Towards an Unwritten Contract of Intel Optane SSD

Kan Wu, Andrea Arpaci-Dusseau and Remzi Arpaci-Dusseau
University of Wisconsin-Madison

Abstract

New non-volatile memory technologies offer unprecedented performance levels for persistent storage. However, to exploit their full potential, a deeper performance characterization of such devices is required. In this paper, we analyze a NVM-based block device – the Intel Optane SSD – and formalize an "unwritten contract" of the Optane SSD. We show that violating this contract can result in 11x worse read latency and limited throughput (only 20% of peak bandwidth) regardless of parallelism. We present that this contract is relevant to features of 3D XPoint memory and Intel Optane SSD's controller/interconnect design. Finally, we discuss the implications of the contract.

1 Introduction

New NVM technologies provide unprecedented performance levels for persistent storage. Such devices offer significantly lower latency than Flash-based SSD and can be a cost-effective alternative to DRAM. One excellent example is Intel's 3D XPoint memory [1,25] from Intel and Micron, available on the open market under the brand name Optane [11]. It is available in various form factors, including Optane memory [9] (a caching layer between DRAM and block device), Optane SSD [10] (a block device), and Optane DC Persistent Memory [8]. Among these devices, the Optane SSD is currently the most cost-effective and widely available option.

Optane SSD offers numerous opportunities for applications. For example, Intel's Memory Direct Technology (IMDT) [7] enables the use of Optane SSD as a DRAM alternative. Use cases of IMDT/Optane SSD include Memcached [4], Redis [5], and Spark [6]. Evaluations of those scenarios demonstrate the potential role of Optane SSD as a cost-effective alternative of DRAM. Optane SSDs are also deployed to support key workloads in Facebook [22, 23], both as a caching layer between DRAM and Flash SSD for RocksDB and for crucial workloads such as machine learning. According to Eisenman *et al.* [23], Facebook stores embeddings of trained neural networks on Optane SSDs.

Using new technology effectively requires understanding its performance and reliability characteristics. For traditional devices, such as hard-drives and Flash-based SSDs, these are well known [16, 17, 20, 21, 26, 30, 33]. However, for new devices like the Optane SSD, much remains unclear, and is thus the focus of our work.

In terms of immediate performance, we summarize six rules that are critical for Optane SSD users. First, to obtain low latency, users should issue small requests ($\geqslant$ 4 KB) and keep a small number of outstanding IOs (**Access with Low Request Scale** rule). Second, different from HDD/Flash SSDs, Optane SSD clients should not consider sequential workloads special (**Random Access is OK** rule). Third, to avoid contention among requests, clients should not issue parallel accesses to a single chunk (4KB) (**Avoid Crowded Accesses** rule). Fourth, to achieve optimal latency, the user needs to control the overall load of both reads and writes (**Control Overall Load** rule). Fifth, to exploit the bandwidth of Optane SSD, clients should never issue requests less than 4KB (**Avoid Tiny Accesses** rule). Sixth, to get the best latency, requests issued to Optane SSD should align to eight sectors (**Issue 4KB Aligned Requests** rule). Finally, when serving sustained workloads, there is no cost of garbage collection in Optane SSD (**Forget Garbage Collection** rule). Overall, these rules are relevant to features of 3D XPoint memory and Optane SSD's controller/interconnect design.

Rule	Impact	Metric	Cause
Access with Low Request Scale	11x	Latency	SSD Controller/Interconnect
Random Access is OK	7x (vs. Flash)	Latency & Throughput	3D XPoint & Controller
Avoid Crowded Accesses	4.6x	Latency & Throughput	SSD Controller/Interconnect
Control Overall Load	5x (vs. Flash)	Latency	3D XPoint Memory
Avoid Tiny Accesses	5x	Throughput	SSD Controller/Interconnect
Issue 4KB Aligned Requests	1.2x	Latency	SSD Controller/Interconnect
Forget Garbage Collection	15x (vs. Flash)	Sustained Throughput	3D XPoint & Controller

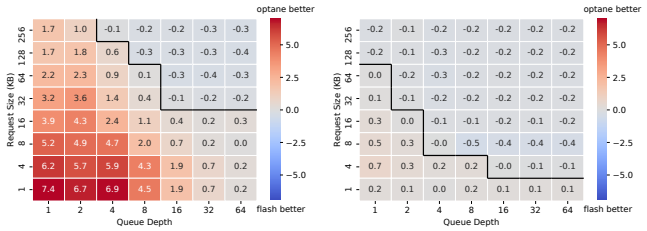
Table 1: Performance Impact of Rule Violations

