# Enhancing Endurance of SSD Based high-performance Storage Systems using Emerging NVM Technologies

Tanaya Roy and Krishna Kant
Temple University
Email: [tanaya.roy,kkant]@temple.edu

***Abstract*—SSD technologies with multiple bits per cell are very attractive for increasing the performance of HPC applications, but suffer from poor endurance. In this paper, we explore how they can be paired with the emerging non-volatile memory (ENVM) technologies to enhance both endurance and performance of the storage systems. While in-memory caching can enhance the performance, it increases the potential for data loss. With the high speed ENVM technologies, it becomes possible to persist writes immediately and thereby eliminate or minimize data loss and yet reduce the stress on the back-end SSD and thereby improve its endurance. In this paper, we show that by an integrated management of DRAM and ENVM caches we can achieve the desired tradeoff between data loss risk, performance, and endurance. For example, with performance as the primary objective, the overall read latency can be reduced to 58% of the original by using Intel Optane cache that is only $\sim 3\%$ of the SSD size. Similarly, with 5 year endurance as the primary objective for both the SSD and Optane storage, the Optane size only needs to be 12% of the SSD size.**

***Keywords*-Emerging Caching; Storage performance; Endurance; Flash Technology, ENVM, High Performance Storage**

## I. Introduction

A diverse and wide range of scientific computing are moving towards cloud, and demand high-performance both in terms of low latency and high throughput [19], [13]. The data intensive computation of these applications places substantial stress on the storage system, which increasingly determine the ultimate performance achieved by these applications.

The primary storage in HPC systems is rapidly evolving from hard disk drive (HDD) to flash technology based solid state storage (SSD) due to their much higher performance and lower energy consumption. However, due to increasing data storage requirements of high-performance computing applications [10], [9], the Flash technology used by SSDs is moving towards multiple bits/cell to deliver higher storage densities at a lower cost. Unfortunately, as the number of bits/cell increases, the endurance of the device (in terms of number program-erase cycles per cell before the cell wears out) goes down rapidly. In particular, while an enterprise class MLC (multi-level cell, or really 2 bits/cell) can provide 10K PE cycles, the most recent QLC (quad level cell or 4 bits/cell) provides endurance only in a few hundred PE cycles. Thus while it is attractive to utilize QLC or TLC (triple level cell or 3 bits/cell) in large HPC storage due to their low cost and dramatically higher performance as compared with HDDs, the endurance can become a highly limiting factor for large IO intensive HPC workloads.

Fortunately, the more recent developments in the storage technologies offer a solution that we explore in this paper. The space of new nonvolatile memory (NVM) technologies is highly active with a range of technologies, all of which are faster than flash, provide much higher endurance, and quite unlike flash allow overwrites. Some examples of these technologies are Phase Change Memory (PCM), Resistive RAM (RRAM), Spin-torque transfer RAM (STTRAM), 3D Xpoint etc. We henceforth refer to them as “emerging” NVM or ENVM technologies. One such ENVM technology has recently been made available commercially by Intel under the name “Optane”, organized as a 3D cross-point structure. It is 10X faster and has 30X higher endurance than Flash (QLC SSD). Currently, it is 20X more expensive; therefore, a suitable use for it in the current storage hierarchy is in form of a nonvolatile cache. Accordingly, we will analyze 3-layer storage hierarchies in this paper consisting of DRAM, Optane, and SSD layers, and examine how a small amount of Optane can provide both high-performance and high overall endurance for the HPC storage hierarchy.

In addition to performance, an important concern in HPC systems is that of resilience, and the resilience issues become more severe as the size of the system increases. One important aspect of resilience is the time it takes to recover from machine crashes. The amount of unsaved data lost is an important aspect of recovery time. With the availability of high speed NVM technologies such as Optane, it is possible to minimize the data loss by minimizing dirty data in memory, and instead persisting the writes into the Optane. While large writes would put a substantial stress on the Optane, small writes can be easily persisted there (and larger writes possibly persisted to SSDs). Thus, in our investigation, we explore three way tradeoff between performance, endurance and data loss.

The amount of write activity in a workload is crucial from the perspective of both the risk of data loss and endurance impact on the storage device. Unfortunately, HPC systems with large page caches often result in IO behavior that is dominated by writes, and these writes can range over a large

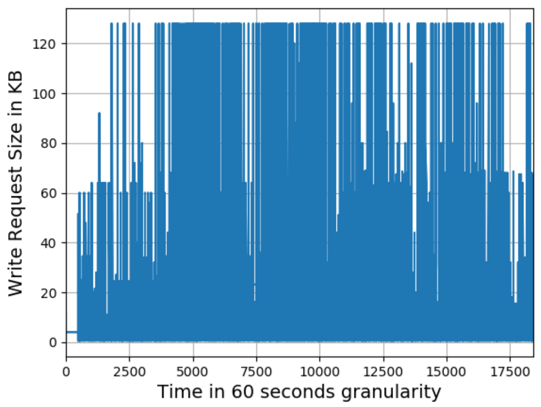


Fig. 1: Write Data Size Variations

size. We observed such characteristics even in the Systor 2017 [7] workload trace which is one of the few rather long-duration public traces available. Fig. 1 shows the write size distribution for this trace. Note that the overall traffic has substantial percentage of reads, however, the large writes make it difficult to accommodate them in a relatively small ENVM. In particular, simply persisting all writes in ENVM will cause it to thrash. On the other hand, frequent update to write data placed at SSD can wear it out quickly. Thus, performance, endurance and data loss are conflicting goals and thereby require a smart mechanism to alleviate data loss risk by persisting write data immediately between ENVM and SSD such that overall endurance of the system is maintained.

Achieving these tradeoffs in HPC storage, several challenges are involved. First, ENVM is required to accommodate both read and write data, irrespective of the underlying caching mechanism, adaptive to the workload characteristics to provide substantial performance. Second, the modified data present in ENVM needs to be updated to the backend SSD in an intelligent fashion such that ENVM always has sufficient space to accommodate incoming DRAM evicted read and write data. Third, to maintain the endurance rating it is required to control the writes on endurance sensitive devices intelligently considering their respective endurance characteristics.

The key contribution of our work is the quantification of the benefits of different policies for exploiting ENVM technologies to enhance the performance and endurance of SSD based high performance storage (HPS). The results can be used to design cost-effective and high-performance SSD storage system by pairing low cost multi-bit per cell SSD technologies with relatively small amounts of ENVM. For the workloads examined, we show that our mechanism can achieve 58% read performance and 28% write latency that of native QLC SSD latency by incorporating ENVM that is only $\sim 3\%$ that of SSD capacity. However, if the objective is to achieve a full 5-year life-time for overall storage system, the required ENVM capacity can be up to 12% that of SSD capacity.

