SDC 19
September 23-26, 2019
Santa Clara, CA

A New Windows Monolithic NVMe SSD Driver In Cloud

Michael Xing
Danyu Zhu
Liang Yang

Microsoft Corporation

Disclaimer

- Performance results presented in this talk are based on a prototype driver in a controlled lab environment.
- Driver is not in production and not intended to replace current Windows storage stack.

Agenda

- Motivations
- Storage Device in Cloud
- Existing Windows Storage Stack
- New Monolithic Cloud SSD Driver
- Performance Test Results

Motivations – Why New Driver

- The popularity of I/O intensive (high throughput / low latency) workloads in cloud leveraging locally attached storage
- Storage hardware technologies advance: NVMe, 3DXP, etc
- Heavy existing software storage stack compared to very low latency and high throughput provided by underlying hardware
- Need innovations on new storage software stack targeting on Cloud environment while preserving maximum user application compatibility
- Explore ideas for the next Windows storage stack update

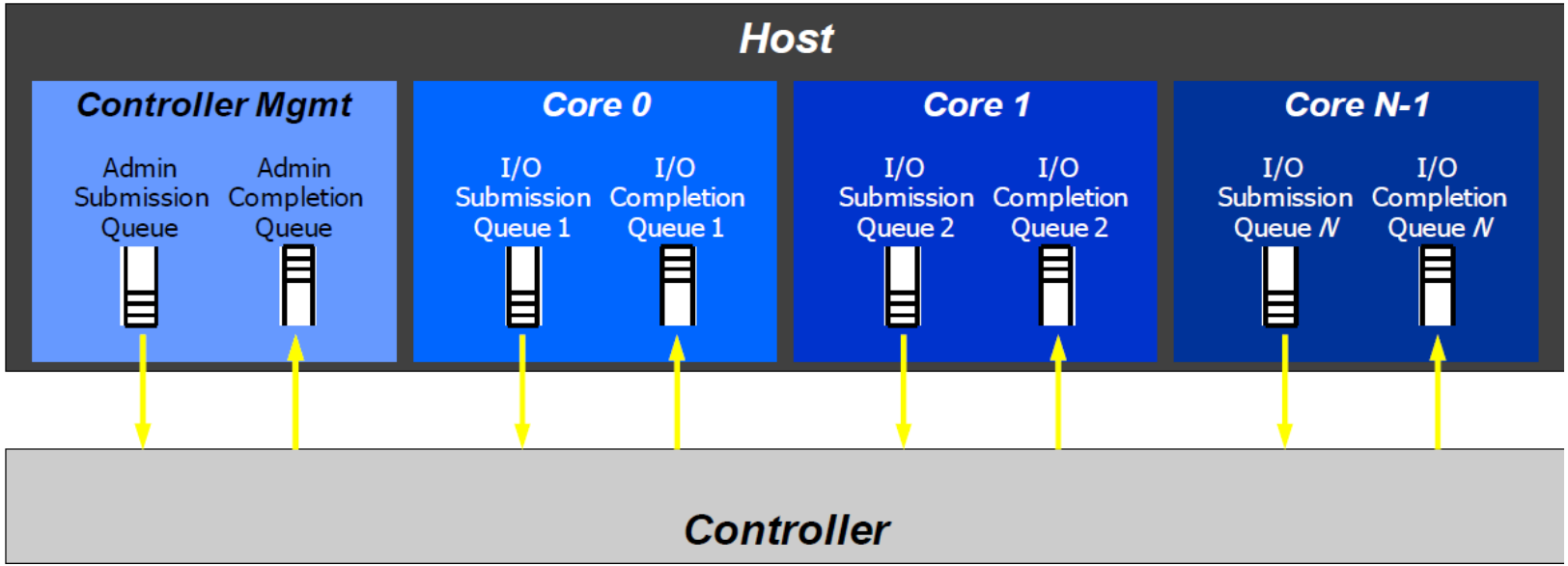
Storage Technologies Advancement

- Datacenter SSD storage bus and media interface are evolving
  - SAS/SATA →PCIe NVMe
  - NAND, 3DXP
- Protocol bandwidth and actual IOPS comparison

