Storage Networking Industry Association
Technical White Paper

Introduction to SNIA Persistent Memory Performance Test Specification

October 2020

Abstract:

This white paper is targeted at storage professionals familiar with the SNIA Performance Test Specifications (PTS), storage and software architects interested in understanding, testing and designing persistent memory into their storage architectures and Persistent Memory (PM) aware software applications and storage architects and marketing managers interested in PM storage solutions.

USAGE

DISCLAIMER

The information contained in this publication is subject to change without notice. The SNIA makes no warranty of any kind with regard to this specification, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this specification.

Suggestions for revisions should be directed to http://www.snia.org/feedback/.

Table of Contents

I. Abstract 6
II. Introduction 7
III. Background – The Need for Performance Test Standards 8
IV. Scope 10
V. Background: Block IO v PM Byte Access 10
VI. Byte v Block IO 11
VII. Different IO Paths for PM access 13
A. Block Access with Sector Atomicity 13
B. Block Access without Sector Atomicity 14
C. Direct Access 14
VIII. PM Test Methodologies & Tests 15
A. DIRTH Test 15
B. REPLAY Test 16
A. INDIVIDUAL STREAMS Test 17
IX. Test Settings & Reference Test Platform 18
X. Conclusions 18
XI. About the Authors & Contributors 19

Table of Figures

Figure 1 - Storage Hierarchy 6
Figure 2 - Storage Hierarchy: Capacity & Speed 6
Figure 4 - Hardware View: Block v Byte Addressable Access 10
Figure 5 - Software View: Block v Byte Addressable Access 11
Figure 6 – Demand Intensity Outside Curve 14
Figure 7 - IOPS & ART v Total OIO 14
Figure 8 - Real Time Plot: IOPS & Response Times Quality of Service 15
Figure 9 - Replay Test: Mmap, Msync, Non Temp W 16
Figure 10 - Ind. Streams Test: Mmap, Msync, Non Temp W 16

I. Abstract

Persistent Memory (PM) is generally defined using the following characteristics:

- Very low latency, achieving memory/DIMM speeds
- Non-volatile, data persists through power cycles and beyond application and system resets
- Byte Addressable, can directly access storage media using byte address
- High performance for applications

Some variants also deliver higher capacity in the memory tier at a lower cost than DRAM. In general, all types will provide higher performance than solid-state drives.

Accordingly, there is significant interest in creating a Performance Test Specification (PTS) for PM. This would encompass test settings, metrics, methodologies, benchmarks, and reference options. These tests would provide reliable results over repeated application.

This white paper targets test development professionals working on storage and memory architectures. The current PM PTS v1.0.1 is applicable to both block IO read/write tests as well as byte-addressable, load/store architecture. The specification is also applicable across a variety of PM such as 3D XPoint, NVDIMM, MRAM, ReRAM, and other media. Follow-on White Paper targets will apply the tests to more specific architectures and define new test benchmarks.

The PM PTS utilizes both synthetic and real-world workloads. Synthetic tests are based on modifications from the SNIA PTS v2.0.1 for NAND Flash SSD. Real-world workload tests are based on the SNIA Real World Storage Workload (RWSW) PTS for Datacenter Storage v1.0.7.

On release, the PM PTS v1.0.1 is intended to allow users to understand and optimize the software stack for PM storage. Interested industry professionals, researchers, and academia are invited to participate in the development of the PM PTS by contacting SNIA at askcmsi@snia.org

II. Introduction

Persistent Memory (PM) is broadly defined as high performance, low latency, byte addressable, non-volatile storage that sits on a cache-coherent link as opposed to traditional Block IO storage that sits on the PCIe bus. Future coherent PCIe implementations could also support PM. PM is expected to occupy a tier in the storage hierarchy below Main Memory DRAM and above NAND Flash based SSDs - see Figure 1 - Storage Hierarchy.

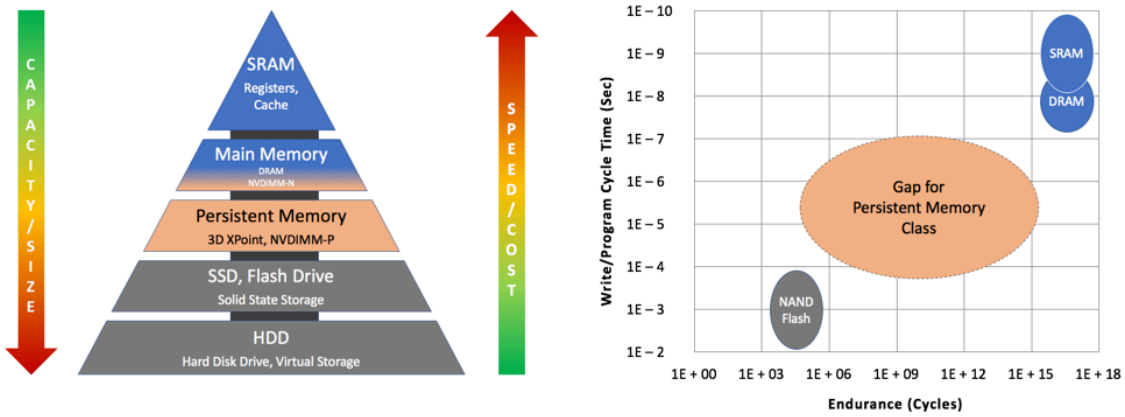

Figure 1 - Storage Hierarchy

PM is cheaper and provides higher capacity than main memory DRAM but is more expensive and faster than SSDs - see Figure 2 - Storage Hierarchy: Capacity & Speed.

