



PCIe SSD 101

An Overview of Standards, Markets and Performance

Members of the SNIA Solid State Storage Initiative
PCIe SSD Committee



The Solid State Storage Initiative

About SNIA

The Storage Networking Industry Association (SNIA) is a not-for-profit global organization made up of some 400-member companies and 7,000 individuals spanning virtually the entire storage industry. SNIA's mission is to lead the storage industry worldwide in developing and promoting standards, technologies and educational services to empower organizations in the management of information. To this end, SNIA is uniquely committed to delivering standards, education and services that will propel open storage networking solutions into the broader market. For additional information, visit the SNIA web site at <http://www.snia.org>.

About SSSI

The SNIA Solid State Storage Initiative (SSSI) was formed to foster the growth and success of the market for solid state storage in both enterprise and client environments. It consists of various subcommittees that are focused on developing technical standards and tools in order to educate users about the advantages of SSS devices. For additional information, visit the SNIA SSSI web site at <http://www.snia.org/forums/ssi>.

About PCIe SSD Committee

This SSSI committee provides guidance to the marketplace on PCIe SSDs. This can take the form of educational materials, best practices documents and SNIA standards. The group also coordinates with other industry organizations involved in PCIe SSD-related activities.

www.snia.org/forums/ssi/pcie



Table of Contents

List of Figures	3
List of Tables	3
1.0 Introduction	4
2.0 Standards	5
3.0 Programming Models for Non-Volatile Memory	16
4.0 PCIe SSD Market	19
5.0 PCIe SSD Performance	26
6.0 Contributors	30

List of Figures

Figure 1 mSATA (left) and M.2 (right) Form Factors	6
Figure 2 SFF-8639 Connector	7
Figure 3 SATA Express Connector	7
Figure 4 NVM Express Queues	10
Figure 5 PQI Domain	12
Figure 6 Queue Elements	13
Figure 7 SATA Express Logo	14
Figure 8 SCSI Express Components	15
Figure 9 Express Bay	15
Figure 10 Basic Storage Programming Model	17
Figure 11 Advanced Storage Programming Model	17
Figure 12 Persistent Memory Programming Model	18
Figure 13 Diagram of the I/O Hardware/Software Stack	23
Figure 14 Enterprise SSD Forecast	25
Figure 15 Random Writes at FOB	26
Figure 16 Random 4KiB Writes – Steady State	27
Figure 17 Random 4KiB Writes – Minimum Response Times	27
Figure 18 Maximum Response Times by Interface	28
Figure 19 Response Times by Interface	28
Figure 20 CPU Utilization Comparison	29
Figure 21 CPU Affinity Settings Impact on Performance	29

List of Tables

Table 1 Key Features of NVMe	9
Table 2 SOP/PQI Key Features	11
Table 3 Summary of PCIe SSD Standards	16



1.0 Introduction

PCI Express (PCIe) Solid State Storage devices and memory are becoming increasingly popular in the high performance computing environment for both client and enterprise storage. The development and deployment of NAND Flash based Solid State Drives (SSDs) has seen the introduction and adoption of SATA and SAS SSDs as replacements for or improvements on traditional magnetic Hard Disk Drives (HDDs). The introduction of even faster PCIe based SSDs with lower latencies has introduced yet another wave of SSD technology and architectures.

The mass storage ecosystem is now being optimized for SSDs. Client applications are anticipating one and two lane PCIe devices embedded on laptop motherboards. Meanwhile, enterprise applications are seeing the evolution of PCIe SSDs as both super fast block IO devices as well as persistent memory.

These developments have led to surges in hardware, software and standards works. This white paper will survey the broad landscape of emerging high performance PCIe storage and the implications of revolutionary applications of these new architectures.

PCIe SSD I/O is a cooperative effort of members of the SNIA Solid State Storage Initiative (SSSI) PCIe SSD Committee. This Committee is comprised of industry leading SSD OEMs, Developers, Test Houses, VARs, Analysts and End Users. Many contributing authors also participate in SNIA technical works including the Non Volatile Memory Programming TWG, the Solid State Storage TWG and SSSI committees.

Additional materials and resources can be found on the SNIA SSSI website at www.snia.org/forums/sssi.



2.0 SSD Performance States

Standards provide interoperability and interchangeability, meaning that devices from different manufacturers work together and can be interchanged with no loss of functionality. In this way, standards drive volume. To achieve these goals, PCIe SSDs must come in standard form factors, utilize standard connectors, and interact with operating systems in an identical way. And the final products must make use of these standard components in the same manner. The following sections will review the standards for the pieces that go together to make a PCIe SSD.

Form Factors

Convergence of mobile and computing devices is driving storage form factor evolution to find the optimal mix of power, performance, mobility and price. Ultrathin notebooks have been introduced to address this mobile computing segment demand for low system profiles with higher performance, while requiring lower power consumption. This ultrathin trend, in turn, encourages storage form factors with low profiles, resulting in the rapid proliferation beyond traditional HDD form factors.

Meanwhile, standard notebooks and desktops continue to push for increasing performance, also with an increased focus on low power. These latter platforms that are less space constrained could make use of the 2.5-inch form factor. This section will survey the various form factors addressing these interface and mobility trends.

HDD-Type Form Factors

The Hard Disk Drive (HDD), introduced by IBM in the 1950's, has been the prevailing storage device for computing applications for decades. Today's most widespread HDD form factors using a SATA interface are:

- 3.5" (EIA-740, SFF-8301, 1999)
- 2.5" (EIA/ECA-720, SFF-8223, 2007)
- 1.8" (SFF-8123)

The 1.8" form factor is used in only a small percentage of the client computing market. Solid State Drives (SSDs) have utilized these form factors in the effort to enable storage solution migration from HDD to SSD. The dimensions for these traditional HDD form factors as used by SSDs are:

- 2.5" (LxWxH)
 - 100.45 mm x 69.85 mm x 7 mm
- 1.8" (LxWxH)
 - 78.5 mm x 54 mm x 5 mm

Enclosed Form Factors

To address the low profile requirement, SFF-8201 (2.5-inch form factor specification) has been updated to include a 5 mm z-height and have the bottom mounting holes optional for 5 mm and 7 mm versions, while SFF-8083 defines a 5 mm HDD form factor with a 20-pin edge card connector.

Modular/Caseless Form Factors

As an emerging technology, client SSDs have adapted to various means of minimizing space constraints of mobile computing platforms through caseless versions of enclosed form factors mentioned above as well as other form factor standards.

mSATA

JEDEC introduced mSATA (MO-300B Variation A) and mSATA Mini (MO-300B Variation B) in 2010. mSATA is mechanically identical to the mini-PCIe card and mates with the existing motherboard socket. This introduction facilitated the use of SSDs in available mini-PCIe slots that could be configured for SATA storage. mSATA has been the small form factor of choice for today's caching SSDs.

See www.jedec.org for form factor specifications.



Figure 1. mSATA (left) and M.2 (right) Form Factors

M.2

Various custom solutions have been created to meet specific platform space limitations. While mSATA has allowed a timely and cost-effective transition for SSDs to modular solution, it has various limitations:

- Up to 3.85 mm z-height
- Single-lane interface
- Space limitation for NAND placements

To address these limitations, another form factor, M.2, has been introduced. The controlling specification resides within PCI-SIG while an informative reference is included in SATA specification 3.2. The PCI-SIG M.2 specification supports multiple add-in cards and modules as well as a number of host interfaces. The SATA-IO specification defines a SATA M.2 SSD. The M.2 module sizes are:

- Type 2230 22 mm x 30 mm
- Type 2242 22 mm x 42 mm
- Type 2260 22 mm x 60 mm
- Type 2280 22 mm x 80 mm
- Type 22110 22 mm x 110 mm



While these modules all have a width of 22 mm, they have varying lengths to allow systems designers expansion and integration flexibility as mobile computing platforms drive thinner and lighter storage solutions.

Visit www.pcisig.com and www.sata-io.org for details on these specifications.

Connectors

Connectors needed to be developed in order to bring PCIe to the drive form factor. There are multiple choices, but they are, for the most part, derivatives of existing drive connectors. All of the connectors support other interfaces in addition to PCIe, thus providing a bridge from traditional drive interfaces to PCIe. This section describes the primary connectors for PCIe SSDs.

SFF-8639

The SFF-8639 is a multi-function connector, supporting:

- 4 PCIe lanes
- Dual port SAS
- Single port SATA
- MultiLink SAS

The specification defines drive and host connectors, and is used in enterprise SSD environments, such as SCSI Express. SFF-8639 host connectors will mate with SFF-8482, SFF-8630, or SFF-8680, and SATA Express device connectors.

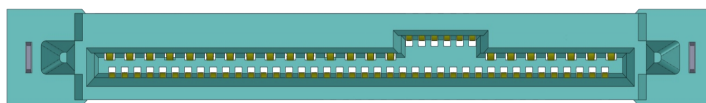


Figure 2. SFF-8639 Connector

SATA Express Connectors

SATA-IO brought PCIe to client storage by defining the SATA Express host and drive connectors containing up to two lanes of PCIe and one SATA port. Unlike the SFF-8639, the PCIe and SATA signals are multiplexed on the same pins, meaning that only PCIe or SATA can be in use at a given time. The SATA Express connectors are mechanically similar to legacy SATA connectors. A SATA Express host connector will mate and function with either a PCIe drive containing a SATA Express connector or a legacy SATA drive. For more information, go to www.sata-io.org/technology/sataexpress.asp.