SSD	SATA	SAS	PCIe NVMe(x4)	PCIe NVMe(x16)
Gen. I	150MBps	300MBps	1000MBps	4000MBps
Gen. II	300MBps	600MBps	2000MBps	8000MBps
Gen. III	600MBps (~100K IOPS)	1200MBps (~250K IOPS)	4000MBps (~1M IOPS)	16000MBps (~3M IOPS)
Gen. IV	N/A	2400MBps (~500K IOPS)	8000MBps (~2M IOPS)	32000MBps (~6M IOPS)

Cloud: NVMe Enables Greater Scalability and Performance

- NVMe has been designed from ground up to capitalize on high IOPS, low latency and internal parallelism of SSDs
- NVMe built-in I/O queues linearly scales I/O initiation and completion w.r.t number of CPUs for high throughput
- NVMe minimizes hardware register access in I/O Path.



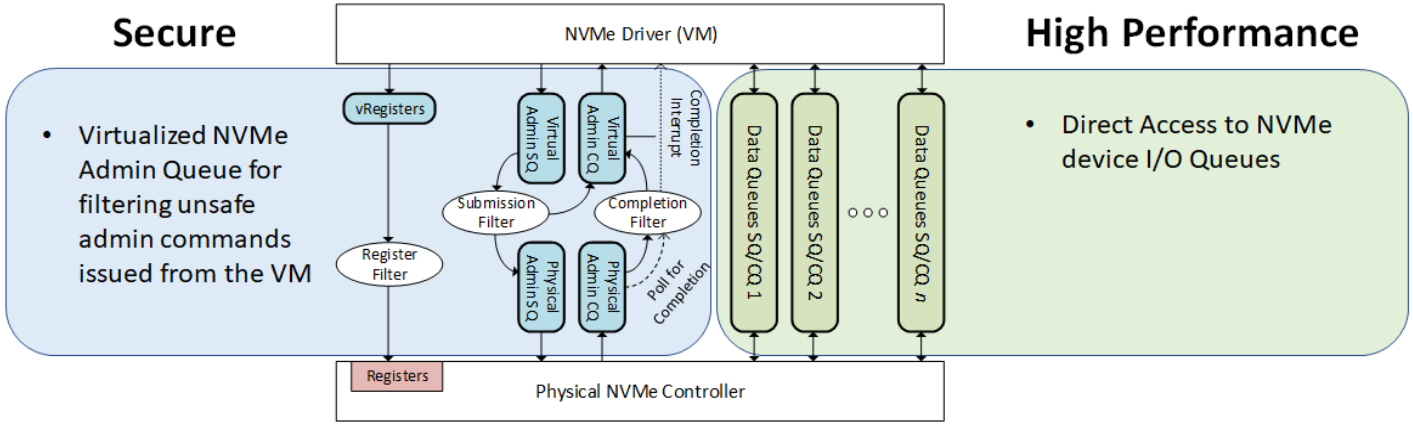
Windows Hyper-V Storage I/O Virtualization Path

- VM storage access mode
  - Emulation path: slower
  - Para-virtualized (synthetic) path: faster
  - Direct hardware assignment: fastest
- VM para-virtualized storage pathlength overhead breakdown
  - IO goes over storage stack twice both in host and VM

Component	Sub-component	Overhead
VM (~20%)	HyperV VM enlightenment (storvsc)	5%
	VM OS storage stack	15%
VM-Host (~20%)	VM-host boundary crossing overhead(vmbus)	20%
Host (~40%)	Hyper-V host component (storvsp, vhdmp)	20%
	Host file system (ntfs)	10%
	Host OS storage stack (storport, miniport)	10%
Hypervisor (~20%)	Interrupt delivery related	15%

