9, 10, 25, 33, 35, 39, 44, 48, 50] |

### *4.1. Use of hard partitioning*

Nam et al. [8] propose a technique which uses SLC as the log buffer to serve the frequently updated data (refer Figure 4). The sequential and random write operations are served by MLC and SLC, respectively. Also, if the sequential write count is more than a quarter of an MLC block, the data in old data blocks and incoming data is merged into a new data block. Also, during block merge operation, valid pages in victim log block are merged with the associated data blocks and the up-to-date data pages are written to new data blocks. To perform overwrite operation in log area, old data is marked as invalid and new data is written in subsequent free pages of log block. This improves performance by virtue of avoiding 'erase-before-write' requirement. When SLC has no free pages, victim block for erase is selected in round-robin manner to achieve wear-leveling. Their technique is suitable for enterprise database applications and can efficiently support random write operations.

Sung et al. [45] propose a technique which seeks to improve performance and lifetime of hybrid SSD. They divide the logical volume into fixed size portions and record the size of write requests in each portion for a fixed time period. Based on this distribution and latency of each request on SLC/MLC, data migration decision is periodically taken to boost read/write throughput in the next period. Frequent small writes are mapped to SLC portion and less-frequent large writes are mapped to MLC portion. Data migration is performed during idle time, e.g., when battery is being charged, which reduces the impact on performance. Also, the amount of migrated data is controlled to reduce degradation in Flash lifetime. Their technique is implemented in the device driver layer of OS and not in FTL, which is closely bound to the Flash organization. Thus, their technique can work with different file-systems and Flash chips from different vendors.

Oh et al. [36] note that due to poor write performance and small write endurance, TLC-only SSD is ineffective as a general storage device. They propose integrating SLC (or MLC) with TLC. Their




DOI: 10.1002/cpe

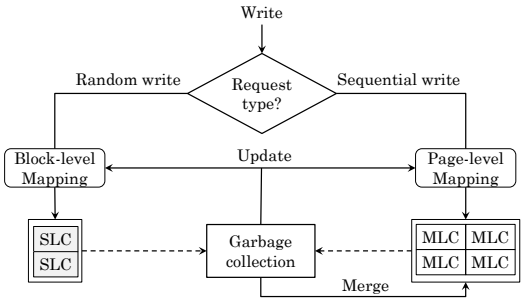



Figure 4. Working of the technique of Nam et al. [8]

technique identifies hot and cold data. Then, based on analytical modeling, they find the fraction of hot data that should be mapped to SLC to boost performance and also balance wear between SLC and TLC. Also, to ensure efficient utilization of SLC, a data-item not accessed for a given time period is migrated to TLC in background. Their technique allows optimizing for performance or lifetime or balancing the two metrics. A limitation of their technique is that the analytical model cannot exactly account for time-varying application characteristics and needs to make unrealistic assumptions such as perfect wear-leveling.

### *4.2. Use of soft partitioning*

Im et al. [35] present a technique which works by detecting the data hotness based on the size of the write. Small writes are mapped to SLC and sequential large writes are mapped to MLC (refer Figure 5). In case SLC runs out of free pages, invalid pages are reclaimed or cold data is moved to MLC. They note that block-level and hybrid mapping involve costly copy operations for merging and hence, these mapping approaches are not suitable for MLC. Further, due to large MLC capacity, page-level mapping necessitates large mapping tables. Hence, they use unit-based page-level mapping [52] which does not require merge operations. In this mapping, each unit has multiple sequential logical blocks and multiple physical blocks can be allocated to every unit. A page can be written at any physical page inside the physical blocks assigned to this unit. This reduces the mapping table size since the page-level mapping record is required only within the unit boundary. The unit size is chosen at design time based on the SLC size.

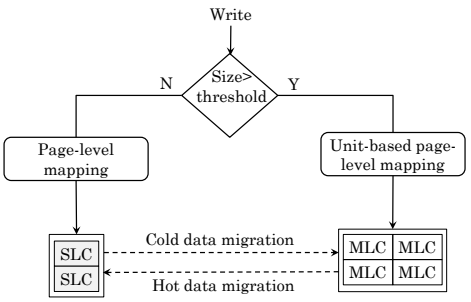



Figure 5. Working of the technique of Im et al. [35]

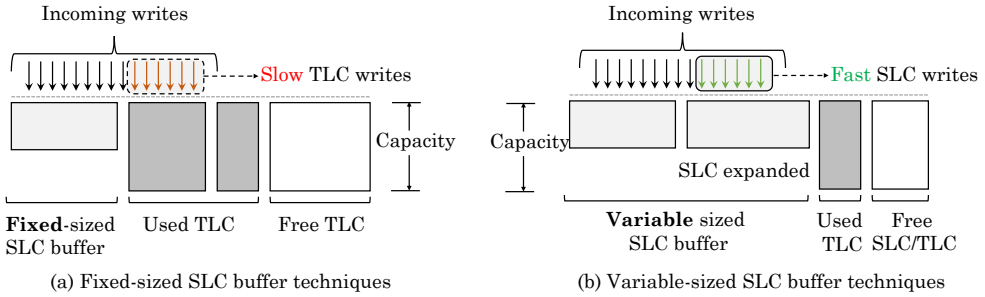
They divide the SLC into a hot and a warm partition. A write request first arrives in the hot partition and after GC in this partition, moves to the warm partition. If warm data sees another write request, the data in warm partition is invalidated and new data is stored in the hot partition. When the warm partition runs out of space, it gives each partition $N$-chances before it is evicted to MLC. By suitably choosing the size of hot and warm lists, the residency of data in SLC can be controlled. The value of $N$ is adjusted based on the update ratio in the warm partition. Their technique provides significant improvement in lifetime and write performance.



DOI: 10.1002/cpe

Jimenez et al. [25] note that in hard partitioning, the SLC-based buffer can fail much sooner than the MLC. They propose selectively writing MLC Flash as an SLC Flash which keeps performance close to that SLC. Their technique chooses between writing in SLC or MLC mode based on the total wear. When a block assigned to the buffer sees more writes than a data block, the blocks are swapped. Thus, the physical location of buffer can be migrated across the device which balances the global wear. Their technique can work with different FTLs and mapping schemes. Their technique provides flexibility to balance wear across the device, provides near-SLC performance and better-than-MLC lifetime with no additional cost and only small reduction in density.

Yang et al. [39] note that TLC blocks can be programmed as SLC blocks which allows exercising tradeoff between capacity and performance based on the access pattern and instantaneous utilization. Since SSD performance is determined by the latency of hot data, by transforming maximum-possible free blocks into SLC, hot data can be served from SLC to boost performance (refer Figure Figure 6(b)) compared to the fixed-size SLC buffer approach (refer Figure 6(a)). In their design, logical address range of a logical block equals the size of TLC block and that of a logical page equals the size of SLC/TLC page. Data of a logical block can be stored as either one TLC block or three SLC blocks. For each logical block, the mapping from logical to physical pages is stored in a page-level mapping table.




(a) Fixed-sized SLC buffer techniques (b) Variable-sized SLC buffer techniques

Figure 6. Fixed-sized SLC buffer versus variable-sized SLC buffer approach ([39])

Since TLC write and read latency values are 8× and 3× (respectively) that of SLC values, they propose giving higher priority to write-intensive blocks than to read-intensive blocks (refer Figure 7(b)). On a write, a block's priority is changed to 'high'. On a read to a 'low' priority block, its priority is changed to 'medium'. When a logical block is scanned by GC and no more block can be reclaimed from this logical block, the priority is reduced from high to medium or medium to low. Thus, in due time, cold blocks get low priority. This allows migrating cold data to TLC and freeing up the space to store higher priority blocks as SLC. Data with higher and lower priorities are stored in SLC and TLC, respectively (refer Figure 7(a)).

On a write request to a logical block, if the number of free SLC pages in this block is sufficient, the write is performed in this block. Otherwise, GC is performed based on the level of storage utilization. In case of low storage utilization, GC is performed within SLC to boost performance at the cost of storage capacity. However, in case of high storage utilization, SLC data are packed and migrated to TLC to increase the overall capacity. TLC is written only through this 'TLC-pack' GC operation which avoids any non-sequential writes violating TLC write constraints. These GC operations are illustrated in Figure 7(c). Overall, their technique improves write performance significantly.