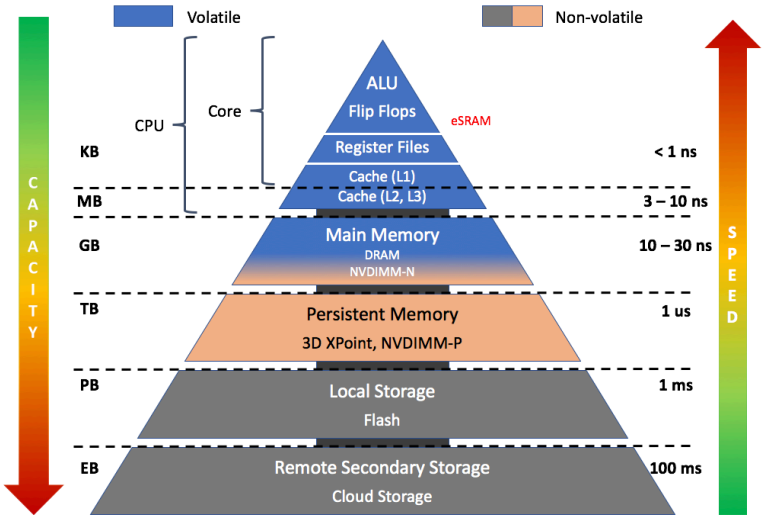

Figure 2 - Storage Hierarchy: Capacity & Speed

This WP focuses on the characterization, optimization and test of PM storage architectures as opposed to PM programming models and software optimization discussed in other SNIA technical works (such as the NVM Programming model which can be viewed at https://www.snia.org/sites/default/files/technical_work/final/NVMProgrammingModel_v1.2.pdf
Examples of PM storage technologies include 3D XPoint, NVDIMM DRAM, Phase Change Memory, MRAM, ReRAM, STRAM and others.

III. Background – The Need for Performance Test Standards

Hard Disk Drives

Early performance specifications for Hard Disk Drives (HDDs) focused largely on the need to define specific synthetic access test patterns to allow buyers to easily compare HDD performance. The first benchmark "corner case stress" tests (such as random 4K write saturation tests) were designed to monitor HDD performance outside the range of normal operation – hence the moniker "corner case stress" tests. Typical benchmark tests measured Random (RND) or Sequential (SEQ) Access of Read or Write (R/W) IOs of various Block Sizes (BS) and reported performance in terms of IO rate (IOPS), bandwidth (MB/s) and response times (or latencies).

While HDD "speeds, feeds and capacities" continued to advance, HDD performance did not keep pace with the corresponding increase in performance of the CPU, main memory, memory caches and overall software/hardware (SW/HW) stack. This gap in performance led to SW/HW stack optimizations to compensate for slower HDD performance. Here, the faster SW/HW stack queues IO requests for the slower HDD storage tier to free the higher performance (and more expensive) CPU and main memory resources to conduct other tasks until the queued HDD IO requests are fulfilled. This compensator was traditionally implemented through DRAM based caches, which proved to be quite expensive.

NAND Flash

This performance separation between the slower HDD performance and the faster SW/HW stack created the need for a faster storage tier and paved the way for the introduction of NAND flash based Solid State Drives (SSDs). SSDs are an order of magnitude, or more, faster than HDDs. However, NAND flash based SSDs also introduce variables that affect the consistent and repeatable measurement of SSD performance.

The increased speed of SSDs and the peculiarities of NAND flash storage (such as R/W asymmetry, FOB (Fresh-Out-of-Box) peak behavior, limited endurance write cycle life, block write/page erase operation, and the need for pre conditioning to measure performance at steady state) required the standardization of the test hardware platform, data path components and drivers, operating system, test software, test methodology, test settings and the specific test procedure used in order to consistently compare performance among different SSD products.

To address these new issues, the SNIA Solid State Storage (SSS) Technical Working Group (TWG) developed and issued a series of SSS Performance Test Specifications (PTS) for client and enterprise class storage. These PTS are intended to set forth standard synthetic benchmark tests and Reference Test Platforms (RTP) to normalize the effect of OS, hardware, test settings, test methodologies and test software on SSD performance results. The SSS PTS v2.0.1 defines a standardized test methodology to prepare and test NAND flash SSDs at Steady State.

Real World Workloads

The advance of storage solution architectures and applications (including storage virtualization, storage tiering, remote and fabric storage, data compression, data dedupe, encryption, open SSD, computational storage, AI and Machine Learning and other optimizations) has highlighted the importance of the test workload in benchmarking SSD performance. SSDs are inherently sensitive to the IO Stream content and intensity, or Queue Depth (QD) of the workload. Different types of IO Streams and the Demand Intensity of the workload (or QD) greatly affects IO, Bandwidth and Response Time

saturation and overall performance. Because application workload content also changes at each layer of the SW stack and abstraction, the capture, analysis and test using real world workloads has become increasingly important to performance benchmarking.