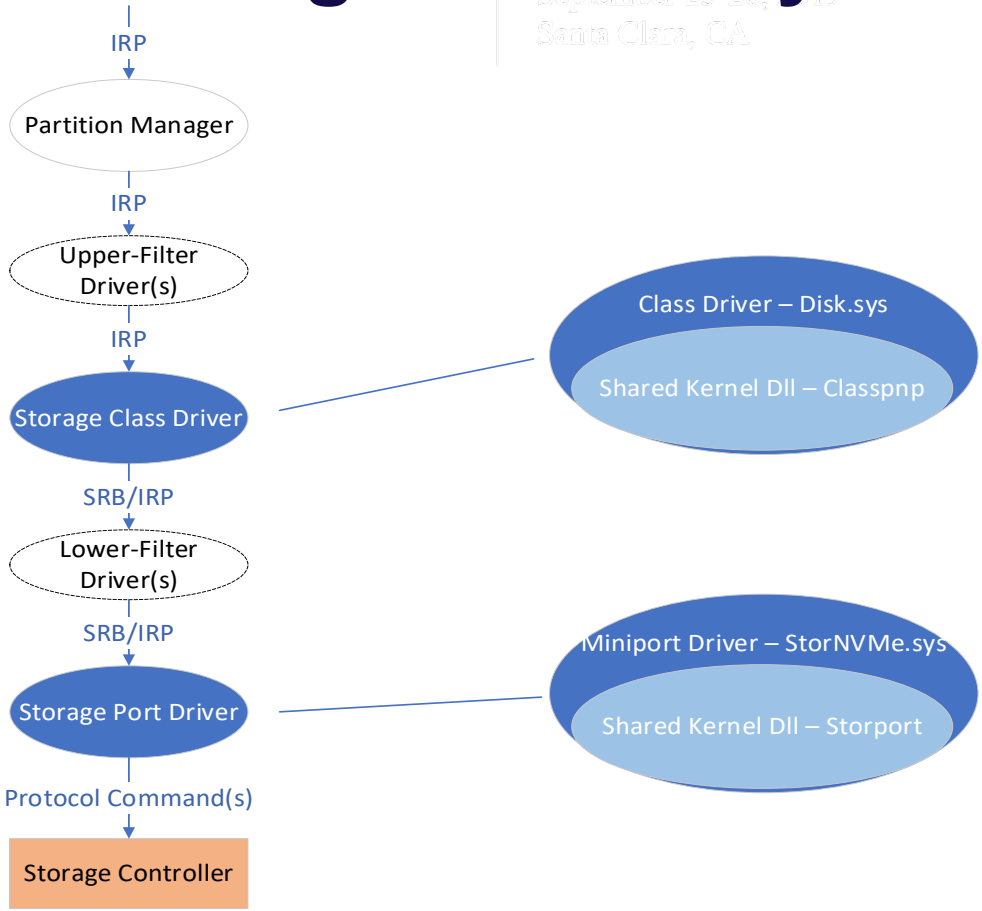
Direct Storage HW Access from VM

- Hyper-V SCSI Passthru is not sufficient to provide maximum performance
  - Cannot avoid Hyper-V para-virtualized path overhead as well as contention between host root and VM VPs
- Hyper-V Discrete Device Assignment (DDA)
  - Allows pass through PCIe devices directly to a Guest VM to avoid overhead with traditional para-virtualized path
  - Security concerns caused by exposing admin queue of NVMe device to (malicious)VM user
- NVMe Direct
  - A high performance and secure Hyper-V PCIe passthrough solution
  - First made available to support Microsoft Azure Lsv2 series VM SKUs