In addition to the real implementation using Optane and QLC SSD, we also built a comprehensive simulation model for this three-level storage hierarchy. The purpose of the simulation is to examine many corner cases and to also explore parameters that would be very difficult or impossible in a real implementation. In particular, we consider SSDs using 1 to 5 bits/cell. The 5 bits/cell technology (called PLC, or penta-level cell) is still in the works and not currently available. Through our simulator, we also explore how various flash technologies affect the amount of ENVM required to meet the intended objective.

Although various solutions to improve overall performance of HPS are proposed introducing SSD and non volatile memory [19], [20], [24], [21], [6], [1], to the best of our knowledge, a comprehensive study of tradeoff between performance, endurance, and data loss by introducing ENVM in HPS, irrespective of underlying caching mechanism, to keep selective read and write data in the storage hierarchy has not been examined in the literature. With ENVM finally becoming available commercially and the Flash technology moving to QLC with rather poor endurance, it is crucial that we consider combinations of the two to not only address the overall performance and endurance simultaneously, but also exploit the opportunity to minimize or eliminate data loss.

The rest of the paper is organized as follows. Section II provides the background and motivation for this work. Section III places our work in the context of the vast amount of literature related to this context. In Section IV we discuss the proposed mechanisms applied on HPS using ENVM technology. The sections V and VI describes proposed policies and experimental evaluation respectively. We conclude the paper in Section VII.

## II. Motivation and Background

The current trend of ENVM, such as Optane based on PCM technology, STTRAM and etc., are available small in size. The large working set cannot be accommodated totally and thus ENVM is proposed to use as cache to enhance the performance. However, what proportion ENVM needs to be used by read and write data to enhance the performance and maintain the endurance of the overall storage hierarchy while reducing the data loss is never studied before. The write characteristics of the Systor 2017 traces on a busy day (22nd February) during 5 hours busy period for Lun0 as in Fig. 2 shows that placing small writes on SSD might wear out the drive fast because of low P/E cycle numbers.

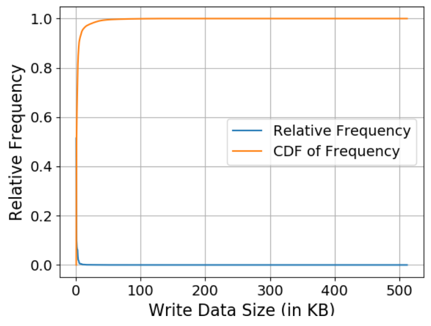


Fig. 2: Write Data Size Frequency

Furthermore, it is necessary to control the number of writes to any endurance sensitive devices. A constant number of allowable writes during any period on these devices might not be of interest. During the high write activity, actual number of writes may exceed the predefined allowable number of writes. Whereas during the non-busy period, the actual number of writes might be lower than the predefined allowable number of writes. Therefore, it is required that the allowable writes should be adaptive with workload write characteristics and device's endurance rating. From Fig. 3, it is observed that there is a homogeneity in the write characteristics of the workload and thus can help in measuring the allowable writes at any time granularity.

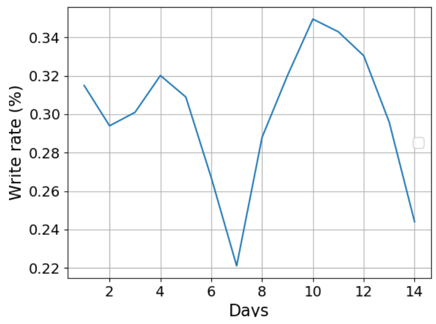


Fig. 3: Write Rate Trend Over a Small Duration

## III. Current ENVM Enhanced Storage Architectures

In this section, we provide a short overview of the literature on a wide range of solutions to enhance the performance of the system beyond the capability of conventional non-volatile memory technologies. We also provide a short overview of the literature on life-time optimization of flash-based storage. The performance enhancement of HPS using SSD and ENVM are also studied here and showed distinct difference from our work.

In recent days, SSD and ENVM are studied actively [15], [18], [17], [26], [11], [22], [5]. Niu et al. [16] provide an extensive literature review on the state-of-the-art research for hybrid storage systems relative to algorithms and policies for caching, data migration, and hot data identification.

Ziqi et al. have designed a hybrid cooperative cache, Hibachi [2], exploring a DRAM cache as read cache only for clean pages, and non-volatile memory as a write buffer for dirty pages. In our work, we allow the ENVM to automatically store clean pages if the capacity allows doing so.

Lin et al. [8] have developed a buffer-aware data migration scheme by exploiting the content information in the buffer cache. They model the process of data migration between HDD and SSD and assess the effectiveness of the buffer-aware data migration scheme to maximize the I/O performance and avoid unnecessary page migrations. In our proposed work, modified data is persisted immediately and writeback is done opportunistically to maximize the I/O performance having in mind endurance requirements for both ENVM and SSD. Tarihi et al. [23] have developed a design that uses a hybrid DRAM-PCM SSD cache design with an intelligent data movement scheme. This design exploits PCM as a DRAM alternative while alleviating its issues such as long write latency, high write energy, and finite endurance based on thorough I/O characterization of desktop and enterprise applications. Our goal instead is to consider a three level hierarchy in which SSD is used as primary storage.

Lu et al [12] have introduced managing performance and endurance of persistent memory together by reducing the overhead caused due to the strict ordering of writes in persistent memory. They have speculated about inter-transaction ordering and intra-transaction ordering of writes and maintaining consistency with a material alteration to the memory log organization and hardware support at cache level. Garcia et al [3] describe a technique to enhance the life-time of non-volatile memory by reducing traffic towards non-volatile memory and write hot-spots. Write traffic is reduced by different cache replacement policies and with a proposed data compression technique to increase cache capacity. Write hot-spots are reduced by distributing writes over non-volatile memory. Our goal is to address a storage hierarchy and explore the effects of persisting every write immediately based on device characteristics to provide high reliability.

Koo et al [6] proposed an optimized progressive file layout method that leverages the advantage of SSDs in HPC cloud environment. In our work we did not touch the SSD file system. Instead we have showed the intelligent data movement can balance the performance and endurance of high-performance storage.

Shi et al [19], [20] have proposed SSDUP and SSDUP+ to use SSD as burst buffer to the underlying HDD as dominant storage device. They have developed a traffic-detection method to detect sequential and randomness in the write traffic to buffer the random writes on SSD. According to SSD's characteristic, random writes tends to wear out fast. In our work, we consider SSD as dominant storage and also maintain its endurance introducing ENVM in HPS hierarchy.