This focus on the content of real world application workloads, and the effects of the SW/HW stack on workload composition and SSD performance, has resulted in the release of the SNIA Real World Storage Workload (RWSW) PTS for Datacenter Storage v1.0.7. This RWSW PTS sets forth standards for the capture, analysis and test of real world workloads.

Non Volatile Memory Programming Model

The continued advance of SW/HW stack performance has also created the need for a new storage tier closer to main memory. SNIA released the NVM Programming Model to address the ongoing proliferation of new non-volatile memory (NVM) functionality and new NVM technologies and defines recommended behavior between various user space and operating system (OS) kernel components supporting NVM. This NVM Programming Model defines the behavior user-space software used to access PM and has driven the creation of standard programming models for persistent memory and associated PM drivers. This has, in turn, created the conditions for yet another new storage tier.

Persistent Memory Storage

Persistent Memory (PM) storage is the most recent storage tier to be developed and resides between main memory DRAM and NVMe SSD storage. This new class of PM storage is in turn driving the need for a PM PTS to set forth industry standard methodologies for both block IO read-write and byte addressable load-store performance optimization and benchmarking.

PM storage and PM applications can access storage using both byte and block addressable IOs via the traditional IO stack or directly from user space. Traditional IO stack access can be done synchronously (pread or pwrite) or asynchronously (libaio). Another interesting attribute of persistent memory user space access is there may be a required cache flush step to make stores persistent, something which is relevant to performance. PM applications also tend to use smaller data transfer sizes (64 bytes to 8KB) as opposed to traditional Block IO transfer sizes. These factors have led to the need to define and standardize PM storage test settings, hardware configurations and test methodologies.

SNIA Vendor Neutrality & Product Agnosticism

The PM PTS v1.0.1 is intended to address the need for defining block and byte addressable performance benchmark methodologies and tests while maintaining the SNIA policy of vendor neutrality and storage architecture agnosticism. Accordingly, while new PM embodiments may only be a single or few sources at this time, every effort is made to maintain vendor neutrality and storage agnosticism.

For example, PM Modules are referred to by their general technology type rather than market name (e.g., 3D XPoint or Data Center PM Modules). Thus, while the first PM storage technologies addressed by the PM PTS are 3D XPoint technology and NVMDIMM-N/P, it is intended that subsequent revisions will include other PM storage architectures and technologies as they become commercially available.

IV. Scope

This PM PTS White Paper (WP) is targeted at storage professionals familiar with the SNIA PTS specifications**,** storage and software architects interested in understanding, testing and designing persistent memory into their storage architectures and PM aware software applications and storage architects and marketing managers interested in PM storage solutions.

V. Background: Block IO v PM Byte Access

In traditional block IO access, data is accessed in data units of some number of sectors expressed as a logical block address (LBA). Typically, the sector size is 512 bytes with the minimum LBA size of 512 bytes (0.5 KB) or 4096 bytes (4 KB). Additional bytes can be added to the sector size to accommodate data integrity fields resulting in 520 byte, 528 byte or other sector size.

Logical blocks are accessed across a storage protocol such as PCIe to the NAND Flash SSD or other storage (Hard Disk, Optical Disk, Tape, etc.). In this case, CPU IO requests cannot directly access the block IO storage and must rely upon a drive protocol to access the LBAs.