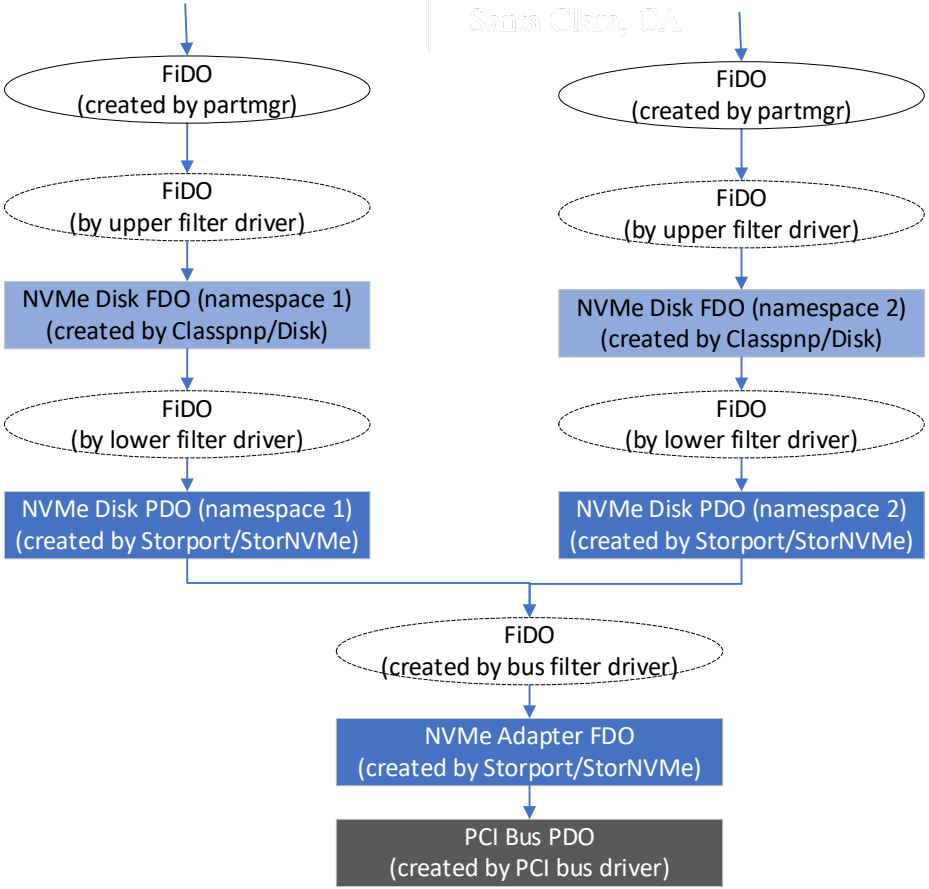


Windows Storage Stack

Windows Storage Stack Layer

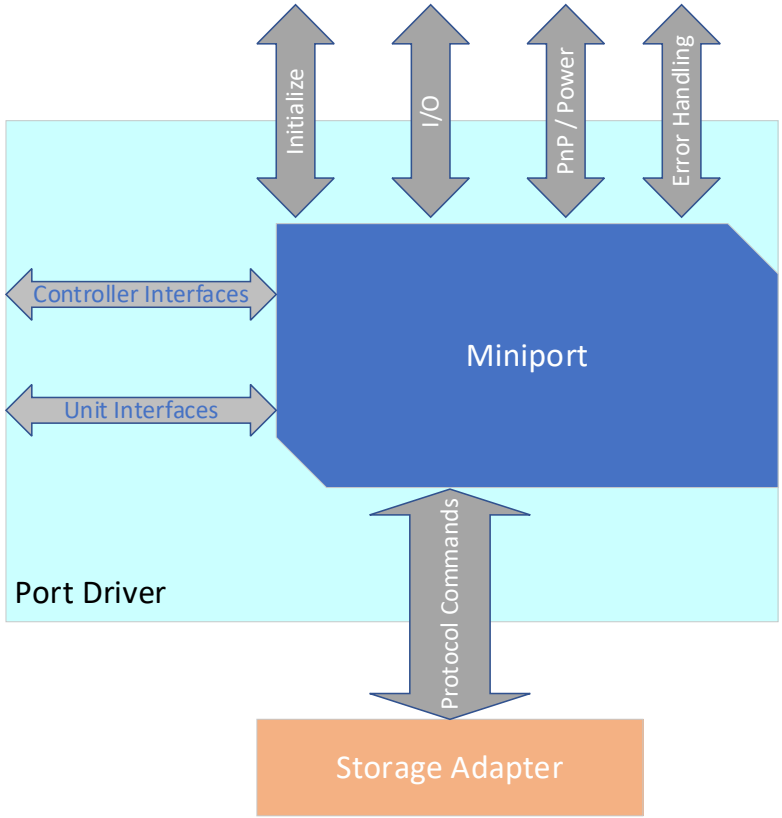


Filter Driver – Modify behavior or add feature



Port Library / Miniport

- Reduce developer work to support new storage adapter
- Port library / Miniport driver
  - Example: Storport and StorNVMe.sys
- Abstract common logic into Port driver; Leave hardware specific logic to Miniport driver.