Subedi et al [21] have introduces Intel Optane device in HPC and compared the performance between Optane SSD and NVMe SSD. In our work, we have used Optane SSD as ENVM top of SSD as backend. Intelligent management of data placement at ENVM and SSD helps to enhance the performance and maintain the endurance of HPS hierarchy.

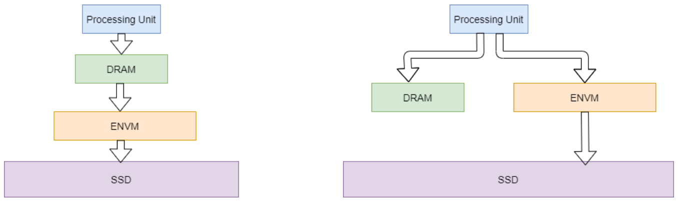
## IV. Enhancing high-performance Storage using ENVM Technology

We examine a high-performance storage hierarchy consists of DRAM at the upper level, and ENVM at the middle level atop flash based SSDs in the backend. As in Fig. 4, an application can use two logical architectures when the ENVM present in the high-performance storage hierarchy. As ENVM comes with low latency and higher endurance than SSD, it can be used along with DRAM serving different purposes such as another level of cache or as write cache minimizing the data loss. Irrespective of the underlying caching policy, different mechanisms on ENVM can be considered to enhance performance and endurance of the overall system.

## A. Performance centric mechanism

A performance centric policy should, at a minimum, minimize over average read and write latency. An additional desirable goal is to reduce tail latency as well; however, this goal could conflict with the goal of reducing average latency.

The DRAM is considered as a simple LRU cache and thus contains the most frequently used data. ENVM is capable of caching all or selective read and write-data evicted from DRAM. Note that it is possible to consider other, more sophisticated, caching policies such as ARC or SARC [14], [4]; however, that's not the objective of this line of research and thus we stick to simple LRU based caching.



(a) Next Level Cache (b) Write Cache

Fig. 4: High-performance Storage Logical Architectures

The write data can be of two types small and large and can be categorized based on the request size. The request size smaller than average write size is considered as small write. Whereas, the request size of average write size and more are considered as large write. The write can happen on ENVM or SSD based on the type of write, small or large. Writeback takes place from ENVM to SSD. However, to make this process efficient, we do not do one element at a time writeback. Instead, we invoke the writeback opportunistically based on the IO queue depth of the SSD. Obviously, an evicted block must stay in ENVM until the writeback is done; therefore, there is a tradeoff between opportunism and making space in the ENVM. The opportunistic writeback is achieved by predicting the gap and scheduling writeback I/O accordingly.

Both lists are maintained at ENVM level and potential move of elements between them only changes the pointers (or metadata, in general).

Keeping track of which data is persisted in ENVM is necessary. Therefore, we assume that there is a journaling file system that treats each data element as a file and thus is responsible for maintaining the mapping necessary to know its' physical location on backend storage. Our priorities are to obtain high-performance and retain desired endurance. Therefore an amount of ENVM should still be reserved for this journaling. How much extra space is needed to keep this metadata depends on the size of data elements, but it is nevertheless a fraction of the size of the area that is devoted to persisting.

## B. Endurance centric mechanism

It is necessary to control the number of writes to the endurance-sensitive device to respect the endurance rating of it. The ENVM can be used to accommodate all or small writes as required and others are accumulated in SSD. Endurance centric mechanism allows write-persist to the device based on an estimated write-budget under the experienced traffic characteristics at a suitable time granularity. The write budget ($\mathrm{W_{est}}$) is calculated based on corresponding device's endurance rating. The traffic characteristic ($\mathrm{TR_{est}}$) is aggregated from the past using exponential smoothing as in Equation 1.

$$\mathrm{T}R_{\mathrm{est(t+1)}} = \alpha * TR_{\mathrm{act(t)}} + (1-\alpha) * TR_{\mathrm{est(t)}} \tag{1}$$

**Algorithm 1** Write-Budget Estimation

```
1: procedure WriteBudget(reward $R$)
2:     $W^{\text{Week}}_{est(\text{t+1})} \leftarrow W^{\text{Week}}_{est(\text{t})} - R$
3:     for $Day \leftarrow Monday \rightarrow Sunday$ do
4:         $TR^{\text{Day}}_{est(t+1)} \leftarrow \alpha * TR^{\text{Day}}_{act(t)} + (1-\alpha) * TR^{\text{Day}}_{est(t)}$
5:     $TR^{\text{Week}}_{est(t+1)} \leftarrow \sum_{Day} TR^{\text{Day}}_{est(t+1)}$
6:     for $Day \leftarrow Monday \rightarrow Sunday$ do
7:         $W^{\text{Day}}_{est(\text{t+1})} \leftarrow W^{\text{Week}}_{est(\text{t+1})} * \frac{TR^{\text{Day}}_{est(t+1)}}{TR^{\text{Week}}_{est(t+1)}}$
```

The write-budget estimation procedure should follow the workload characteristics and adapt to the change in workload over time. The estimated write-budget should be adaptive; high enough during the write-busy period to accommodate maximum possible large write-persist at the SSD level and thus provide ENVM the opportunity to control the space for immediate persisting. The Write-Budget estimation procedure is represented in Algorithm 1. Given the capacity and endurance of the SSD, a coarse-grain write-budget is calculated initially. According to the workload characteristic the write budget is distributed over finer time granularity and the surplus amounts of writes at coarser granularity are adjusted over the next cycle. We consider a week as the time period with coarser time granularity and a day as the time period when we consider finer time granularity. The adjustment of excess or surplus is calculated by measuring the difference between actual number of write-data persists at the device over a week ($W^{Week}_{act}$) and the estimated number of write-data persists in the other device over a week ($W^{Week}_{est}$) as Equation 2. The surplus is adjusted by measuring the actual number of writes on the device over a week ($W^{Week}_{act}$) and the estimated number of writes on the device over a week ($W^{Week}_{est}$) as Equation 2. This endurance centric mechanism can be applied considering any of the devices. We applied this mechanism on SSD and Optane to enhance the SSD life-time.

$$R = \sum_{Day} W^{\text{Day}}_{act(t)} - W^{\text{Week}}_{est(\text{t})} \tag{2}$$

## V. Proposed Policies

We consider three different policies in managing the high-performance storage hierarchy presented in Fig. 4 including ENVM serving different purposes such as another level of cache or as write cache minimizing the data loss. The performance and endurance centric mechanisms are applied to these policies accordingly. The first policy is "Oblivious Caching" without regard to clean or dirty data. The second and third policies make this distinction, but the second policy does not do immediate persisting of all modified data in ENVM, whereas the third policy does. These policies are described in more detail in the following.

