2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)

Performance Analysis of 3D XPoint SSDs in Virtualized and non-Virtualized Environments

Jiachen Zhang, Peng Li, Bo Liu, Trent G. Marbach, Xiaoguang Liu*, Gang Wang*
Nankai-Baidu Joint Lab, College of Computer Science, *Nankai University*, Tianjin, China
{jczhang, lipeng, liubo, trent.marbach, liuxg, wgzwp}@nbjl.nankai.edu.cn

***Abstract*—Intel's Optane SSD recently came to the market as the pioneer of 3D XPoint based commercial devices. They have much lower latency (about 14 µs) and better parallelism properties than traditional SSDs, and as such are set to replace NAND flash SSD in commercial settings. To best serve cloud and enterprise data centers' higher performing storage demands, it is necessary to know the performance characteristics of the new devices in both virtualized cloud environments and traditional non-virtualized environments. In this paper, we present an analysis of Optane SSDs based on a large number of experiments. We use several micro-benchmarks to gain knowledge of Optane's basic performance metrics. We also discuss the impact of state-of-the-art storage stacks on the performance of Optane SSDs. By analyzing the test results, we provide configuration suggestions for storage I/O applications using Optane SSDs. Lastly, we evaluate the real-world performance of Optane SDDs by running MySQL database based experiments. All the experiments are performed in non-virtualized and virtualized environments (Linux and QEMU) with a comparison study between the Optane SSD and a SATA NAND flash-based SSD.**

***Index Terms*—3D XPoint, solid-state drive (SSD), non-volatile memory (NVM), virtualization, performance analysis**

I. Introduction

Storage hardware is at present rapidly developing. NAND flash-based solid state drives (SSDs) have become very popular in enterprise storage systems as a secondary storage medium, due to their superiority over rotating hard disk drives (HDDs). Emerging non-volatile memory technologies (NVMs) like PCM, STT-RAM, ReRAM and 3D XPoint are also under active development. NVMs have lower latency, higher bandwidth and longer lifetime than NAND flash devices.

As displayed in Fig. 1, NAND flash-based SSDs are usually connected through SATA or PCIe, while NVM based devices can be connected through either DIMM or PCIe, and so compete with both SSD and DRAM devices. There has been a lot of studies focusing on NVMs [1]. However, most of them were based on NVM emulators, as NVM based devices were previously not commercially available. Recently 3D XPoint based Intel Optane SSDs (Optane) were released into the market. These devices are connected through the PCIe interface and are based on NVMe protocol specification [2]. Although the PCIe interface is not revolutionary, the underlying 3D XPoint memory has many different characteristics with NAND flash. Thus, we are interested in how Optane behaves in the following aspects:

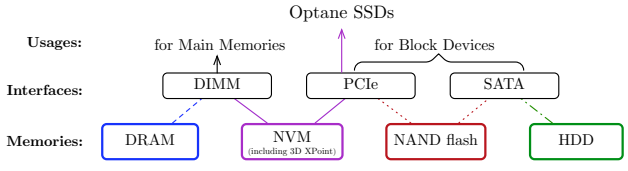


Fig. 1: Memories and their usage models.

1) Current modern Operating System (OS) storage stacks, which include layers such as block I/O subsystem and file systems, were initially optimized for rotating disks and subsequently for NAND flash-based SSDs [3]. Are these technologies efficiency enough for Optane?
2) Virtualized execution environments, which are the back-ends of cloud services, have become indispensable for data centers. However, their storage stacks are more complex due to the additional virtualization layers and the prolonged I/O path. How do Optane devices perform in the virtualized environments?
3) Storage I/O intensive applications, such as relational database management systems (DBMS), are mainly bottlenecked during storage. Further to this, previous configuration settings have been optimized using extensive research to operate on traditional devices. How is the application performance improved when using Optane, and are the previous configurations still tenable for use with Optane?

To answer these questions, we present extensive experiments on an Intel Optane SSD 900P in Linux, a non-virtualized physical environment (PE), and in QEMU, a virtualized execution environment (VE). For comparison, we also perform each experiment on a NAND flash-based SATA SSD (NAND). For performance experiments, we use both micro-benchmarks and real world application MySQL OLTP benchmarks.

The rest of the paper proceeds as follows. Section II discusses the storage stacks, Section III performs measurements on the basic metrics. We give the suggestions on using Optane in Section IV and evaluate Optane equipped MySQL in Section V. Finally, Section VI presents the related work, and Section VII summarizes our work in this paper.

II. The Impacts of System Software

A. Storage Stacks

Computer systems such as storage systems typically use multiple levels of indirections, owing to the high flexibility demanded for system expansion. However, the communication

978-1-5386-7308-9/18/$31.00 ©2018 IEEE
DOI 10.1109/ICPADS.2018.00018

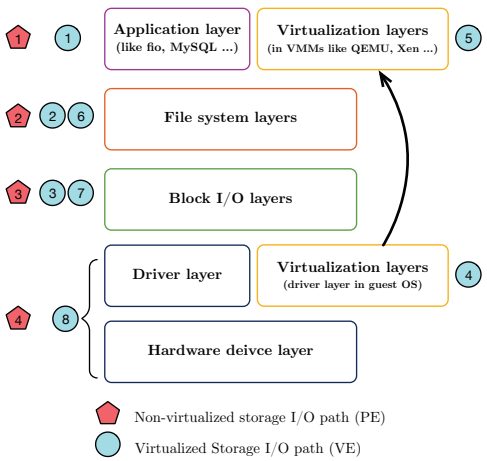


Fig. 2: I/O paths in PE or VE storage stacks.

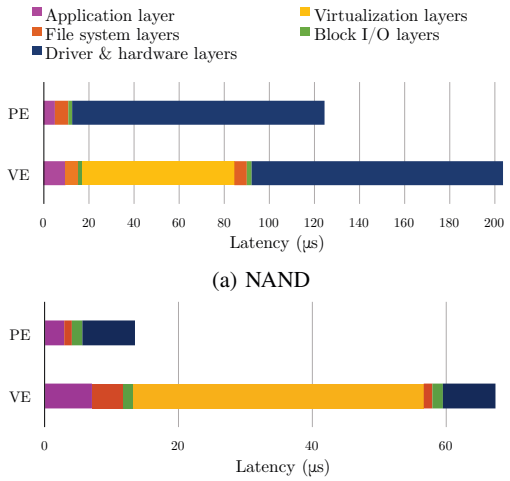


Fig. 3: Composition of a 4 KB random read I/O request latency in VE and PE storage stacks.

among different layers will inevitably result in extra performance overheads [4], thus increase the request latency.