Windows Storage Stack Evolution

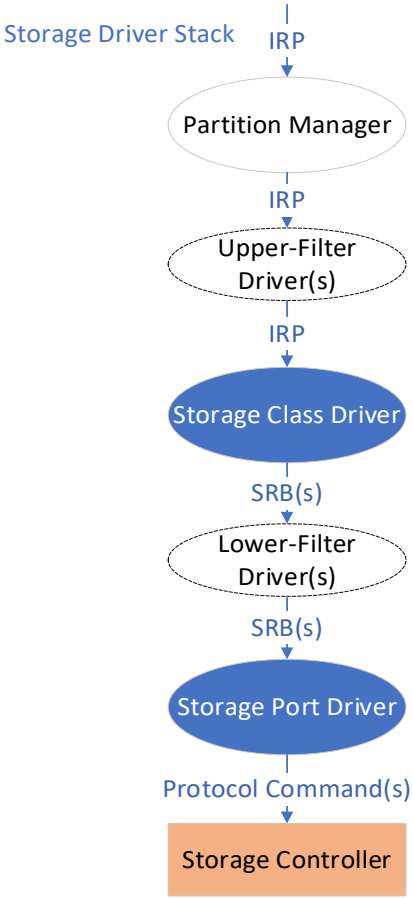
- Designed to be layers of drivers with each layer having a specific responsibility
  - Bus function driver, device function driver, and filter drivers
- SCSI protocol based stack with translation to other protocols at the bottom of the stack
- Multiple Port Driver Models; Consolidated to Storport model
  - SCSIport, ATAport, Storport
- Advantage of Windows storage stack
  - Single storage driver model supports diverse hardware for different markets (e.g. client, enterprise)
  - Reduce storage driver development complexity by providing class and port driver libraries
  - Layered model provides flexibility of filter driver development
- Cloud environment supports limited choices of hardware owned by cloud providers
  - Opportunity for a new driver model with reduced software overhead