The unwritten contract provides numerous implications for systems and applications. According to the Access with Low Request Scale rule and the Control Overall Load rule, the design of heterogeneous storage systems including Optane SSD and Flash SSD must be carefully considered. Many rules (e.g., Random Access is OK) present opportunities and new challenges to external data structure design for Optane SSD. Finally, the Random Access is OK rule may enable new applications on Optane SSD that don't work well on existing technologies.

The rest of this paper is structured as follows. We describe the unwritten contract and explain the insights for each rule in Section 2. We provide the implications of the unwritten contract in Section 3, conclude in Section 4 and point to potential research directions in Section 5.

2 The Unwritten Contract of Optane SSD

In this section, we define the rules of the unwritten contract. We offer experiments (available at https://github.com/sherlockwu/OptaneBench) to infer



(a) Read (b) Write

Figure 1: Avg Latency of random workloads, Optane vs. Flash

the rules. We summarize the impact and cause of each rule in Table 1. We begin with six rules related to immediate performance like latency and throughput. We then present rules for sustainable performance.

2.1 Access with Low Request Scale

The Optane SSD is based on 3D XPoint memory which is said to provide up to 1,000 times lower latency than NAND Flash [2]. Does 3D XPoint memory always lead to better performance for Optane SSD compared to Flash SSD? We answer this question with our first rule: to obtain low latency, Optane SSD users should issue small requests and maintain a small number of outstanding IOs. This rule is needed not only to extract low latency but also enough to exploit the full bandwidth of the Optane SSD.

We uncovered this rule when quantitatively comparing Intel Optane SSD 905P (960GB) with a "high-end" Flash SSD: Samsung 970 Pro (1TB) [12]. From our experiments, we found that **Optane SSD does not show improvement for workloads with many outstanding requests**.

We compare Optane and Flash SSDs with random read-only and write-only workloads. Each workload has two variables: request size and queue depth (QD) (or number of in-flight I/Os). Figure 1 compares the two devices in terms of average access latency. The temperature T in each rectangle shows the scaled difference between the latencies of the two systems; $T > 0$ indicates Optane has smaller latency, while $T < 0$ indicates Flash SSD has smaller latency. Specifically, $T = \frac{L_{higher} - L_{lower}}{L_{lower}}$. As shown, Optane SSD and Flash SSD outperform each other in different cases.

For read-only workloads, Optane SSD has lower latency than Flash SSD when request size and queue depth are small. Specifically, when the request size $\leqslant$ 16KB, Optane SSD is better than Flash SSD. Thanks to 3D XPoint memory's low access latency, the read latency from Optane SSD can be 8.4x faster than Flash SSD. However, when QD$>$ 8 and request size$>$ 16KB, Flash SSD achieves lower latency, by as much as 40%. This difference occurs because the latency of Optane SSD increases linearly with higher queue depths (e.g., the latency of a 4KB random read with QD$=$ 64 is 8x slower than QD$=$ 8 and is 11x slower than QD$=$ 1).

Similarly, for write-only workloads, Optane SSD has lower latency for small requests and low QD, while Flash SSD is

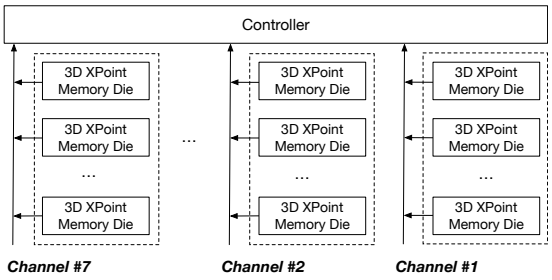
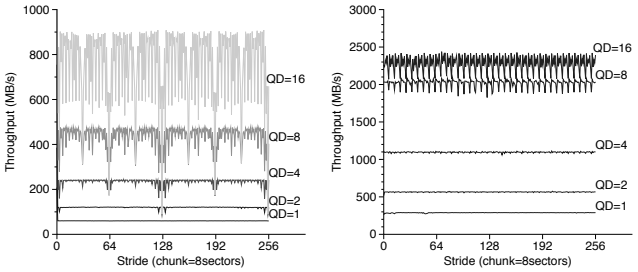