The typical storage stacks of a PE and a VE are illustrated in Fig. 2, labeled by pentagon and circle symbols respectively. The ordered numbers in the figure indicate the layers that an I/O request is processed through. For example, in a PE, a read request will pass from the top Application Layer to the bottom Hardware device layer, numbered with $1-4$. In a VE, the condition becomes more complex, with a virtual machine (VM) running in an application called virtual machine monitor (VMM) based on the host OS.

The request, issued by an application in the VM, will first go through the layers of the VM storage stack and then gets through that of the host server, numbered with $1-8$. The two additional virtualization layers (numbered with 4 and 5) act as the bridge between the VM and the host server.

The overall latency of a storage stack consists of hardware-related latency incurred by hardware device layer and driver layer, and software-related latency incurred by the others parts of the stack.

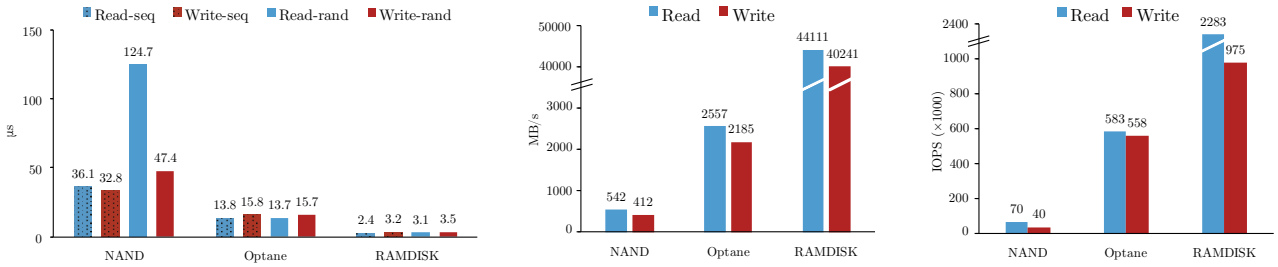
In the past, data centers primarily relied upon PEs. Together with the disk speeds at the time, this meant that hardware latency heavily dominated the total request latency [5]. However, the situation has become more complex recently in two aspects: (1) VEs have increased in popularity, prolonging the software latency with the increased number of layers, and as a result decreasing the proportion for the hardware latency. (2) Higher speed devices (e.g., Optane) have become available, which reduce the hardware latency greatly. To give clear insight on how to optimize the total latency, we measure the latency breakdown incurred by different layers in both the PE and VE with NAND and Optane devices equipped, which is detailed in the following section.

B. Latency Breakdown

In this section, we evaluate the latency breakdown when using NAND and Optane, in both PEs and VEs. We use the same experimental setup like that in Section III. We also measure the random read requests as they are common in the real-world workloads with the help of `fio` [6] and `blktrace` [7]. The latencies of each layer are directly measured or calculated. The results are shown in Fig. 3, where the colors of each layer correspond to that in Fig. 2. We make several observations:

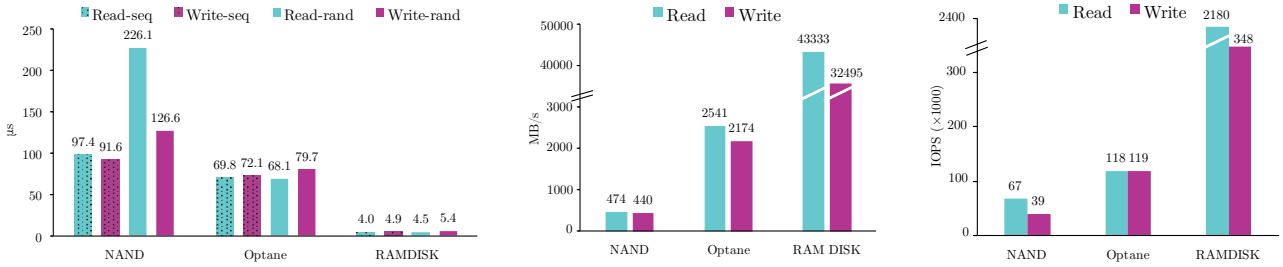
1) As expected, using Optane decreases the latencies in both environments. Specifically, since the hardware latency when using Optane is 7.9 µs, which is 103.8 µs faster than that of NAND, the overall latencies will decrease by 67.0% and 89.1% in PE and VE respectively.
2) Latencies in the VE are higher than the PE with both devices due to the extension of the VE's I/O path, which increases the software latency. Specifically, a large proportion of the software latency is due to the virtualization layers. As a result, the software latency in the VE is responsible for 45.1% and 88.3% of the total latency when using NAND and Optane respectively. As a result, the overall latency of the NAND device using a VE (203.7 µs) is 3 times that of when using a PE (67.3 µs), and that factor increases to 10 times when using Optane (124.5 µs in VE and 13.5 µs PE).
3) In the PE, the software latency of NAND is negligible compared with its overall latency, with only a proportion of 10.3%. However, this proportion increases to 41.6% for Optane, owing to the lower hardware latency that Optane achieves.

Directed by these facts, we propose that reductions in the overall request latency can be best achieved by: (1) The adoption of new devices, which helps to decrease hardware latencies. (2) Optimizations of the virtualization layers in the VE, which will reduce the software latencies in the VE significantly. (3) Optimizations of the traditional OS storage



(a) Latency (b) Bandwidth (c) IOPS

Fig. 4: Micro-benchmarks of NAND, Optane and RAMDISK in general Linux environment (PE).



(a) Latency (b) Bandwidth (c) IOPS

Fig. 5: Micro-benchmarks of NAND, Optane and RAMDISK in virtualized QEMU environment (VE).

stacks in the PE, which will reduce the request latency in both PE and VE.

III. Basic Metrics

In this section, we show how Optane behaves regarding the basic metrics of latency, IOPS and bandwidth by making comparisons with two kinds of devices, NAND and RAMDISK, in both the PE and VE. The detailed information of these devices is listed in Table I. Optane is a PCIe-attached device using 3D XPoint technology, while NAND is a traditional SATA-attached device based on NAND flash, and RAMDISK is an emulated device using DRAM.

We further explore the performance curve between latency and IOPS in Section III-C, and present a summary.

TABLE I: Hardware devices.

	NAND	Optane	RAMDISK
Device Model	Intel S3510	Intel Optane 900P	Micron DDR4
Interface	SATA 3.0	PCIe 3.0 ×4	DIMM
Memory Medium	NAND flash	3D XPoint	DRAM
Capacity	480 GB	480 GB	64 GB
Dollars per GB	0.68	1.16	13.13

A. Experimental Setup

We use `fio` to perform the tests by generating an appropriate number of read or write threads with an appropriate I/O request sizes, which will be detailed in Section III-B.

All the experiments are conducted on an X86 server, with configuration information listed in Table II. For the PE, all

TABLE II: Server Configurations.