### *A. Policy-I: Oblivious Cache (OC)*

In this policy, the ENVM is used as another level of independent larger cache next to DRAM cache. As ENVM has lower latency than SSD, the average read and write latency can be improved. However, since this policy does not limit the amount of modified data in DRAM or how long it resides there before being evicted and written back to ENVM or SSD, the policy carries data loss risk due to system crashes. We assume that both the ENVM and SSD implement suitable mechanisms to maintain consistency in the event of system crash. Since the ENVM is used as a DRAM-like cache, it can use a simpler mechanism to ensure all or nothing semantics for the data and data structures to manage the cache. Assuming that the SSD implements a file system, appropriate journaling mechanism would handle consistency issues for it.

The performance of this policy will vary with different caching policies, but our focus is not caching policies per se. Therefore, we will use simple LRU or a LRU variant for both DRAM and ENVM level. All data evicted from the DRAM caches goes into the ENVM cache but evictions from the ENVM can be discarded if they are clean. Evictions of modified blocks from ENVM must be written back to SSD. Note that this policy does not explicitly consider the endurance of SSD or ENVM while doing the writes.

In this policy, a lookup operation travels first to DRAM, next on a cache miss to ENVM and finally resolves at SSD upon missing out of the ENVM. On DRAM eviction all data are stored in ENVM cache. If the number of modified data touches the concurrency threshold, prioritized writeback writes data to the SSD. Otherwise, an opportunistic writeback is done.

### *B. Policy-II: Next Level Cache Without Immediate Persisting (NLCWOIP)*

In this policy also the ENVM is used as the next level cache on top of the SSD as backend. The data evicted from the DRAM cache to be written to the ENVM prioritizes write data over read data unlike Policy I. A portion of ENVM, adaptive to the workload characteristics, is reserved for storing relevant clean data for future use for lowering the read latency. Like oblivious cache, this policy also risks data loss due to system crashes due to same reason.The consistency and system crash handling is also same as Policy I.

The performance of this policy is enhanced by incorporating opportunistic and conditional writeback mechanism like policy I. Also, speculating relevant clean data for storing in ENVM avoids unnecessary evictions from ENVM Cache (and hence writebacks to the SSD). The data in DRAM with higher the access frequency acquires higher chance to be stored in ENVM. The selection of relevant data involves very simple computation and hence does not add much to the latency.

In this policy, DRAM Cache miss works similarly as Policy I. Upon eviction, all modified data are stored in ENVM, whereas only the clean data selected as relevant gets stored. Also conditional and opportunistic writeback mechanisms are involved.

### *C. Policy-III: Next Level Cache With Immediate Persisting (NLCWIP)*

In this policy, we persist all modified data immediately either to ENVM or SSD. In this policy the DRAM works as read cache for clean data whereas the ENVM works primarily as write cache for the SSD. The ENVM is also used as extended read cache adaptive to the workload characteristics. The non volatile and low latency characteristics of ENVM allow it to work as a write cache alongside the DRAM. Moreover, adapting to the workload characteristic, the low latency aspect of ENVM makes it suitable for storing relevant clean data evicted from the DRAM cache. This policy omits the data loss risk by persisting modified data immediately in the ENVM cache or SSD.

The performance of this policy is enhanced by incorporating conditional writeback along with opportunistic writeback mechanism to have available space for modified data to persist immediately. Also speculating relevant clean data for storing avoids unnecessary eviction from ENVM Cache. Both of these techniques help to lower the read and write latency. Unlike the other two policies, the policy III incorporates endurance centric mechanism. There is a common phenomenon that small writes appear much more frequently than large writes. The large and small both write affects device life-time based on their endurance characteristics. Thus to maintain the endurance of the ENVM and SSD, the small writes are persisted in ENVM and large writes in SSD. Also intelligent write-budget procedure is applied to maintain the endurance rating of the considered device.

In this policy, a read miss results in ENVM lookup and on success data are accessed from there or from SSD otherwise. In case of write hits in ENVM, writes are persisted there. For write miss, if the write data is large, it is persisted on SSD respecting the write budget. For small writes, it is persisted in ENVM always, DRAM eviction and writeback mechanisms are same as described earlier for other policies.

## VI. Experimental Evaluation

### A. Implementation Setup

In order to explore the proposed mechanisms, we built an experimental setup consisting of an available Optane SSD module and a SSD. Both were of rather modest size; however, they are adequate to explore the issues of highly variable and write intensive workloads that HPC systems migh face. At the time of our experimental setup, the available Optane modules were only 32GB, and we paired one of those with an Intel 1 TB QLC SSD and 1 TB Samsung EVO TLS SSD. The DRAM buffer was sized accordingly. This was done on an Intel 6-core i5-8500 CPU @ 3.00GHz machine. Because of small size of Optane we limited DRAM cache to 4GB. The implementation intercepts the IOs and forces them through our own DRAM cache (different from system's buffer cache). The Optane SSD does not necessarily require a file system and we used it without one, exactly like memory. The metadata relating to the Optane was kept mostly in DRAM for efficiency; however, there is adequate information in the blocks stored in Optane so that its contents can be reconstructed in case of a machine crash. The SSD hosts a normal EXT4 Linux file system with normal journaling mechanisms. We drove this system by generating IO requested based on several storage IO traces.

The endurance rating measured in GB per day are 100, 110 and 252.5 GB for Optane, QLC and TLC respectively.

The crash management of the system is considered as follows. For DRAM, it is expected that the normal data limiting mechanisms will be in place; however, for the purposes of these experiments, no such restrictions played. Ensuring the all or nothing semantics for a single write is much simpler in our case, particularly due to the availability of Optane, where the metadata are recorded for only the ongoing write to recover from its interruption. With Optane used as a memory cache without a FS, it involves a rather trivial metadata recording. For policy III, the journaling system on the SSD file system comes into play only for evictions from Optane. It is even possible to simplify journaling for SSD FS (since the data is safely stored in Optane and can persist there until the eviction is properly handled in its entirety). However, SSD FS is not touched in this work.

Additionally, a comprehensive simulation model for three-level storage hierarchy built. The key purpose of this simulator was to consider PLC and how it affect the amount of ENVM required to meet the target.

