Implementing Virtual NVMe for Flash-Storage Class Memory

Jinpyo Kim, Sr. Staff Engineer, VMware

Murali Rajagopal, Storage Architect, VMware

Virtual NVMe: Motivations (1)

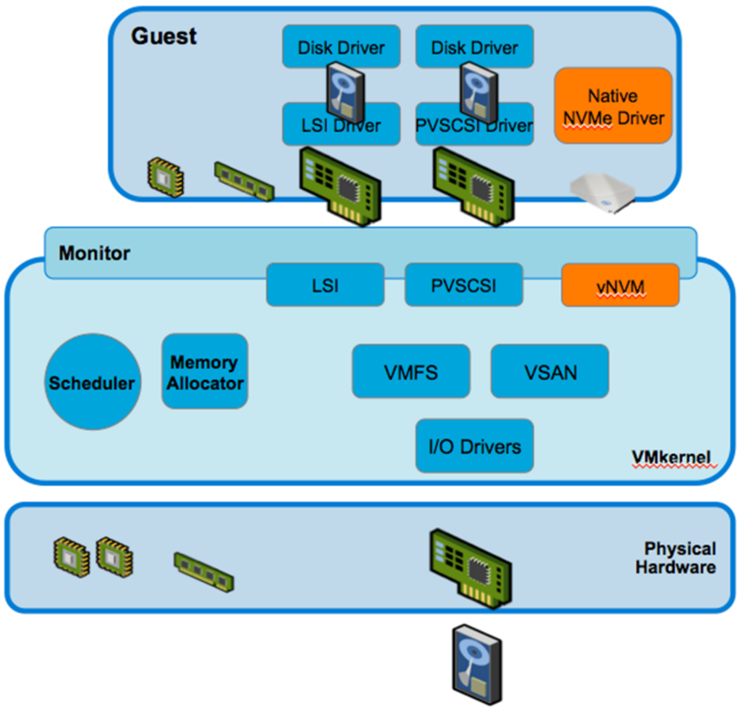
- **Increasing demands for high performing virtual storage devices**
  - **Virtual storage appliances, databases, big data, cloud native and low latency apps**
- **Supporting highly scalable I/Os (multi-Q)**
  - **16 queues, total 4K queue depth**
- **Reducing in-guest I/O stack overhead**
  - **Avoid unnecessary in-guest SCSI stack overhead for NVMe and NVDIMM fast block storage**

Virtual NVMe: Motivations (2)

- **Reducing storage virtualization overhead on ESX**
  - **Reducing translation overhead in ESX storage stack**

- **Enabling more features**
  - **Namespace management: in-band storage provisioning**
  - **Directive support (stream, I/O determinism)**
  - **Encryption**
  - **Data protection**

Virtual NVMe: High Performance Guest Block I/O



- **Guest NVMe driver**
  - No VMWare overhead to maintain drivers for all Oses
  - **Lower CPU cost per IO in guest I/O stack**
  - **Scalable I/Os with Multi-Q support**
- **NVMe emulation**
  - NVMe register interface emulations
  - Queue/name space managements
  - BE(Back End) for VMFS VMDKs, **vSAN and VVoL vmdks, NVM/NVMe native virtual storage objects**

Virtual NVMe: Implementation

- **Virtual NVMe Device on vSphere 6.5 or later**
  - **NVMe v1.0e compliant virtual device (mandatory admin and I/O commands)**
  - **Hot Add/Remove Support (max.15 namespaces and 4 controllers)**
  - **16 Queues/Interrupts (1 admin + 15 I/O queues), total 4K queue depth**
  - **PC BIOS/EFI boot support, No custom device driver in guest OS (Linux, Windows, and so on)**