Chang et al. [9] present a technique which selectively performs SLC writes in MLC blocks to meet workload SLOs without requiring over-provisioning and also reducing SLC write counts. It is noteworthy that meeting SLO is different from simply boosting performance and requires more careful management. Also, SLC/MLC mode selection for a request impacts both its own latency and the latency of upcoming requests. Their scheduling algorithm works on the observation that if a schedule is viable with $k$ SLC writes for some write accesses, it remains viable on serving exactly first $k$ requests in the SLC mode. Incoming reads/writes are inserted in respective queues in order of their deadlines, which is its arrival time plus the target latency. By default, writes are performed in



DOI: 10.1002/cpe

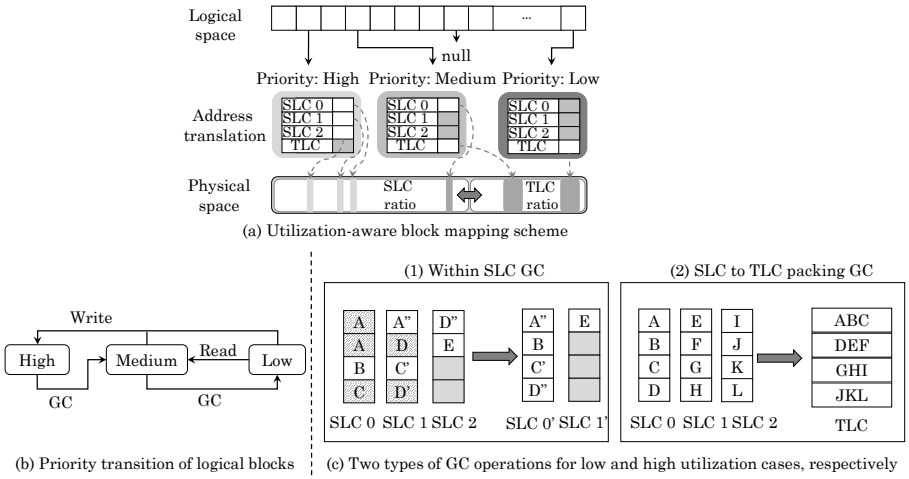



Figure 7. Illustration of (a) block-mapping, (b) priority transition and (c) GC operations in the technique of Yang et al. [39]

MLC mode, but if any queued request will fail to meet its deadline, an incoming write is performed in the SLC mode. Further, if actual latencies fail to meet the SLO requirements, their technique decreases the target latency and/or increases SLC-mode writes for meeting SLO requirements. In their technique, read requests are prioritized unless write queue saturates since reads have higher impact on user-experience and they are served by the "earliest-deadline first" scheme.

To avoid the need of on-demand GC, their technique performs GC when the storage is idle and remaining free block count falls below a threshold. This threshold is updated based on SLC and MLC write counts to balance capacity and performance. In case of bursty writes or low remaining space due to aggressive use of SLC-mode writes, GC is performed in on-demand manner. GC scans both SLC and MLC blocks and chooses those with the lowest amount of live data as victims. Their technique can achieve or approach the SLO target without requiring over-provisioning, whereas MLC-only device is unable to meet this target. Also, the increase in erase operations due to their technique is small.

Lee et al. [48] propose a soft-partitioning technique which seeks to provide performance of SLC with capacity of MLC. When a write arrives, their technique chooses SLC/MLC mode for programming with a view to balance the performance and lifetime. To boost performance, their technique writes maximum possible data in SLC-mode, however, excessive use of SLC can degrade capacity and lifetime. To address this, their technique moves valid pages in SLC blocks to MLC blocks to create free space and allows controlling the fraction of writes performed in SLC-mode. For a write request, first the SLC/MLC mode is chosen and data is temporarily buffered in SLC or MLC write buffer. Then, the write is performed similar to that in log-structured file systems [53]. On eviction from write buffer, a data-item is written to log block of SLC or MLC region in that mode. As shown in Figure 8, if there are two SLC blocks and one free block, then the valid pages of both SLC blocks can be copied to the free block with MLC-mode programming. Then, both SLC blocks can be erased, which provides two free blocks. Free space reclamation is invoked only when the available free space falls below a threshold. This approach reduces data migration compared to early reclamation during idle time and reduces performance penalty compared to on-demand reclamation. A limitation of their technique is that it uses SLC for storing both hot and cold data. Also, it assumes the wear of SLC-mode to be equal to that of MLC-mode which prevents fully leveraging Flash memory endurance.

Zhang et al. [50] exploit PPP feature of SLC to reduce write overhead by implementing in-place delta compression. They note that since a given location is frequently written in a short interval (e.g., metadata updates, revision of file content etc.), writes to the buffer have high temporal redundancy. This allows using delta compression to reduce the write overhead to the buffer. Previous techniques



*Prepared using* cpeauth.cls
DOI: 10.1002/cpe

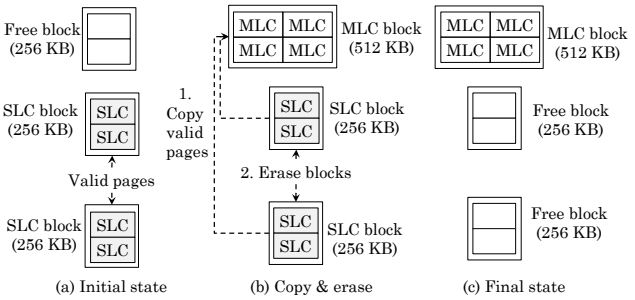



Figure 8. Free-space reclamation in the technique of Lee et al. [48]

store the base (original data) and subsequent deltas in different physical pages, which requires maintaining a large amount of mapping information. Also, a read access needs to now fetch the base and deltas from different pages which leads to read amplification and read latency penalty. Assuming a 4KB sector and 16KB page, their technique writes a sector to an SLC page in compressed form. Since per-sector compression leaves some cells in the page unutilized, they use PPP feature of SLC to use these cells for later storing the deltas. Thus, the base and deltas are stored in the same SLC page. If a page is filled after certain number of updates or the number of deltas reach a threshold, a new physical page is allocated, the latest version data is written to a new page and delta compression is reset for future udpates. This ensures that only one page needs to be accessed for reading a data-item. They also propose a hybrid ECC for dealing with the different sizes of base and deltas. They show that their technique reduces write traffic to SLC pages with negligible latency and area overhead.

## 5. BUFFER DESIGN AND MANAGEMENT TECHNIQUES

The use of a particular memory technology or cell design as a buffer has a significant impact on the overall performance, energy efficiency and cost of the hybrid SSD. For this reason, we now review hybrid SSD management techniques in term of their buffer architecture and management policies (refer Table VIII).

### *5.1. Using SLC as a buffer*

Several techniques use SLC as a buffer to serve write operations which improves performance and also provides higher device lifetime. However, since SLC has lower density, the MLC/TLC pool is used for storing remaining (e.g., cold) working set size, which improves capacity and lowers the cost.

Jung et al. [34] propose a technique in which many SLC chips are used as non-volatile write buffer. Their technique designates one SLC chip to store data from the host till it gets full, which is termed as “foreground operation”. The remaining SLC chips perform merge and garbage-collection operation with MLC chips, termed as “background operation”. To improve the efficiency of buffering, the blocks within SLC are divided into three partitions: sequential log blocks, random log blocks, and data blocks. When the log block partition in foreground SLC is fully consumed, another SLC chip is selected for foreground operation and this SLC is now used for background operation. Background and foreground operations happen on different channels which avoids conflicts. Their mapping technique is built-on the FAST (fully associative sector translation) technique [54] and seeks to allow SLC to handle both random and sequential write access operations.

Im et al. [38] propose a technique which uses SLC as log buffer with page-mapping scheme and MLC as data block with block-mapping scheme. Use of block-level mapping in MLC helps in reducing the size of mapping table since the MLC partition has larger number of pages than the SLC partition. When SLC has no empty space, GC migrates cold data to MLC or reclaims the space



DOI: 10.1002/cpe

Table VIII. Classification based on the buffer architecture

| Category | References |
|---|---|
| SLC Flash | [7, 8, 18, 25, 32–35, 38, 39, 50] |
| MLC Flash | [40] |
| DRAM | [41, 49] |
| SCM | [15, 24, 42, 47] |
| Both DRAM and SCM | [46] |

from invalid pages. Hot pages are kept in SLC area. To minimize migration cost, the pages selected for migration are chosen based on the number of valid pages in corresponding MLC data block with which the migrated pages are merged.

Based on the number of updates to a page after last GC operation, each page is classified into hot, warm and cold. Then, based on the number of pages in a data block whose corresponding page in log buffer is hot/cold/warm, each data block is also classified as hot/warm/cold. Based on this, page-level merge is performed only with cold and warm blocks. Then, log pages whose linked data blocks are cold, are moved to data blocks. For pages in log buffer whose connected warm data blocks have more than a threshold valid pages, log blocks are not merged with data blocks to postpone MLC page copy operations. Further, if a log block has less than a threshold number of valid pages, then, its valid pages are moved to another log block. Finally, blocks with no valid pages are erased. Sequential large chunks of data are bypassed from SLC since they are usually written only once. They show that their technique manages SLC area efficiently and improves performance significantly. The limitation of their technique is the use of a special 'fusion Flash memory' which has lower performance than general SLC and MLC chips [8].