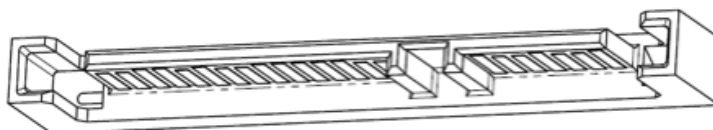


Figure 3 SATA Express Connector

Connector specifications may be found at www.sffcommittee.org.



Register Interfaces

It is taken for granted that a user can swap one storage device for another from a different manufacturer and see no difference in functionality. Of course, this assumes that both drives have the same interface, whether it is SATA, SAS or PCIe. A less obvious requirement is that both devices communicate with the host operating system in the same way. A standard register interface enables this level of interchangeability. This section describes the currently available standard register interfaces applicable to PCIe drives.

AHCI

The Advanced Host Controller Interface (AHCI) specification describes the register-level interface for a host controller for Serial ATA. AHCI was designed assuming the latencies of HDDs, and thus is not optimized for the orders of magnitude quicker response times of SSDs. Key features include:

- One command queue, 32 commands
- Single interrupt with no steering
- Must have synchronization lock before issuing a command

On the plus side, AHCI is supported by every operating system that utilizes SATA without the need for a separate driver, thus providing compatibility with the very large installed base of SATA software. Despite the software compatibility, it is generally expected that only early implementations of PCIe SSDs will utilize AHCI, and subsequent products will quickly migrate to a newer register interface that will optimize PCIe SSD performance.

For more details on AHCI, go to www.intel.com/content/www/us/en/io/serial-ata/ahci.html.

NVM Express (NVMe)

Legacy storage interfaces are optimized for rotating media technology. The increasing popularity of NVM technologies such as NAND Flash created demand for a host controller interface optimized for NVM. NVM Express is a scalable host controller interface architected to meet the needs of enterprise and client systems using PCIe solid state drives.

NVMe is efficient, scalable and high performance. The growth of multi-core processor systems means more multi-threading, and therefore more operations in flight. NVMe meets the needs of multi-core architectures by scaling up to 64K queues with 64K commands per IO queue. Enterprise capabilities like end-to-end data protection, virtualization and enhanced error reporting are included.



Table I summarizes the important features of NVMe:

FEATURE	BENEFIT
Streamlined command set	Enables less complex implementations (Total of 13 Required Commands 10 Admin, 3 IO)
Command issue/completion path requires no uncacheable / MMIO register reads	Eliminates performance bottleneck
Command issue path requires a maximum of one MMIO register write	Optimized for performance
More queues	Better fit for multi-core architectures (64K queues)
Larger queues	Scales to enterprise workloads (64K commands per queue)
Per queue priorities	Supports differentiated Quality Of Service
Efficient small random IO	Enables higher IOPS (Example: All information to complete a 4KB read request is included in the 64B command itself)
Support for MSI/MSI-X	MSI-X enables efficient targeting of individual interrupts to a specific core in a multi-core architecture
Interrupt aggregation	Increases efficiency by reducing interrupts/command to less than one
End to end data protection	Enables enterprise level RAS (compatible with T10 DIF and DIX standards)
Enhanced error reporting	Enables enterprise level RAS (PCIe Advanced Error Reporting capable)
Efficient support for SR-IOV	Hypervisor performance bottleneck eliminated
Multi-Path IO support	Enables dual port SSD usage with redundant host connections or multi-host connections
Autonomous power transitions	SSD initiated power savings without complex OS SW requirements
Lower number of clocks per IO	Saves power as the CPU & SSD can return to a sleep state sooner
Multiple namespaces	Enables multi-host access protection mechanism
Doorbell registers	Enables scalable number of Submission/Completion Queues

Table I Key features of NVM

Queues

Queue pairs are an integral mechanism in NVMe. Queues reside in host memory. The queue pair consists of a Submission Queue and a Completion Queue. The host inserts commands into the Submission Queue. The SSD controller inserts command completions into the Completion Queue. Multiple Submission Queues can share a single Completion Queue.

Host management and control of an SSD uses a dedicated queue pair and Admin Command Set. The Admin Submission Queue is used to abort commands, create and delete IO Submission and Completion Queues and other management tasks.



NVMe enables scaling the number of queues to match system configuration and workload requirements. In a multi-core system, a queue pair per core can be assigned to balance the workload across processors for heavy workloads. For lighter workloads, multiple Submission Queues may share the same Completion Queue. The architectural flexibility of NVMe allows the system architect to tune the system to efficiently support a wide array of workloads.

NVMe uses circular queues with a fixed entry size due to the implementation efficiency of this approach. Queues are addressed via Head and Tail pointers. The host places commands of fixed size (64 bytes) in a Submission Queue. The host then writes the associated Submission Queue Tail doorbell register of the SSD. The SSD reads command entries in the Submission Queue in the order the commands were placed in the Submission Queue. Out of order execution of the commands can occur within the SSD based on vendor-unique optimizations.

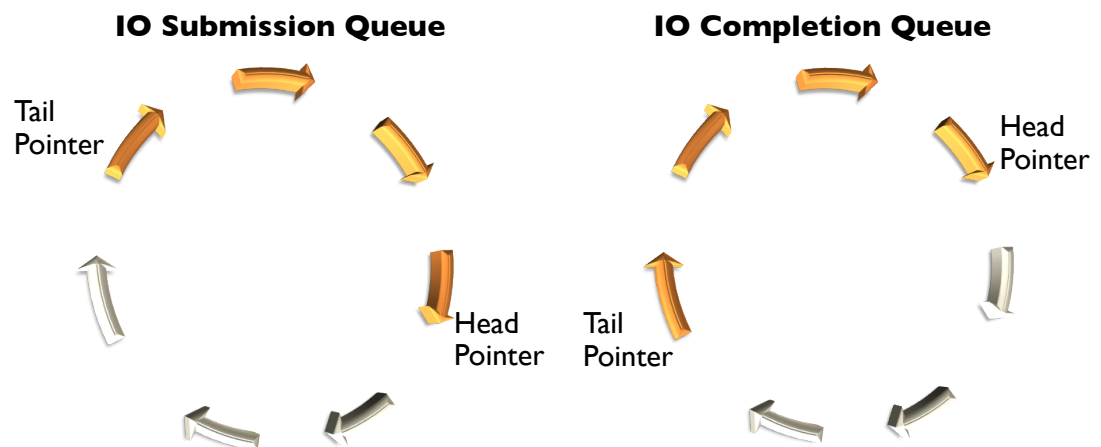


Figure 4. NVMe Express Queues

The host specifies the location to be used for data transfers via Physical Region Page (PRP) pointers within the command. If the locations of data for transfer is larger than can be described by the two PRP pointers within a command, then a pointer to a PRP list is used instead of a direct pointer to the data.

The SSD writes a Completion Queue with status information after command execution is done. The host identifies the status entry via a Submission Queue identifier and a command identifier it used in the command the host placed in the Submission Queue. The host writes the Completion Queue Head pointer after it has processed Completion Queue command status entries.

Drivers for Windows and Linux currently exist for NVMe. Solaris, VMWare, and UEFI drivers are in development at time of publication.

For more additional information, go to www.nvmexpress.org.

SCSI Over PCIe (SOP) / SCSI Over PCIe Queuing Interface (PQI)

The T10 standards committee has defined SOP as the way the SCSI commands and responses are packaged into Information Units (IUs) or messages. Another T10 standard, PQI, is used to efficiently transfer these messages with minimal latency. These are the protocol components of the SCSI Express interface.



Table 2 lists important features of the SOP/PQI interface standard.

