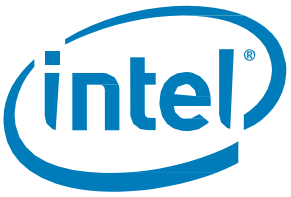
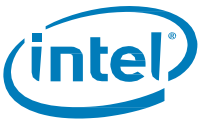


Performance Benchmarking for PCIe* and NVMe* Enterprise Solid-State Drives

White Paper

February 2015

Order Number: 330909-003US



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Testing performed internally by Intel.

Configurations used by Intel in the testing environment, using Intel test methodologies and hardware: IOMeter* tests run on Dell server R920 system with Intel® Xeon® CPU E7-4809 v2 @ 1.90 GHz (4 processors), 64-bit Operating System, x64-based processor, 4 sockets, 24 cores, 48 logical processors, Hyper-V support, 256 GB RAM, Microsoft* Windows* Server® 2012 R2 64 bit O/S.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as IOMeter*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance when combined with other products.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Material in this presentation is intended as product positioning and not approved end user messaging.

Intel does not control or audit the design or implementation of third party benchmark data or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmark data are reported and confirm whether the referenced benchmark data are accurate.

*Other names and brands may be claimed as the property of others.

Intel® is a trademark of Intel Corporation in the U.S. and other countries.

Copyright © 2015 Intel Corporation. All rights reserved.

Overview

Benchmarking performance for today's high-performance Enterprise PCIe* and NVMe* Solid-State Drives (SSDs), such as Intel® SSD DC P3x00 series, involves more than verifying a few data points on the product datasheet. To get a meaningful measurement and clearly determine the expected performance, a comprehensive measurement methodology must include a range of operating regions.

Performance measurements should span mixed workloads over a range of queue depths rather than rely solely on the all-read and all-write test cases. This is because the effects of mixing reads and writes are not obvious and sometimes non-linear. Hence, the measured values from all-read and all-write test results cannot be combined to compute the expected mixed-workload results.

Many enterprise and data center environments are sensitive to service time outliers, so performance measurements should also cover latency attributes. An SSD that delivers extremely high performance or input/output operations per second (IOPS) at an unacceptably high latency might not be useful for such enterprise applications or usage.

The host system's ability to attain the device IOPS is also important. When accessing the SSD, the host CPU efficiency contributes to the system cost of the IOPS, and can itself limit the overall attainable performance or IOPS.

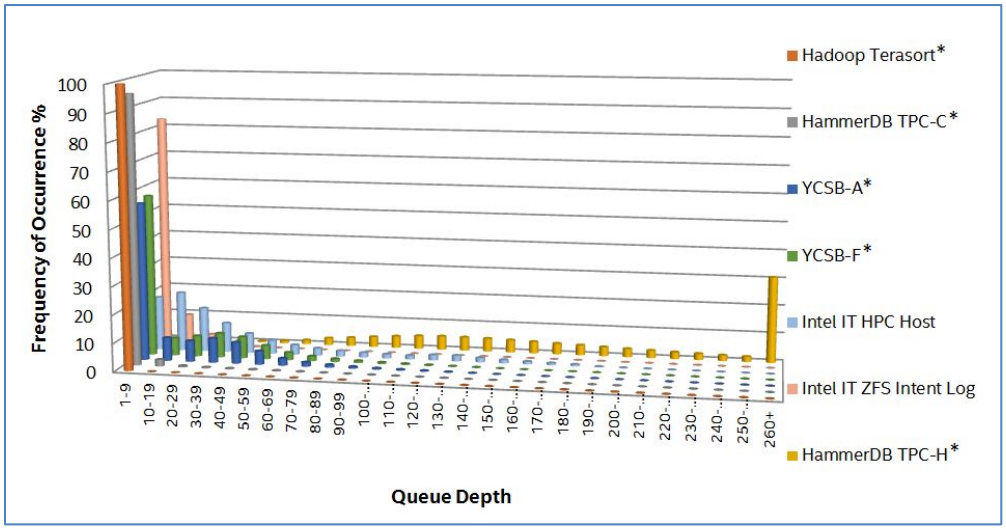
This paper describes the performance benchmarking methodology and considerations for the latest PCIe and NVMe enterprise SSDs for effective evaluation.