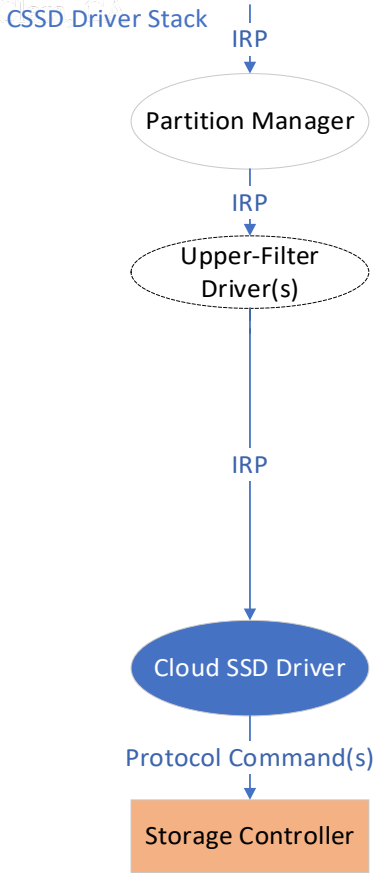
Cloud SSD Driver Stack

Cloud SSD Driver Design

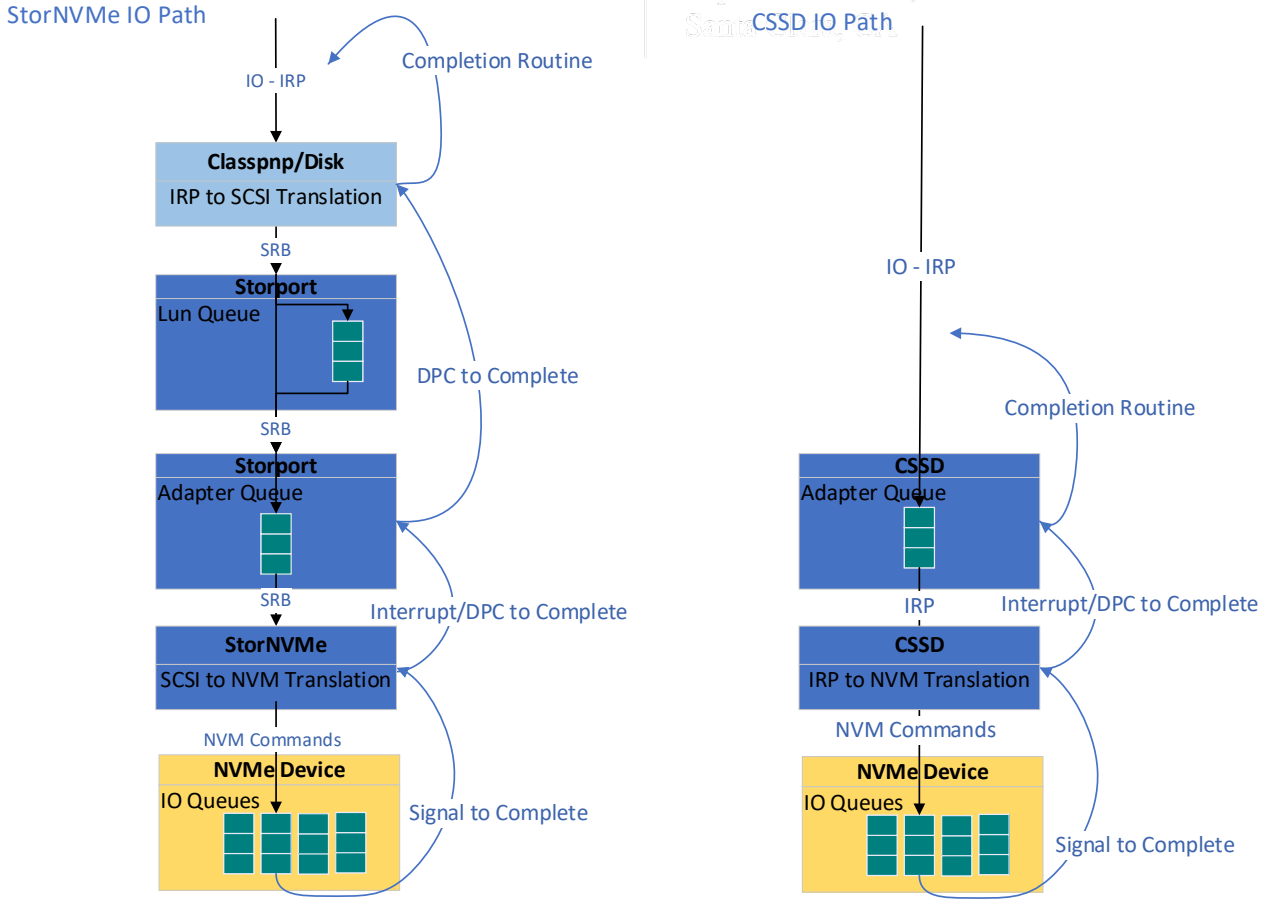
- Focus on cloud environment for specific customer scenarios – IO intensive, CPU sensitive
  - Not intend to replace existing Windows storage stack
  - Target NVMe device only
- More efficient IO path
  - Reduce driver stack depth by merging multiple storage drivers into a monolithic driver
  - Translating IO from IRP to NVMe command and sending to device directly in context of original IRP
  - Improve queuing latency by reducing IO queue layers
  - Only support features needed in cloud environment
- Support asynchronous IOCTL paths
  - Allow vendor specific read/write IO command for new hardware through IOCTL path as fast I/O path
- Modular design by dividing Cloud SSD driver into two parts:
  - Core Library: contains general storage device functionality. - Cssdcore.lib
  - Driver: contains NVMe protocol-specific functionality. - Cssd.sys
- Support both interrupt and polling IO mode

Cloud SSD Driver Stack





Windows NVMe IO Path and Queuing



Cloud SSD Limitation

- New driver is designed for controlled environment only, such as Azure Cloud
  - Feature is not on par with general purpose driver
- Target NVMe SSDs only, no support of other storage bus interfaces
- No support for lower disk class filter drivers

Performance Evaluation

Test configuration

- Physical machine configurations

System	Microsoft Azure Gen6 2-Socket 96 logical processor with Hyper-Threading on
Processor	Intel Xeon Scalable Skylake Platinum 8168 Processor 2.70GHZ
Memory	192G 2666MHZ DDR4
Storage	Intel PCIe Gen 3.0 x8 HHHL P4608 6.4TB NVMe SSD (1.6M IOPS per device)
Host OS	Azure Cloud OS