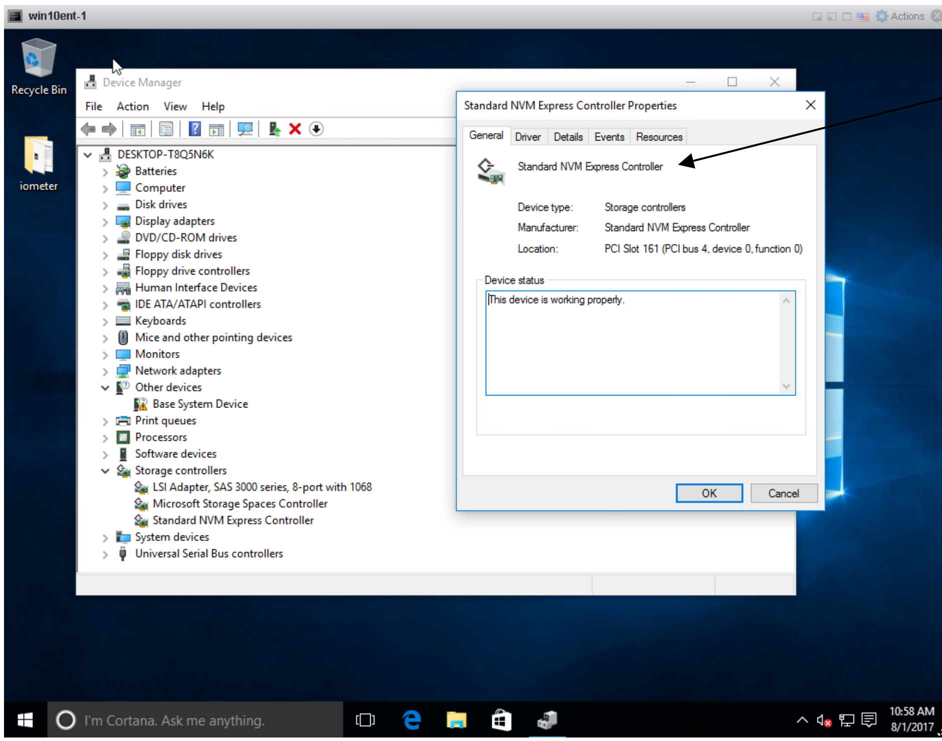
Supported NVMe Commands

	Admin Commands
●	Create I/O Submission Queue
●	Delete I/O Submission Queue
●	Create I/O Completion Queue
●	Delete I/O Completion Queue
●	Get Log Page
●	Identify
●	Abort
●	Set Features
●	Get Features
●	Asynchronous Event Request
	Firmware Activate (optional)
	Firmware Image Download (opt)
	Format NVM (optional)
	Security Send (optional)
	Security Receive (optional)

	NVM I/O Commands
●	Read
●	Write
●	Flush
	Write Uncorrectable (optional)
	Compare (optional)
●	*Dataset Management (optional)*
	Write Zeros (optional)
	Reservation Register (optional)
	Reservation Report (optional)
	Reservation Acquire (optional)
	Reservation Release (optional)