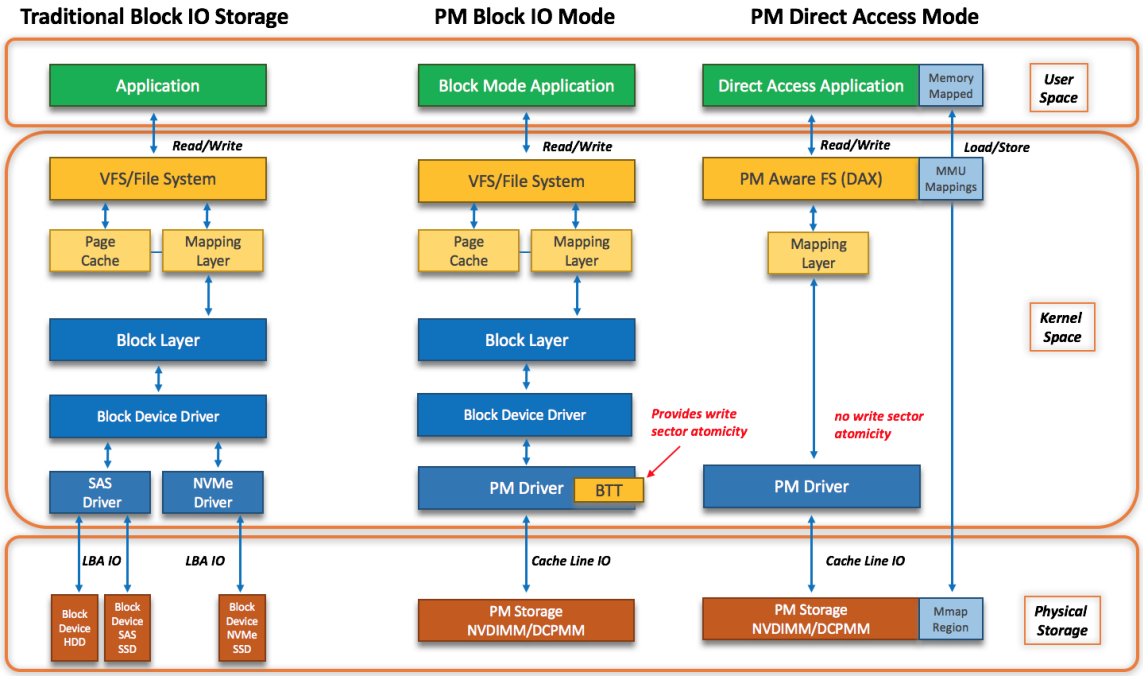

Figure 3 - Traditional Block IO v PM Direct Access Mode

In byte access load-store, the CPU (and IO request) can directly access storage media using byte address access to cache lines (such as 64 byte cache lines). When using a PM device in block IO mode, a PM aware driver converts block IO requests into byte addressable (cache line) memory copies. This conversion is transparent to the user and results in an increase in "traditional" block IO performance when applied to a PM across the PM Driver.

For PM Aware applications, cache lines can be directly accessed by the CPU and result in much higher performance than block IO over a PM driver or traditional block IO across PCIe.

VI. Byte v Block IO

Block IO Access. The fact that CPU and storage are separated has some implications, the most important being the fact that the CPU cannot talk *directly* to the storage subsystem. By directly, it is meant that an instruction executed by the CPU cannot issue a read/write (i.e., IO) operation directly against a storage device. Instead, the CPU needs to send *requests* to a device over an IO bus using a protocol, such as PCIe. Since this communication is handled by a driver, a call to the operating system (OS) is inevitable (often times, including the mandatory context switch to the OS, there may also be a context switch to other process to run on the same CPU core). To hide some of the latencies involved, IO requests are made in blocks of some predetermined large size such as 4 kibibytes (KiB) (4096 bytes). Moreover, data may need to be copied to an intermediate Dynamic Random Access Memory (DRAM) buffer in user space, which is different from the page cache where the application accesses it using regular load and store instructions.

Byte Access. The previous paragraph describes block access. In contrast to block IO access, byte access allows the CPU to talk directly to the media. In some places, you see this type of media being referred to as *byte-addressable*. An example of byte-addressable media is Double Data Rate (DDR) memory. Any of the persistent memory technologies that sit on the memory bus are also byte-addressable. Figure 4 shows the difference between these two modes of access at the hardware level while Figure 5 looks at these differences at the software level.

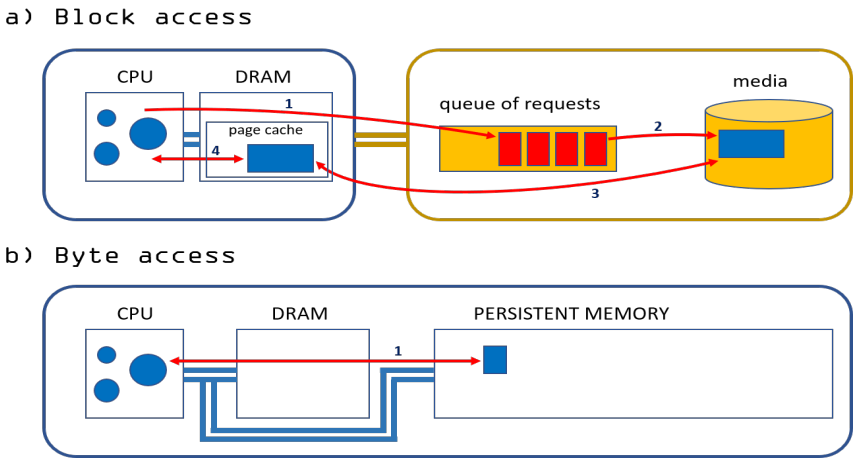

Figure 4 - Hardware View: Block v Byte Addressable Access

Hardware Steps – Block IO access path. The numerals in Figure 4 specify different steps in the IO path. In the case of Block access, Figure 4a, all IO operations require the CPU to:

(1) Queue requests in the device;
(2) Run requests by the device itself which may trigger additional step(s) – see next.
(3) Direct Memory Access (DMA) operation to read blocks from DRAM and write them to the storage media, or the other way around where the page cache can be bypassed. For the sake of simplicity, Figure 4 only considers the case with page cache. It is also important to consider that Block IO typically takes long enough that the OS will switch away to other runnable processes while waiting for the device to complete the IO, then switch back when the IO is done.
(4) The application simply reads the data if the operation was a read, or writes the data before issuing a write operation.

Hardware Steps – Byte IO access path. In Figure 5b Byte access, the CPU can directly access the data in the device at cache line granularity and in a single step.