### *5.2. Using MLC as a buffer*

Hachiya et al. [40] design MLC/TLC Flash based hybrid SSD and use MLC Flash as a buffer [40]. Their technique stores hot data in MLC pool to achieve high performance. To ensure efficient utilization of MLC pool, their technique selects extremely cold data and migrates it to TLC pool. Since TLC has limited write endurance and high write latency, storing rarely accessed data in TLC leads to significant reduction in accesses to TLC which improves its lifetime. Compared to MLC-only SSD, their technique provides improvement in performance, energy and cost efficiency.

### *5.3. Using SCMs or DRAM as a buffer*

Given the low capacity of SLC Flash and low endurance and high latency of MLC Flash, use of them as a buffer presents challenges. To address these limitations, some works propose using non-volatile SCMs (e.g., PCM, ReRAM) or volatile memories (e.g., DRAM) as a buffer. We now discuss several such works.

Yim et al. [15] propose using NVRAM (e.g., PCM) as a write buffer in an SLC/MLC hybrid SSD. Their technique demarcates write buffer into three portions. The first portion is used for serving small writes (e.g., 4KB writes for file system metadata) which reduces the host delay. These data-items are copied to Flash memory when it is lying idle. The second portion is used for reducing the number of writes performed by Flash (refer Figure 9).

Their technique uses hash tables stored in volatile buffer to test whether the requested page is present in the buffer. If so, the page is directly updated in the buffer. Otherwise, if the page had a write in the near past (e.g., last $K$ writes), the page is considered to be hot, otherwise, it is marked as cold. Also, based on the type of file system (e.g., FFS (fast file system) or FAT (file allocation table)), pages belonging to metadata can be explicitly marked as hot. Then, only hot pages are stored in the write buffer. For a write directed to a page stored in buffer, the page is directly updated which reduces the number of writes to Flash. A page evicted from buffer is stored in SLC, whereas all



DOI: 10.1002/cpe

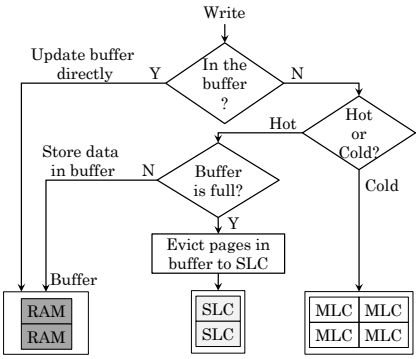



Figure 9. Working of the technique of Yim et al. [15]

cold pages are stored in MLC. The third portion of buffer facilitates byte-granularity reads to enable 'execute-in-place' capability. On a read access, if the page is found in the buffer, the data is provided to host with low latency, otherwise, the page is loaded from Flash to the buffer and also delivered to the host. Overall, their technique achieves write performance of SLC with density and cost of MLC memory.

Park et al. [41] present a technique which uses chained-block (CB) to design hybrid SLC-MLC SSD. Their technique stores the chained-block mapping table, page mapping table, file system meta-data, and user data with frequent short writes in SLC. Long writes are stored in MLC. Both SLC and MLC store data as separate chained-blocks but have similar organization (refer Figure 10).

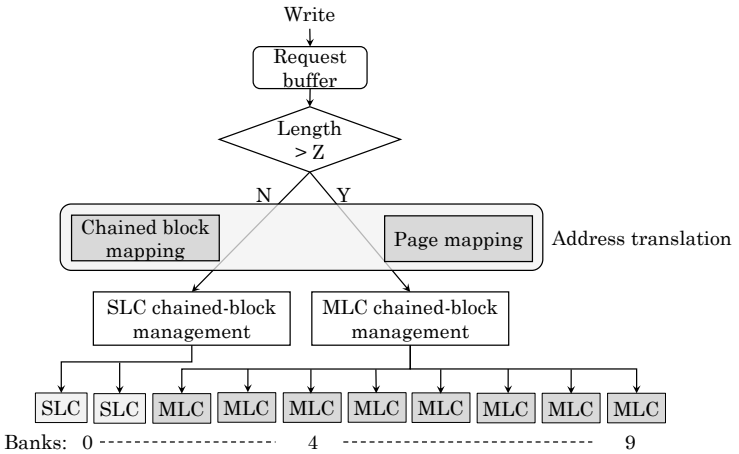



Figure 10. Working of the technique of Park et al. [41] (Z is a threshold)

Number of physical blocks in SLC chained-block is same as the number of SLC chips and each of them resides in a different SLC chip. MLC chained-block length is higher than that of SLC chained-block, which compensates for its higher latency. After one write, if an associated write comes within a fixed time window, these writes are bundled into one bulk-write in the write buffer (which is stored in a RAM, e.g., DRAM). When the time window is over, or a read comes or the buffer saturates, data in the buffer is stored in either an SLC or an MLC chained-block depending on length of the bundled write.

They use two kinds of mapping tables: (1) a CB mapping table which provides physical block address inside a logical CB and (2) a page mapping table which provides address of the requested page inside the logical CB. Finally, Flash is accessed using chip-ID, physical block address and physical page address. When a logical CB is updated, an SLC or MLC CB is allocated to the logical CB, depending on whether the write request is smaller or larger than a threshold. SLC CB waits for



DOI: 10.1002/cpe

more writes for a certain time, after which the data is moved to MLC CB. Figure 11 summarizes their address translation approach. While using 80% MLC chips, they achieve write performance close to that of the SLC chips. The limitation of their technique is the use of complex mapping scheme and use of two types of Flash memory as the update area which degrades performance. Also, since the lifetime is limited by the portion with smallest lifetime, they use 10-30% SLC portion to increase its lifetime above MLC portion which increases SSD cost and the size of address translation table.

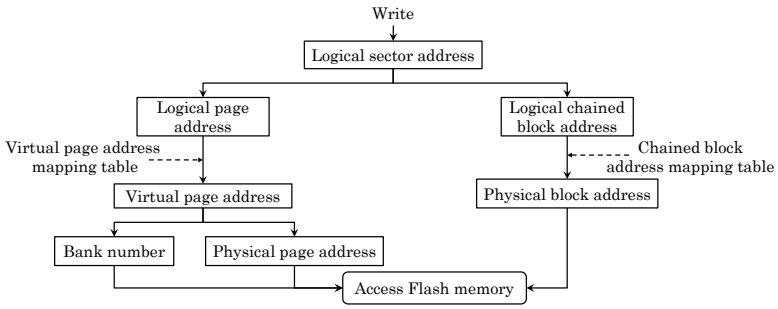



Figure 11. Address translation process of Park et al. [41] technique

Lu et al. [42] propose a hybrid SSD design which uses PCM and SLC as primary and secondary update areas, respectively and MLC as the main data storage (refer Figure 12). They use PCM for buffering and updating the write data. SLC and MLC are organized as superblock designs [41] and their page size needs to be equal. The technique of Park et al. [41] uses one MLC superblock for data, one MLC superblock for sequential updating and multiple SLC superblocks for random updating. In the technique of Lu et al. [42], an LSBA can have only one MLC superblock as data area and any number of SLC superblocks for random updating and thus, their technique does not use MLC superblock(s) as the update area.

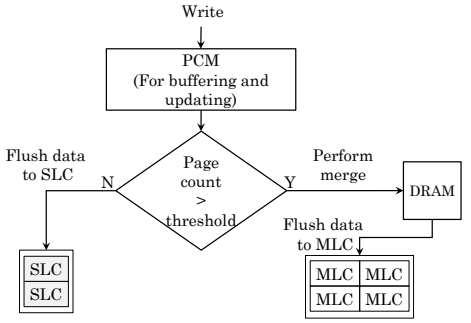



Figure 12. Working of the technique of Lu et al. [42]

If PCM has free space to store incoming write data, the requests are packed with the existing buffered data into multiple page-clusters. Otherwise, single or multiple page-cluster(s) are flushed to hybrid Flash to create space in PCM. Depending on whether the number of pages to flush is more or less than the threshold, the data is flushed to MLC or SLC, respectively. For flushing to MLC, a DRAM buffer of the size of one MLC superblock size is used. The flushed data and valid pages from corresponding LSBA of Flash are all stored in the DRAM buffer (refer Figure 13(a)). In DRAM buffer, the pages are sorted based on LPN and written to a free MLC superblock (refer Figure 13(b)). Now, this becomes LSBA's data area and the old superblock is erased. Thus, MLC superblock is not used as the update area which reduces the amount of mapping information. The DRAM buffer is turned-off when idle.