Figure 2: RAID-like Architecture



(a) Flash SSD (b) Optane 905P SSD

Figure 3: Detecting Interleaving Degree

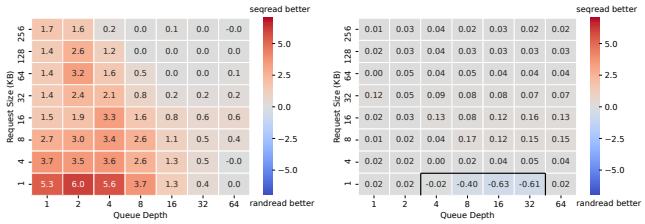
again better in the opposite cases; however, the difference for writes is not as high as for reads (Optane is up to 1.7x faster than Flash). This result is due to Flash SSDs log-structured layout and buffering optimizations. Flash SSD outperforms Optane SSD when request size$>$ 4KB and QD$>$ 2, which includes most of our tested workloads.

The device's internal parallelism dictates its behavior when serving workloads with high request scale. We are thus motivated to uncover the internals of the Optane SSD. Like Flash SSD, Optane SSD utilizes a RAID-like organization of memory dies (Figure 2). Through a fine-grained experiment [18, 19] we examine a critical parameter for internal parallelism: the interleaving degree, or number of channels. We maintain a read stream with stride S from the devices, where S is the distance between two consecutive chunks (a chunk is 4KB or 8 sectors as shown in Section 2.6). Figure 3 presents the throughput of workloads with various strides. An individual line represents workloads with the same QD.

Optane SSD has a significantly smaller interleaving degree (7) than Flash SSD (128). In Figure 3, the distance between the lowest dips in each line indirectly indicates the interleaving degree of the device. For Optane SSD, we observe the pattern is 7, though the difference between dips is not visible until QD$=$ 8. Our finding agrees with the hardware description of Optane SSD [3]: it has a controller connected to seven channels, each of which is connected to memory dies.

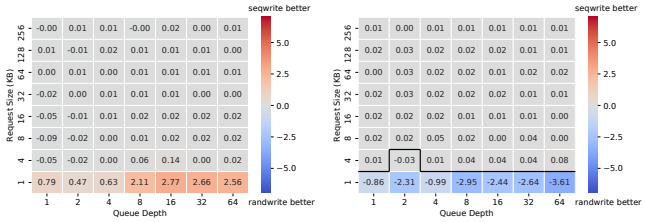
The tested Flash SSD reaches its lowest throughput every 128 chunks. For high queue depths (e.g., 16), it presents a richer pattern with dips at different levels, indicating copious levels of parallelism (channel, package and die).

Overall, **Optane SSD has limited internal parallelism, compared to Flash SSD**. This characteristic explains the



(a) Flash-based SSD (b) Optane 905P SSD

Figure 4: Avg Read Latency Random vs. Sequential



(a) Flash-based SSD (b) Optane 905P SSD

Figure 5: Avg Write Latency Random vs. Sequential

high latency of Optane SSD for workloads with a large request scale, and supports the need to access the Optane SSD with a low request scale.

The limited internal parallelism of the Optane SSD impacts its throughput in two ways. First, the tested Flash SSD achieves larger maximum read ($3500MB/s$) and write ($2700MB/s$) bandwidth than the Optane SSD ($2500MB/s$); it outperforms Optane SSD serving workloads with large request scale. Flash SSD's richer internal parallelism enables it to serve more parallel requests. Second, Optane SSD can achieve peak throughput when serving small requests with low queue depth. This is due to Optane SSD's limited internal parallelism which requires only a small number of requests to utilize all of its resources. Hence, the rule to access the device at a low scale not only guides users to obtain low access latency but also is enough to achieve full bandwidth.

The influence of contention in Optane SSD is modest compared to that in Flash SSD. The dips in Figure 3 are due to concurrent requests contending for shared resources (e.g., channels). In Flash SSD, contention significantly restricts overall throughput; for example, with $QD = 16$, $S = 127$ chunks, read throughput is 88 MB/s, which is only 6% of the maximum throughput with the same queue depth. Although parallel requests to Optane SSD can also introduce contention (limiting throughput to within 86% of maximum), this influence is much less than that in Flash SSD.