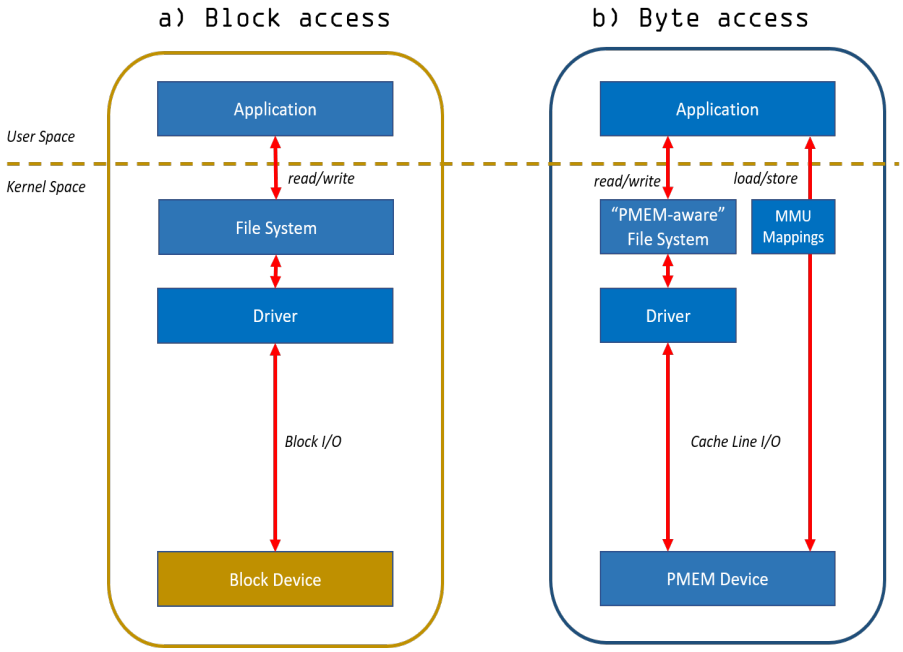

Figure 5 - Software View: Block v Byte Addressable Access

Software Steps – Difference between Block and Byte Access. In Figure 5, we can see the two main differences between block and byte access at the software level.

1) The first difference is that all IO done in Figure 4b is always done at cache line granularity, even when blocks are read or written through a file system.

2) The second difference is that it is possible to bypass the OS by memory mapping a file. The application can, in this case, access the file through loads and stores, as well as flush data out of the CPU caches, directly from user space.

While Figures 4 and 5 do not cover all of the possible ways in which applications can do IO, they are adequate for the sake of this discussion. As an example, consider the case of an application that memory maps a file in the block access case, Figure 5a, and then accesses the data using load and store instructions. Even in that case, the file system still needs to fetch those blocks from the device and store them in DRAM. In fact, accessing bytes from a non-cached (but mapped) block is essentially the same as issuing a read request to the device for that whole block. Also, the file system is involved when the application wants to flush pending writes out of DRAM buffers to make sure they are persistent.

Before moving on, let's summarize the important lessons to take away from this section.

In block access:

1. Access to data in the media is done in blocks.
2. The CPU cannot access the data directly.
3. The OS is always needed in the IO path.

In byte access:

1. Access to data in the media is always done at cache line granularity, even when IO is done through a file system calling read/write. Operations larger than a cache line are broken up.
2. The CPU can access the data directly.
3. System calls to the OS can be avoided by memory mapping files, which allows the user to issue loads, stores, and flush data out of the CPU caches completely from user space.

VII. Different IO Paths for PM access

As discussed, PM is both fast (low latency) and persistent (nonvolatile) and occupies the storage hierarchy gap between the fastest SSD found in the market and DRAM memory (see Figure 1). In terms of latency, PM response times are much closer to DRAM's than to SSD latencies. This powerful combination of low latency and persistence provides additional IO access advantages to the storage architect and software developer.

Application developers can take full advantage of PM in the type of IO accesses used. Without PM, developers may need to rely upon workarounds such as using large SEQ block sizes to mask high (slow) non PM access latencies associated with smaller block RND and SEQ accesses. However, with PM, developers can now rely on faster, smaller block RND and SEQ access as well as memory copies (load and store instructions) directly from user space without invoking a system call to the OS.

Nevertheless, PM can still be used with traditional IO stacks as well as allowing use with existing non PM aware applications. The following subsections introduce the different modes in which PM can be used by applications.

A. Block Access with Sector Atomicity

It is possible to use persistent memory as storage without changing your application or file system, as long as you have the proper PM drivers and tools installed in the system.

As was mentioned before, all IO done against persistent memory is done at cache line granularity (typically 64 byte). This means that all IO operations are converted by the driver into memory copies. PM does not provide power fail write atomicity for applications and file systems that rely on write atomicity at the block/sector level. This means that applications' data can be corrupted by torn sectors in the event of a crash or power failure (because x86 architectures only guarantee 8 bytes will not be torn by crash or power failure).

To avoid such a scenario, the PM driver supports a mode allowing atomic sector writes called *sector mode*. In this mode, the driver maintains a data structure called the Block Translation Table (BTT) to make sure torn sectors do not occur (see Figure 3).