In case no SLC superblock is free, GC is performed and a victim is selected in the following order. (1) If an LSBA is found such that the number of used pages in all SLC superblocks in the LSBA update area exceeds the pages of single MLC superblock in LSBA data area, then at least one SLC



DOI: 10.1002/cpe

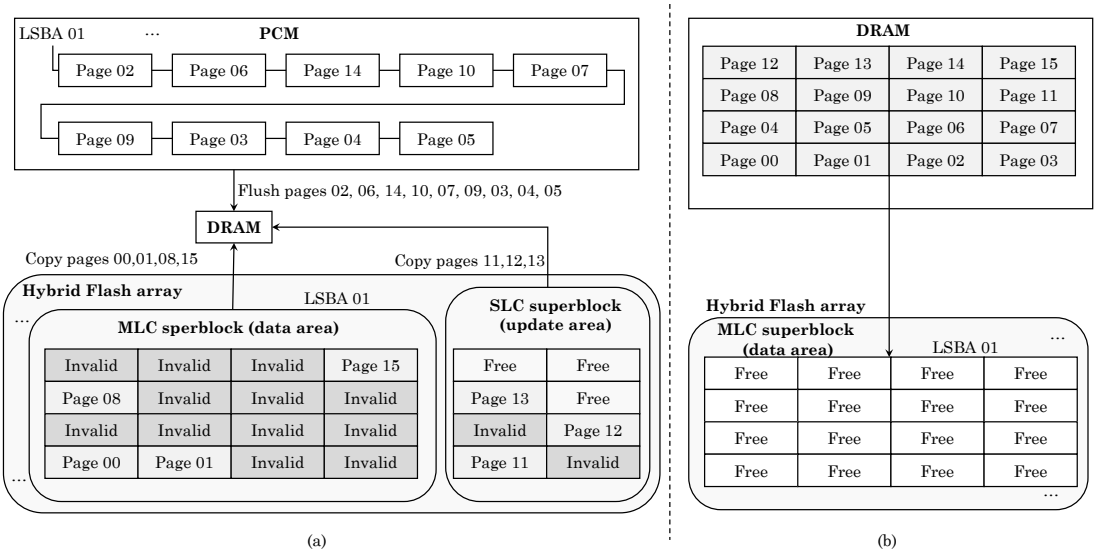



Figure 13. Illustration of flush operation in technique of Lu et al. [42] (a) Valid pages are gathered from PCM, data area, and update area for flushing to DRAM. (b) The pages stored in DRAM are sorted and transferred into data area and the original updated data area is erased

superblock can be reclaimed after merging SLC superblocks. (2) Otherwise, an LSBA is searched which has SLC superblocks with more invalid pages than the total pages in one SLC superblock. (3) If no such LSBA is found, it implies that SLC superblocks are uniformly allocated in LSBAs. Hence, an SLC superblock (update area) is selected and merged with an MLC superblock (data area). Then, valid pages are written to the newly allocated empty MLC superblock and reclaimed SLC superblock(s) are erased. Overall, their hierarchical update approach improves lifetime and write performance of SSD significantly.

Park et al. [46] note that since the data stored in volatile-RAM (VRAM) buffer may be lost on a power failure or system crash, most systems use `pdflush` mechanism to periodically (e.g., 30 seconds in Linux) write updated data to storage, even when the buffer has free space. This, however, contributes ~80% of the writes to Flash memory and thus, increases the Flash writes significantly. They propose designing the buffer cache with both VRAM (e.g., DRAM) and NVRAM (PCM or STT-RAM), as shown in Figure 14. On a read to a block, the data is cached in the VRAM buffer which provides low-latency for read-intensive data. On a write due to I/O or `pdflush` operation, the block is moved to the NVRAM buffer. This ensures data consistency without requiring writes to the Flash memory. Writes to Flash happen only when a victim is evicted from the NVRAM buffer.

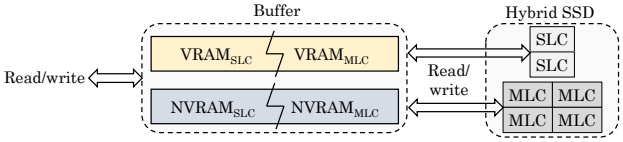



Figure 14. Working of the technique of Park et al. [46]

Their technique divides both VRAM and NVRAM buffers into SLC and MLC portions. For each of the four portions ($VRAM_{SLC}$, $VRAM_{MLC}$, $NVRAM_{SLC}$ and $NVRAM_{MLC}$), the metadata of recently evicted blocks are tracked. By seeing hits to them and based on the read/write latency to SLC and MLC, the size of four portions is adjusted. The sizes are enforced at the time of buffer replacement. On a miss in the buffer, their technique checks if a free block is present in the VRAM (or NVRAM) on a read (or write) reference. If no, a block is evicted from VRAM (or NVRAM) and the requested block is stored in place of free block. Within VRAM, the victim is chosen from $VRAM_{SLC}$ or $VRAM_{MLC}$ depending on which portion currently exceeds its quota.



DOI: 10.1002/cpe

Similar procedure is applicable to NVRAM. Blocks evicted from NVRAM are flushed to Flash. By assigning the cache space based on the I/O behavior, access pattern of buffered blocks and memory characteristics, their technique provides large improvement in I/O performance. Also, by virtue of using NVRAM in the buffer cache, it promises high reliability of file data.

Cho et al. [51] propose a technique which seeks to maximize parallel execution of sequential write requests since random request do not benefit from parallelism. Their technique uses an adapted version of hybrid super-block mapping where a hybrid mapping approach is used along with super-block design. In this approach, mapping of logical to physical address is done using super-blocks. Then, the VPN table for each physical super-block address is accessed for tagging the logical page number. Thus, there is mapping between LSBAs to PSBAs. VPN maps LPN in PSBA and provides the channel and bank IDs. This allows parallel writing of pages in an LSBA. Their technique buffers the data in LSBA and flushes them to Flash chips for maximizing interleaving.

As shown in Figure 15, if the requested data is already present in the buffer, the requested data is updated. Otherwise, if the buffer has sufficient free space, the incoming data is pushed to the top slot. However, if the buffer does not have sufficient space, a victim block is found. Victim entry is selected using LRU policy and to account for the spatial locality, number of update pages in a superblock are also considered. If MLC channel is busy due to merge operations, a random entry is selected as a victim, otherwise, a sequential entry is selected. Random and sequential data are flushed to SLC and MLC, respectively and the threshold for determining the random or sequential write is selected based on the number of MLC and SLC banks. Their technique reduces the number of erase operations and also improves write performance.

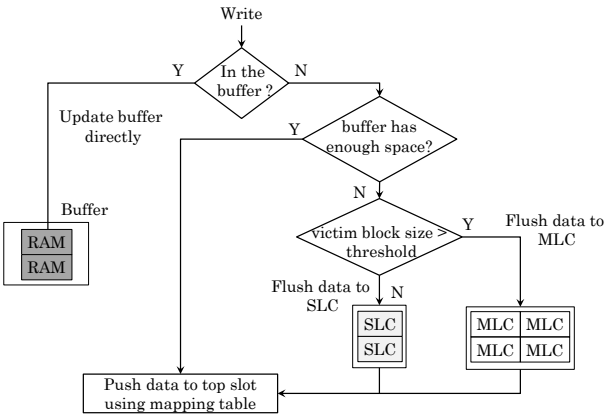



Figure 15. Working of the technique of Cho et al. [51]

Sun et al. [47] note that although NVRAMs provide higher performance and endurance than Flash, they also have higher cost and hence, the amount of NVRAM used in a hybrid SSD needs to be carefully chosen based on several factors, e.g., area, cost, latency and application behavior. By analyzing these parameters, they observe that to achieve high SSD performance, read/write latencies of NVRAMs need to be below $1\mu$s. Also, write latency has higher impact on SSD energy than read latency. They study the impact of different NVRAM area cost models (viz., optimistic and pessimistic), NVRAM:Flash capacity ratio, NVRAM read:write latency ratio, etc. and also study strategies for mapping hot/cold and sequential/random data to NVRAM or Flash.