FEATURE	BENEFIT
SCSI Command set	Easily integrates with existing storage and management protocol stacks and ecosystems. Can be streamlined for high performance operation
Command issue/completion path requires no uncachable / MMIO register reads	Eliminates performance bottleneck
Command issue path requires a maximum of one MMIO register write	Optimized for performance
More queues	Better fit for multi-core architectures (64K queues)
Larger queues	Scales to enterprise workloads (64K queue elements per queue)
Per queue priorities	Supports differentiated Quality Of Service
Efficient small random IO	Enables higher IOPS (Example: All information to complete a 4KB read request is included in the 48B IU)
Support for MSI/MSI-X	MSI-X enables efficient targeting of individual interrupts to a specific core in a multi-core architecture
Interrupt aggregation	Increases efficiency by reducing interrupts/command to less than one
End to end data protection	Enables enterprise level RAS per the T10 DIF and DIX standards
Enhanced error reporting	Robust SCSI error reporting and Advance PCIe error reporting
Efficient support for SR-IOV	Hypervisor performance bottleneck eliminated
Multi-Path IO support	Enables dual port SSD usage with redundant host connections or multi-host connections
Autonomous power transitions	SSD initiated power savings without complex OS SW requirements
Lower number of clocks per IO	Saves power as the CPU & SSD can return to a sleep state sooner
Multiple LUN capable	Standard SCSI feature enables multi-host access protection mechanism
Doorbell registers	Enables scalable number of Input/Output Queues

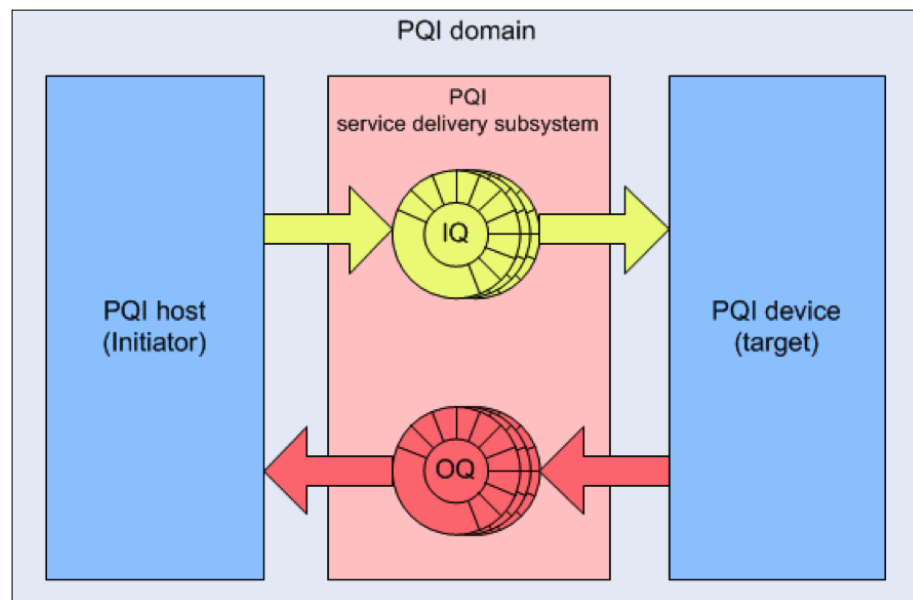
Table 2 SOP/PQI Key Features

Information units (IUs) are the formatted messages that are exchanged between an initiator and a target device. There are several types of IUs. They fall into four general categories:

1. SCSI Request/Response IUs
2. General Management Request/Response IUs
3. Bridge Management Request/Response IUs
4. Administrative Request/Response IUs

IUs associated with data transfers also contain a scatter gather list (SGL) or a pointer to a SGL located elsewhere in memory.

IUs are passed between SCSI Initiators and targets using circular queue pairs, as seen in Figure 5. Consistent with normal SCSI conventions, directional orientation is described from a target device's point of view. The Input Queue contains IUs being passed to the target device, while Output Queue contains IUs being passed to the initiator. These are analogous to the Submission and Completion queues in NVMe, and operate in a similar manner. PQI uses Producer Index (PI) and Consumer Index (CI) pointers similar to the Head and Tail queue pointers in NVMe.



Source: T10 Document PQI-r06i

Figure 5 PQI Domain

Queues are composed of equal queue elements typically 16-64 bytes each. IUs start at the beginning of queue elements. There is no more than one IU in a queue element. Any unused bytes after an IU in a queue element are ignored up until the end of the queue element. IUs may span more than a single queue element if the IU is larger than a queue element. Figure 6 illustrates this relationship.

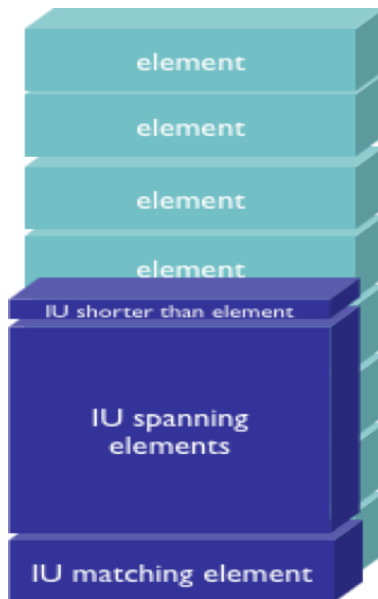


Figure 6 Queue Elements

There are two types of queues in PQI. An Administrator Queue pair is used to perform management of the queuing interface. There is only one Administrator Queue pair per device. A typical Administrator Queue pair function would be to create Operational Queues used to perform data transfer and task management functions. There can be any number of input and output Operational Queues up to the maximum number supported by a particular design, and they do not need to be in pairs. These Operational Queues can be assigned to specific cores

Once a driver has set up its operational queues typical operation is as follows: The IU describing the data transfer is placed into the input queue by the initiator and a doorbell is rung letting the device know there is an entry. The target device then reads the IU from the input queue and performs the requested data transfer via DMA directly to/from the system memory. Upon completion, a response IU is placed in the output queue by the target device and an interrupt generated. For efficiency, multiple queue entries can be written or read at the same time as well as coalescing of interrupts to reduce CPU overhead.

Drivers for Linux and Windows are in development at the time of publication.

For more additional information, go to www.t10.org or www.scsita.org.

PCIe SSD Device Standards

Various groups have defined the components of PCIe-based storage – form factors, connectors, register interfaces – but engineers require yet more information to design an SSD with a PCIe interface. To meet this need, two device-level standards have been created – SATA Express and SCSI Express.

SATA Express

SSDs and Solid State Hybrid Drives (HDDs with an on-drive Flash cache) are already pushing the SATA interface bandwidth limitation of 600 MB/s; a higher-speed interface is needed for these Flash-based devices. PCIe offers multi-lane scalability, enabling speeds greater than SATA. Rather than define a next-generation interface speed for SATA, SATA-IO opted for PCIe.



Source: SATA-IO

Figure 7 SATA Express Logo

SATA Express defines electrical and mechanical requirements for client storage to employ a PCIe interface using standard disk drive form factors. SATA Express standardizes PCIe as a client storage device interface and defines device and host connectors to support up to 1GB/s and 2GB/s interface speeds via one or two lanes of PCIe.

SATA-IO brought PCIe to client storage by defining SATA Express host and drive connectors supporting up to two lanes of PCIe or one SATA port. SATA Express leverages existing connector, form factor and interface standards to facilitate backward compatibility as well as cost-effective transition of client storage from SATA to PCIe. A SATA Express host will function with either a legacy SATA drive or a PCIe device.

Applications not requiring the speed of PCIe will continue to be served by existing SATA technology.

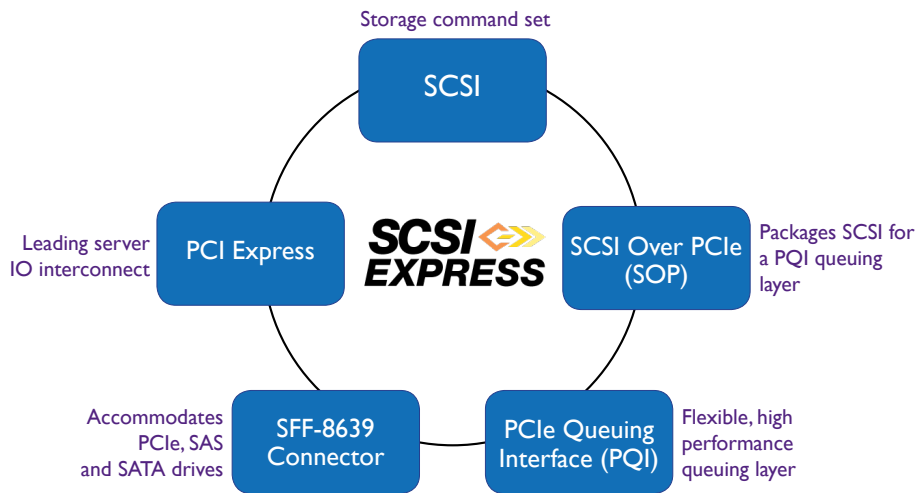
Although not defined by the SATA Express specification, there are two choices for register interface / command set – AHCI and NVMe. Even though AHCI provides some level of SATA software compatibility, it is expected that most client PCIe SSDs will eventually use NVMe, to optimize PCIe SSD performance.

For more information, refer to www.sata-io.org/technology/sataexpress.asp.

SCSI Express

Logical SCSI protocols are at the core of nearly all storage interfaces used by computer systems today, including USB, Memory stick, Firewire, Infiniband, iSCSI, SAS, Fibre Channel, and many others. SCSI Express combines this proven SCSI protocol with PCIe to create an industry standard path to PCIe storage. It delivers enterprise class features for PCIe-based storage devices in a standardized ecosystem. It is able to take advantage of the lower latency of PCIe to improve application performance through an efficient combination of message formats and queuing methods. It does this while integrating into a unified management and programming interface that is the core of enterprise storage solutions.