Performance Analysis

NAND Flash-based SSDs achieve much of their performance by harnessing the concurrency available from the multiple Flash components from which the SSD is constructed. To harness this concurrency, the SSD must be presented with multiple queued commands, which can be distributed to multiple components. The attainable performance is therefore sensitive to the level of concurrency, also known as Q*ueue Depth*. Random Mixed workload represents common enterprise use case.

Figure 1 shows the distribution of queue depths for a few sample data center workloads measured in the Intel environment to illustrate examples of the operating region.

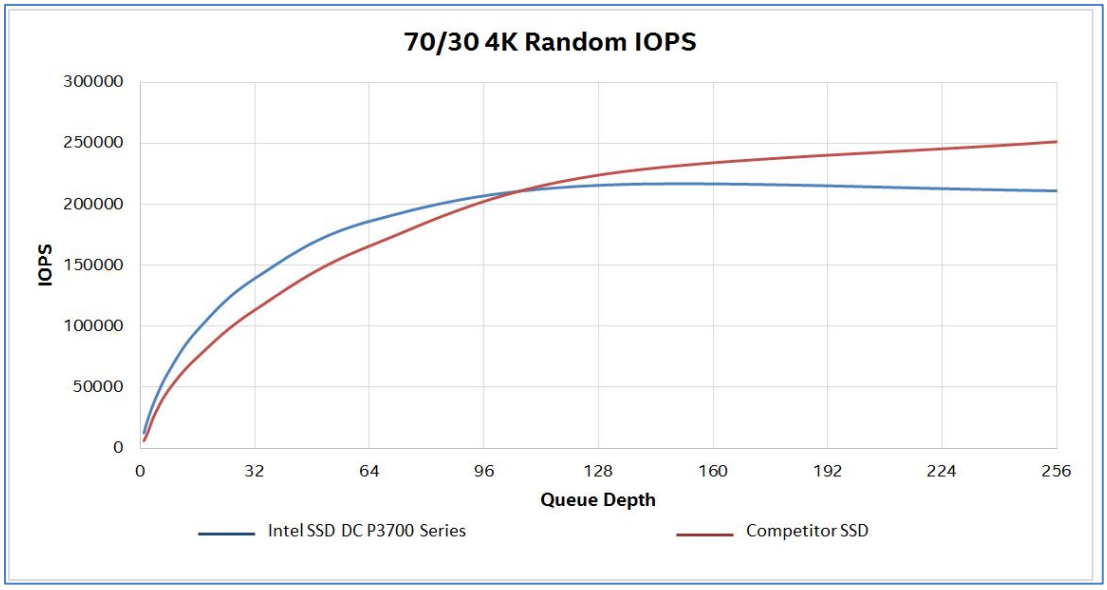
Figure 1: Queue Depths for Typical Data Center Workloads* (Source: Intel)



*Other names and brands may be claimed as the property of others. Intel® is a trademark of Intel Corporation in the U.S. and other countries.

Figure 2 shows the measured random mixed workload performance as a function of queue depth for an Intel SSD DC P3700 Series and a Competitor SSD.

Figure 2: Performance as Function of Queue Depths for Two NVMe SSDs (Source: Intel)



The above graph shows that as the queue depth increases, an increasing number of concurrent Flash components are utilized, thus increasing the performance. However, this increase is not linear with queue depth because random access will not distribute perfectly across the multiple dies in the SSD. As queue depth increases, there are more cases of commands landing on the same Flash component. As a result, the performance asymptotically approaches the saturation as the queue depth gets large.

For the two NVMe SSDs, the Intel SSD DC P3700 Series has higher performance at lower queue depths, while the Competitor SSD has higher performance at higher queue depths. The effective performance realized in the system depends on the queue depth that the system exercises.

The net effective performance of the SSD is the measured performance curve as a function of queue depth overlaid with the distribution of queue depths for the applied workload. Only the last workload in Figure 1 would benefit from the high queue depth performance of the Competitor SSD in Figure 2.

Performance and Latency

As described in the previous section, the performance scaling is non-linear and approaches saturation asymptotically. The service time, or *latency*, for I/Os increases with the increasing queue depth because each I/O must also wait for all I/Os ahead of it in the queue to complete. The latency also increases because some I/Os target Flash components that might already have other I/Os outstanding to it. As a result of the random distribution of requests, collisions occur naturally as the concurrency increases.