*1) ENVM specific Implementation Detail:* To accommodate both read and write data ENVM maintains two separate lists clean list (CL) and modified list (ML). Both lists are maintained at ENVM level and potential move of elements between them only changes the pointers (or metadata, in general).

Two seperate I/O queues: demand queue that serves the read request and writeback queue are considered. In general, the demand queue gets higher priority, but empty demand queue triggers opportunistic writeback if there is enough time estimated through gap prediction $\text{Gap}_{\text{est}}$ (Equation 3) - the exponentially smoothed value of the amount of time between two requests with $\alpha = 0.5$ so that estimated and actual have same influence. The number of data elements that are selected to be written back $\text{WB}_{\text{element}}$ is calculated from average access latency and the estimated $\text{Gap}_{\text{est}}$ as shown in Equation 4.

$$\text{Gap}_{\text{est(t+1)}} = \alpha * \text{Gap}_{\text{act(t)}} + (1-\alpha) * \text{Gap}_{\text{est(t)}} \tag{3}$$

$$WB_{\text{element}} = \frac{\text{Gap}_{\text{est}}}{\text{SSD average access latency}} \tag{4}$$

Also, conditional prioritized writeback functionality is used to ensure that there is enough space for the ML. The concurrency threshold is defined in equation 5. The conditional writeback is triggered when the ML size exceeds hysteresis percentage applied on the concurrency (S).

$$\text{S} = \frac{\text{Write miss count}}{\text{Write miss count + Evict miss count}} \tag{5}$$

To determine the relevant data evicted from DRAM and to store in the CL follows a simple frequency based mechanism in this context. A clean evicted data element will be selected as relevant data if they have more access frequency during DRAM life-time than any element currently lying on ENVM CL.

*2) Write Budget specific Implementation Detail:* Based on traffic characteristics, the write budget is estimated using Algorithm1. The smoothed value with $\alpha$ being 0.5 is used as in Equation 1 to consider same influence for estimated and actual value of traffic experienced.

### B. Evaluation Metrics

With performance-centric measures, we would like to gauge how effective our policies are to attain desired performance. Whereas in case of endurance, we have measured effectiveness of the write traffic control at the endurance-sensitive device. Therefore, different metrics are considered for measuring data loss, performance and endurance as described below.

**Data loss metric**:

- Average Data loss Risk: Average amount of modified data lying in DRAM, which can be lost due to system failure, is measured in .

**Performance centric metrics**:

- Average Read Latency ($Avg_{\mathbf{RL}}$): Weighted average of DRAM read latency ($L_D$), ENVM average read latency ($L_{NR}$) and SSD average read latency ($L_S$) together with latency ($L$) due to DRAM eviction, writeback to SSD from Optane defined as Equation 6 and measured in microseconds.

$$\begin{aligned} Avg_{\mathbf{RL}} &= H_D * L_D + H_N * L_{NR} \\ &+ (1 - H_D - H_N) * L_S + L, \end{aligned} \tag{6}$$

where $H_D$ and $H_N$ are the number of time DRAM and ENVM are accessed.

- Average Write Latency ($Avg_{\mathbf{WL}}$): Weighted average of DRAM access latency ($L_D$), ENVM write latency ($L_{NW}$) and SSD average access latency ($L_S$) together with latency ($L$) due to DRAM eviction, writeback to SSD from Optane defined as Equation 7 and measured in microseconds.

$$\begin{aligned} Avg_{\mathbf{WL}} = H_D * L_D + H_N * L_{NW} \\ +(1 - H_D - H_N) * L_S + L, \end{aligned} \tag{7}$$

  where $H_D$ and $H_N$ are the number of time DRAM and ENVM are accessed.
- Tail Read Latency: 99% latency experienced during read operation for various sizes of requested data and measured in microseconds.

**Endurance centric metrics**:

- ENVM Write: Amount of data writes on ENVM and measured in GB.
- SSD Write: Amount of data writes on SSD and measured in GB.

### C. Dataset Characteristics

In order to study endurance and data loss issues relevant to HPC, we sought storage system traces that are relatively write heavy and have significant variation in write sizes. Unfortunately, most of the publicly available traces or the benchmarks such as SpecSFS do not provide trace with these characteristics. For example, we considered the five different SRCMap storage workloads in [25] and available at SNIA's website. However, due to absence of large variation in request size, they are not interesting for testing and thus not reported results for this workload here. The Systor 2017 trace [7] is a much longer trace and consist of wide variations of request size. This workload represents virtual desktop infrastructure (VDI) where a number of users log in remotely and use typical desktop applications (e.g., Microsoft Word, Excel, etc.). A separate VM is created for each user and is given resources depending on the user's requirements. The workload is generally read intensive but has huge variations of write request size from 512 bytes up to 128KB. This trace considers

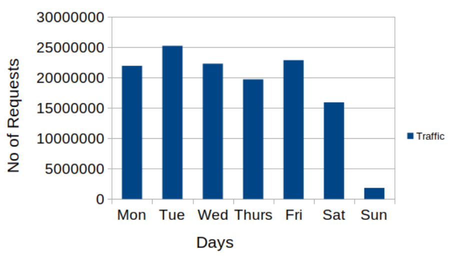


Fig. 5: VDI Average Traffic Characteristics

2706 I/O traces from 6 different block storage devices (LUNs 0,1,2,3,4, and 6), each of them experiencing almost the same workload behavior. Anyone of them can be representative of the set of LUNs and we have considered LUN0 in this context. The typical week long experienced traffic characteristics of this trace are shown in Fig. 5. Based on the number of requests experienced per day, a day long traffic can be classified as busy, moderate and low traffic day.

### D. Results and Discussion

We have evaluated our proposed policies by measuring the evaluation metrics for performance, endurance and data loss. Also, we have compared our proposed policies with a base policy configuration comprising of DRAM as the cache to the SSD without any ENVM.

*1) Data loss Evaluation:* The average data loss risk is measured in KB during busy, moderate and low traffic days for all policies. It is clear from Fig. 6 that both policies I and II experience equal average data loss risk. In contrast, Policy III has zero average data loss risk due to immediate data persist.

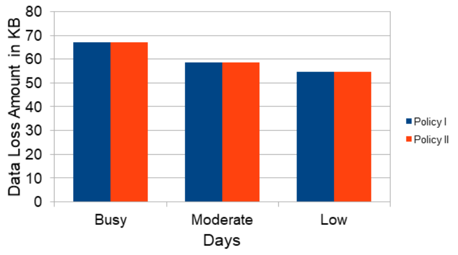


Fig. 6: VDI Average Data Loss Characteristics

*2) Performance Evaluation:* The figures 7 and 8 represent average read and write latency for three different policies during busy, moderate and low traffic conditions. The comparative study uses 1 TB QLC and Optane of size 32Gb, which is $\sim 3\%$ of the size of the QLC.