SCSI Express consists of five primary components, as shown in Figure 8. The first is the SCSI command set. The second is SCSI Over PCIe (SOP). SOP defines the way that the SCSI commands and responses are packaged into Information Units (IUs) or messages. The third is the PCIe Queuing Interface (PQI). In order to transfer these IUs across the PCIe bus, SCSI Express uses PQI to efficiently transfer these messages with minimal latency. The fourth component is the SFF-8639 multifunction connector; which accommodates multiple interfaces. We refer to a drive bay that uses this connector as an Express Bay. The fifth component is the x4 PCIe interface that is wired to the SFF-8639 connector.



Source: SCSI Trade Association

Figure 8 SCSI Express Components

For more additional information, go to www.t10.org or www.scsita.org.

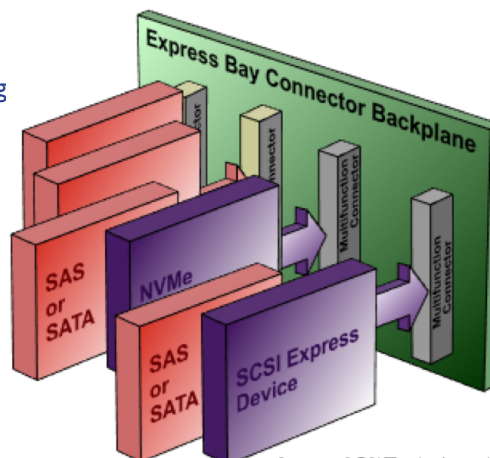
NVMe PCIe SSD

It is also possible to create an enterprise class PCIe SSD with NVMe as the register interface, utilizing the enterprise level features of NVMe. Such an SSD would typically use the SFF-8639 connector and would provide up to four lanes of PCIe as the host interface.

Express Bay

Express Bay is a serviceable drive bay, typically found in a server or storage enclosure that utilizes the SFF-8639 connector and is provisioned for multi protocol use and provides additional power and cooling. Typically an Express Bay will be provisioned to support SAS, SATA and PCIe devices. These devices are serviceable, utilizing drive carriers compatible with the OEMs enclosures. In order to get the maximum performance in the least amount of space, these bays will provide additional power (up to 25W per device) and the additional cooling required by the higher power dissipation. Use of the additional power is negotiated between the system and device. Both the system and the device need to support the appropriate power management protocols.

For more additional information, go to www.scsita.org



Source: SCSI Trade Association

Figure 9 Express Bay



Standards Summary

The following chart summarizes the elements of PCIe SSD standards.

ITEM	GROUP → STANDARD	CLIENT PCIe SSD	ENTERPRISE PCIe SSD
PHYSICAL INTERFACE	PCI-SIG → PCI Express SATA-IO → SATA Express SCSI Trade Assn → SCSI Express	up to 2 LANES Drive up to 4 LANES M. Card	Up to 4 LANES
COMMAND SET	Intel → AHCI NVM Express Group → NVMe T10 → SOP/PQI	AHCI or NVMe	SOP / PQI Or SOP / NVMe Or NVMe
HOST CONNECTOR	SATA-IO → SATA Express SSD FF WG → multifunction connector Small FF Committee → SFF-8639	SATA Express Host Or M.2 Host Or SFF-8639	SFF-8639
DEVICE CONNECTOR	SATA-IO → SATA Express SSD FF WG → multifunction connector Small FF Committee → SFF-8639	SATA Express Drive Or M.2 Card Edge	SFF-8639

Table 3 Summary of PCIe SSD Standards

3.0 Programming Models for Non-Volatile Memory

With few exceptions, host software does not interact directly with storage devices. Operating systems (OSes) provide file systems and related services that allow multiple applications to simultaneously utilize storage independently from the storage hardware and protocols. The majority of software use a basic storage programming model designed to run on any disk (HDD, drive factor SSD, or PCIe SSD) with no logic specific to disk hardware. Some applications extend the basic programming model to utilize storage hardware features, primarily to enhance performance or to assure data is written to persistent media. This section presents information about these programming models, and how they are being enhanced for SSDs.

The other area of software enhancements for NVM is support for persistent memory (PM) - NVM hardware designed to be treated by software similarly to system memory. Persistent memory may be deployed as memory or as PCIe cards. New programming models which isolate software from hardware differences are also described.

Basic Storage Programming Model

As a general rule, applications can operate on data of any length, starting at any address in system memory. But the data must be stored in fixed-sized blocks on disks. Many applications use a basic storage programming model where file systems manage the differences between memory and disk; file systems typically do this using a memory cache. Where possible, application read and write operations are directed to the cache rather than accessing the disk.



The programming model used for “basic storage” programming is:

- Applications create or modify data using CPU registers
- Data is temporarily stored in volatile memory
- To assure the data is durable across system and application restarts, applications write the memory contents to files
- The file system utilizes a cache (also stored in volatile memory) to improve access to non-volatile disks

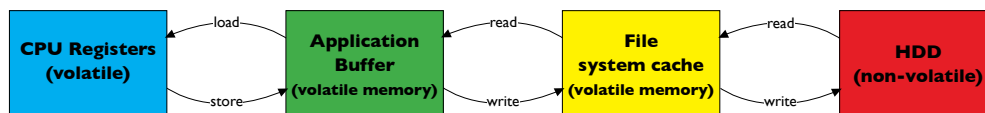


Figure 10 Basic Storage Programming Model

Advanced Storage Programming Model

Some applications use an advanced storage programming model that bypasses the file system cache for higher performance (by implementing a cache in the application) or to be certain the application is informed when write operations to disks fail.



Figure 11 Advanced Storage Programming Model

NVM Extensions

Today's SSDs also require block-oriented IO. The same programming models are used for SSDs with a few SSD extensions. Some examples of extensions requested by application developers:

- **Alignment** - Until recently, software could optimally read and write to disks starting at any block address. As a result, partition management software allowed users to create partitions starting at any block address. Some newer disks (including most SSDs) internally use 4096 byte granules, but present 512 byte blocks to the software. This allows software to start read or write operations at any block address, but optimal performance or endurance is achieved when IOs start on eight block granules (addresses evenly divisible by eight). Some partition manager software allows users to override default settings to set the optimal partition alignment, but users are not always aware they need to make this adjustment. Failure to adjust the partition alignment may result in significant performance degradation for SSD write operations (users typically see a 10x performance increase in SSD writes when partitions are manually aligned compared to using the default behavior of the Linux fdisk partition manager application; see <http://www.ibm.com/developerworks/library/l-4kb-sector-disks/>). If operating systems or file systems provide a way to determine optimal disk alignment granules, partition management software can assure optimal alignment without user intervention.

- **Atomic write** operations guarantee that either “all specified blocks are written” or “no blocks are written” – this behavior allows applications to assure consistency in on-disk data with improved performance (see <http://www.fusionio.com/blog/atomic-writes-accelerate-mysql-performance/>). Different SSDs support different capabilities related to atomic writes. At this time, some database software allows a user to configure the software to utilize atomic write capabilities of SSDs. But if the software configuration does not correctly match the SSD, performance may be degraded or reliability may be compromised. Enhancements to operating systems allowing applications to determine atomic write features will avoid these user misconfiguration issues.

Persistent Memory Programming Models

Two programming models are evolving for persistent memory (PM). One approach is to provide a software layer that makes PM appear to be an SSD. This provides the benefits of higher-performance NAND to existing applications created for traditional HDDs or SSDs. But emulating SSDs does not allow software to fully exploit persistent memory advantages.

As mentioned above, the applications using disk storage programming models create/modify data in CPU registers, move the data to volatile memory, then move the data to disks for persistence across restarts. In the other approach, PM introduces the option of skipping step 3 if the memory in step 2 is persistent. This has the benefits of avoiding the overhead of moving data between volatile memory and persistent storage, avoiding the overhead introduced when file systems (or applications) adapt application-oriented data sizes to disk block sizes.

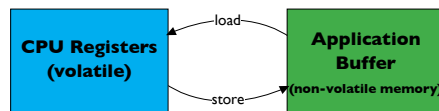


Figure 12 Persistent Memory Programming Model

Applications may need to be rewritten to use this new programming model. Volatile memory is not tied to specific applications; applications allocate memory by size and the run-time environment finds a large-enough range of memory and maps it to the application’s virtual memory. When the application restarts, the physical address of the allocated memory may be different. In order to use persistent memory, the application needs to access the same data (the same physical address range). This implies the need to name PM address ranges. Furthermore, applications should not be allowed to access PM ranges assigned to other applications. So new programming models are being defined to allow operating systems to control access to PM, but bypass extraneous copy-to-disk overhead.

The SNIA NVM Programming Technical Working Group is defining NVM programming models to address these NVM extensions for SSDs and Persistent Memory. Storage programming models are similar, but not identical across operating systems. The programming models being defined provide an interoperable, high-level approach to implementing NVM extensions, but allow operating system developers to implement support in a way that is consistent with related OS functionality. Similarly, application developers should be able to use a common, interoperable design to use these NVM extensions, even if the supporting code varies across operating systems. This common, interoperable design should help accelerate application support for NVM extensions.



4.0 PCIe SSD Market

It is a well known fact that the number of applications benefiting from SSDs is growing rapidly. The advent of PCIe as yet another interface choice complicates this picture, but also provides opportunities. This section discusses the client & enterprise (or data center) SSD markets, touching on interfaces and other factors that differ between the two.