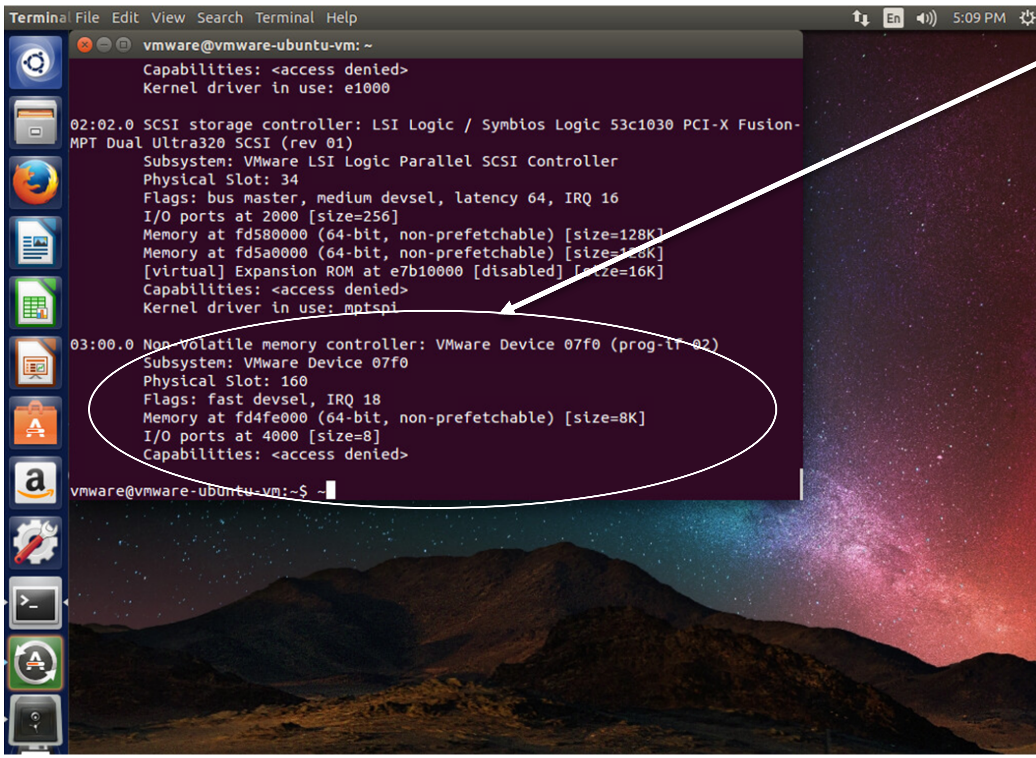
● **Implemented.**

Windows 10 device manager view



- Identified with in-box NVMe device driver

Ubuntu 14.04 LTS lspci output



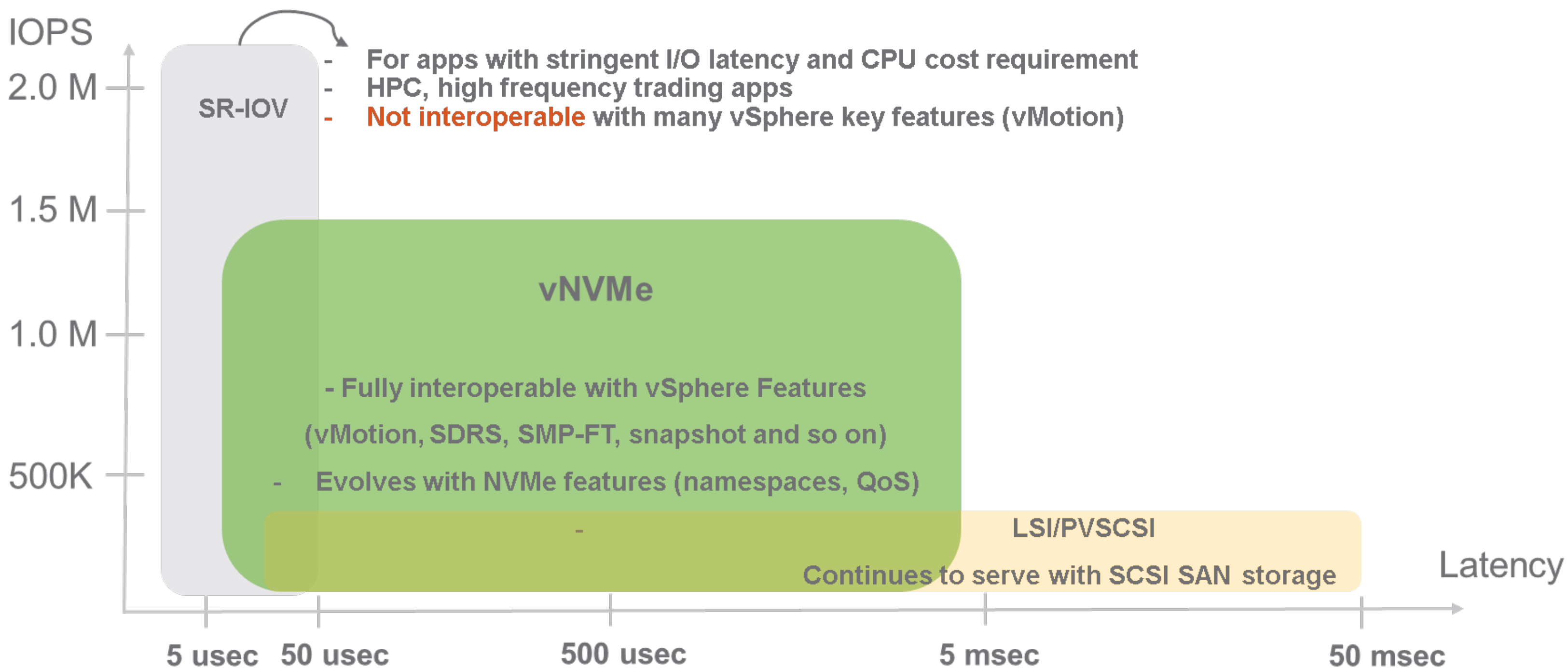
- Identified with in-box NVMe device driver