2.2 Random Access is OK

With hard drives or SSDs, clients often expect better performance from sequential than random accesses. With Optane SSD, this is no longer true. **Optane SSD is a random access block device**, where clients can observe the same performance for random and sequential workloads.

We study Optane SSD's average latency when serving random and sequential workloads; throughput and tail latency yield similar results. Each workload maintains four worker threads, while each thread issues IOs randomly or in a sequential stream. As shown in Figures 4 and 5, on Optane SSD, for requests $>$ 1KB, random and sequential workloads achieve comparable performance. The difference between sequential and random latency is within 17% for reads and within 5% for writes.

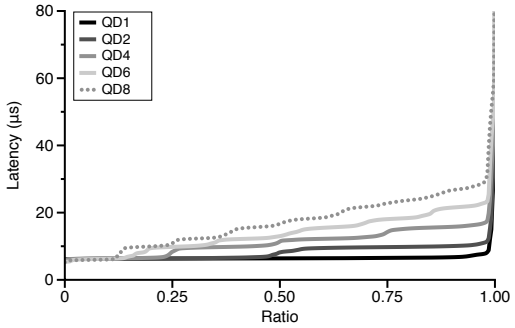


Figure 6: Parallel Sector Reads to a Chunk(Lat. Distribution)

In contrast, Flash SSD prefers sequential over random reads, especially at a low request scale; sequential reads can be 7x faster. Flash SSD achieves much better sequential performance due to prefetching and simplified address translation. For writes, Flash SSD presents similar random and sequential latency due to log-structuring. However, as we will show in Section 2.7, log-structuring introduces significant overhead when the device fills; Optane does not have such concerns.

The Random Accesse is OK rule in Optane SSDs occurs due to the ability to perform in-place updates in 3D XPoint memory. In Optane SSD, there is no difference in address translation costs for random versus sequential workloads; in Section 2.7, we will show that the mapping policy in Optane SSD is based on logical addresses. In addition, as indicated by our read latency study, there is no prefetching for sequential reads within Optane SSD.

Workloads with 1KB requests are special on Optane SSD compared both to other request sizes and to Flash SSD. We investigate this in the next rule.

2.3 Avoid Crowded Accesses

The Optane SSD contains shared resources (e.g., channels). To avoid contention, the Avoid Crowded Accesses rule dictates that clients of Optane SSD should never issue parallel accesses to a single chunk (4KB). We uncover this rule by investigating the 1KB workload performance shown in Figure 4 and 5. In Optane SSD, sequential 1KB accesses can increase latency by 63% for reads and by 3.6x for writes, compared to random 1KB accesses.

We study the difference between random and sequential accesses for small requests by performing parallel accesses to a single 4KB chunk. In this experiment, we randomly choose chunks from the device, then issue P parallel reads to different sectors within one chunk. Figure 6 presents the distribution

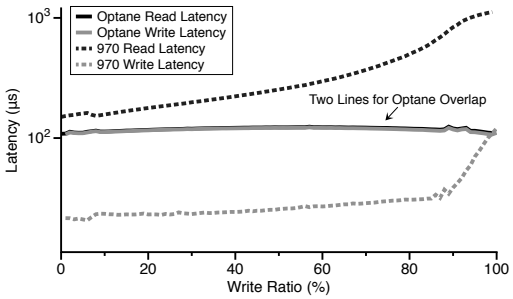


Figure 7: Latency of Mixed Reads and Writes

of latencies for different values of P. We observe a "stair" pattern for QD>1. For each line, the number of levels equals the queue depth and the steps occur at evenly-spaced intervals.

This pattern indicates queuing and/or contention across the issued parallel requests. **Parallel small requests to a single chunk introduce contention**. This experiment illustrates why sequential 1KB workloads have worse performance than random 1KB workloads: although random 1KB accesses *may* introduce contention, sequential 1KB accesses *must* introduce contention.

2.4 Control Overall Load

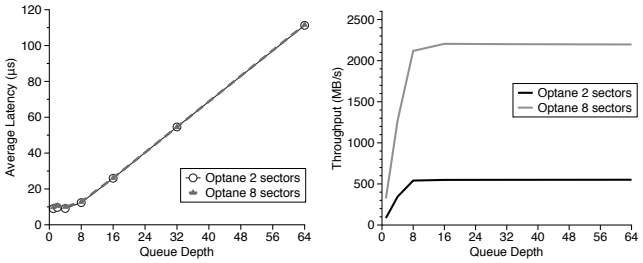
To achieve optimal latency from Optane SSD, the client must control the overall load of both reads and writes. This rule indicates distinct performance characteristics between Optane SSD and Flash SSD.