Client

The client SSD market is moving from an “either/or” model in which a PC uses either an SSD or an HDD but rarely a mix of the two, to a model in which an SSD and HDD are paired using software to move hot data from the HDD into and out of the SSD or in which a HDD may contain some cache Flash memory.

For these dual storage applications, the SSD has extraordinarily low write loads, because the majority of its capacity is used for files that are seldom changed. This means that the SSD controller need not be optimized for a high write load since this use case is never likely to occur.

In the caching case, the write load becomes significantly higher as data is being moved from the HDD to the Flash memory in proportion to the size of the Flash memory – a small storage capacity of Flash memory will require more writes than a large Flash cache.

Client SSDs also have far less stringent requirements for data integrity, and end-to-end data protection and very low bit error rates are not as important as they are for enterprise SSDs. In addition, the client user is typically insensitive to pauses in SSD operation; the same cannot always be said for enterprise environments.


Enterprise

Enterprise SSDs undergo significantly more stress than client SSDs yet they must perform better than their client counterpart. These SSDs are expected to operate continuously under high write workloads while providing consistent latency, low bit error rates and high performance. End-to-end data protection is the norm.

Since many enterprise applications do not utilize SSDs for a content library, but rather as performance enhancing cache for near-line storage or DAS, the usage model tends to put them under a very high write workload; about 50% writes and 50% reads. This creates significant wear, and opportunities for interference between internal housekeeping algorithms and externally commanded writes, similar to the caching case mentioned for the client SSDs above.

Cost Motivators

For high-performance applications, PCIe is a better candidate for SSDs than conventional HDD interfaces like SATA and SAS, but PCIe offers little or no speed advantage over HDD interfaces in cost-sensitive applications for reasons that will be explained shortly.



There are different ways to design a PCIe SSD depending on the device's performance and cost goals. The highest-speed SSDs are designed with a high degree of parallelism to overcome difficulties stemming from NAND Flash's slow write process and "erase-before-write" requirement. Writes are further slowed by the fact that NAND is written in 4-8k byte pages but erased in blocks containing thousands of these pages. To achieve the highest performance, SSDs place several chips in parallel, allowing various housekeeping tasks performed in some chips from interfering with access in other chips.

Since these high-speed SSDs use numerous chips, their cost naturally has a bottom limit. A good rule of thumb is to assume that the minimum price for a NAND chip/channel is \$4.00. NAND manufacturers do their best to render chips obsolete after they have gone below this price point. This means that the cheapest PCIe SSD must be \$4 X the number of channels. Client SSDs typically use between four and sixteen channels to achieve SATA performance, and most enterprise PCIe SSDs use 20-50 channels. This would put an absolute lower price limit of around \$20 on a client PCIe SSD and \$100 for an enterprise PCIe SSD.

Keep in mind that a low-end client PCIe SSD would perform no better than its SATA counterpart. That is, the storage architecture cannot support the data rates that PCIe is capable of. As a consequence there is no speed advantage of using the PCIe interface, rather than a SATA interface.

If a designer attempted to design an SSD with a lower cost structure then fewer channels would need to be used, negatively impacting performance. It is more reasonable to assume that PCIe SSDs will be used in performance applications rather than in price-sensitive client caching applications.

Market Motivators

The following is a brief outline of key markets and factors that will drive increasing PCIe SSD adoption.

Application Acceleration

PCIe SSDs are largely used today to improve the speed of IO in systems that do not count on shared storage. In this DAS-based approach, the administrator stores the time-sensitive data or programs onto the SSD, making sure to purchase a large enough PCIe SSD to contain the entire data set and/or program.

This is not a complex affair as long as the sizes of the data set and code are static, and the server is dedicated to a dedicated task or set of tasks.

Server Caching

A very rational way to use a small SSD in an application with a large data set or program is to use caching software alongside the SSD. This software moves the most critical data into the SSD as it is needed, evicting it from the SSD once it is no longer as necessary to the task at hand.

Caching software is growing in popularity. There are already over ten products shipping from as many companies, each with certain strengths over others. All of them allow the user to make the best use out of a small, inexpensive SSD with data that is too large to be completely contained within the SSD. This generally puts additional overhead on the host microprocessor and thus sacrifices some of the possible performance gain for lower system costs.



User Benefits

With either of these approaches – putting the entire critical data set in Flash, or using caching algorithms to manage a smaller SSD to appear to be large – users are able to accelerate IO to such a degree that they often can move past a difficult situation that was caused by some other system IO bottleneck. The majority of enterprise SSD “early adopters” found themselves using SSDs for exactly this reason.

Virtualization

Virtualization is growing in popularity and has served to improve the cost structure of most IT departments. Virtualization does this by allowing all tasks to be performed on any server in the cluster and sharing storage in a storage array depending upon changing user needs. Tasks move into and out of the servers in the cluster according to a priority scheme and are never tied down to a single machine. As a consequence of virtualization, system storage utilization has improved significantly over the last few years.

On the other hand, virtualization has created new problems as well. Fortunately, SSDs help to solve these problems allowing users to get the most out of their virtualized system.

The “Virtualization Blender”

The most common problem in virtualized systems is the nature of the IO stream presented to the storage array.

In an un-virtualized system, in which each server uses its own un-shared storage and performs a single task, the IO stream from each server to its own storage follows relatively predictable patterns, and these patterns could be anticipated by the storage hardware to produce data and absorb writes in an optimized way. If that didn't achieve the user's performance goals, software often could be hand-tuned to improve overall performance.


In a virtualized system all servers' IO from all applications are mixed together randomly, forcing the IO stream to become a muddled chaos of random events making it difficult to organize writing and reading data on the disk drives. This chaos makes disk accesses very slow, since accesses are occurring all over the disk rather than in a more thoughtful pattern.

SSDs do a good job of handling small random IO, so the difficulty of keeping up with the Virtualization Blender becomes far easier through the use of SSDs in front of HDDs to help organize the data before it is transferred to HDDs.

Boot Storm Issues

Another phenomenon created by desktop virtualization has become known as “The Boot Storm.” In an office with tens or hundreds of virtualized PCs that all count on the SAN for their storage there is a problem when all the PCs are turned on simultaneously. Although this may happen with the daily arrival of the office staff, it can happen much more often when it is triggered by a power outage, causing delays and loss of data.

Booting a PC involves a large number of disk accesses, and when a centralized SAN is servicing all of these accesses, the high workload of random IO demand can cause performance to drop to a very low level.



There are two common approaches to solving the boot storm issue. The first is to add some localized storage to each PC, so that the PC boots from the local storage. This is problematic when the software needs to be upgraded, since upgrades must be performed on each PC individually, and also adds cost and inefficiency to the overall enterprise. The second approach is to increase the bandwidth of the SAN, usually by using SSDs with the boot data stored on them, so that the boot storm is serviced rapidly enough to avoid any noticeable delay.

Shared Storage versus DAS

Since network latency is often higher than the latency of an SSD, the fastest place to put any SSD would be within the server as Direct-Attached Storage (DAS). This works against virtualization, which requires all storage to be shared. While this approach provides a lot of speed, it doesn't fully tap all that the SSD can offer: A virtual machine cannot move from server to server unless shared storage is used for all stored programs and data.

Fortunately, many companies are working on solutions to this problem, including:

- Making DAS appear to be memory rather than storage
- Developing caching systems whose policies prevent data in the SAN from becoming stale
- Implementing systems that use a "back door" network (or out of band communication link) to manage coherency between DAS in each server in the cluster.

Total Cost of Ownership

One of the best ways to evaluate the use of SSDs is to look at the total cost of ownership (TCO) of an HDD-based system compared to that of its SSD-based counterpart. In many cases the SSD may be a more expensive alternative at purchase, but will reap rewards in power, cooling, floor space, or reliability that cause it to be the lower-cost alternative over the lifetime of the system.

SNIA SSSI has created an Excel spreadsheet model of enterprise storage system costs in which a user can input various parameters like SSD and HDD performance specifications, cost per kWh of electricity, IOPS required by the system, etc. to determine whether SSDs or HDDs are a better choice for a particular user's system.

The SNIA TCO Model can be downloaded free of charge at: <http://www.snia.org/forums/ssi/programs/TCOcalc>.

OpEx versus CapEx

One difficulty in the adoption of a TCO model in many organizations is that the capital costs come out of one organization's budget while operating costs (like power, floor space and cooling) are often paid for through a completely different cost center. In this type of organization capital spending decisions have no built-in consideration for ongoing costs, and the solution that had the lowest initial cost will be selected over one that can best benefit the company over the longer term.

The only way past this problem is to somehow make operating costs a part of the purchase decision, perhaps by pushing the decision to a higher level of management, or by giving the solution purchasers some responsibility for operating costs. In other words, this is an organizational issue, not a technical one.



Legacy Software

Prior to the introduction of SSDs in the data processing environment, software was speed-optimized for HDD based storage but did not anticipate the use of high-speed Flash-based storage. As a result, many IO routines take longer to execute than the access time of the SSDs with which they interact. This has become a key bottleneck in many SSD implementations and it is starting to receive attention from leading SSD suppliers.