Increasing proportion of virtualization overhead in I/O latency

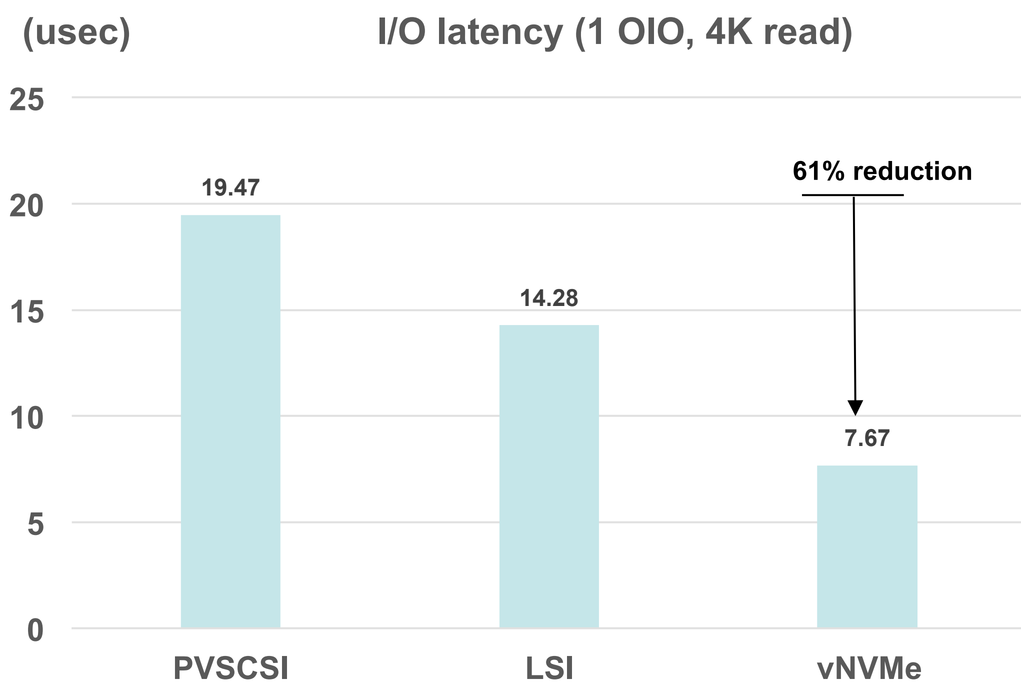
Mean Device Latency (usec)	Devices	Virtualization overhead* (%)
> 100 usec	SATA SSD (Mid range)	< 13%
> 35 usec	Enterprise SATA/PCIe SSD	< 27%
> 15 usec	NVMe SSD	< 43%
< 5 usec	FlashDIMM, NVMe PCM	< 60%