CPU	Intel Xeon E5-2609 1.70 GHz ×2
CPU cores	8 ×2
Processor cache	32 KB L1i, 32 KB L1d, 256 KB L2, 20 MB L3
DRAM	128 GB
OS	RHEL 7.0, kernel version 4.14 (same in VMs)
VMM	QEMU 2.10
File system	XFS

devices are formatted with XFS file system. For the VE, we create raw format image files on both Optane and NAND devices as the backends of the VM storage. The RAMDISK is created directly in the VM by a guest OS, as memory virtualization is more efficient than storage I/O virtualization (due to the prolonged VE I/O path). Note that in the VE, the additional overheads of RAMDISK are mainly caused by memory virtualization rather than the additional storage stack layers. We also format these devices with XFS in the VE.

For all experiments, each result is averaged over a 30s execution and the size of the data stored in the drive is maintained at 20 GB (originally set randomly).

B. Latency, Bandwidth and IOPS

a) I/O Latency: To measure the averaged latency of an I/O request, we run `fio` in single thread mode and generate 4 KB requests one at a time.

The results of the PE and VE latency experiments are shown in Fig. 4 (a) and Fig. 5 (a) respectively. We use *Read-seq* and *Write-seq* to denote the sequential read and write tests, and

Read-rand and *Write-rand* to denote the random read and write tests respectively. In the PE, as expected, Optane achieves lower latencies than NAND, with a reduction of 67.4% on average for all four request types. Additionally, the latencies of Optane are more balanced between sequential and random requests, differing with a factor of at most 12.7%, while there is a 2 to 3 times difference between the NAND requests[1]. In the VE, the latency increases in all situations, however, Optane's performance suffers the most from the additional virtualization layers. For example, on average, the random read latency of Optane in the VE is 5 times that of it in the PE, while for NAND, the multiplication factor decreases to 2 times.

RAMDISK is the fastest device in all cases, owing to the more rapid memory speed obtained by using DRAM. However, Optane is only 5 times slower than RAMDISK on average while NAND is 10 to 40 times slower than RAMDISK. RAMDISK also achieves relatively balanced latencies (e.g., about 3 µs in PE and 5 µs in VE), showing a similarity between Optane and RAMDISK.

b) Transfer Bandwidth: We calculate the upper bounds of the I/O throughput (transfer bandwidths) by sending multiple large request units (128 KB) over multiple threads to the device.

Fig. 4 (b) and Fig. 5 (b) show the bandwidths in the PE and VE. As can be seen, the bandwidth of Optane is over 2 GB/s, approximately 5 times more than NAND's bandwidth. As Optane is connected through PCIe 3.0 ×4 interface, whose theoretical bandwidth is 3.94 GB/s, and NAND is connected through SATA 3.0, whose theoretical bandwidth is 600 MB/s, the interfaces do not strongly bound the bandwidths. In the VE, the longer transfer time of the larger 128 KB units along with the use of multiple threads act to hide the additional virtualization latencies.

Although RAMDISK has the best bandwidth between each of the memory types, the performance boost by using Optane compared with NAND is still meaningful. The higher bandwidth of Optane indicates it can be used as persistent file cache in offline workloads such as backup system. Also, higher bandwidth may change optimization policies of some storage services like transparent data compression, data deduplication and erasure coding. We present a more detailed analysis in Section IV-C by a case study of transparent compression.

c) Maximum IOPS: To measure the maximum IOPS of each device, we run `fio` in multi-thread mode (64 threads), which generates random read and write requests with a small 4 KB I/O size. We calculate the IOPS by averaging the number of successful requests per seconds. Using small unit size saves I/O bandwidth; thus we can expect to achieve the highest IOPS before reaching the hardware transfer bandwidth.

Fig. 4 (c) shows that Optane works significantly better than NAND in the PE, with an order of magnitude higher

[1]Theoretically, the write operation of NAND should be more expensive than the read operations due to the NAND flash write amplification factor [8], we believe the reason random write is faster than random read here is due to the write buffering in our device [9].

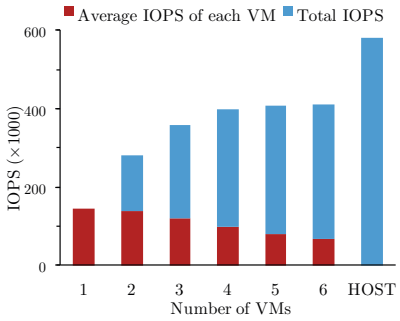


Fig. 6: Read IOPS of multiple VMs based on one Optane.

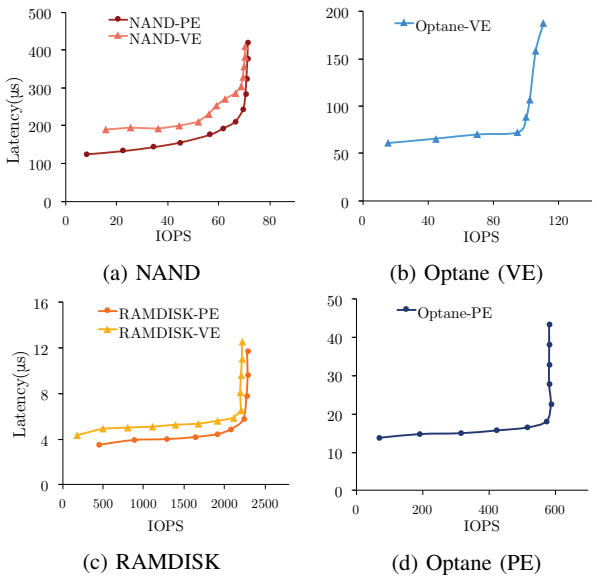
IOPS (from 8 to 11 fold increase). The theoretical upper bound on IOPS for a given parameter setting is given by its corresponding bandwidth divided by 4096 (the I/O size). NAND obtains 51.7% and 38.8% of the upper bound for read and write, Optane obtains 91.2% and 102.15%[2], and RAMDISK obtains 20.7% and 9.7%. It is clear from these results that large thread and request counts negatively effect NAND and RAMDISK devices, but do not seem to affect the Optane device. These results indicate that Optane works well in a parallel setting.

Besides, there is only a small gap between read and write operations for Optane but a large gap for NAND. This indicates that Optane is naturally suitable for different read and write situations while NAND will perform poorly for write-intensive systems due to the write amplification factor of NAND flash. RAMDISK performs best in IOPS with 2 to 4 times higher than Optane, owing to the high parallelism of the memory bus.

In VE, as shown in Fig. 5 (c), Optane only obtains about 120k IOPSs, one-fifth of that in the PE. In contrast, NAND gets nearly the same IOPSs as that in the PE. This indicates in the VE, the parallelism of Optane is still underutilized.