We discover this rule by looking into the performance of Optane SSD serving mixed reads and writes. In the experiment, we issue random 4KB requests, varying the percentage of writes from 0% to 100%, with QD$=64$ (large enough to achieve full throughput for both Optane SSD and Flash SSD). Figure 7 shows the access latency of Optane SSD and Flash SSD. **Within Optane SSD, reads and writes are treated equally.** Specifically, on Optane SSD, each type of request is served with the same latency and the latency is related to the overall load, not to the percentage of writes.

Flash SSD exhibits distinct characteristics for read- versus write-dominated workloads. On the left side of Figure 7, for Flash SSD, the read latency is similar to that in read-only workloads with the same queue depth (38% slower than Optane SSD). However, the write latency is similar to that of a pure-write workload with very low queue depth (and only 19% of that on Optane SSD). With an increasing number of writes, reads to Flash SSD achieve poor latency due to the influence of writes; when the workload is write-dominated, read latency can be as high as $1.1ms$ (10x Optane access latency).

2.5 Avoid Tiny Accesses

Does the byte-addressabilily of 3D XPoint memory enable efficient tiny accesses to Optane SSD? We answer this question with our rule to Avoid Tiny Accesses: to exploit bandwidth of the SSD, the client must not issue requests less than 4KB.



(a) Latency (b) Throughput

Figure 8: Performance of Small Requests

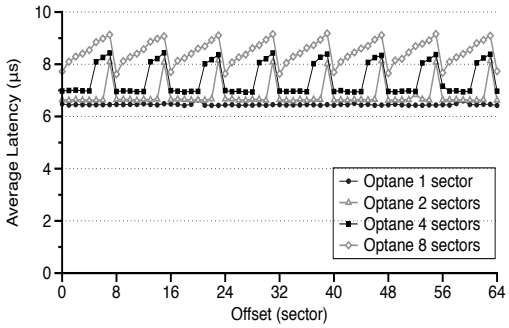


Figure 9: Identifying Influence of Misalignment

Figure 8 shows the latency and throughput of random reads less than 4KB, with separate lines for two sectors and eight sectors requests. As shown, the latency of two sector requests is the same as eight sector (4KB) requests. However, the throughput of tiny requests is limited by the maximum IOPS supported by Optane SSD (575K); for 1KB requests, throughput is only 20% of the full bandwidth of the device. Given these two results, **there is no benefit in issuing requests smaller than 4KB to Optane SSD**.

2.6 Issue 4KB Aligned Requests

To achieve the best latency, requests issued to Optane SSD should always align to eight sectors. We present the difference between aligned and misaligned requests. In the experiment, we measure the latency of individual read requests (QD$=1$); each read is issued to a position $A+offset$, where A is a random position aligned to 32KB and *offset* is a 512-byte sector within that 32KB.

Figure 9 shows the read latency of requests issued to different offsets, averaged over a half million requests to the same offset. Each line represents a workload with a different request size. In contrast to what one might expect given 3D XPoint's byte-addressability, **Optane SSD favors aligned requests.** Requests of one sector have the same latency no matter the offset, and larger requests aligned to eight sectors always get the lowest latency. For a request crossing the boundary of 4KB, its latency is linearly correlated to the part it issues to the second chunk after the boundary. The difference between the high and low latencies of 4KB requests is 21%. Note that the eight-sector chunk here is not related to a concept like a page or block in Flash SSD.

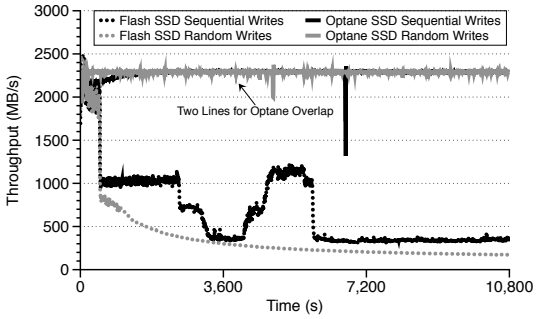
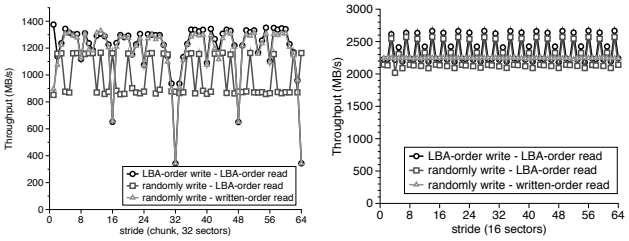