* Growing latency proportion of vmm/vmkernel IO latency in total IO latency
(with the assumption of vmm/vmkernel io latency = 15 usec, guest io stack = 5 usec)

Virtual NVMe Performance/Functional Coverage
(Compared to SR-IOV and LSI/PVSCSI)



VM I/O latency for DIMM memory backed storage*



- FIO (1 OIO, 4K 100% read)
- 4 vCPU Ubuntu 16.04 VM
- 1 memory backed virtual disk
- Intel E5-2687W v3 @3.1 GHz
- vSphere 6.5 GA

- **Compared to PVSCSI, 61% of I/O latency was reduced with vNVMe device.**

* Subject to change depending on measurement setup

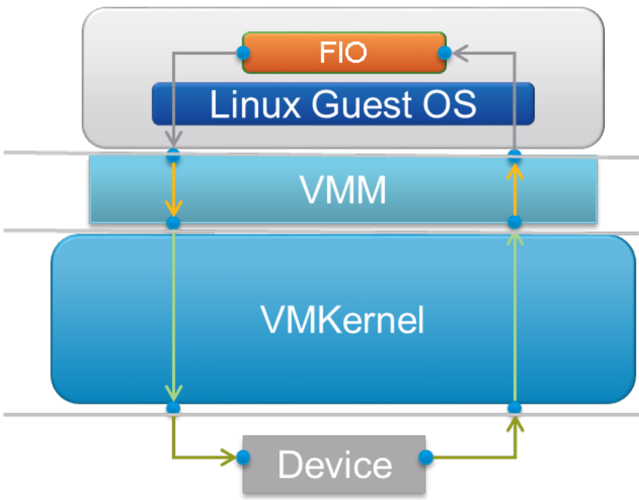
Taking a closer look at vNVMe I/O latency (4K read)*

```
benchmark: (g=0): rw=randread, bs=4K-4K/4K-4K/4K-4K, ioengine=libaio, iodepth=1
fio-2.2.10
Starting 1 process
Jobs: 1 (f=1): [r(1)] [100.0% done] [518.3MB/0KB/0KB /s] [133K/0/0 iops] [eta 00m:00s]
benchmark: (groupid=0, jobs=1): err= 0: pid=3119: Thu Jul 14 11:54:42 2016
  read : io=28845MB, bw=492282KB/s, iops=123070, runt= 60000msec
    slat (usec): min=4, max=2286, avg= 6.64, stdev= 1.77
    clat (usec): min=0, max=2185, avg= 0.96, stdev= 2.42
     lat (usec): min=5, max=2288, avg= 7.67, stdev= 2.70
    clat percentiles (usec):
     |  1.00th=[    0],  5.00th=[    0], 10.00th=[    0], 20.00th=[    0],
     | 30.00th=[    0], 40.00th=[    0], 50.00th=[    0], 60.00th=[    0],
     | 70.00th=[    1], 80.00th=[    1], 90.00th=[    6], 95.00th=[    6],
     | 99.00th=[    7], 99.50th=[    7], 99.90th=[   13], 99.95th=[   17],
     | 99.99th=[   18]
    bw (KB  /s): min=310856, max=540456, per=99.63%, avg=490460.92, stdev=81735.37
```