Getting Away from Disk-Based Software Stacks

Figure 13 illustrates a standard computer IO stack and some of the ways SSD software has been written to bypass some of the slower levels of the software.

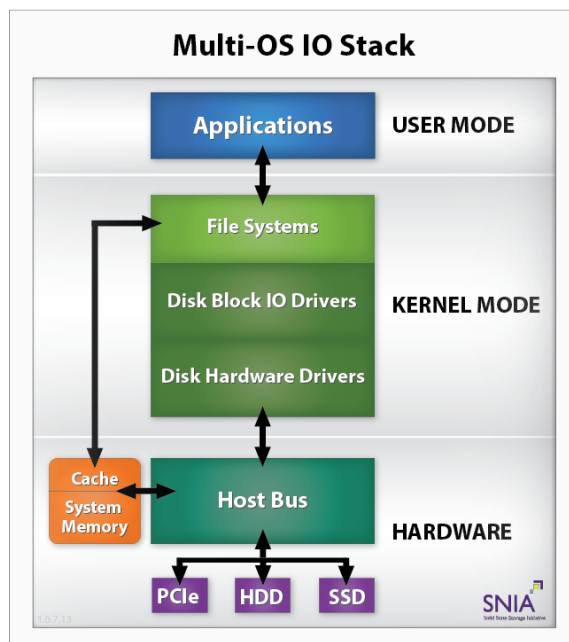



Figure 13 Diagram of the IO Hardware/Software Stack

This is a relatively complex topic that was thoroughly explained in a SNIA presentation called "How IOs Traverse the IO Software/Hardware Stack" which can be accessed through the SNIA webcast channel at: http://www.snia.org/news_events/multimedia#webcasts.

Customer Concerns

The use of PCIe interfaces on SSDs is still very new, and this has resulted in implementations that are unique to each vendor. While the first PCIe SSDs required the use of system resources to manage the SSD's Flash memory, many later models are completely self-contained, allowing the system to treat them as it would any hardware RAID connected through the PCIe interface.

There are benefits to both approaches that will not be detailed here. Be warned, however, that PCIe SSDs are not yet interchangeable as are SAS or SATA SSDs, leading to sole-sourced drives and the implications to availability that stem from that situation.



SNIA is working with various standards bodies (SATA-IO, STA, PCIe SIG) to standardize the way in which the PCIe interface is used to access storage. These efforts are making good headway and should result in important standards being issued in the near future.

One important drawback in using PCIe SSDs compared to SSDs with the SAS interface is the fact that PCIe presents a single point of failure through its single external interface, where the SAS interface provides two ports in case one fails.

This is a special issue that will need to be addressed by PCIe SSD users in high availability applications. The likely solution is to treat this in the same way as system memory is treated, since it, too, has a connection as a single point of failure. When the server/memory/PCIe SSD complex is treated as a unit, then the HA approach for this unit can be determined without special considerations for the PCIe SSD by itself. Another approach is redundancy, such as mirroring, allowing one unit to take over if its mirror experiences trouble.

Market Outlook

Implications of the PCI-SIG Roadmap

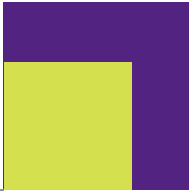
The PCIe interface is governed and supported by the PCI-SIG (Special Interest Group). The PCIe 3.0 standard has been optimized for high-speed solid state drives (SSDs) used in enterprise and client applications. The PCIe 3.0 technology allows multiple communication lanes, each lane having a throughput of 8 giga-transitions per second (GT/s). The net data rate of four-bidirectional lanes is 32 giga-bits per second (Gb/s). Four PCIe 3.0 lanes provide a 4 GB/s bidirectional data rate, which enables fast performance from SSD storage devices.

PCIe 4.0 will double the bidirectional communication lane speeds to 16 GT/s with restrictions on direct connection run lengths. Four lanes of PCIe 4.0 will provide up to 8 GB/s bidirectional data rate and will support even faster devices including storage products. The PCIe 4.0 specification should be completed in 2014 or 2015 with initial products introduced using this specification in the 2017 time frame.

PCIe 3.0 has enabled SATA Express and SCSI Express, as well as PCIe for mobile devices.

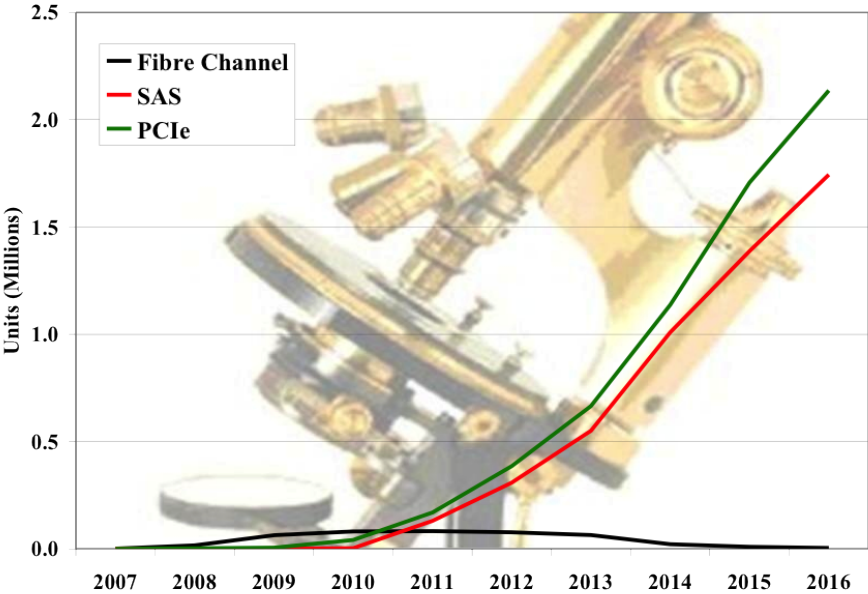
PCI-SIG announced a low power capability to support high speed interfaces for battery powered mobile platforms. This new mobile PCIe interface is the result of a collaboration between MIPI and PCI-SIG. MIPI is a trade group that develops interface specifications for mobile devices.

The M-PCIe standard uses the same MIPI M-PHY as Universal Flash Storage (UFS), and is similarly targeted at mobile phones and tablets. M-PCIe scales up in performance by adding PCIe lanes. Mobile products using the M-PCIe interface can be expected within 12-18 months from the specification release.



Five Year Forecast

The forecast in Figure 14 Enterprise SSD Forecast is from the 2012 Objective Analysis report: "The Enterprise SSD: Technologies & Markets" and reflects the anticipated growth of PCIe SSDs in enterprise applications.



Source: Objective Analysis 2012

Figure 14 Enterprise SSD Forecast

This forecast shows unit shipments of PCIe, SAS, and Fibre Channel SSDs, with PCIe SSDs leading SAS, and Fibre Channel gradually fading away.

By 2016, the end of the forecast, Objective Analysis forecasts over 2 million PCIe enterprise SSDs to ship.



5.0 PCIe SSD Performance

As the latest development in NAND Flash solid state storage devices, PCIe SSDs have higher performance (more IOPS and Bandwidth) and are faster (lower response times) than SAS and SATA SSDs and HDDs. The higher performance is made possible by a combination of more advanced PCIe SSD product architectures, a shorter IO driver stack (no SCSI layer) and the emergence of faster host systems (PCIe 3.0 at 8Gb/s per lane motherboards with 8 core CPUs).

The following charts show the relative performance of PCIe SSDs compared to SAS & SATA SSDs and HDDs – IOPS and Response Times. Six PCIe SSDs, six SAS SSDs, fourteen SATA SSDs, a 15K RPM SAS HDD and a 7200 RPM SATA Solid State Hybrid HDD (with 8GB of on-drive Flash) were measured.

The general range of performance for random 4KiB writes between different classes of SSDs clearly show the higher performance of PCIe SSDs.

Figure 15 shows various classes of SSD and HDD drives at FOB (Fresh out of Box after a device Purge) settling to Steady State. A Purge is a command that puts the SSD in the state “as if no writes have occurred” by resetting the Program/Erase state of the NAND Flash.