Figure 10: Throughput of Sustained Writes



(a) Flash-based SSD (b) Optane 905P SSD

Figure 11: Detecting Mapping Policy

2.7 Forget Garbage Collection

We now study the long-term performance of Optane SSD. First, we explore its performance when the device gets full. According to our experiments, there is no need to worry about garbage collection in Optane SSD.

We examine sustained 4KB random and sequential writes on Optane SSD and Flash SSD over three hours. The device is completely unmapped using the trim command before each experiment; the two devices tested share a similar capacity (960GB vs. 1TB). We maintain QD$=$ 32 for each workload.

Figure 10 presents sustained write performance. For Flash SSD, after the device becomes full, write throughput drops significantly because subsequent writes constantly trigger garbage collection. After about 6000 seconds, write throughput stabilizes: sequential throughput is around $350MB/s$ and random throughput is $170MB/s$ (only 7% of the maximum throughput). The throughput of sequential writes is better than random because of lower garbage collection cost. **Different from Flash SSD, Optane SSD maintains maximum throughput for sustained writes**. The flat throughput for Optane SSD indicates no cost for garbage collection.

Finally, we study the mapping policy (LBA→PBA) in Optane SSD by comparing three workload variations. The first is the same workload as for the interleaving experiments in Figure 3: blocks are first written in logical address order and then read back in that same LBA order (LBA-order write:LBA-order read). The second workload preconditions the working zone with random writes (random write:LBA-order read). The third workload preconditions with random writes, but then reads in the order in which the chunks were written (random write:written-order read). Figure 11 shows the throughput of the three workloads.

For Flash SSD, when the read order does not match the write order, the throughput pattern disappears; therefore, its mapping policy is not based on LBA. Flash SSD uses a mapping policy based on written-order (log-structured) and therefore the throughput pattern only occurs when we read according to the written order; this is why Flash SSD requires garbage collection. Optane SSD behaves quite differently; no matter how we precondition the device, the pattern occurs when reading according to LBA. Hence, **Optane SSD likely adopts LBA-based mapping.** This design is enabled by 3D XPoint memory's capability to perform in-place updates. As a result, Optane SSD doesn't require garbage collection and can deliver sustainable performance over time.

3 Implications From the Contract

The following implications can be drawn from our unwritten contract. We have two audiences in mind; first, those who design systems for Optane SSD; second, those who combine Flash and Optane in a hybrid setting.

The Random Access is OK rule suggests possible restructuring of external data structures on Optane SSD. Previous designs try hard to convert unstructured accesses into sequential ones, which is now less necessary. Applications which behave poorly on Flash thus become potential consumers of Optane. The No Crowded Accesses rule, No Tiny Access rule, and Alignment rule suggest pitfalls that fine-grained data structures must be aware.

Heterogeneous storage also needs careful design. The motivation for hybrid designs comes directly from the Low Request Scale rule: Flash SSD outperforms Optane SSD in some cases. Optane SSD provides low access latency for workloads with low request scale, while workloads with high request scale might prefer Flash SSD. For workloads with mixed reads and writes, the Control Overall Load rule suggests that read-dominated workloads should be deployed on Flash SSD to achieve low write latency. However, write-intensive workloads prefer the Optane SSD, thus avoiding excessive read latency (influenced by writes in classic Flash SSD). Finally, our results for sustainable performance suggest that when devices become full (and thus would cause garbage collection on an SSD), Optane may be a better choice.

4 Conclusion

We analyze a popular NVM-based block device: the Intel Optane SSD. We formalize the rules that Optane SSD users need to follow. We provide experiments to present the impact when violating each rule, and examine the internals of Optane SSD to provide insights for each rule. The unwritten contract provides implications and points to directions for potential research on NVM-based devices.

Acknowledgments

We thank the anonymous reviewers and Anirudh Badam (our shepherd) for their feedback. This material was supported by the Microsoft Gray Research Lab and funding from NSF grants CNS-1763810 and CNS-1838733.