To explore how to make full use of Optane in the VE, we deploy multiple VMs, which use Optane as there storage backend. We run `fio` benchmarks simultaneously in these VMs and test the accumulated IOPS among all the VMs. The results are illustrated in Fig. 6, where the blue bar denotes the total IOPS, and red bar denotes the averaged IOPS. We observe that although modern VMMs provide inadequate support for Optane regarding parallelism for a single VM, Optane still can play a role in VEs when multiple VMs are deployed. As the number of VMs was increased, the IOPS improved sublinearly until the number of VMs was 4. When using 6 VMs, the IOPS is saturated at about 410k IOPS, which is comparable to the IOPS in the PE (580k). This means that if properly utilized, Optane can be a promising solution within cloud data centers.

[2]The reason why this number is slightly larger than 100% is because the theoretical upper bound we use is also measured in the previous section, and the measurement error is unavoidable.



(a) NAND (b) Optane (VE)

(c) RAMDISK (d) Optane (PE)

Fig. 7: Relationship between IOPS and latency.

C. The Performance Curve (between Latency and IOPS)

To better understand the parallel nature of these devices, we also test the relationship between IOPS and latency of random read requests and draw latency curves, see Fig. 7.

As the IOPS rate increases, the device experiences a greater total demand. The average latencies in the traditional device gradually become worse as the IOPS increases, whereas the Optane and RAMDISK devices are more resilient to this demand. Once the IOPS achieves the upper bounds, the latencies increase rapidly on all devices [10] [11].

As can be observed in Fig. 7 (a), (c) and (d), as the IOPS rate achieves 95% of the upper bound, the latencies of NAND, RAMDISK and Optane has increased by 80%, 54% and 25% from its base levels, respectively. This indicates Optane internally spends less time addressing the data. As a result, the latency of Optane can stay optimal in high concurrency workloads. Moreover, when Optane reaches its IOPS bound (about 580k), the corresponding throughput is about 2.2 GB/s, which is very close to its transfer bandwidth (2.5 GB/s). On the contrary, NAND gets a 280 MB throughput when reaching its upper bound IOPS of 70k, which is much smaller than its bandwidth (542 MB/s). Thus, we conclude that Optane has better parallelism and scalability and does better for high concurrency workloads. Fig. 7 (a), (b) and (c) show the results of NAND, Optane and RAMDISK in VE. The performance of NAND and RAMDISK is only a little worse than that in the PE, while the performance of Optane drops a lot in the VE. However, the latency of Optane is still more steady as the IOPS rate increases. When the IOPS rate achieves 95% of the upper bounds, latency of NAND, Optane and RAMDISK increase by 158%, 50% and 35% respectively.

Through these measurements, we believe although Optane costs twice the price per GB than the NAND device, it is still a more competitive option than NAND for modern data centers. On the other hand, compared with NAND, the performance of Optane drops more in VEs. This indicates to make better use of Optane in cloud data centers, further research on scheduling and configuring problems should be performed. However, as mentioned above, as one VM cannot make full use of the Optane's parallelism, using one Optane device for multiple VMs is more appropriate in cloud data centers presently.

IV. Impacts on Storage Optimizations

A. File Cache

File caches are used to bridge the performance gap between storage devices and main memories. Using DRAM as a file cache is a common optimization in the storage module of I/O intensive applications. When using a file cache, the average read I/O latency can be expressed as follows:

$$\text{Latency} = t_{\text{I/O}} \times (1 - H) + t_{\text{load}} \times H \tag{1}$$

where H denotes the cache hit rate, $t_{\text{I/O}}$ denotes the average latency reading from the disk, and t_{load} denotes the time spent fetching data from the file cache. Generally, the term $t_{\text{load}} \times H$ can be omitted as t_{load} is very small compared with $t_{\text{I/O}}$. When using Optane, file caches are still necessary but less important as its speed ($t_{\text{I/O}}$) is closer to DRAM (t_{load}). In addition, as Optane is orders of magnitude faster than HDDs, Optane itself can be used as a cache between HDDs and main memories [12].

B. I/O Granularity

In I/O intensive applications like DBMSs and key-value stores, data is generally organized by fixed-length contiguous chunks, which we call pages. A page is the smallest unit of data that is fetched or stored.

Traditionally, when using HDDs, the large seek time t_{seek} dominates the request response time, which results in a large page size choice. For example, ZFS [13] by default uses 128 KB. And as a common experience, when using faster devices like SSDs, a smaller page size will be optimal [14] [15]. However, this experience may be wrong when switching the device from HDDs and low-end SSDs to Optane, and we aim to explore the underlying reason through sophisticated mathematical analysis.

An application serves a query request using 3 steps: (1) interpreting the request to obtain the corresponding pages that will be transferred, (2) exchanging the pages between application layer and underlying software layers of the storage stack, and (3) exchanging the data between the bottom software layer and storage devices. We use T_{app}, T_{stk} and T_{dev} to denote the time cost corresponding to the above three steps respectively. Thus, the total time cost for a request can be described by

$$T = T_{\text{app}} + T_{\text{stk}} + T_{\text{dev}}. \tag{2}$$

Note that in the underlying storage stack, data is organized in pages too. We use d_a and d_s to be the page size in the application layer and the underlying storage stack layer respectively. Practically, for the sake of performance, d_a is a

multiple of d_s, indicating that requesting one application page involves $\frac{d_a}{d_s}$ underlying storage stack pages. In addition, a total of $\lceil \frac{d}{d_a} \rceil$ application pages will be transferred if the requested I/O size is d. Thus, we have

$$T_{\text{app}} = t_{\text{app}} \lceil \frac{d}{d_a} \rceil, \tag{3}$$

$$T_{\text{stk}} = t_{\text{stk}} \frac{d_a}{d_s} \lceil \frac{d}{d_a} \rceil, \tag{4}$$

where t_{app} and t_{stk} denotes the time cost of a page in the application layer and the underlying software storage stack. Let b be the bandwidth of the storage device, and t_{seek} be the seek time of the storage device, whose value is equal to the difference between the random latency and sequential latency. Thus for the third term in (2), we have

$$T_{\text{dev}} = \lceil \frac{d}{d_a} \rceil (t_{\text{seek}} + \frac{d_a}{b}). \tag{5}$$

Applying (3), (4) and (5) into (2), we have the time cost for a request r_i with I/O size d_i as follows:

$$T = (t_{\text{app}} + t_{\text{stk}} \frac{d_a}{d_s} + t_{\text{seek}} + \frac{d_a}{b}) \lceil \frac{d_i}{d_a} \rceil \tag{6}$$