Kwon et al. [49] propose a scheme which seeks to improve SSD performance and balance the wear of SLC and MLC Flash. Their scheme identifies hot data based on two factors: frequent updates to a data-item and irregular allocation. Irregular allocation refers to the fact that the LPA of the small hot data is not sequential to the previous LPA. Hot and cold data are stored in SLC and MLC registers, respectively from where they are transferred to SLC and MLC chips, respectively. Writing and updating of data in the registers is performed similar to previous techniques [55, 56]. DRAM is used for data that cannot be identified as cold or hot. If the write request does not update existing data and is not sequential, LSN of write request is stored in DRAM buffer if DRAM buffer



DOI: 10.1002/cpe

contains less than a threshold number of LSNs. However, when DRAM buffer contains more than a threshold number of LSNs, their scheme checks if the LSN of the write request is in sequence with the LSNs stored in the DRAM buffer. If so, they are transferred to the MLC registers to be written to MLC chips. Otherwise, the data is stored in DRAM since its sequential/random nature is still not known. Overall, their technique reduces total write and erase operations and also achieves wear-leveling.

Matsui et al. [24] propose a technique which uses SCM-based buffer for a MLC/TLC-based hybrid SSD (refer Figure 16). They classify the data as hot, cold and frozen, which refers to frequently, infrequently and rarely accessed (respectively) data. Then, hot, cold and frozen data are stored in SCM, MLC and TLC, respectively. Further, based on whether the length of a write request is greater than or lower than a threshold (8KB), the data are classified as random or sequential, respectively. Then, random data is stored in SCM whereas sequential data is stored in MLC. Furthermore, during GC in MLC, the oldest block is selected for reclamation. If the block has many valid pages, these data are considered frozen and they are migrated to TLC. However, if the block does not have many valid pages, its valid pages are moved to other blocks in MLC itself and the block is erased. Thus, data are not directly written to TLC. Their design improves performance for wide variety of applications with minimal increase in SSD cost.

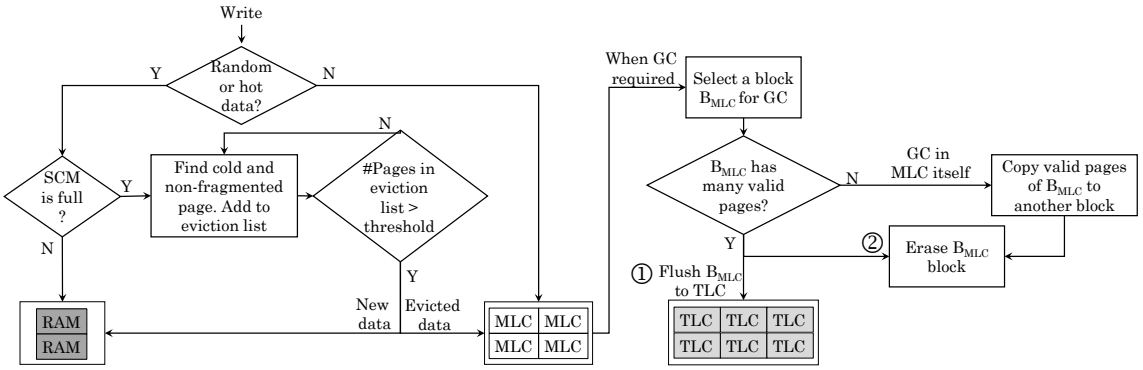



Figure 16. Working of the technique of Matsui et al. [24] (RAM refers to an SCM, e.g., ReRAM)

## 6. OPTIMIZATION OBJECTIVES AND ALGORITHMS

In this section, we discuss SSD management techniques in terms of their optimization metric and solution algorithms. Several hybrid SSD management techniques use optimization/solution heuristics such as k-means clustering [18], genetic algorithm [10], integral controller [43] and analytical models [10, 36, 44].

### *6.1. Techniques for managing failed blocks*

Jimenez et al. [33] note that in traditional MLC Flash, a block is discarded as soon as its error rate exceeds the correction capability of ECC. They propose 'revitalizing' this block by using it as an SLC block which boosts lifetime at the cost of capacity. Figure 17 shows the working of the baseline and the technique proposed by Jimenez et al. [33] over their lifetime. In both cases, initially four portions are allocated: data portion, buffer portion (organized as SLC-mode), free blocks and bad blocks. Initially, bad block portion is empty. In the baseline design, over the lifetime, the blocks which become faulty are moved to the bad block portion and the free block portion gradually becomes empty. This is shown by the 'intermediate state' on the left side in Figure 17. When the SSD has no free blocks, it is assumed to reach its end, as shown by the 'final state' on the left side in Figure 17.



DOI: 10.1002/cpe

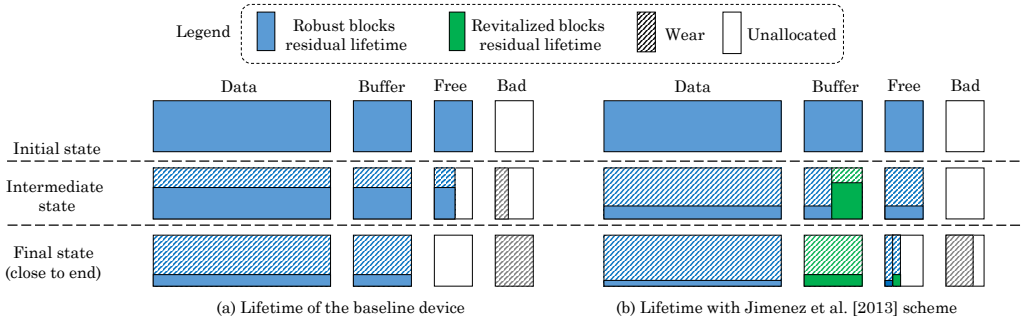



Figure 17. Jimenez et al. [33] technique versus the baseline architecture

In their proposed technique shown on the right side of Figure 17, the buffer and the free block portions can have both revitalized and robust blocks, however, the data portion can have only robust blocks. Revitalized blocks are kept in free block portion and completely failed blocks are directly migrated to the bad block portion. As shown in the right side of Figure 17, over time, robust blocks get replaced by revitalized blocks in the buffer. While allocating a block from free set, the buffer preferentially allocates revitalized blocks, which reduces the pressure on robust blocks. Thus, as long as enough robust blocks are available for the data portion, the SSD stays alive which increases the overall lifetime, as shown in bottom right side of Figure 17.

Since buffer partition sees higher write intensity than data partition, revitalized blocks help in reducing write pressure on free blocks and the remaining robust blocks. Management of revitalized blocks is performed similar to those of bad blocks and they are distinguished from robust blocks by use of a flag. Buffer portion preferentially allocates revitalized blocks instead of robust blocks to reduce pressure on them. Their technique increases device lifetime without harming performance or requiring additional storage.

It is noteworthy that the technique of Jimenez et al. [33] converts MLC blocks into SLC when they exhaust their lifetime to benefit from the *high endurance* of SLC blocks. By comparison, other soft partitioning techniques perform mode-transition even when MLC blocks are healthy to benefit from the *low latency/energy* of SLC blocks.

### *6.2. Techniques for managing ECC*

Hsiehv et al. [32] propose storing ECC in data area of SLC (2KB) instead of spare area of MLC (128B), which allows using stronger ECC, unconstrained by spare area size limitations. Since MLC does not allow partial-page programming (PPP), storing ECC in MLC data area is not effective, since ECC size is much lower than that of a page. With PPP, every SLC page can be written four times as 1/4 page size chunks each time, and thus, ECC can be stored in SLC without requiring erase operations. Also, higher write endurance and lower error rate of SLC ensures higher reliability of ECC.

Multiple contiguous SLC pages make an ECC area (refer Figure 18). Also, multiple MLC pages (e.g., one MLC block) form one mapping unit and the ECCs for all pages in a unit are stored in corresponding ECC area for saving space since the ECC size is much lower than the data area size. Since multiple writes are required for filling an MLC block and completing the ECC area, they use SLC itself as a buffer area for temporarily storing ECCs of previously written MLC pages before committing entire ECC area to SLC. As MLC blocks only support sequential programming, when the last page of an MLC block is programmed, all ECCs in buffer area can be committed to an ECC area. After updating ECC mapping table, buffer area is reclaimed.