5 Discussion

The unwritten contract introduces some open questions requiring discussion and future research:

I **Applicability of This Contract:** This contract targets the Intel Optane SSD 905P. It is not clear of the applicability of this contract; will this contract stay accurate for all NVM-based SSDs? will Intel Optane persistent memory follow similar rules? This contract needs to be verified once new devices are available.

II **Move to Optane SSD:** According to the contract, Optane SSD does not promise benefits for all applications. Which applications suit the move to Optane SSD naturally? What may need modification to exploit the benefits of Optane SSD? An analysis of applications on Optane SSD is required.

III **Rethink External Data Structure Design:** Do previous external data structures/algorithms get the optimal performance from Optane SSD? To answer this question, we think it is worth to first review "Is there any tradeoff we made to transform unstructured accesses into sequential ones?" This tradeoff likely exists in single machine graph processing systems. There is already recent work [24] following this path (but for NVMe SSD). On the contrary, can we move in-memory data structures to Optane SSD directly? A study is required. Previous work on NVM [14, 29, 34] can be inspiring.

IV **Optane SSD + Flash SSD:** How to make full usage of hybrid Optane SSD and Flash SSD systems? Is previous work [22] using Optane SSD as a simple caching layer appropriate? Are there opportunities to make caching more efficient? Prior efforts on Flash SSD [31, 32] may trigger thinking in this area.

V **Split Accesses:** Led by the unwritten contract, we think the guide to take advantage of both Flash SSD and Optane SSD is to split accesses to the device which best suit them. Some directions are interesting to look into. 1) At what granularity should we implement this splitting? Split workloads to devices? Or make the external data structure span devices? 2) How to support the splitting automatically? There can be challenges like dynamic workloads and access dependencies. Can machine learning play a role here? Related work includes [15, 28]

VI **Wearing out:** There is limited open knowledge about the wearing out of 3D XPoint memory. How severe is it? How is wear-leveling performed in Optane SSD? How is it done in Optane Persistent Memory?

VII **The whole stack:** The storage hierarchy is changing. The stack includes not only DRAM and HDD/SSD, but also Optane SSD and Optane Persistent Memory. What does this richer hierarchy mean for the operating system [13], the file system [27], and applications?

References

[1] 3D XPoint. https://en.wikipedia.org/wiki/3D_XPoint.

[2] 3D XPoint Technology. https://www.micron.com/products/advanced-solutions/3d-xpoint-technology.

[3] Decompsition of Intel Optane SSD 900P. https://www.anandtech.com/show/12136/the-intel-optane-ssd-900p-480gb-review.

[4] IMDT Use Case, Memcached. https://www.intel.com/content/www/us/en/support/articles/000026359/memory-and-storage/data-center-ssds.html.

[5] IMDT Use Case, Redis. https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/imdt-solution-brief-in-memory-data-store.pdf.

[6] IMDT Use Case, Spark. https://www.intel.com/content/www/us/en/software/apache-spark-optimization-technology-brief.html.

[7] Intel Memory Drive Technology. https://www.intel.com/content/www/us/en/software/intel-memory-drive-technology.html.

[8] Intel Optane DC Persistent Memory. https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html.

[9] Intel Optane Memory. https://www.intel.com/content/www/us/en/architecture-and-technology/optane-memory.html.

[10] Intel Optane SSD. https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/data-center-ssds/optane-dc-p4800x-series.html.

[11] Intel Optane Technology. https://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology.html.

[12] Samsung 970 Pro. https://www.samsung.com/semiconductor/minisite/ssd/product/consumer/970pro/.

[13] Daniel Bittman, Matt Bryson, Yuanjiang Ni, Arjun Govindjee, Isaak Cherdak, Pankaj Mehra, Darrell D. E. Long, and Ethan L. Miller. Twizzler: An Operating System for Next-Generation Memory Hierarchies. Technical Report UCSC-SSRC-17-01, University of California, Santa Cruz, December 2017.

[14] Daniel Bittman, Darrell DE Long, Peter Alvaro, and Ethan L Miller. Optimizing Systems for Byte-Addressable *NVM* by Reducing Bit Flipping. In *17th USENIX Conference on File and Storage Technologies (FAST 19)*, 2019.