Applications usually have two types of requests, which can be called point requests and range requests. Point requests only involve a data amount much smaller than the page size, and so $\lceil \frac{d_i}{d_a} \rceil$ will be 1. Range requests involve a large amount of data that involve many pages, and so $\lceil \frac{d_i}{d_a} \rceil$ is approximately equal to $\frac{d_i}{d_a}$. We assume a workload involves N query requests, which consists of m point requests and n range requests, indexed by r_i ($1 \le i \le N, N = m + n$), whose requested I/O sizes are represented by $D = \{d_1, d_2, \ldots, d_m, d_{m+1}, \ldots, d_N\}$. Therefore, the total time cost for the given workload can be expressed by

$$\begin{aligned} T_D =& \sum_{i=1}^{m} T_i + \sum_{i=m+1}^{N} T_i \\ =& \sum_{i=1}^{m} (t_{\text{app}} + t_{\text{stk}} \frac{d_a}{d_s} + t_{\text{seek}} + \frac{d_a}{b}) \\ &+ \sum_{i=m+1}^{N} (t_{\text{app}} + t_{\text{stk}} \frac{d_a}{d_s} + t_{\text{seek}} + \frac{d_a}{b}) \frac{d_i}{d_a}. \end{aligned} \tag{7}$$

Let $\bar{d}$ to be the average I/O size of the range requests. Then we have

$$\begin{aligned} T_D =& m(t_{\text{app}} + t_{\text{stk}} \frac{d_a}{d_s} + t_{\text{seek}} + \frac{d_a}{b}) \\ &+ n(t_{\text{app}} + t_{\text{stk}} \frac{d_a}{d_s} + t_{\text{seek}} + \frac{d_a}{b}) \frac{\bar{d}}{d_a} \end{aligned} \tag{8}$$

In (8), we find the derivative of T_D of the page size (d_a)

$$\frac{\partial T_D}{\partial d_a} = m(\frac{t_{\text{stk}}}{d_s} + \frac{1}{b}) - \frac{n(t_{\text{app}} \bar{d} + t_{\text{seek}} \bar{d})}{d_a^2} \tag{9}$$

and there will be a extreme point of d_a

$$d_a = \sqrt{\frac{n(t_{\text{app}} + t_{\text{seek}}) \bar{d}}{m(t_{\text{stk}}/d_s + 1/b)}}. \tag{10}$$

Formula 10 gives a way to find the ideal page size, depending on the context of the application. When switching to the high-end SSDs like Optane, the hardware latency t_{seek} and $1/b$ is sufficiently small and is comparable with the software latency. As a result, hardware latency no longer dominates Formula 10, software factors t_{app} and t_{stk} (such as application latency, application workloads, virtualization latency and OS I/O subsystem latency) will dominate the best choice of page sizes in equation (10). As software factors are very different between execution environments and workloads, not only the experience which tends to choose smaller page size for faster devices becomes invalid, the best choice will also be much easier to change. Thus when using Optane, we should not choose a small page size only based on its low hardware latency, more workload-related analysis and tests are needed.

C. Storage-orient Computing

Storage-oriented computing tasks like transparent compression, deduplication and erasure coding, are incorporated into some I/O intensive applications for space efficiency, high performance and reliability [16] [17] [18]. These tasks work in harmony with traditional slower storage devices. However, when using the new devices like Optane, some of the advantages may disappear, and they may even lead to poor performance. We take transparent data compression as an example.

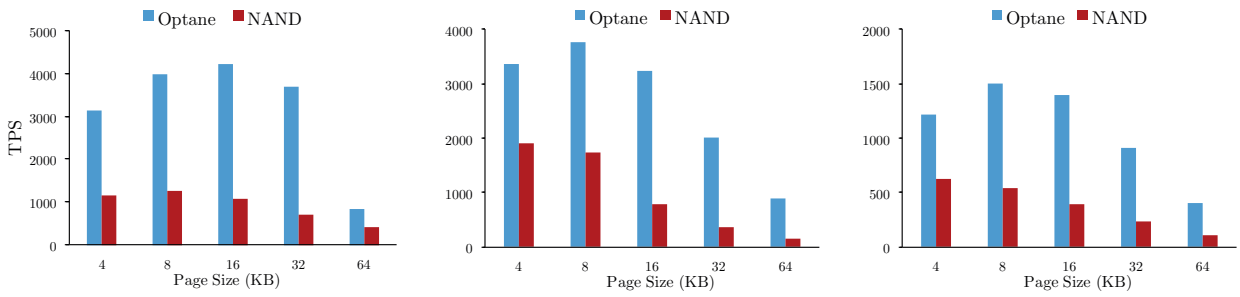
Data compression has been widely used in I/O intensive applications such as ZFS [13], NTFS [19] and MySQL [20], owing to the benefits it provides as follows: (1) less storage space needed, (2) lower I/O (i.e. disk and network) bandwidth for data fetching, and (3) higher cache hit rate.

TABLE III: I/O bandwidths and data compression throughputs (MB/s).

I/O Devices	**Read**	**Write**
NAND	542	412
Optane	2557	2185
Algorithms	**Decoding**	**Encoding**
LZ4	2013	356
Snappy	915	269
zlib deflate	133	23

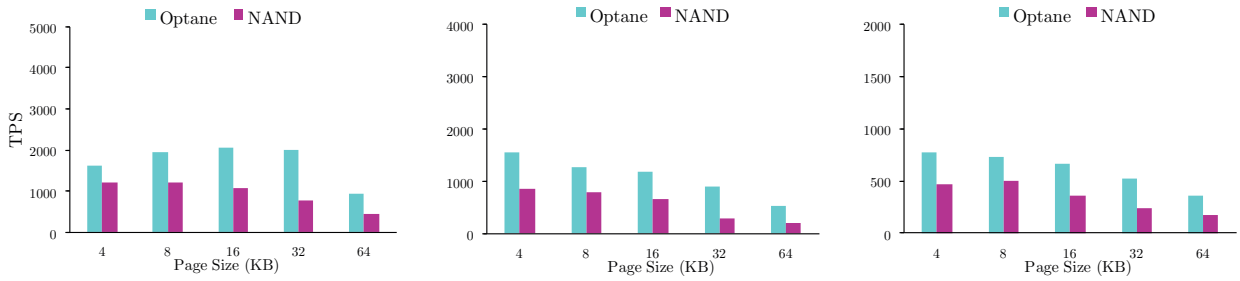
We list the I/O bandwidth and coding throughput of several popular lossless compression algorithms in Table III. The I/O bandwidth is taken from Section III-B, while coding throughputs are tested by `lzbench` [21]. The compression algorithms are tested with the granularity of chunks (128 KB). Traditionally, when using HDDs or NAND devices, decoding is much faster than I/O reading, which can eliminate the I/O traffic and benefit the cache capacity, and encoding will not bottleneck the I/O writing due to the widely used write buffering. When it comes to Optane, both encoding and decoding are slower than I/O operations, which indicates that using data compression may lead to poor performance.