B. Block Access without Sector Atomicity

Modern file systems, such as ext4 and xfs in Linux, are persistent memory-aware since Linux kernel version 4.2 and hence will work fine with persistent memory media, which does not provide power fail write atomicity. The mode to use PM without power fail write atomicity is called fsdax.

Keep in mind, the protection that persistent memory-aware file systems provide only relates to the file system metadata, and never the application's data. If your application relies on write atomicity for sectors of data, you may need to redesign your application. Likewise, if you use the device without a file system (as some databases do), the application will also need to be aware of this fact.

C. Direct Access

Direct access, or DAX, is a "special case" of fsdax where applications can memory map files and read/write to them through memory copies directly from user space. How DAX can be configured in a system differs by OS. In Linux, for example, this is accomplished by mounting a persistent memory-aware file system with the option "DAX".

Here, DAX means direct access from the point of view of applications (remember that the CPU always has direct access to persistent memory devices).

It is important to point out that the DAX option for fsdax was designed for programming against persistent memory devices following the NVM Programming Model (NVMPM) standard developed by the Storage and Networking Industry Association (SNIA). In other words, it was designed specifically to allow applications to access bytes in place through a direct pointer to the media, instead of copying them to a local buffer first. In fact, doing the latter may just produce an extra, and probably unnecessary, data copy between two memory devices (DRAM to PM) on the same bus.

The DAX option, apart from allowing applications to access the persistent media directly from user space when files are memory-mapped, also automatically bypasses the page cache (it must do so). It should be mentioned that bypassing the page cache has always been possible with traditional file systems. For example, in the case of POSIX, page cache can be bypassed by opening files with O_DIRECT. The difference here is that in the traditional case:

1. All IO is still done to/from user-space buffers (no direct pointers to media).
2. The OS is still in the IO path.

Although DAX can be considered a better way to bypass the page cache when using PM devices in general, whether you should use DAX or O_DIRECT may depend on your application.

VIII. PM Test Methodologies & Tests

The PM PTS sets forth test methodologies and tests designed to benchmark various types of Persistent Memory using both Synthetic and Real World Workloads. These tests are based, in large part, on SNIA SSS PTS v2.0.1 for SSDs and the RWSW PTS for Datacenter Storage v1.0.7. In both cases, modifications are made to accommodate the behaviors of PM storage as opposed to those of NAND Flash SSDs. For example, PM does not display the write hysteresis of NAND Flash and thus eliminates the need for much of the test process steps associated with a device Purge, pre-conditioning to steady state and block size/demand intensity sequencing in test flows.

The Synthetic tests in the draft PM PTS v0.3 are:

1. DIRTH Single Stream test – single Block Size/RW mix IO Stream applied across a range of Demand Intensity
2. DIRTH Multiple Stream test – multiple Block Size/RW mix IO Streams applied across a range of Demand Intensity

The Real World Workload tests in the draft PM PTS v0.3 are:

1. Replay test – apply the sequence and combination of IO Streams and QDs observed in the real world workload capture
2. Individual Streams test – testing each of the observed and selected IO Streams from the real world workload as single Block Size/RW mix saturation test.

A. DIRTH Test

The DIRTH test is an acronym for the Demand Intensity Response Time Histogram test. It is a cornerstone test of the SNIA PTS for SSDs and the Real World Storage Workload PTS for Datacenter Storage because it can be used to apply any fixed stimulus – synthetic benchmark or real world workload IO streams – to assess storage performance (IO, Bandwidth, Response Time, CPU saturation) across a range of Demand Intensity.

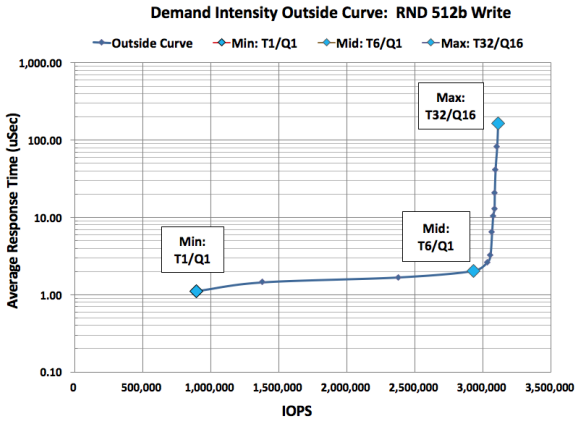

Figure 6 – Demand Intensity Outside Curve

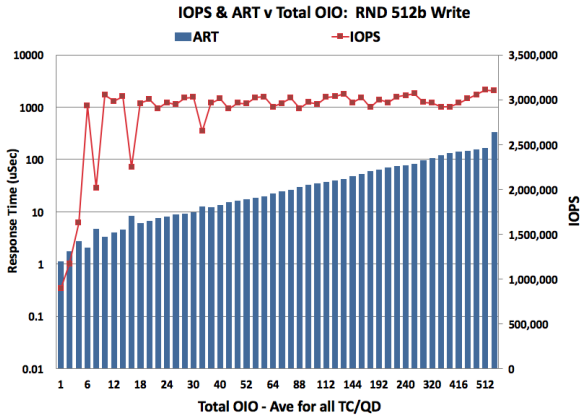

Figure 7 - IOPS & ART v Total OIO

Any storage performance measurements depend, in large part, on the Demand Intensity of the workload. In this case, Demand Intensity (DI) refers to the number of Outstanding IOs (or more simply put jobs, threads or requests) that are applied to the storage – if there is insufficient DI, there are not enough requests to get to the maximum IO or Bandwidth performance of the storage; if there is excessive DI, bottlenecks caused by too many IOs

simultaneously trying to access the storage will result in elevated response times and decreased IO rates.