[15] Christian Black, IT Michael Mesnier, and Terry Yoshii. Solid-State Drive Caching with Differentiated Storage Services. *Intel White Paper*, 2012.

[16] Simona Boboila and Peter Desnoyers. Performance Models of Flash-based Solid-State Drives for Real Workloads. In *2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2011.

[17] Luc Bouganim, Björn Thór Jónsson, Philippe Bonnet, et al. uFLIP: Understanding Flash IO Patterns. In *Proceedings of the fourth Conference on Innovative Data Systems Research (CIDR '09)*, Pacific Grove, California, January 2009.

[18] Feng Chen, Rubao Lee, and Xiaodong Zhang. Essential Roles of Exploiting Internal Parallelism of Flash Memory Based Solid State Drives in High-speed Data Processing. In *Proceedings of the 17th International Symposium on High Performance Computer Architecture (HPCA-11)*, pages 266–277, San Antonio, Texas, February 2011.

[19] Timothy E Denehy, John Bent, Florentina I Popovici, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. Deconstructing Storage Arrays. In *ACM SIGARCH Computer Architecture News*, volume 32. ACM, 2004.

[20] Peter Desnoyers. Empirical Evaluation of NAND Flash Memory Performance. *ACM SIGOPS Operating Systems Review*, 44(1), 2010.

[21] Peter Desnoyers. What Systems Researchers Need to Know about NAND Flash. In *Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems*, 2013.

[22] Assaf Eisenman, Darryl Gardner, Islam AbdelRahman, Jens Axboe, Siying Dong, Kim Hazelwood, Chris Petersen, Asaf Cidon, and Sachin Katti. Reducing DRAM Footprint with *NVM* in Facebook. In *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018.

[23] Assaf Eisenman, Maxim Naumov, Darryl Gardner, Misha Smelyanskiy, Sergey Pupyrev, Kim Hazelwood, Asaf Cidon, and Sachin Katti. Bandana: Using Non-volatile Memory for Storing Deep Learning Models. *arXiv preprint arXiv:1811.05922*, 2018.

[24] Nima Elyasi, Changho Choi, and Anand Sivasubramaniam. Large-Scale Graph Processing on Emerging Storage Devices. In *17th USENIX Conference on File and Storage Technologies (FAST 19)*, 2019.

[25] Frank T Hady, Annie Foong, Bryan Veal, and Dan Williams. Platform Storage Performance With 3D XPoint Technology. *Proceedings of the IEEE*, 105(9), 2017.

[26] Jun He, Sudarsun Kannan, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. The Unwritten Contract of Solid State Drives. In *Proceedings of the Twelfth European Conference on Computer Systems*. ACM, 2017.

[27] Youngjin Kwon, Henrique Fingler, Tyler Hunt, Simon Peter, Emmett Witchel, and Thomas Anderson. Strata: A Cross Media File System. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017.

[28] Tian Luo, Rubao Lee, Michael Mesnier, Feng Chen, and Xiaodong Zhang. hStorage-DB: Heterogeneity-aware Data Management to Exploit the Full Capability of Hybrid Storage Systems. *Proceedings of the VLDB Endowment*, 5(10), 2012.

[29] Virendra J Marathe, Margo Seltzer, Steve Byan, and Tim Harris. Persistent Memcached: Bringing Legacy Code to Byte-Addressable Persistent Memory. In *9th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 17)*, 2017.

[30] Steven W. Schlosser and Gregory R. Ganger. MEMS-based Storage Devices and Standard Disk Interfaces: A Square Peg in a Round Hole? In *Proceedings of the 3rd USENIX Symposium on File and Storage Technologies (FAST '04)*, San Francisco, California, April 2004.

[31] Kefei Wang and Feng Chen. Cascade Mapping: Optimizing Memory Efficiency for Flash-based Key-value Caching. In *Proceedings of the ACM Symposium on Cloud Computing*. ACM, 2018.

[32] Gala Yadgar, Michael Factor, and Assaf Schuster. Cooperative Caching with Return on Investment. In *2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2013.

[33] Gala Yadgar and Moshe Gabel. Avoiding the Streetlight Effect: I/O Workload Analysis with SSDs in Mind. In *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '16)*, Denver, CO, June 2016.

[34] Pengfei Zuo, Yu Hua, and Jie Wu. Write-Optimized and High-Performance Hashing Index Scheme for Persistent Memory. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018.