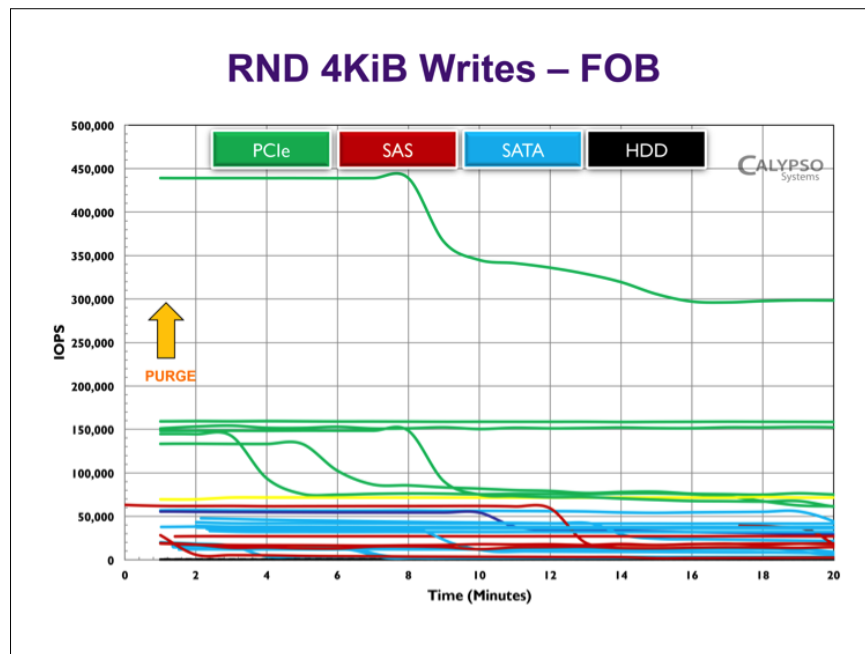


Figure 15 Random Writes at FOB

Notes:

1. The charts in this section use the same nomenclature as the Solid State Storage Performance Test Specification (PTS). All memory sizes in the PTS are base 2 (KiB, MiB or GiB) while reporting rates are base 10 (KB, MB or GB). See the PTS (www.snia.org/pts) for more details.
2. Data was collected by Calypso Systems, Inc. using the SSS PTS 1.1 on an SSSI Reference Test Platform (www.snia.org/forums/sssi/rtp).



Figure 16 shows the Steady State region after 500 minutes of random 4KiB writes.

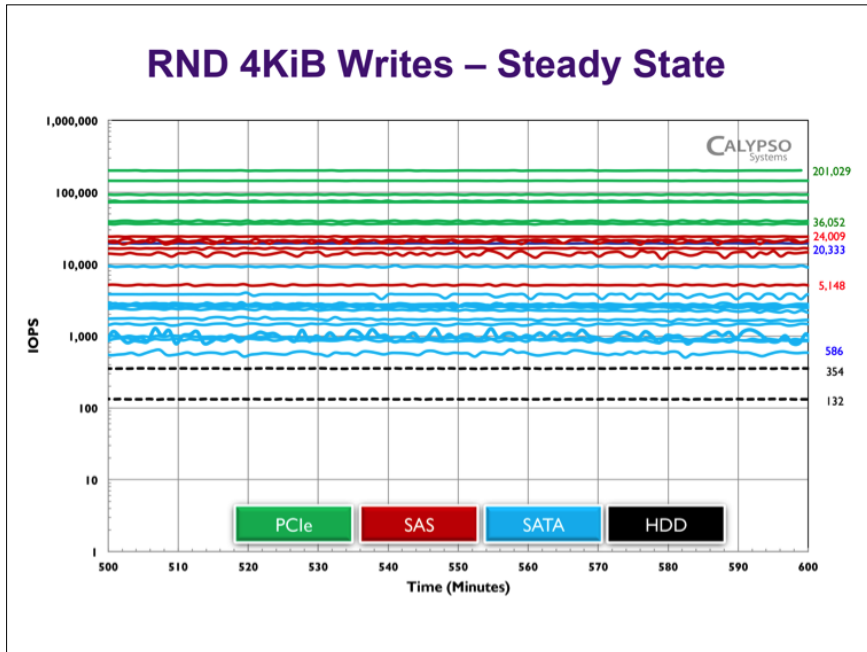


Figure 16 Random 4KiB Writes – Steady State

Figure 17 shows minimum response time classified by interface type. As expected, PCIe devices show the lowest response time.

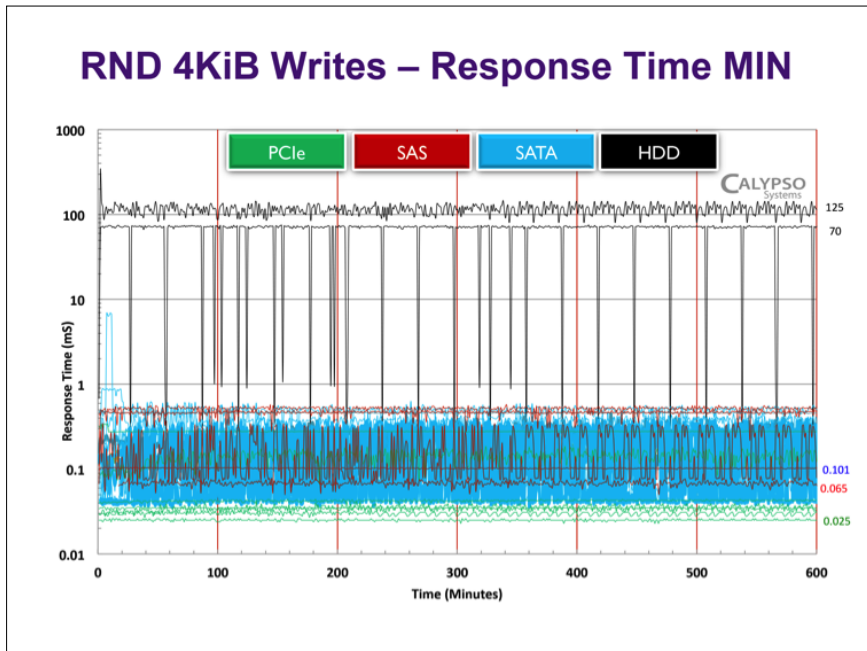


Figure 17 Random 4KiB Writes – Minimum Response Times



Figure 18 shows maximum response time classified by interface type. As expected, PCIe and SAS devices show the lower response times..

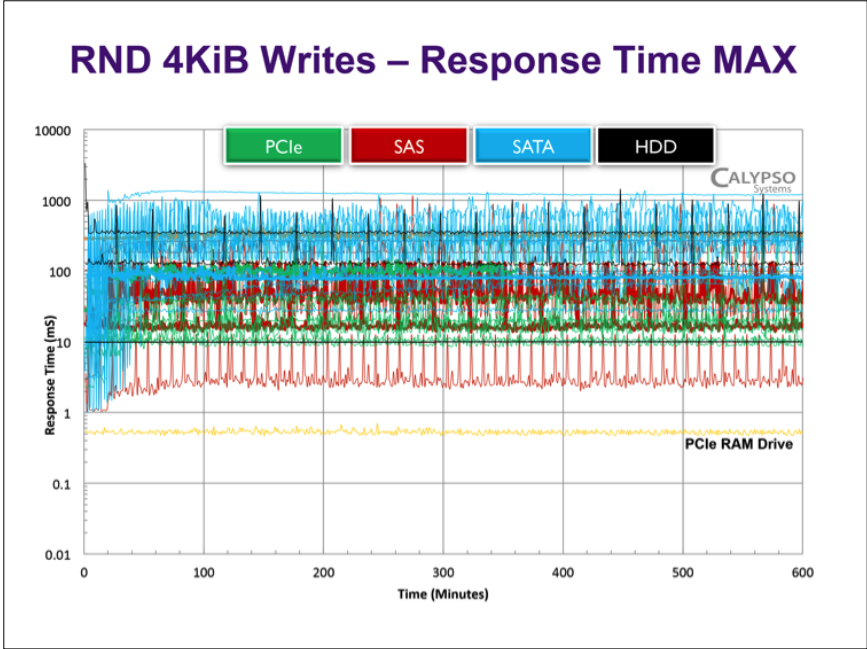


Figure 18 Maximum Response Times by Interface

Figure 19 shows the minimum response time classified by interface type, but plotted in a different way. One can clearly see that PCIe devices have the lowest times, with both SATA and SAS devices showing higher time values.

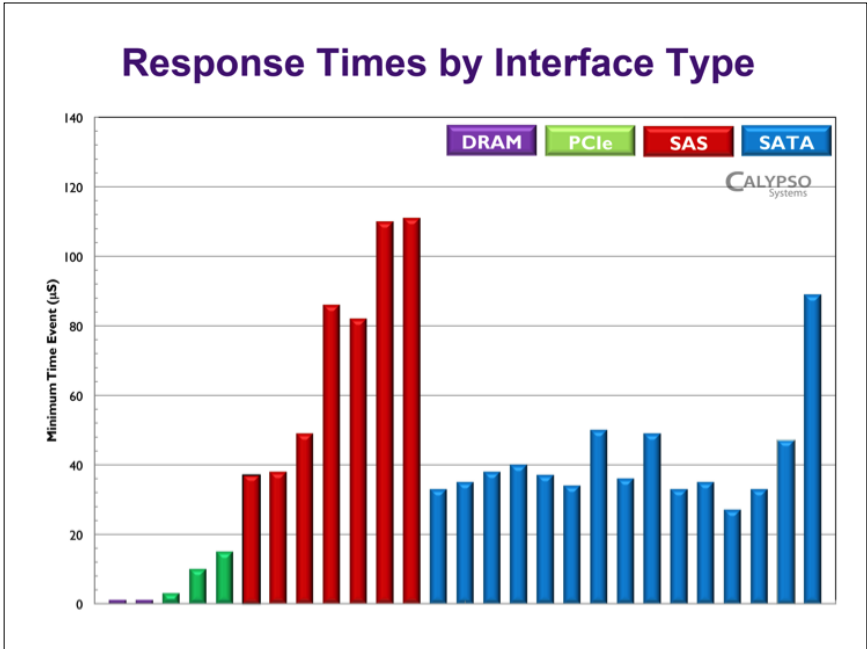


Figure 19 Response Times by Interface



Figure 20 shows a comparison of CPU utilization for two different PCIe SSD devices. One can see that some PCIe SSDs utilize the host CPU and other resources to increase performance.