We consider two typical application scenarios where compression may not benefit the system performance when using Optane: First, compressed data can improve the cache hit rate



(a) Read-only (b) Write-only (c) Mixed read and write

Fig. 8: MySQL OLTP performance using various page size in PE.



(a) Read-only (b) Write-only (c) Mixed read and write

Fig. 9: MySQL OLTP performance using various page size in VE.

because more data can be cached after compression. However, with high-speed devices, extra latency may be incurred when decompressing the cached data, which offsets the benefits that high cache hit rate brings. Second, transparent compression is traditionally used in systems to reduce the amount of I/O bandwidth, as I/O latency dominates the total request latency rather than computing. This hypothesis won't hold for high-speed devices, where the proportion of decompression cost is considerable. Thus, we conclude that when designing or configuring applications equipped with Optane, data compression is a trade-off between storage space saving and the performance.

V. Tests in Database (MySQL)

It is evident that a straightforward shift to Optane can improve the performance of any workload. However, in this section, we focus on how Optane behaves in real-world applications with different configurations. We select MySQL [20], a widely used relational DBMS to perform our tests. The tests are designed with the intention of previous analysis and conclusions.

We run the MySQL online transaction processing (OLTP) tests generated by the benchmark tool `Sysbench` [22] on MySQL server version 5.7. We also make some configuring decisions for better observations: We use the default storage engine InnoDB with direct I/O enabled , thus the influence of the OS page cache is eliminated. We by default generate 20 GB data, use 32 MB file cache (named buffer pool in MySQL InnoDB) and 16 client threads are connected to MySQL. When doing mixed read&write tests, the ratio of read and write requests is 7 : 2, which follows the default setting of `Sysbench` OLTP benchmarks. Other setups are the same as in Section III. The performance is measured in transactions per second (TPS, higher is better).

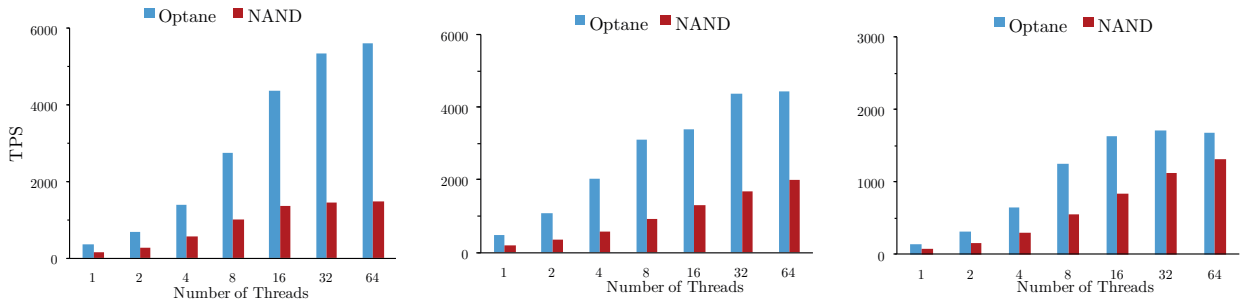
A. Page Size

As explained in Section IV-B, when switching from NAND to Optane, the early experience [14] [15] which recommends smaller page sizes for faster devices is no longer valid. This conclusion is verified in this experiment.

In MySQL InnoDB, all data is organized by fixed size chunks called pages, and the pages are also the minimal I/O units. So we test the OLTP performance as the page size changes between 4 KB and 64 KB. We show the results in Fig. 8 and Fig. 9. Optane still does much better than NAND in all situations, but in the VE, the TPSs of Optane decrease much more than NAND, which is caused by the large virtualization overheads. We focus more on the best page size when using Optane.

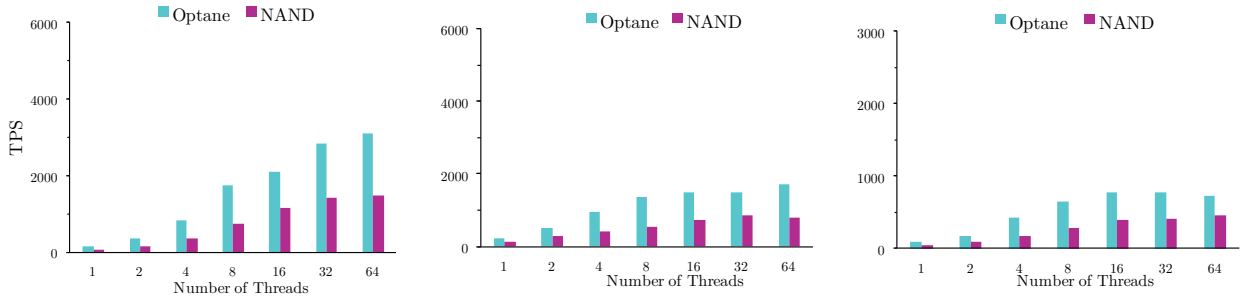
TABLE IV: Best MySQL page sizes (KB).

Device	Read	Mixed R&W	Write
Optane	16	8	8
Optane (VE)	16	4	4
NAND	8	4	4
NAND (VE)	8	8	4



(a) Read-only (b) Write-only (c) Mixed read and write

Fig. 10: MySQL OLTP performance using various thread number in PE.



(a) Read-only (b) Write-only (c) Mixed read and write

Fig. 11: MySQL OLTP performance using various thread number in VE.

We list the best page sizes in Table IV. For example, in Table IV, the best page size of the read-only situation is 16 KB for Optane and 8 KB for NAND in both environments, and in the VE the best page of Optane is smaller than that in the PE. As the Optane outperforms the NAND device, and Optane in the PE outperforms it in the VE, both examples of the results are in violation of the early experience. Thus when configuring MySQL equipped with Optane, tests or detailed analysis based on real-world workloads are needed to find the best page size.

B. Scalability

A common situation is where a MySQL server is connected to multiple clients simultaneously. In this situation, the requests are concurrently submitted to the InnoDB storage engine. Thus, we simulate this situation by using multiple threads benchmark provided by `Sysbench`. The results in the PE and the VE are shown in Fig. 10 and Fig. 11 respectively.