They use adaptive ECC scheme which provides BCH (Bose-Chaudhuri-Hocquenghem) codes with 9, 14, 19 and 24 bits correction capability for every 512 bytes. Initially, 9-bit ECC is allocated to each page in an MLC block. Since different cells in a page see different write patterns, they show different error rates. Thus, depending on error rate or amount of wear of an MLC page, stronger ECC is allocated to it. They assume a multi-channel design where ECC and user data are simultaneously



DOI: 10.1002/cpe

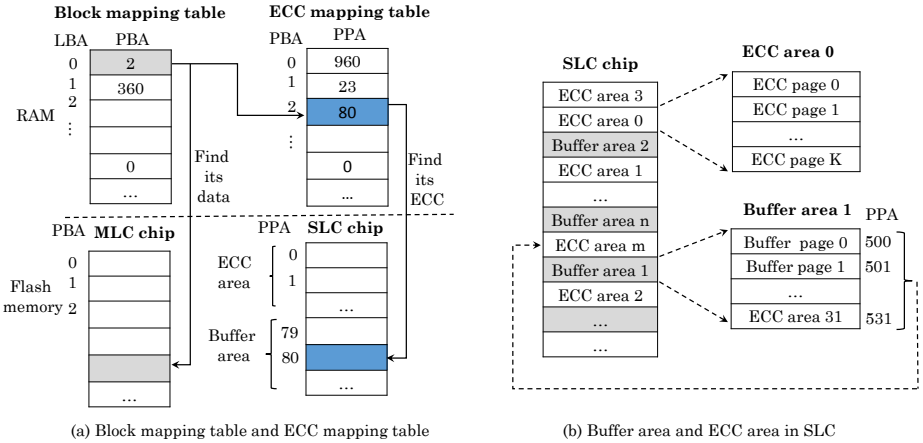



Figure 18. Management of block mapping table, ECC mapping table, and buffer area in the technique of Hsiehv et al. [32]. (a) Both the MLC chip and the ECC mapping table are indexed by PBA. Both the block mapping table and the ECC mapping table are maintained in RAM. (b) Buffer areas and ECC areas are physically interchangeable in SLC. However, buffer/ ECC pages within the buffer/ECC areas must be physically contiguous

written to SLC and MLC through different channels. They show that with a 32GB MLC-based SSD and 1GB SLC, their technique improves SSD lifetime significantly. It is noteworthy that "low-density parity-check" (LDPC) codes are used much more widely in state-of-the-art SSD controllers than the BCH codes.

### *6.3. Use of k-means clustering*

Chang et al. [18] note that small writes occur frequently and are directed to a small amount of data. Non-hot data are referenced infrequently by large writes. Further, data are either very small or very large and not medium-sized. Based on this, they use $k$-means ($k = 2$) clustering to find two peak frequencies in request size distribution. This provides the threshold of small writes and all such small and hot writes are directed to SLC. They manage SLC as a circular list where new data arrives at the head pointer location and free space is reclaimed from the tail pointer location since its update frequency is lower (refer Figure 19). In case of no space, head pointer moves to the next block. Due to the out-of-place writes, recent data appears close to the head pointer. When the head pointer moves ahead of the tail pointer by more than certain blocks, the tail block is reclaimed and the tail pointer advances. This organization reduces copy operations during GC and also ensures uniform wear across SLC blocks.

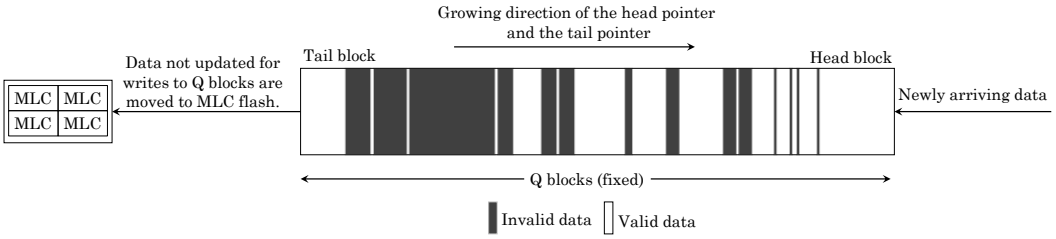



Figure 19. Use of SLC flash as a circular log space in the technique of Chang et al. [18]

SLC is managed using page-level mapping and the mapping information is stored in a hashed table. MLC is managed with a two-level mapping scheme. In level-1 mapping, logical blocks are mapped one-to-one to physical blocks. Such physical blocks are termed 'data block'. Since in-place writes are forbidden, a spare physical block with all free space is allocated on first sector write to a logical block. New data are written to the first page of the spare block and the next sector write



DOI: 10.1002/cpe

data are appended to its free space. This block is termed a log block and in case of no remaining free space, another log block is allocated. The ordered list of one data block and multiple log blocks make a block chain. In level-2 mapping, disk sectors of a logical block are mapped to MLC pages in associated block chain.

When SLC wear exceeds MLC wear, SLC accepts only very hot data or updates to existing data; remaining data are directed to MLC (refer Figure 20). To account for the changing working set, only the metadata of non-update write request is recorded in SLC. If the write-request appears again, then the data is stored in SLC, otherwise, the metadata is removed when the corresponding block is erased.

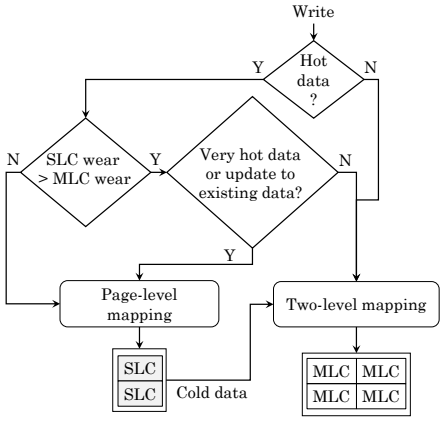



Figure 20. Working of the technique of Chang et al. [18]

During GC, before moving a valid sector from SLC to MLC, the MLC checks if its corresponding logical block has been allocated to any of the log blocks. If yes, the sector is written to the MLC flash, otherwise, the sector is copied to the head block of SLC. By virtue of postponing writing of random data to MLC, this approach reduces GC overhead in MLC, although it also reduces SLC capacity available for storing the hot data. To compensate this, during GC in MLC, valid data are collected from both log blocks and SLC. They show that using 256MB SLC with 20GB MLC provides nearly double the performance compared to an MLC-only SSD while also saving energy.

### *6.4. Use of genetic algorithm*

Liu et al. [10] propose a technique for improving lifetime and performance of SLC/TLC based SSD. In their technique, TLC stores user data whereas mapping tables are stored in SLC since they have high access frequency (refer Figure 21). A write request with lower or higher than a threshold size is considered random or sequential, respectively. Sequential writes are further divided into hot and cold depending on the access frequency. Random and hot sequential writes are stored with page-level mapping and the cold sequential writes are stored with block-level mapping.

They allocate the number of SLC blocks based on the relative instantaneous erase counts of SLC and TLC. Since this simple mathematical model may not accurately capture behavior of different applications, they also use a genetic algorithm to obtain a heuristic solution. The population for the algorithm is formed by choosing different combinations of three parameters, viz., hot data threshold, random data threshold and SLC-mode block ratio. The fitness functions are (1) ratio of erase count between SLC and TLC and (2) ratio of effective capacity of hybrid Flash to that of TLC-only Flash. Multiple generations are simulated until the best-fit solution is found. Their technique brings large improvement in lifetime and performance. Also, the genetic algorithm provides higher lifetime than the mathematical-formula based approach although the genetic algorithm also incurs higher storage and latency overhead due to simulating multiple generations.




DOI: 10.1002/cpe

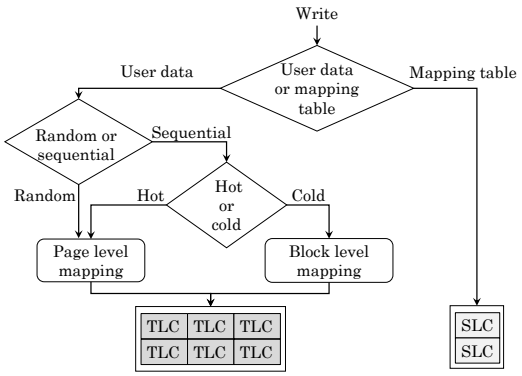



Figure 21. Working of the technique of Liu et al. [10]

### 6.5. *Use of integral controller*

Murugan et al. [43] propose a hybrid SSD management technique which works by identifying hot data. Pages with write-count higher than a threshold are termed as hot and remaining pages with non-zero writes as termed as warm since they have potential to become hot. The threshold is periodically updated to include pages responsible for top $K\%$ (e.g., $10\%$) of writes in the list of hot pages. Then, based on latency and write endurance of SLC and MLC, an integral controller decides the fraction of hot writes that are directed to MLC and the remaining hot writes go to SLC. All cold writes go to MLC. They use wear-leveling techniques within SLC and MLC. Due to the change in working set, a previous hot data-item may become cold and hence, it is migrated from SLC to MLC. As time passes, the amount of data migrated from SLC to MLC is gradually reduced to lower the lifetime degradation. Their technique improves both write performance and SSD lifetime.