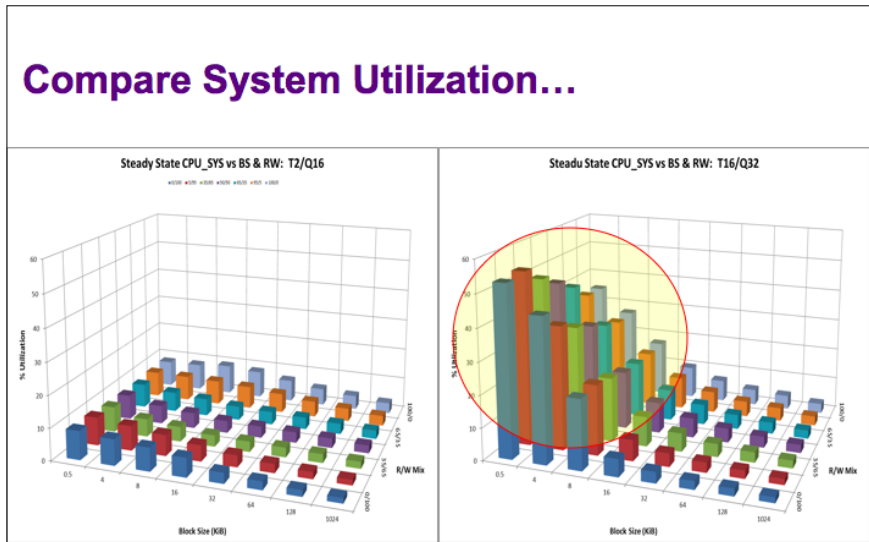


Figure 20 CPU Utilization Comparison

Figure 21 shows the effect of CPU affinity settings on performance. In this chart, the PCIe SSD card was tested with some or all CPU core(s) set for use with the device or reserved for use by the system.

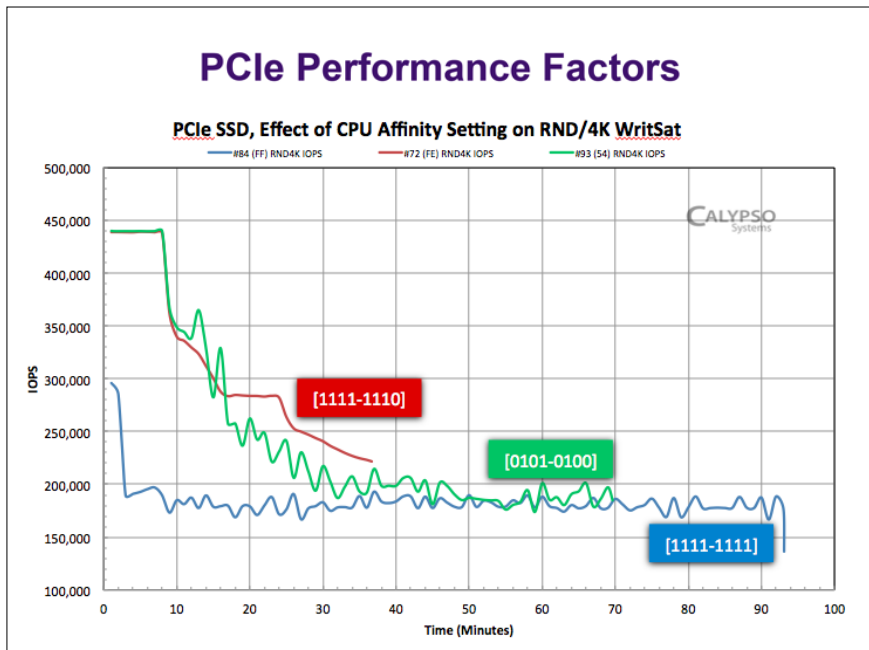


Figure 21 CPU Affinity Settings Impact on Performance

An overarching conclusion can be drawn from this series of charts. The drastic increase in IO rate and decrease in response time of PCIe SSDs allows visibility of the effects of the test environment such as thread count, queue depth, IO driver and CPU utilization on overall performance.



6.0 Contributors

Thanks to the following for their contributions to this white paper:

Eden Kim, CEO

Calypso Systems, Inc., www.calypsotesters.com

Eden is the Chair of the SSSI PCIe and TechDev committees and Chair of the SSS TWG. Eden has received past SNIA awards for Outstanding Volunteer 2010 and Industry Impact Player 2011. Calypso is the SSSI PTS Test House and the Calypso RTP/CTS was used to develop and validate the PTS 1.0 and 1.1.

Tom Coughlin, President

Coughlin Associates, www.tomcoughlin.com

Tom Coughlin is Chair of the SSSI Education Committee, prior Chair of the Marketing Committee and is a consultant. Tom writes reports on digital storage and applications, puts on the annual Storage Visions and Creative Storage Conferences and is General Chair of the annual Flash Memory Summit.

Walt Hubis, Storage Standards Architect

Fusion-io, www.fusionio.com

Walt is responsible for defining new industry standards for solid state storage solutions. Walt is active in SNIA, T10, and other storage standards and storage security organizations and has authored several key patents in RAID and other storage related technologies.

Chuck Paridon, Master Storage Performance Architect

Hewlett Packard, www.hp.com

Chuck is a 26 year veteran of computer and storage subsystem performance analysis. He is responsible for the development and delivery of HP field performance training. A member of the SNIA SSS and GSI TWGs, and the Storage Performance Council, He holds a BS in ME and an MS in Computer Science.

Doug Voigt, Distinguished Technologist

Hewlett Packard, www.hp.com

Doug Voigt is in the Chief Technologist's office of HP's Storage Division. He has 35 years of experience in HP's disk, disk array and storage management product lines. Doug has degrees in computer science and electrical engineering from Cornell University and 31 US patents.

Paul von Behren, Software Architect

Intel, www.intel.com

Paul von Behren is a Software Architect at Intel Corporation. Currently he is co-chair of the SNIA NVM Programming Technical Work Group. His background includes software for managing Fibre Channel, iSCSI, and SAS storage devices, multipath management software, and RAID systems.



Jack Hollins, Strategic Marketing Manager

LSI, www.lsi.com

Jack is a member of the SNIA SSSI PCIe SSD committee. He is involved in other industry groups including SATA-IO, T10, T13, JEDEC, SSD Form Factor Working Group, and NVM Express. Jack has a MSEE degree from U.C. Berkeley and a BSEE degree from Rice University.

Paul Wassenberg, Director of Product Marketing

Marvell Semiconductor, www.marvell.com

Paul is a member of the SNIA SSSI PCIe Committee. He chairs the SNIA SSSI and is involved in other industry groups including SATA-IO, T10, JEDEC, UFS, and NVM Express. Paul has BSEE and MBA degrees from San Jose State University.

Kristen Hopper, Product Marketing Manager

Micron Technology, Inc., www.micron.com

Kristen Hopper is a SNIA SSSI PCIe Committee member and Product Marketing Manager at Micron responsible for Client SSDs. Kristen has been involved in SATA-IO working groups and is an IEEE Senior Member. Kristen has a B.S.E.E. with Distinction from Cornell University and an M.S.E.E. from U.C. Berkeley.

Doug Rollins, Senior Applications Engineer (Enterprise SSDs)

Micron Technology, Inc., www.micron.com

Doug Rollins is the named inventor in 13 U.S. patents and has been recognized by both the Storage Networking Industry Association (SNIA) and Intel Corporation for outstanding technical achievement. Mr. Rollins earned his BA degree in mathematics.

Jim Handy, Director

Objective Analysis, www.objective-analysis.com

Jim has over 25 years of experience in semiconductors, including analysis at Gartner Dataquest plus marketing and design positions at Intel, National Semi, and Infineon. Author of "The Cache Memory Book," he is a patent holder with MBA and BSEE degrees. He is well-versed in all aspects of memory and SSDs.

Tony Roug, Owner

Rougs.com LLC, www.rougs.com

Tony Roug does technical consulting at Rougs.com LLC and is a founder of ZigStor. He has 5 years of experience in solid state storage technology and usage models as a principle solution architect at Intel and Virident. He has an MS in computer engineering from Carnegie Mellon University.

Marty Czekalski, Sr. Staff Program Manager

Seagate Technology, www.seagate.com

Marty Czekalski is a member of several SNIA TWGs. In addition, he is the President of the SCSI Trade Association and is involved with multiple industry groups, including: T10, SFF, JEDEC, and PCI-SIG.



Storage Networking Industry Association

425 Market Street, Suite 1020 • San Francisco, CA 94105 • Phone: 415-402-0006 • Fax: 415-402-0009 • www.snia.org

© Copyright August 2013 Storage Networking Industry Association. All Rights Reserved.