The advantage of the DIRTH test is that the user can simultaneously assess multiple dimensions of performance: IOs, Bandwidth (BW), Response Times (RT), CPU Usage and Demand Intensity. Generally, when DI is low, IOPS are lower but Response Times are faster. Conversely, when DI is high, IOPS and Bandwidth are higher but associated Response Times are slower and CPU System % usage is higher. See Figures 6 & 7.

The key value of this test is to see at what level of DI that the storage attains peak IO/BW/RT performance and at what point RTs saturate and become increasingly slow or CPU System % usage becomes increasingly high.

By plotting IO, BW & RTs against Response Times, the test operator can plot both: a) Demand Intensity Outside Curves that shows IO/BW as a function of DI, as well as b) IO/BW/RT/CPU % against total Outstanding IOs (or DI) to show the expected performance of the storage device subjected to the workload across a range of Demand Intensity (or Users/Thread Count). See Figures 7.

B. REPLAY Test

The Replay test applies the sequence of IO Streams and QDs observed in the original real world workload IO Capture to the test storage. Defined in the RWSW PTS 1.0.7 for DC Storage, the Replay test re-creates the sequence of changing IO Streams and QDs from the IO Capture workload. Each step of the Replay test is applied to the test storage for a period of time (or step duration) as desired by the test operator.

The Replay test allows the reader to see the target test storage performance subjected to the workload IO Streams and QDs captured from the real world application storage server. Key performance indicators, such as IOPS, Bandwidth, Response Times, Queue Depths, CPU Usage %, power consumption, temperature, LBA Range address, TRIMs, write amplification factor and other metrics can be viewed. See Figure 8 below.

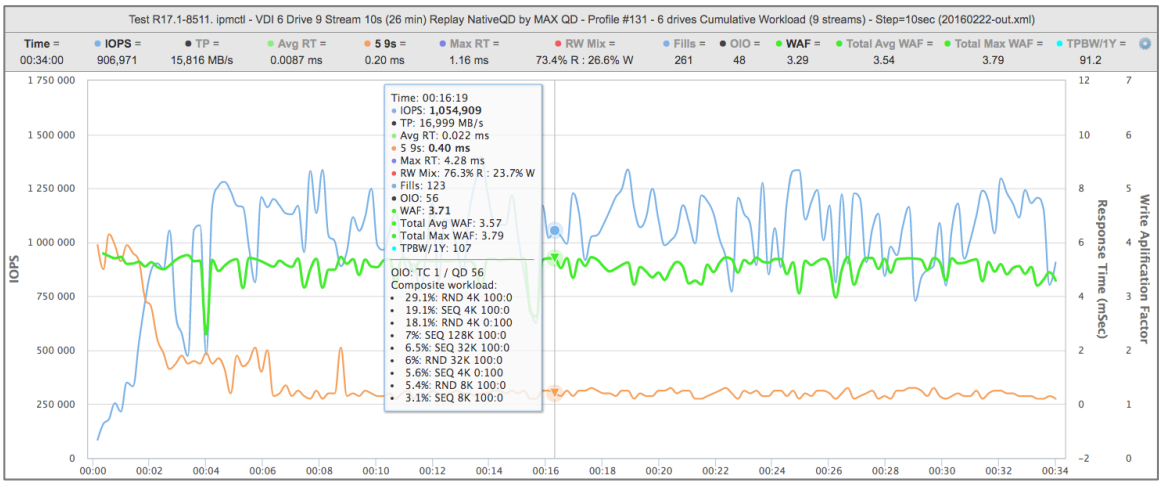

Figure 8 - Real Time Plot: IOPS & Response Times Quality of Service

Replay test performance can be reported on a Real Time Plot that shows performance over time (each time step of the Replay test) or can be reported as a single summary value (such as average IOPS over some portion or all of the time steps of the Replay test).

The key value of the Replay test is to show how test storage responds to the changing combinations of IO Streams and QDs observed in the real world workload capture as opposed to synthetic corner case tests that measure a single IO Stream or fixed combination of IO Streams. As such, Replay tests can show you how well your test storage may respond to the intended real world application workload.

A. INDIVIDUAL STREAMS Test

The Individual Streams test measures the performance of each individual IO Stream observed and tested in the Replay test. By applying the WSAT (Write Saturation) test, the test operator can measure the steady state performance of each individual IO Stream and compare that value to manufacturer or test specification values for corner case tests.