### 6.6. *Use of analytical models*

Wang et al. [44] note that use of a fixed size SLC buffer can harm performance and naively increasing the capacity of SLC buffer can increase GC overhead due to the requirement of reclaiming/moving large number of blocks. Also, this fixed-size buffer approach fails to account for the variation in application characteristics. They develop an analytical model of write costs in hot and cold regions based on the application behavior, GC overhead and existing region utilizations. GC cost increases with rising utilization since more valid pages need to be migrated from a victim block. Based on these factors, the optimum capacities of two regions are decided.

Due to the change in working set or incorrect identification of hot/cold data, a data-item may be placed in incorrect region. They note that when a hot data page is wrongly placed in cold region, a future update to the page can be performed in hot region and the previous copy in cold region can be invalidated, thus, no data migration is required to correct the placement. However, when a cold data page is incorrectly placed in the hot region, it needs to be migrated to the cold region. They use page-mapping scheme. A hot block uses only fast pages whereas all pages are used in a cold block.

When free blocks in a region fall below a threshold, GC is performed. There are three possible GC operations: (1) taking an erased block from other partition which changes the capacity and utilization of both partitions (2) moving valid data in victim blocks to other partition which changes utilization but not capacity of each partition and (3) moving valid data within a partition which does not change partition utilization. For selecting one of three GC operations, their after-GC performance is evaluated and the one achieving the smallest write cost is selected and performed. Their technique improves performance over a fixed-sized partitioning technique.



DOI: 10.1002/cpe

## 7. CONCLUSION AND FUTURE OUTLOOK

The design and management of storage devices have a significant impact on the performance, energy and cost efficiency of all computing systems ranging from embedded systems to data-centers and supercomputers [57]. In this paper, we presented a survey of the management techniques for hybrid SLC/MLC/TLC Flash-based SSD devices. We conclude this paper with a brief mention of future research directions.

Flash memory suffers from read disturb issue whereby a read to one row of cells affects the threshold voltages of unread cells in other rows of the same block. Addressing this will be highly important especially as Flash scales to low feature-size. Further, in face of increasing memory capacity demands, 3D Flash appears as a promising solution to continue to scale capacity within area budgets. While 48-layer 3D Flash products are already in market, 256-layer Flash is expected to become available in near future [6]. Compared to 2D Flash, 3D Flash memory provides higher data write performance and reliability, and lower cost per byte. The challenges in using 3D Flash is that it incurs higher initial investment and manufacturing cost and may provide lower yield compared to 2D Flash. Addressing these challenges will be vital for ensuring adoption of 3D Flash in commercial SSDs.

File system maintains metadata for files which can provide valuable hints to SSD-management techniques, however, most existing techniques fail to benefit from this information. For example, file-system level information can be useful for selecting appropriate blocks for garbage collection, or as hot/cold blocks for migration to suitable SSD partition. Clearly, a coordinated file-system and SSD management approach can provide much higher rewards than isolated management techniques for them.

Most proposed techniques use a fixed value of algorithm parameters such as threshold for finding hotness of data. While this approach works well when the programmer/designer has insights into nature of applications, for the general case of workloads with random combination of applications or different ratio of read/write operations, use of fixed parameters may not work well. Clearly, runtime adaptation of control parameters is vital for the proposed techniques to be effective for real-world scenarios.

Most existing simulators have been developed for evaluating homogeneous (e.g., MLC-only) SSD devices. Since each simulator mimics a particular device/machine configuration and makes certain assumptions, using these simulators for evaluating hybrid SSD devices may not be fully accurate. Creating hybrid SSD specific simulators and hardware prototypes will be significantly boost research in this field and will provide even more meaningful insights.

Based on Moore's law, the number of cores in a system are steadily increasing. Conventional hybrid SSD management techniques focus only on improving overall throughput which was sufficient for single-core systems. In multi-core systems, several other system-level objectives and constraints become prominent, such as fairness, QoS (quality of service), implementing priorities between applications running on different cores, non-uniform latencies due to data-migration operations in hybrid SSDs, etc. However, current hybrid SSD management policies are oblivious of these goals and constraints. Accounting for these metrics in the management policy of hybrid SSDs is important for ensuring their adoption in future multi- and many-core systems.

Most existing SSD management techniques have been proposed and evaluated in context of CPUs. In recent years, GPUs have been intensively used for a range of big-data analytics, database and scientific applications [58]. Since GPUs are not standalone computing units that can directly access the external storage, they need the help of host-side storage software stack for accessing the data on SSDs [59]. This, however, exacerbates the overheads of file-resident data movement and prevents fully exploiting the potential of both GPU and SSD [60]. As GPUs become primary citizens of the computing world, novel strategies to directly connect GPUs to hybrid SSDs and exploit specific features of SLC/MLC/TLC in boosting efficiency of GPU applications will be highly rewarding.

Traditional error-intolerant computing approach necessitates precise computation and storage. However, the intrinsic error-tolerance of applications and limitations in cognitive capabilities



DOI: 10.1002/cpe

of users provides scope for approximate computing and storage [61], where accuracy can be sacrificed to boost performance and energy efficiency. This approach can provide large benefits in Flash memories, for example, permitting imprecise writes in MLC/TLC Flash allows significantly reducing the number of program and verify cycles, which lowers write latency/energy and improves write endurance. Going forward, an effective approach for imprecisely storing and processing data from hybrid SSDs will be highly effective in meeting the challenges of 'big data'.

## REFERENCES

[1] Mittal S, Vetter JS, Li D. A survey of architectural approaches for managing embedded dram and non-volatile on-chip caches. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 2015; **26**(6):1524 – 1537.

[2] Chang LP, Kuo TW. Efficient management for large-scale flash-memory storage systems with resource conservation. *ACM Transactions on Storage (TOS)* 2005; **1**(4):381–418.

[3] Liu D, Wang Y, Qin Z, Shao Z, Guan Y. A space reuse strategy for flash translation layers in SLC NAND flash memory storage systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 2012; **20**(6):1094–1107.

[4] Hsieh JW, Kuo TW, Chang LP. Efficient identification of hot data for flash memory storage systems. *ACM Transactions on Storage (TOS)* 2006; **2**(1):22–40.

[5] Vetter J, Mittal S. Opportunities for nonvolatile memory systems in extreme-scale high performance computing. *Computing in Science and Engineering* 2015; **17**(2):73 – 82.

[6] Tom Coughlin. How Many Layers Are Possible In 3D Flash? `https://goo.gl/HtDd5B` 2017.

[7] Jimenez X, Novo D, Ienne P. Software controlled cell bit-density to improve nand flash lifetime. *Design Automation Conference*, ACM, 2012; 229–234.

[8] Nam BW, Na GJ, Lee SW. A hybrid flash memory ssd scheme for enterprise database applications. *International Asia-Pacific Web Conference (APWEB)*, IEEE, 2010; 39–44.

[9] Chang CW, Chen GY, Chen YJ, Yeh CW, Eng PY, Cheung A, Yang CL. Exploiting write heterogeneity of morphable mlc/slc ssds in datacenters with service-level objectives. *IEEE Transactions on Computers* 2017; **PP**(99):1–1.

[10] Liu D, Yao L, Long L, Shao Z, Guan Y. A workload-aware flash translation layer enhancing performance and lifespan of tlc/slc dual-mode flash memory in embedded systems. *Microprocessors and Microsystems* 2017; :–.

[11] HP. SSD endurance 2015. URL `https://goo.gl/xjWzDD`.

[12] MICRON. 3D NAND Flash Memory 2016. URL `https://goo.gl/dmI86v`.

[13] ADVANTECH. Flash type comparison for SLC/MLC/TLC and Advantechs Ultra MLC technology 2016. URL `https://goo.gl/ewWzea`.

[14] MICRON. TLC MLC and SLC Devices 2017. URL `https://goo.gl/FJo45c`.

[15] Yim KS. A novel memory hierarchy for flash memory based storage systems. *Journal of Semiconductor Technology and Science* 2005; **5**(4):262–269.

[16] Silicon Motion. Silicon Motion Announces Launch of Three New SSD Controllers Optimized for Managing MLC NAND Flash 2008. URL `https://goo.gl/J2psoF`.



DOI: 10.1002/cpe

[17] Mittal S, Vetter J. A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 2016; **27**(5):1537–1550.

[18] Chang LP. A hybrid approach to NAND-flash-based solid-state disks. *IEEE Transactions on Computers* 2010; **59**(10):1337–1349.

[19] Park D, Du DH. Hot data identification for flash-based storage systems using multiple bloom filters. *MSST*, 2011; 1–11.