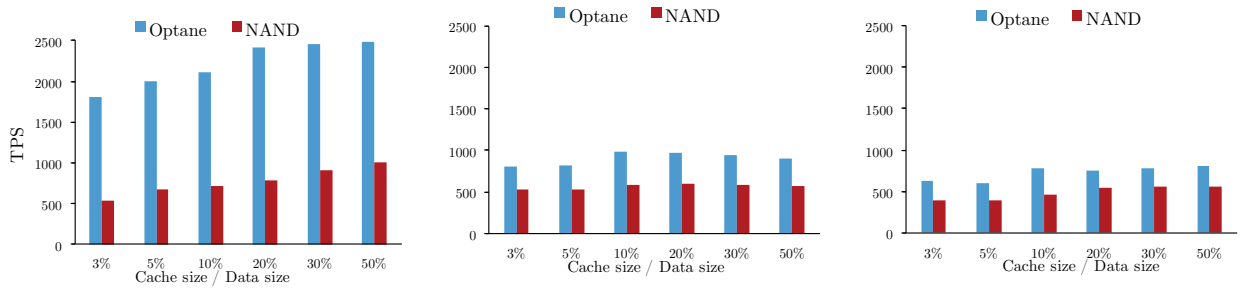
Using Optane can get better TPSs in all situation, as expected. As shown in Fig.10 (a), when only one thread connected, the TPS of Optane is only 2.6× higher than NAND. When the thread number is 64, the TPS of Optane is 3.8× higher than NAND. For the write-only case and mixed read&write-case in Fig.10 (b) and (c), Optane gets better scalability when the thread number is smaller than 8. We also believe the good scalability of NAND is because of the write buffering in our NAND hardware, as explained in Section III. Thus, we conclude that Optane has better scalability in MySQL workloads, especially for read workloads.

Similar to previous tests, Optane does much worse in the VE due to the large overheads of the virtualization layers. As mentioned in Section III, although one VM cannot take full advantages of the parallelism of Optane, we can use multiple VMs with MySQL running on them, to get overall higher scalability. This situation is also more common in cloud computing services such as Database as a Service (DBaaS).

C. Cache Size

As mentioned, file cache is commonly used to bridge the performance gap between storage levels. However, as the performance gap between DRAM and Optane is smaller, using DRAM as the cache of Optane will benefit less than when using it with NAND or HDDs. The file cache of MySQL InnoDB is called the buffer pool, and we configure it between 3% to 50% of the size of the data , use Gaussian distribution and mixed read&write access pattern to test the OLTP performance.

The results in the PE and the VE are shown in Fig. 12 (a) and Fig. 13 (a) respectively. Transfering from 3% data cached to 50% data cached resulted with Optane in the PE increasing its TPS 1.4-fold, NAND in the PE increased 1.9-fold, Optane in the VE increased 1.3-fold, and NAND in the VE increased 1.5-fold. Therefore, the cache is more important for slower NAND both in the VE and in the PE.

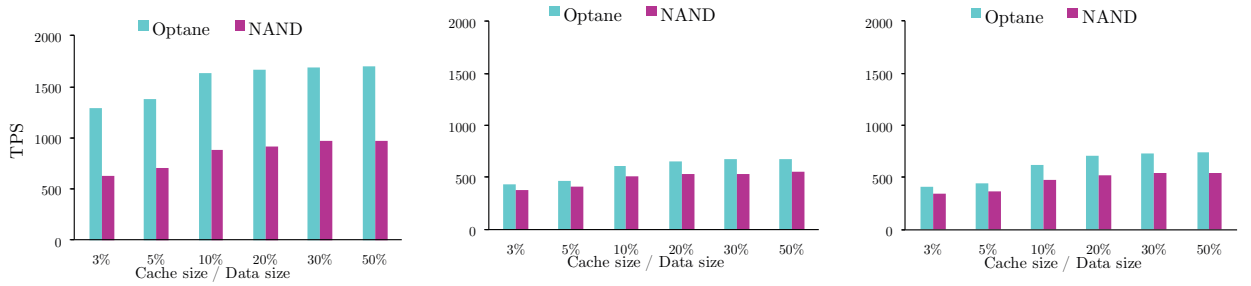


(a) Uncompressed table (page size is 16 KB)

(b) Compressed table (pages are compressed from 16 KB to 8 KB)

(c) Compressed table (pages are compressed from 16 KB to 4 KB)

Fig. 12: MySQL OLTP performance using various cache size in PE.



(a) Uncompressed table (page size is 16 KB)

(b) Compressed table (pages are compressed from 16 KB to 8 KB)

(c) Compressed table (pages are compressed from 16 KB to 4 KB)

Fig. 13: MySQL OLTP performance using various cache size in VE.

D. Table Compression

We discussed the transparent compression for I/O intensive applications in Section IV. Many database storage engines, including MySQL InnoDB, have a compression feature [23] [24]. The MySQL InnoDB supported page-level compression is called table compression. When it is enabled, InnoDB will try compressing each page into a smaller page, such as compressing the 16 KB pages into 4 KB pages. The compressed pages will be stored in files or cached in the buffer pool.

Performance boosting caused by compression can only happen when using slow HDDs, and the purpose of using table compression in SSDs is only to save storage space. As displayed in Fig. 12 (b)(c) and Fig. 13 (b)(c), we set the compressed page size 8 KB and 4 KB, and test the OLTP performance when cache size increases in PE and VE. Compared with the uncompressed situation in Fig.12 (a) and Fig. 13 (a), using table compression feature slightly decreases the performance of NAND equipped with MySQL, while the decrease of Optane equipped with MySQL is significant.

Therefore, in MySQL InnoDB, configuring the table compression feature is a trade-off between storage space saving and the performance for Optane and NAND in the PE. In the VE, using compression is so costly that Optanes performance drops to NAND levels.

VI. Related Work

Many research works have evaluated the performance of emerging SSDs. Xu et al. [25] conducted the first in-depth performance analysis of NVMe drives, comparing HDDs, SATA SSDs and NVMe SSDs. Son et al. [15] evaluated the performance of NVMe SSDs in the context of Linux file systems and database workloads, making comparisons between NVMe SSDs and SATA SSDs. Hady et al. [12] introduced 3D XPoint technology and evaluated its performance using Optane, focusing mainly on the usages of 3D XPoint based devices. Wu et al. [26] evaluated Optane high-performance computing (HPC) applications, making comparisons between Optane and HDDs. We, in this paper, are more interested in the contrast between NAND flash-based SATA SSDs and 3D XPoint based Optane SSDs. We think the comparison between a SATA SSD and a PCIe 3D XPoint SSD is meaningful because SATA connected SSDs are still widely used today in many data centers, and the NVMe protocol for the PCIe interface is becoming increasingly popular.

As cloud computing continues to develop and storage devices are becoming faster, work has been done to optimize the storage stack of VMMs. QEMU used to only support one I/O thread to serve all VM I/O requests, thus causing an I/O scalability problems. Later, this was improved by the virtio-blk-dataplane feature which makes use of a dedicated I/O thread for each device [27]. But as the NVMe SSDs came

into markets, the I/O bottlenecks of QEMU reappeared, thus a lot of optimizations has also been done, such as using multiple I/O threads per devices [28] [29], using the user-space NVMe driver [30], and polling the asynchronous I/O completion [31]. As Optane SSDs have come into the market, we believe gaining knowledge of how Optane behaves in a state-of-the-art virtualized environment is meaningful for cloud data centers preparing to use the new devices.