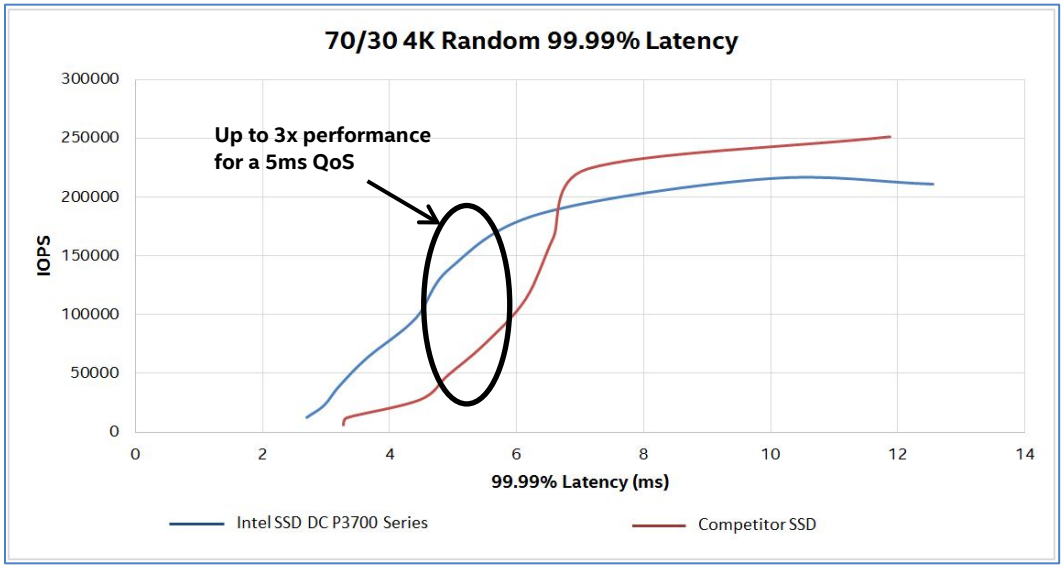
These collisions cause higher variability in the service times because some requests might be serviced immediately as they happen to target an idle Flash component, while other requests might target a Flash component that already has several other requests stacked up on it.

Average latency is not a metric worth examining because it is a re-statement of the average IOPS. For a given queue depth, the average latency is equal to the inverse of the average IOPS multiplied by that queue depth.

For applications that are sensitive to latency, the maximum latency and latency outliers provide more meaningful metrics. Rather than an absolute maximum, we often focus on the 99.99 percentile latency (the outliers) as a figure to convey the latency variability. This is essentially the amount of time that 99.99% of the IOPS take to complete (conversely, only 0.01% of the IOPS have a latency longer than this value). This is also referred to as Quality of Service (QoS) metric.

Figure 3 shows examples of relationships between mixed workload performance and 99.99 percentile latency (QoS) for the same two NVMe SSDs as in Figure 2 (Intel SSD DC P3700 Series and a Competitor SSD).

Figure 3: Relationship between 99.99% QoS and Performance (Source: Intel)



Note that once the curves reach the SSDs' maximum attainable IOPS (asymptotic values from Figure 2), then only the latency increases without further increases in performance. Although the Competitor SSD has a higher maximum attainable IOPS, it has higher latencies across the range up to the point where the Intel SSD DC P3700 Series' IOPS approach saturation and no longer increase. If an application is targeting a 5ms tolerable latency (for example, the Intel SSD DC P3700 Series SSD with the lower max IOPS), it can actually deliver up to 3X higher net IOPS to that latency sensitive application within the 5ms QoS constraint (roughly 150K IOPS compared to roughly 50K IOPS).

Mixed Workload Performance Computation

For Flash-based SSDs, there is typically a large asymmetry between the attainable read IOPS and attainable write IOPS. This asymmetry has a dramatic effect on the realizable mixed performance that is often more pronounced than what might be assumed. While different application scenarios present different mix ratios, one typical enterprise application's mix ratio often used for evaluation purposes is 70/30 (70% of the IOPS are reads and 30% are writes). The way to compute mixed workload performance is presented in this section.

The expected best case mixed IOPS performance is not the weighted combination of the corresponding read and write IOPS values. For example, if a drive has read IOPS saturation of 500,000 *IOPS* and a write IOPS saturation of 100,000 *IOPS*, the theoretical best case 70/30 mixed IOPS at saturation would **not** be:

Equation 1

$$0.7 \times 500{,}000 + 0.3 \times 100{,}000 = 380{,}000\ IOPS$$

This weighted blend of the IOPS is not the correct math to yield the expected blended IOPS values.