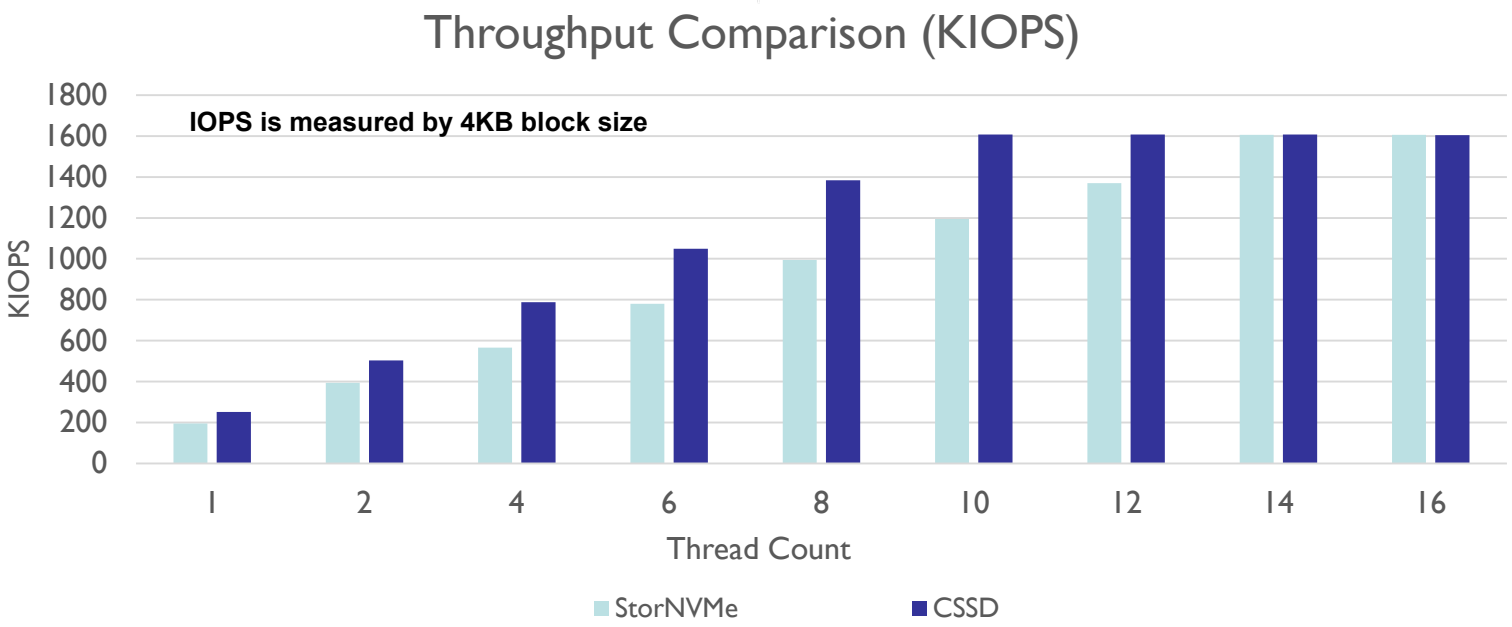
- Virtual machine configurations

Virtual Processors	24VMVP
Memory	64G RAM
Storage	Intel PCIe Gen 3.0 x8 HHHL P4608 6.4TB NVMe SSD Attached by NVMe Direct
Guest OS	Windows Server 2019

Test Tool and Experimental Settings

- Experimental Settings
  - Each NVMe device is used as a raw disk (raw I/O)
  - 1 diskspd instance per disk with various of IO worker threads
  - 128 Queue Depth per thread
  - 4KiB random read 4KiB aligned non-buffered I/Os
- Test tool diskspd 2.0.18a: A open source(link) storage load generator and performance test tool from Microsoft
- Performance comparison metrics
  - Total throughput: IOPS
  - Path length: Cycles/IO
  - Latency for single thread, single queue depth: ms

Throughput IOPS comparison



- CSSD provides up to 40% IO throughput gain
- CSSD fully utilizes disk throughput with 10 IO threads while StorNVMe needs 14 IO threads