Traditional storage manufacturer performance test specifications tend to highlight a few commonly used metrics such as RND 4K Read/Write (for IOPS), SEQ 128K Read/Write (for Bandwidth) and RND/SEQ 4K Read/Write Latency (or Response Time at a single outstanding IO).

However, as it quickly becomes apparent, real world workload IO Stream content is rarely comprised of a single IO Stream. Further, the major IO Streams of a real world workload usually contain nontraditional benchmark BS/RW mix such as 1K, 2K, 0.5K, 396K and other BS fragments. In those instances when the real world workload *does* contain traditional BS streams (such as RND 4K R/W or SEQ 128K R/W), those IO Streams are usually a small percentage of the total IO Streams of the real world workload.

Figure 9 below shows the IOPS, Average Response Times and 5 9s Quality of Service (QoS) in 3 modes (Mmap, Msync and Non Temporal Writes) for a Datacenter Storage Replay test where all of the IO streams are averaged over the duration of the test. Figure 10 below shows the performance of each individual IO stream observed in the Replay test where each IO Stream is tested individually to Steady State.

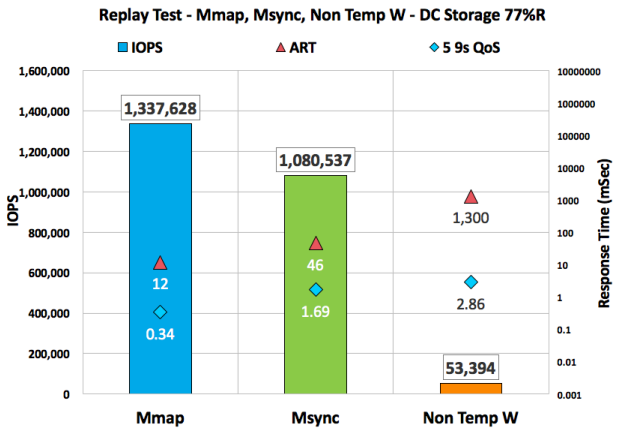

Figure 9 - Replay Test: Mmap, Msync, Non Temp W

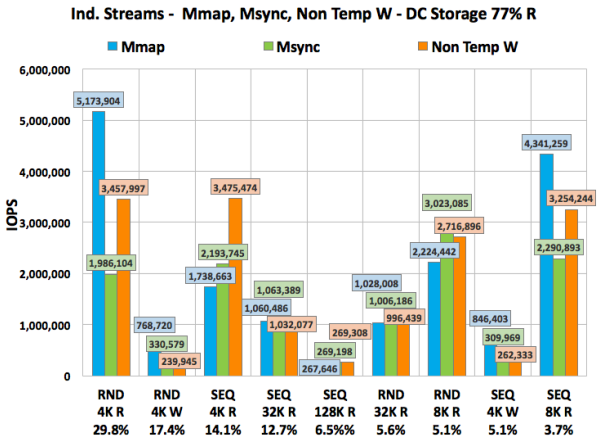

Figure 10 - Ind. Streams: Mmap, Msync, Non Temp W

The key value of the Individual Streams test is to: 1) show which BS/RW mix IO Streams are in the real world workload; 2) show the performance of each of those individual IO Streams at steady state; and 3) compare real world workload IO Stream performance to manufacturer specification benchmark values for the same single BS/RW mix IO Stream.

IX. Test Settings & Reference Test Platform

The PM PTS will define a PM Reference Test Platform (PM RTP) for each type of PM covered by the specification. The RTP is only a recommended test platform and not a requirement of the PM PTS and is listed in an effort to normalize the hardware and software environment and test settings to allow consistent, repeatable and comparable performance benchmark data.

The PM RTP will be posted on the SNIA website and will be updated from time to time to reflect modifications to the PM PTS and the release of more advanced commercial platforms on software.

X. Conclusions

Persistent Memory storage products are filling the gap in the storage hierarchy by offering fast, low latency and persistent storage that moves memory closer to the CPU. The PM PTS is intended to introduce the concepts of Persistent Memory, PM Storage and the software and hardware requirements for use, development and integration of PM Storage and PM aware applications.

Early adoption of 3D XPOINT and NVDIMM-N can be seen in applications that require large in memory data sets (e.g. in-memory database and metadata and logging for AI mission critical systems), fast nonvolatile storage tiers and easily accessible storage space for high speed, computationally intensive applications.

This white paper is intended to help storage architects, software developers and marketing professionals to understand how PM fits in the storage hierarchy and to help define a technical and marketing roadmap for new PM aware applications.

Interested parties are invited to participate in this exciting area. Questions and comments can be directed to askcmsi@snia.org. By joining SNIA, individuals and companies can participate in SNIA Technical Working Groups and Initiatives to help drive the creation and adoption of Persistent Memory standards, evangelize PM storage architectures and contribute to PM educational materials.

XI. About the Authors & Contributors

The SNIA Solid State Storage Technical Working Group, which developed and reviewed this document, recognizes the significant contributions made by the following individuals and companies:

- Eduardo Berrocal, Intel
- Jim Fister, The Decision Place
- Eden Kim, Calypso Systems, Inc.
- Chuck Paridon, dxc
- Andy Rudoff, Intel