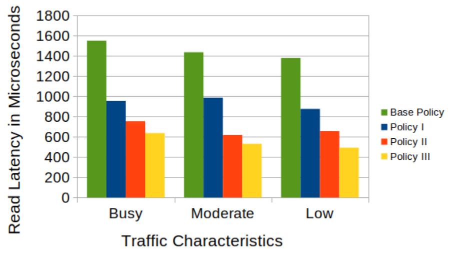


Fig. 7: VDI Average Read Latency

The Fig. 7 shows that the absence of ENVM results in much higher read latency than any of our proposed policies. The frequent access to SSD on DRAM miss is the reason such high latency is observed. Policy I improves the latency by caching the DRAM evicted elements, whereas Policy II improves it even more by caching the relevant DRAM evicted elements that lowers the number of unnecessary eviction from ENVM. Among our proposed policies, the Policy III works the best, improving latency by 58% as compared to the base policy for busy traffic, as it leverages the whole DRAM and a significant portion of ENVM to cache the read data. Policy III also eliminates data loss by immediately persisting all writes. In contrast, even though Policy II gets to use more ENVM for read data (by virtue of not having to persist every write), it falls short because of modified but infrequently used data lying in DRAM. This aspect obviously depends on the workload characteristics and speed of DRAM relative to that of Optane.

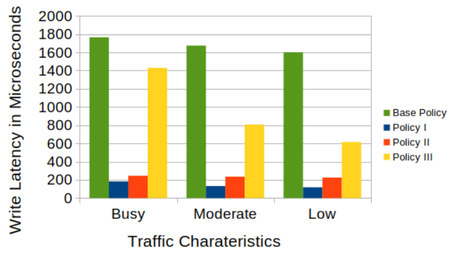


Fig. 8: VDI Average Write Latency

The Fig. 8 shows that the absence of ENVM brings huge write latency as the modified data upon eviction from DRAM is written directly to SSD. For Policy I and II, all writes are done at DRAM and hence results in much lower write latency than Policy III, where small writes are persisted on ENVM and large writes on SSD. During the busy period, large writes appear more than moderate and low traffic days. Therefore, Policy III experiences more write latency during busy traffic than moderate and low. However, Policy III improves 28% latency during busy day than base policy. Policy II experiences more latency than I because of a greater number of writebacks from ENVM to SSD.

As tail read latency for different request sizes can provide more insight, we have measured 99% read latency experienced for three different policies during busy traffic. The request sizes in the workload are starts from 512B and go up to 128KB. Here we have considered request size multiple of 4KB. The 99% read latency for various request size for three different policies are shown in figures 9. All three policies yield approximately the same read tail latency, which argues for policy III since it eliminates data loss. The tail latency without ENVM (not shown) will obviously correspond to going directly to SSD and will be very high. The tail latency obviously increases with the transfer size in all cases.

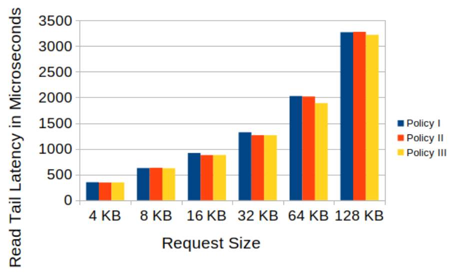


Fig. 9: VDI 99% Tail Latency for Read Request

*3) Endurance Evaluation:* Under the base policy, the VDI workload can degrade the endurance rating of the SSD due to large write bursts. For this, we measure the number of writes that exceed the daily write limit for the SSD. We call these as **critical writes** and measured in GB. Writes below this limit are called **safe writes** and measured in GB. Fig. 10 shows the safe writes and critical writes for the base policy. This figure depicts that a significant number of critical writes are experienced during busy and moderate traffic. It is desired to limit the amount of write data on SSD.

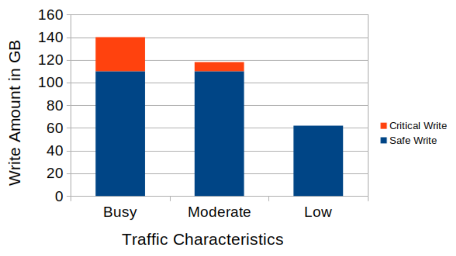


Fig. 10: VDI SSD write for Base Policy

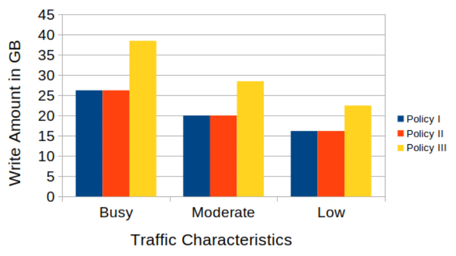


Fig. 11: VDI SSD write amount for different Policies

Fig. 11 shows the observed SSD writes for all the three policies. In Policy I and II, all writes are initially done at DRAM and frequent writes on same DRAM elements reduces SSD writes though risk of data loss remains. For both of these policies, the ENVM accumulates all or selective data evicted from DRAM and eventually passes modified data to the SSD during writeback. In case policy III, write budget scheme is applied on SSD and large writes always go to the SSD. These are in addition to the writeback from ENVM. Therefore, amount of SSD writes are more than Policy I and II and does not exceed the safe write limit showed in Fig. 10.

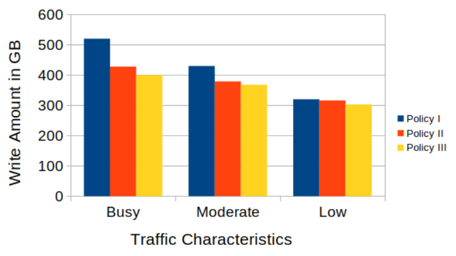


Fig. 12: VDI ENVM write amount for different Policies

Fig. 12 shows the total number of ENVM writes for all three policies. In case of policy I and II, ENVM writes are much higher than Policy III. In case of policy I, a larger number of writes on ENVM happen because of caching all DRAM evicted read and write data. Policy II does better but still accumulates both small and large writes in ENVM on eviction from the DRAM. Policy III works the best and lowers writes by 22% than Policy I by persisting only small writes and storing relevant read from eviction from DRAM. In fact, all three policies violates the ENVM endurance rating, which is 110 GB per day for 32 GB Optane. Thus, to maintain endurance rating for both ENVM and SSD, one needs a careful balancing act which can be achieved not only by Policy variation but also by controlling the size of the ENVM and SSD. In particular, by making the ENVM about 12% of the SSD capacity, we can maintain endurance of both for all policies for this workload.