Latency comparison

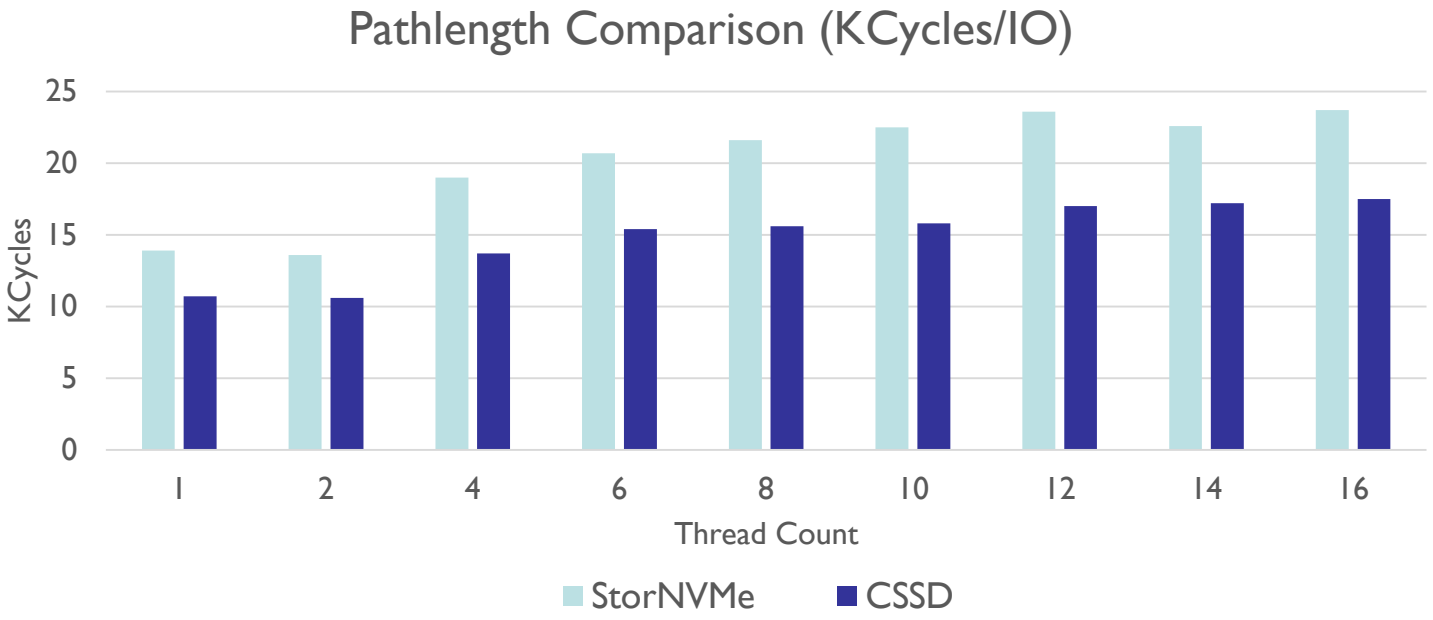
- Single thread, single queue depth

	Throughput (IOPS)	PathLength (Cycles/IO)	Avg. Latency (ms)	3-nines Latency (ms)
StorNVMe	46000	30900	0.200	0.121
CSSD	51000	24300	0.190	0.119
CSSD vs. StorNVMe	11%	-21%	-5%	-2%

- 16 threads, 128 queue depth

	Throughput (IOPS)	PathLength (Cycles/IO)	Avg. Latency (ms)	3-nines Latency (ms)
StorNVMe	160500	23700	1.27	2.76
CSSD	160800	17500	1.27	2.06
CSSD vs. StorNVMe	0%	-23%	0%	-25%

Pathlength Comparison



- Pathlength is measured by CPU cycles per IO
- CSSD provides up to 30% pathlength reduction

Conclusions

- Rapid storage hardware advancement could take advantage of highly efficient IO path
- Existing Windows storage stack has opportunities to be further optimized for cloud workloads
- Prototype driver explores ideas for future Windows storage software stack innovations with demonstrated performance benefits and preserves maximum user application compatibility
- A cloud-specific driver enables faster deployment of high performance VM SKUs in cloud

Q & A