The 3D XPoint DIMM form, which called Optane DC persistent memory (PM) [32], was said to be broad available in the near future. Because of the blurring of storage and memory, PMs will significantly change the storage architectures and programming model on both hardware and software, which will be a long-term evolution. In this paper, we do not discuss the 3D XPoint-based PM devices, instead focusing on the technology that is recently released.

VII. Conclusion

3D XPoint based Optane SSDs outperform NAND flash-based SSDs in all aspects, such as lower latency, higher parallelism and higher bandwidth. We comprehensively analyzed the performance characteristics of the 3D XPoint SSDs. According to the analysis, traditional optimizations like file caching, transparent data compressing either become less effective, or even reduce the performance of the new devices. The situation is more complex in the virtualized environments, further research should investigate due to the poor support of current VMMs for these high-performance storage devices. Our future work will make specific optimizations of OSs, VMMs and I/O intensive applications based on these characteristics.

Acknowledgement

This work is partially supported by NSF of China (61602266, 61872201, U1833114), Science and Technology Development Plan of Tianjin (17JCYBJC15300, 16JCYBJC41900) and the Fundamental Research Funds for the Central Universities and SAFEA: Overseas Young Talents in Cultural and Educational Sector.

References

[1] S. R. Dulloor, *Systems and applications for persistent memory*. PhD thesis, Georgia Institute of Technology, 2015.

[2] "NVMe." https://www.nvmexpress.org/. [Online; accessed 10-July-2018].

[3] M. Bjørling, J. Axboe, D. Nellans, and P. Bonnet, "Linux block IO: introducing multi-queue SSD access on multi-core systems," in *Proceedings of the 6th International Systems and Storage Conference*, p. 22, 2013.

[4] P. Bonnet, "What's up with the storage hierarchy?," in *Conference on Innovative Data Systems Research*, 2017.

[5] K. Kant, "Data center evolution: A tutorial on state of the art, issues, and challenges," *Computer Networks*, vol. 53, pp. 2939–2965, 2009.

[6] "Flexible I/O Tester." https://github.com/axboe/fio. [Online; accessed 10-July-2018].

[7] A. D. Brunelle, "Block I/O layer tracing: blktrace," in *Gelato-Itanium Conference and Expo*, 2006.

[8] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. S. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," in *Proceedings of the USENIX Conference on Annual Technical Conference*, pp. 57–70, 2008.

[9] "Intel SSD DC S3510 Series Specifications." https://ark.intel.com/products/86197/Intel-SSD-DC-S3510-Series-480GB-2_5in-SATA-6Gbs-16nm-MLC. [Online; accessed 10-July-2018].

[10] A. Gulati, G. Shanmuganathan, I. Ahmad, C. Waldspurger, and M. Uysal, "Pesto: online storage performance management in virtualized datacenters," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, p. 19, 2011.

[11] J. Basak and M. Bharde, "Dynamic provisioning of storage workloads," in *Proceedings of the 29th Large Installation System Administration Conference*, pp. 13–24, 2015.

[12] F. T. Hady, A. Foong, B. Veal, and D. Williams, "Platform storage performance with 3D XPoint technology," *Proceedings of the IEEE*, vol. 105, pp. 1822–1833, 2017.

[13] "ZFS." https://en.wikipedia.org/wiki/ZFS. [Online; accessed 10-July-2018].

[14] Y. Son, H. Kang, J.-Y. Ha, J. Lee, H. Han, H. Jung, and H. Y. Yeom, "An empirical evaluation of enterprise and SATA-based transactional solid-state drives," in *Proceedings of the 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 231–240, 2016.

[15] Y. Son, H. Kang, H. Han, and H. Y. Yeom, "An empirical evaluation and analysis of the performance of NVM express solid state drive," *Cluster Computing*, vol. 19, pp. 1541–1553, 2016.

[16] T. Makatos, Y. Klonatos, M. Marazakis, M. D. Flouris, and A. Bilas, "Using transparent compression to improve SSD-based I/O caches," in *Proceedings of the 5th European Conference on Computer systems*, pp. 1–14, 2010.

[17] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, "A comprehensive study of the past, present, and future of data deduplication," *Proceedings of the IEEE*, vol. 104, pp. 1681–1710, 2016.

[18] O. Khan, R. C. Burns, J. S. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads.," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, p. 20, 2012.

[19] "NTFS." https://en.wikipedia.org/wiki/NTFS. [Online; accessed 10-July-2018].

[20] "MySQL." https://www.mysql.com/. [Online; accessed 10-July-2018].

[21] "lzbench." https://github.com/inikep/lzbench. [Online; accessed 10-July-2018].

[22] A. Kopytov, "Sysbench: A system performance benchmark." https://github.com/akopytov/sysbench, 2004.

[23] S. Aghav, "Database compression techniques for performance optimization," in *Proceedings of the 2nd International Conference on Computer Engineering and Technology*, vol. 6, pp. V6–714, 2010.

[24] J. Ma, B. Yin, Z. Kong, Y. Ma, C. Chen, L. Wang, G. Wang, and X. Liu, "Leveraging page-level compression in MySQL - a practice at baidu," in *Proceedings of the 14th IEEE International Symposium on Parallel and Distributed Processing with Applications*, pp. 1085–1092, 2016.

[25] Q. Xu, H. Siyamwala, M. Ghosh, T. Suri, M. Awasthi, Z. Guz, A. Shayesteh, and V. Balakrishnan, "Performance analysis of NVMe SSDs and their implication on real world databases," in *Proceedings of the 8th ACM International Systems and Storage Conference*, p. 6, 2015.

[26] K. Wu, F. Ober, S. Hamlin, and D. Li, "Early evaluation of Intel Optane non-volatile memory with HPC I/O workloads," *arXiv preprint arXiv:1708.02199*, 2017.

[27] S. Hajnoczi, "Towards multi-threaded device emulation in QEMU," in *KVM forum*, 2014.

[28] T. Y. Kim, D. H. Kang, D. Lee, and Y. I. Eom, "Improving performance by bridging the semantic gap between multi-queue ssd and I/O virtualization framework," in *Proceedings of the 31st IEEE Symposium on Mass Storage Systems and Technologies*, pp. 1–11, 2015.

[29] D. Zhang, H. Wu, F. Xue, L. Chen, and H. Huang, "High performance and scalable virtual machine storage I/O stack for multicore systems," in *Proceedings of the 23rd IEEE International Conference on Parallel and Distributed Systems*, pp. 292–301, 2017.

[30] F. Zheng, "Userspace NVMe driver in QEMU," in *KVM forum*, 2017.

[31] S. Hajnoczi, "Applying polling techniques to QEMU," in *KVM forum*, 2017.

[32] "Intel Optane Technology." https://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology.html. [Online; accessed 10-July-2018].