Given the lack of publicly available HPC storage traces, we modified the Systor Trace to incorporate large write feature, typical of HPC. The write sizes in Systor trace are magnified to two (2X), four(4X) and sixteen(16X) times. Fig. 13a shows the amount of safe and critical writes on SSD due to different large writes during busy period when base policy is applied. It is clear that the SSD endurance rating is affected substantially for 4X and 16X cases under the baseline policy.

Next we examine how three policies impact SSD writes for different magnified write traces. This is shown in Fig. 13b. For 2X magnified trace, all policies still results in safe write on SSD. However, for 4X modified trace, policy III results in small amount of critical write. In case of 16X magnified trace, all these three policies results in huge amount of critical writes. This suggests to have a bigger amount of SSD with high endurance rating to accommodate large amount of writes.

To examine the behavior of different policies, we also consider Samsung 860 EVO TLC SSD along with DRAM without considering ENVM. The capacity of TLC is restricted to 75% capacity of 1TB QLC assuming the same number of cells. In case of the VDI workload, the endurance is not violated by persisting all writes on TLC SSD directly. The average read and write latency experienced, as in the Fig. 13c, is much more than the latency of QLC pairing with ENVM as explained in figures 7 and 8. Thus, inserting Optane in the storage hierarchy is still useful from the perspective of performance while limiting data loss.

As we approach towards more bits per cell and cell types in NAND Flash technology, 5 bits per cell (PLC), it is interesting to investigate the estimated ENVM need to maintain the endurance rating. We simulated policy III during considered busy write traffic trend for different flash technologies to compare the estimated ENVM sizes for absorbing critical writes corresponding to SLC, MLC, TLC, QLC and PLC based on the values publicly hosted (*https://blocksandfiles.com/2019/08/07/penta-level-cell-flash/*). The capacities of SLC, MLC, TLC and PLC are considered as 25%, 50%, 75% and 125% capacity respectively of 1TB QLC assuming the same number of cells. The P/E cycles for different flash technologies and cell types are considered as publicly hosted. The ENVM required for different flash technologies and cell types are represented in the Fig. 14b. For different flash technologies, SLC requires smallest amount of ENVM to maintain the overall endurance compared to MLC, TLC, QLC and PLC. With different cell types, 1Xnm requires more ENVM for SLC, TLC, QLC and PLC than 2Xnm and 3Xnm. The Fig. 14a shows the p/e cycle for different flash technologies and cell types. Though PLC gets much more blocks to be written than SLC, The sharp drop in P/E cycle from SLC to PLC is not compensated by the relative increase in the number of blocks to be written and thus overall TBW drops consistently with the technology progress from SLC to PLC. In other words, so long as the SSD is large enough to hold workloads data, having more bits/cell hurts in terms of endurance. However, the required ENVM comes with price. The Fig. 14c represents the estimated price associated with required amount of ENVM and same capacity of different SSD flash technologies. It is noticeable that the combination of MLC and ENVM is much lower than the other combinations.

## VII. Conclusions

In this paper, we presented the opportunities and challenges in using emerging NVM technologies to manage the tradeoffs between performance and endurance characteristics of low endurance SSD based high-performance storage. The availability of high speed technologies like Intel Optane also allows immediate persist of modified data and thereby eliminate (or at least substantially reduce) data loss potential and also simplifies resilience. We discussed several ways of combining NAND flash technologies and ENVM technologies. The immediate persist actually reduces the read latency, which is often more critical than the write latency. Of course, the overall write latency goes up (since all writes must eventually go to the SSD) but this is not important since a write completion response can be returned to the application as soon as the ENVM persist is done. The high-performance computing applications often involve large writes and the real price of immediate persist is in potentially degrading the endurance of the ENVM. However, since EVNM has very high PE cycle rating, a somewhat larger size can often address the problem.

In the future, we plan to explore other policies for storage hierarchies involving SSD and ENVM, including byte addressable ones, and will also explore performance for a larger set of workloads.

## References

[1] Feng Chen, David A Koufaty, and Xiaodong Zhang. Hystor: making the best use of solid state drives in high performance storage systems. In *Proceedings of the international conference on Supercomputing*, pages 22–32, 2011.

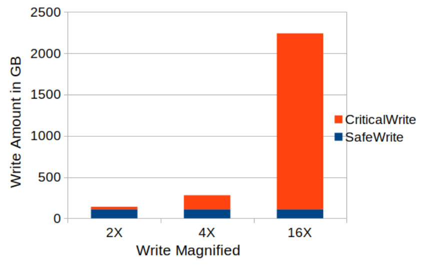
[2] Ziqi Fan, Fenggang Wu, Dongchul Park, Jim Diehl, Doug Voigt, and David HC Du. Hibachi: A cooperative hybrid cache with nvram and dram for storage arrays. In *Proc. 33rd Int. Conf. Massive Storage Syst. Technol.(MSST)*, 2017.

[3] Andrés Amaya García, René de Jong, William Wang, and Stephan Diestelhorst. Composing lifetime enhancing techniques for non-volatile main memories. In *Proceedings of the International Symposium on Memory Systems*, pages 363–373. ACM, 2017.

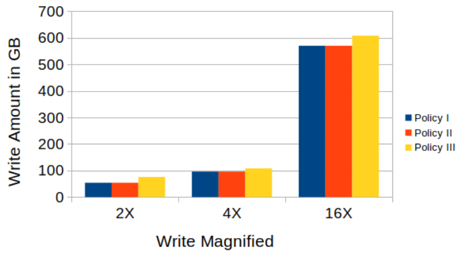
[4] Binny S Gill and Dharmendra S Modha. Sarc: Sequential prefetching in adaptive replacement cache. In *USENIX Annual Technical Conference, General Track*, pages 293–308, 2005.

[5] Kiran Kumar Gunnam. Wear leveling in non-volatile memories, October 12 2017. US Patent App. 15/627,135.

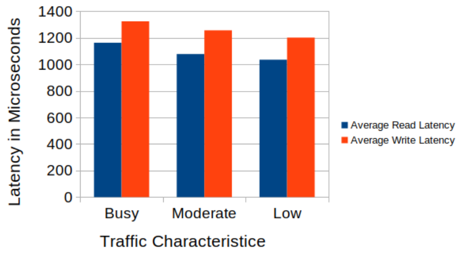
[6] Donghun Koo, Jik-Soo Kim, Soonwook Hwang, Hyeonsang Eom, and Jaehwan Lee. Adaptive hybrid storage systems leveraging ssds and hdds in hpc cloud environments. *Cluster Computing*, 20(3):2119–2131, 2017.