The proper method for estimating the expected best case 70/30 mixed IOPS is to take the amount of time it takes to perform 0.7 reads, plus the amount of time it takes to perform 0.3 writes, and then invert that resulting total time to yield IOPS. So for our earlier example, the expected best case mix would be:

Equation 2

$$\frac{1}{0.7 \times \frac{1}{500{,}000} + 0.3 \times \frac{1}{100{,}000}} = 227{,}000\ IOPS$$

This is contrary to the 380,000 IOPS, the incorrect math produced in Equation 1. If the drive was capable of 500,000 read IOPS, but the writes were reduced from 100,000 to 75,000 (drop of 25K IOPS), the resulting 70/30 blend would drop to 185,000 IOPS – that is, a 25K write IOPS reduction produced a blended drop of 42K IOPS. If that drop were to be made up by increasing the read IOPS, then the read IOPS performance would have to be raised from 500,000 to 1,750,000 IOPS:

Equation 3

$$\frac{1}{0.7 \times \frac{1}{1{,}750{,}000} + 0.3 \times \frac{1}{75{,}000}} = 227{,}000\ IOPS$$

Tradeoffs between reads and writes in determining the expected mixed IOPS performance can be unexpected and highly pronounced. Therefore, measurements should be directly made to the mixed IOPS performance rather than rely on the read and write performance figures to be used solely as performance attributes. For mixed workloads, it is often not the case that increased read IOPS can effectively compensate for a low write IOPS capability.

The following table summarizes the mixed workload computation based on all-read and all-write equations. It is clear that write performance has a much higher impact on effective mixed workload performance than the read performance.

Performance Metric	Case 1	Case 2	Case 3
Read IOPS	500,000	500,000	1,750,000
Write IOPS	100,000	75,000	75,000
Mixed 70-30 IOPS	227,273	185,185	227,273

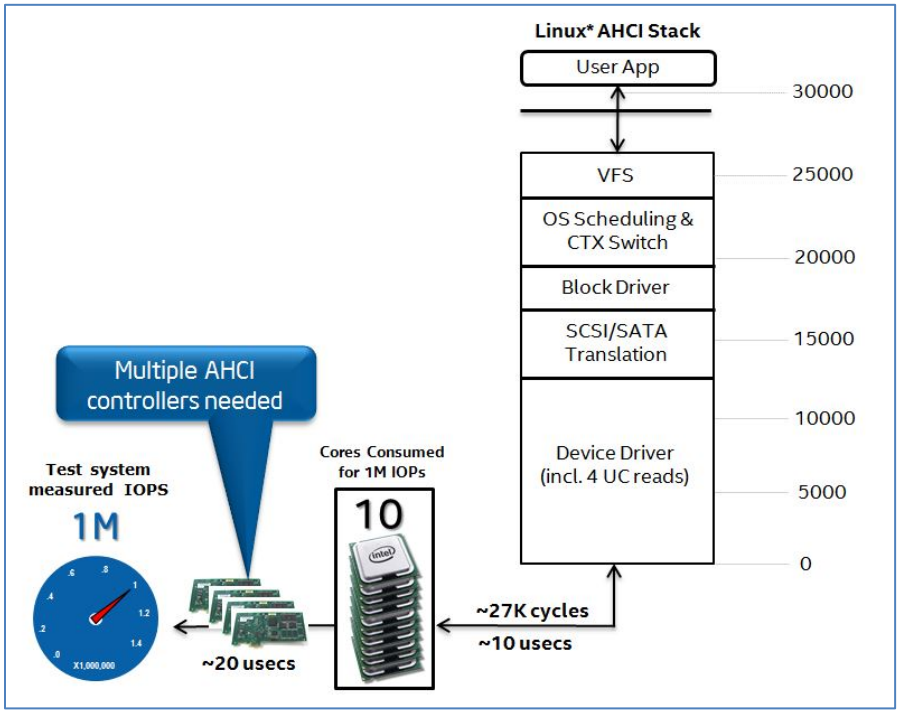
System Efficiency: PCIe/AHCI vs. NVMe Interface

For storage devices that support high performance, the system must realize that performance depends on the device performance attributes as well as the host capabilities. Each IOP takes some amount of host processing time to accomplish, and the host CPU can bottleneck on its processing before the device saturates on its ability to service the requests. In addition, hosts presumably perform some operations on the IOPS it generates, so it is important to leave some CPU processing power to accomplish work other than generating the I/Os.