[20] Chang YM, Chang YH, Kuo TW, Li YC, Li HP. Disturbance relaxation for 3d flash memory. *IEEE Transactions on Computers* 2016; **65**(5):1467–1483.

[21] Liu D, Wang T, Wang Y, Qin Z, Shao Z. Pcm-ftl: A write-activity-aware nand flash memory management scheme for pcm-based embedded systems. *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, IEEE, 2011; 357–366.

[22] Kim Y, Tauras B, Gupta A, Urgaonkar B. Flashsim: A simulator for nand flash-based solid-state drives. *Advances in System Simulation, 2009. SIMUL'09. First International Conference on*, IEEE, 2009; 125–131.

[23] Alsalibi AI, Sumari P, Alomari SA, Al-Betar MA. Performance and reliability concern scheme for efficient garbage collection and wear leveling on flash memory-based solid state disk. *Microsystem Technologies* 2016; :1–15.

[24] Matsui C, Yamada T, Sugiyama Y, Yamaga Y, Takeuchi K. Optimal memory configuration analysis in tri-hybrid solid-state drives with storage class memory and multi-level cell/triple-level cell nand flash memory. *Japanese Journal of Applied Physics* 2017; **56**(4S):04CE02.

[25] Jimenez X, Novo D, Ienne P. Libra: Software-controlled cell bit-density to balance wear in nand flash. *ACM Trans. Embed. Comput. Syst.* 2015; **14**(2):28:1–28:22.

[26] Samsung. Samsung Unveils its Third Fusion Semiconductor - Flex-OneNAND 2007. URL `http://www.samsung.com/semiconductor/about-us/news/4194`.

[27] Geoff Gasior. Micron's M600 SSD accelerates writes with dynamic SLC cache 2014. URL `https://goo.gl/Q3hicc`.

[28] Geoff Gasior. Samsung's 840 EVO solid-state drive reviewed TLC NAND with a shot of SLC cache 2013. URL `https://goo.gl/oSDk49`.

[29] Mittal S, Wang R, Vetter J. DESTINY: A Comprehensive Tool with 3D and Multi-level Cell Memory Modeling Capability. *Journal of Low Power Electronics and Applications* 2017; **7**(3):23.

[30] Mittal S. A survey of power management techniques for phase change memory. *International Journal of Computer Aided Engineering and Technology (IJCAET)* 2016; **8**(4):424–444.

[31] Mittal S. A survey of techniques for architecting processor components using domain wall memory. *ACM Journal on Emerging Technologies in Computing Systems* November 2016; **13**(2):29.

[32] Hsieh JW, Chen CW, Lin HY. Adaptive ECC Scheme for Hybrid SSD. *IEEE Transactions on Computers* Dec 2015; **64**(12):3348–3361.

[33] Jimenez X, Novo D, Ienne P. Phoenix: reviving mlc blocks as slc to extend nand flash devices lifetime. *DATE*, 2013; 226–229.

[34] Jung S, Song YH. Hierarchical use of heterogeneous flash memories for high performance and durability. *Consumer Electronics, IEEE Transactions on* 2009; **55**(3):1383–1391.



DOI: 10.1002/cpe

[35] Im S, Shin D. Comboftl: Improving performance and lifespan of mlc flash memory using slc flash buffer. *Journal of Systems Architecture* 2010; **56**(12):641–653.

[36] Oh Y, Lee E, Choi J, Lee D, Noh SH. Hybrid solid state drives for improved performance and enhanced lifetime. *Symposium on Mass Storage Systems and Technologies (MSST)*, 2013; 1–5.

[37] Chen R, Qin Z, Wang Y, Liu D, Shao Z, Guan Y. On-demand block-level address mapping in large-scale nand flash storage systems. *IEEE Transactions on Computers* 2015; **64**(6):1729–1741.

[38] Im S, Shin D. Storage architecture and software support for slc/mlc combined flash memory. *Proceedings of the 2009 ACM symposium on Applied Computing*, ACM, 2009; 1664–1669.

[39] Yang MC, Chang YH, Tsao CW, Liu CY. Utilization-aware self-tuning design for tlc flash storage devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* Oct 2016; **24**(10):3132–3144.

[40] Hachiya S, Johguchi K, Miyaji K, Takeuchi K. Hybrid triple-level-cell/multi-level-cell nand flash storage array with chip exchangeable method. *Japanese Journal of Applied Physics* 2014; **53**(4S):04EE04.

[41] Park JW, Park SH, Weems CC, Kim SD. A hybrid flash translation layer design for slc–mlc flash memory based multibank solid state disk. *Microprocessors and Microsystems* 2011; **35**(1):48–59.

[42] Lu N, Choi IS, Ko SH, Kim SD. An Effective Hierarchical PRAM-SLC-MLC Hybrid Solid State Disk. *International Conference on Computer and Information Science (ICIS)*, IEEE, 2012; 113–118.

[43] Murugan M, Du DH. Hybrot: Towards improved performance in hybrid slc-mlc devices. *International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, IEEE, 2012; 481–484.

[44] Wang W, Pan W, Xie T, Zhou D. How many mlcs should impersonate slcs to optimize ssd performance? *Proceedings of the Second International Symposium on Memory Systems*, MEMSYS '16, ACM: New York, NY, USA, 2016; 238–247.

[45] Sung M, Kim K. Asymmetric flash volume management. *Consumer Electronics, IEEE Transactions on* 2012; **58**(2):455–461.

[46] Park J, Lee E, Bahn H. Dabc-nv: A buffer cache architecture for mobile systems with heterogeneous flash memories. *Consumer Electronics, IEEE Transactions on* 2012; **58**(4):1237–1245.

[47] Sun C, Iwasaki TO, Onagi T, Johguchi K, Takeuchi K. Cost, capacity, and performance analyses for hybrid scm/nand flash ssd. *IEEE Transactions on Circuits and Systems I: Regular Papers* Aug 2014; **61**(8):2360–2369.

[48] Lee S, Kim J. Improving performance and capacity of flash storage devices by exploiting heterogeneity of mlc flash memory. *Computers, IEEE Transactions on* 2014; **63**(10):2445–2458.

[49] Kwon SJ, Chung TS. Data pattern aware ftl for slc+mlc hybrid ssd. *Design Automation for Embedded Systems* 2015; **19**(1):101–127.

[50] Zhang X, Li J, Wang H, Zhao K, Zhang T. Reducing solid-state storage device write stress through opportunistic in-place delta compression. *FAST*, 2016; 111–124.



DOI: 10.1002/cpe

[51] Cho IP, Ko SH, Yang HM, Kim CG, Kim SD. A dynamic buffer management of hybrid solid state disk for media applications. *Proceedings of the International Conference on IT Convergence and Security 2011*, Springer, 2012; 267–279.

[52] In Jh, Kim Hj, Lee Ky, Chung Ts. Method of remapping flash memory 2009. US Patent 7,516,295.

[53] Rosenblum M, Ousterhout JK. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems (TOCS)* 1992; **10**(1):26–52.

[54] Lee SW, Choi WK, Park DJ. Fast: An efficient flash translation layer for flash memory. *Emerging Directions in Embedded and Ubiquitous Computing*. Springer, 2006; 879–887.

[55] Kim J, SEOL J, MAENG S. A buffer management issue in designing ssds for lfss. *IEICE Transactions on Information and Systems* 2010; **E93.D**(6):1644–1647.

[56] Jin S, Kim J, Kim J, Huh J, Maeng S. Sector log: Fine-grained storage management for solid state drives. *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, ACM, 2011; 360–367.

[57] Mittal S. Power management techniques for data centers: A survey. *arXiv preprint arXiv:1404.6681* 2014; .

[58] Mittal S, Vetter JS. A Survey of Methods for Analyzing and Improving GPU Energy Efficiency. *ACM Computing Surveys* 2015; **47**(2):19:1–19:23.

[59] Mittal S, Vetter J. A Survey of CPU-GPU Heterogeneous Computing Techniques. *ACM Computing Surveys* 2015; **47**(4):69:1–69:35.

[60] Zhang J, Donofrio D, Shalf J, Kandemir MT, Jung M. Nvmmu: A non-volatile memory management unit for heterogeneous gpu-ssd architectures. *2015 International Conference on Parallel Architecture and Compilation (PACT)*, 2015; 13–24.

[61] Mittal S. A survey of techniques for approximate computing. *ACM Computing Surveys* 2016; **48**(4):62:1–62:33.

[62] Sharma D. System Design for mainstream TLC SSD: Meeting the performance challenge. `goo.gl/cYRX3P` 2014.

[63] Hruska J. How Do SSDs Work? 2017. URL `https://goo.gl/Rx48QH`.



DOI: 10.1002/cpe