(a) SSD writes for Base Policy with Magnified Trace

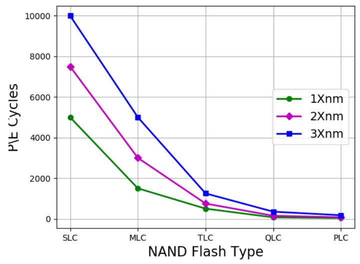


(b) SSD writes for different policies with Magnified Trace

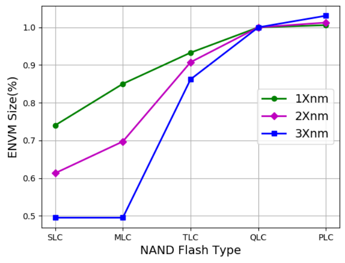


(c) Avg. Read & Write Latency for TLC with Systor Trace

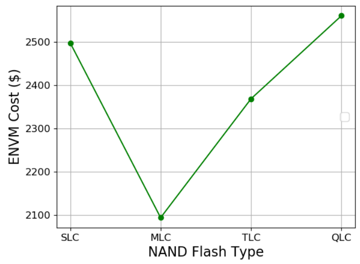
Fig. 13: Results for Write Magnified Trace and TLC SSD with Systor Trace



(a) P/E Cycles vs. bits/cell



(b) Estimated ENVM size relative to QLC size to maintain same Endurance



(c) Estimated ENVM cost to maintain same Endurance

Fig. 14: ENVM vs. Flash Technologies

[7] Chunghan Lee, Tatsuo Kumano, Tatsuma Matsuki, Hiroshi Endo, Naoto Fukumoto, and Mariko Sugawara. Understanding storage traffic characteristics on enterprise virtual desktop infrastructure. In *Proceedings of the 10th ACM International Systems and Storage Conference*, SYSTOR '17, pages 13:1–13:11, New York, NY, USA, 2017. ACM.

[8] Mingwei Lin, Riqing Chen, Li Lin, Xuan Li, and Jingchang Huang. Buffer-aware data migration scheme for hybrid storage systems. *IEEE Access*, 6:47646–47656, 2018.

[9] Ning Liu, Jason Cope, Philip Carns, Christopher Carothers, Robert Ross, Gary Grider, Adam Crume, and Carlos Maltzahn. On the role of burst buffers in leadership-class storage systems. In *012 ieee 28th symposium on mass storage systems and technologies (msst)*, pages 1–11. IEEE, 2012.

[10] Yang Liu, Raghul Gunasekaran, Xiaosong Ma, and Sudharshan S Vazhkudai. Server-side log data analytics for i/o workload characterization and coordination on large shared storage systems. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 819–829. IEEE, 2016.

[11] Jerry Lo, Lan D Phan, and Cliff Pajaro. Managing wear leveling and garbage collection operations in a solid-state memory using linked lists, April 29 2014. US Patent 8,713,066.

[12] Youyou Lu, Jiwu Shu, Long Sun, and Onur Mutlu. Improving performance and endurance of persistent memory with loose-ordering consistency. *IEEE Transactions on Parallel and Distributed Systems*, 2017.

[13] Huong Luu, Marianne Winslett, William Gropp, Robert Ross, Philip Carns, Kevin Harms, Mr Prabhat, Suren Byna, and Yushu Yao. A multiplatform study of i/o behavior on petascale supercomputers. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, pages 33–44, 2015.

[14] Nimrod Megiddo and Dharmendra S Modha. Arc: A self-tuning, low overhead replacement cache. In *FAST*, volume 3, pages 115–130, 2003.

[15] Junpeng Niu. Hybrid system analysis and design for scale-out storage environments, 2018. PhD Thesis, DR-NTU, Nanyang Technological University Library, Singapore.

[16] Junpeng Niu, Jun Xu, and Lihua Xie. Hybrid storage systems: A survey of architectures and algorithms. *IEEE ACCESS*, 6:13385–13406, 2018.

[17] Moinuddin K Qureshi, Michele M Franceschini, Ashish Jagmohan, and Luis A Lastras. Preset: improving performance of phase change memories by exploiting asymmetry in write times. *ACM SIGARCH Computer Architecture News*, 40(3):380–391, 2012.

[18] Reza Salkhordeh, Onur Mutlu, and Hossein Asadi. An analytical model for performance and lifetime estimation of hybrid dram-nvm main memories. *arXiv preprint arXiv:1903.10067*, 2019.

[19] Xuanhua Shi, Ming Li, Wei Liu, Hai Jin, Chen Yu, and Yong Chen. Ssdup: a traffic-aware ssd burst buffer for hpc systems. In *Proceedings of the international conference on supercomputing*, pages 1–10, 2017.

[20] Xuanhua Shi, Wei Liu, Ligang He, Hai Jin, Ming Li, and Yong Chen. Optimizing the ssd burst buffer by traffic detection. *arXiv preprint arXiv:1902.05746*, 2019.

[21] Pradeep Subedi, Philip E Davis, Juan J Villalobos, Ivan Rodero, and Manish Parashar. Using intel optane devices for in-situ data staging in hpc workflows. *arXiv preprint arXiv:1807.09651*, 2018.

[22] Ming-Yu Tai, Subhash Balakrishna Pillai, Yung-Li Ji, and Haining Liu. Memory apparatus and methods thereof for write amplification aware wear leveling, July 5 2018. US Patent App. 15/396,548.

[23] Mojtaba Tarihi, Hossein Asadi, Alireza Haghdoost, Mohammad Arjomand, and Hamid Sarbazi-Azad. A hybrid non-volatile cache design for solid-state drives using comprehensive i/o characterization. *IEEE Transactions on Computers*, 65(6):1678–1691, 2016.

[24] Mahdi Torabzadehkashi, Ali Heydarigorji, Siavash Rezaei, Hosein Bobarshad, Vladimir Alves, and Nader Bagherzadeh. Accelerating hpc applications using computational storage devices. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1878–1885. IEEE, 2019.

[25] Akshat Verma, Ricardo Koller, Luis Useche, and Raju Rangaswami. Srcmap: Energy proportional storage using dynamic consolidation. In *FAST*, volume 10, pages 267–280, 2010.

[26] Chundong Wang and Weng-Fai Wong. Observational wear leveling: an efficient algorithm for flash memory management. In *Proceedings of the 49th Annual Design Automation Conference*, pages 235–242. ACM, 2012.