Therefore, it is important that high-performance storage devices have attributes that are favorable to host CPU efficiency and keep CPU utilization low for processing the I/Os. New interfaces, such as NVMe, are devised explicitly for such low-host CPU utilization, which are highly efficient in processing I/Os.

Figure 4 shows how the CPU spends cycles in PCIe/AHCI interface in the various layers in the operating system stack, as well as in the device driver itself.

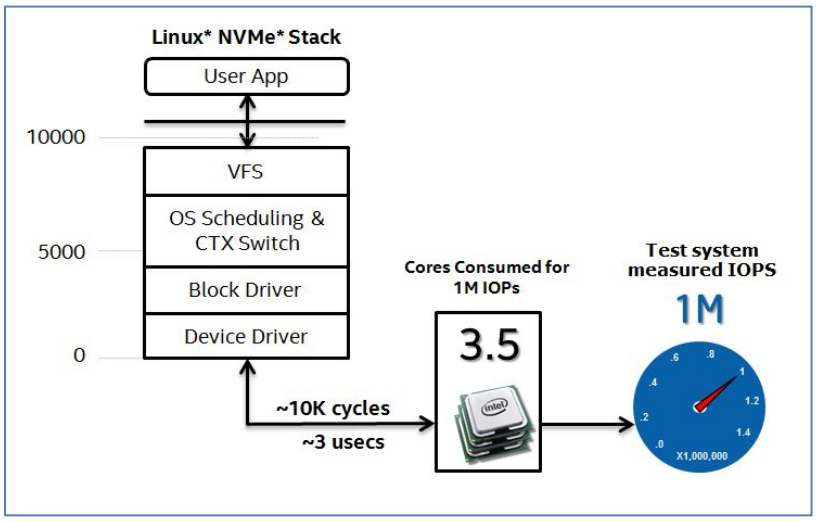
Figure 4: CPU Cycles for IOPS Using AHCI (Source: Intel)



Lab measurements of the number of CPU clock cycles it takes to complete a disk request when comparing the Linux* AHCI and NVMe interfaces has shown that the CPU efficiencies for different interfaces can vary over a large range. A Linux AHCI implementation is measured at approximately 27,000 CPU cycles (or roughly 10 µs).

If an I/O takes 10 µs of CPU time to accomplish, then a CPU will saturate at 100% utilization at 1/10 µs IOPS, or just 100,000 IOPS (of course, modern processors include several cores so this might not be the system limit). A system achieving 1 million IOPS using that solution would consume all the processing capability of 10 CPU cores. A typical Linux NVMe implementation on the other hand, measures at less than 10,000 cycles (or roughly 3 µs) and consumes a third of the host CPU as the AHCI implementation. With one third the CPU time spent, an NVMe device leaves more CPU cycles left for generating more I/O requests and for processing the data from its generating I/Os.

Figure 5: CPU Cycles for IOPS Using NVMe



When measuring the attributes of high-performance storage devices, measurement of the impact the devices have on the host is also important in driving clarity in the realizable system storage performance. Therefore, host CPU utilization should be included in measurements of high-performance storage devices. A device that delivers higher IOPS, but consumes all host CPU in doing so, might not realize better overall system storage performance than a device that might have slightly lower raw IOPS capability, but delivers those IOPS while being highly efficient with use of the host CPU.

For storage devices that support high IOPS, the system must realize that those IOPS not only depend on the device performance attributes, but also the host capabilities. Each I/O takes some amount of host processing time to accomplish, and the host CPU can bottleneck on its processing before the device saturates on its ability to service the requests. In addition, hosts presumably perform some operations on the IOPS it generates, so it is important to leave some CPU processing power to accomplish work other than generating the I/Os.

Conclusion

To enable an effective evaluation, it is important to consider different performance metrics for benchmarking today's high-performance enterprise PCIe and NVMe enterprise SSDs. These metrics are mixed workload performance at different queue depths, performance as a function of QoS, and system configuration.

Testing Methodology: IOMeter

Performance data used in the paper is measured with a new version of IOMeter. This version is an enhancement over version 1.1.0[1], and is available as open source software at following locations:

www.IOMeter.org/doc/downloads.html

sourceforge.net/projects/IOMeter/files/IOMeter-stable/1.1.0/

1. For the performance analysis presented in this paper, the latency bins were modified and extended in latest IOMeter version 1.1.0. These are found in `IOGrunt.cpp` starting at line 1047.