Guest VM perceived I/O latency

*** Subject to change depending on measurement setup**

VM I/O latency breakdown for vNVMe (DIMM-backed block storage)*



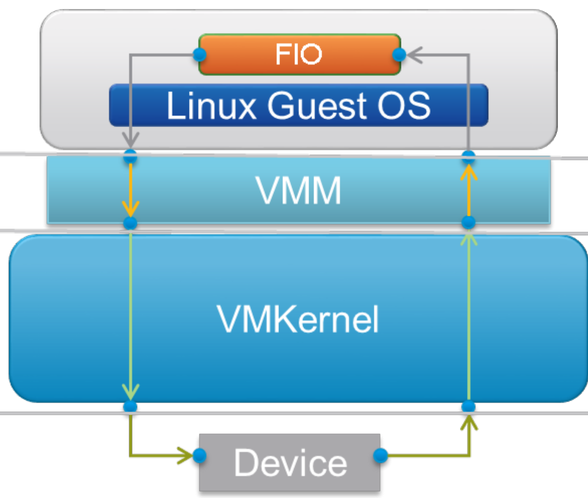
	on DIMM
total	7.6 usec
Time in guest OS	3.5 usec
Time in VMM	1.1 usec
Time in VMkernel	2.5 usec
Time in device	0.5 usec

- **When best possible backing storage (memory-backed block storage) is used, time spent in hypervisor storage stack can be reduced to ~3 usec.**

*** Subject to change depending on measurement setup**

Flash Memory Summit

VM I/O latency breakdown for vNVMe (NVMe SSD block storage)*



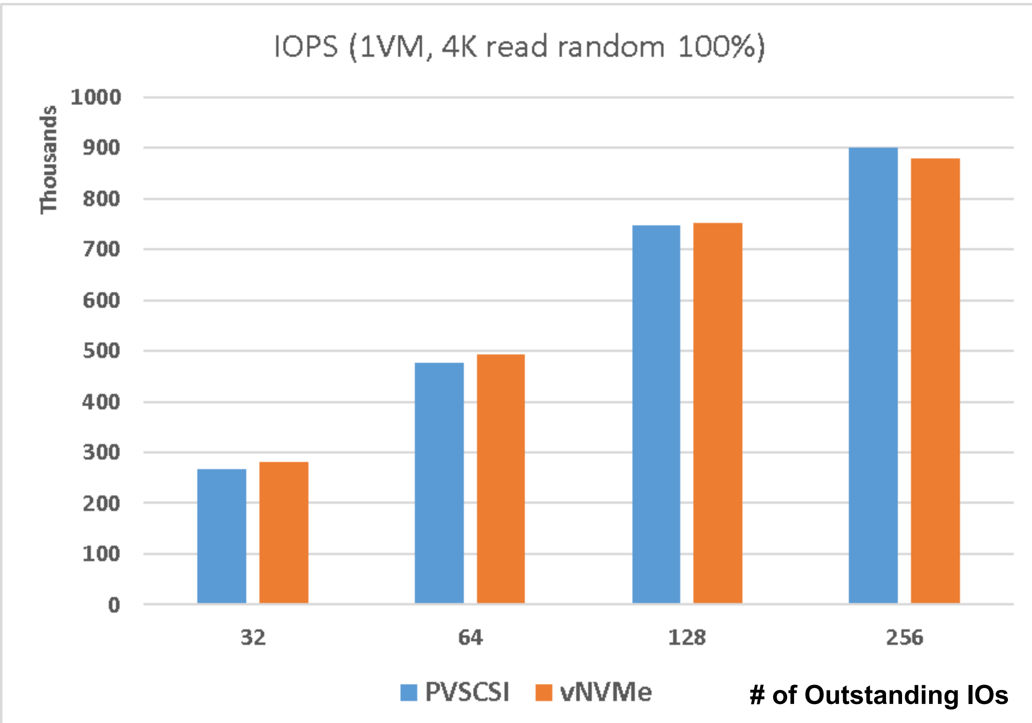
total	29.7 usec
Time in guest OS	3.5 usec
Time in VMM	1.2 usec
Time in VMkernel	10 usec
Time in device	15 usec

on NVMe SSD

- **When NVMe SSD block storage is used, time spent in hypervisor storage stack is around ~12 usec.**
- **Further storage stack optimization opportunities for NVMe storage**

*** Subject to change depending on measurement setup**

IOPS scalability (NVMe SSD block storage)*

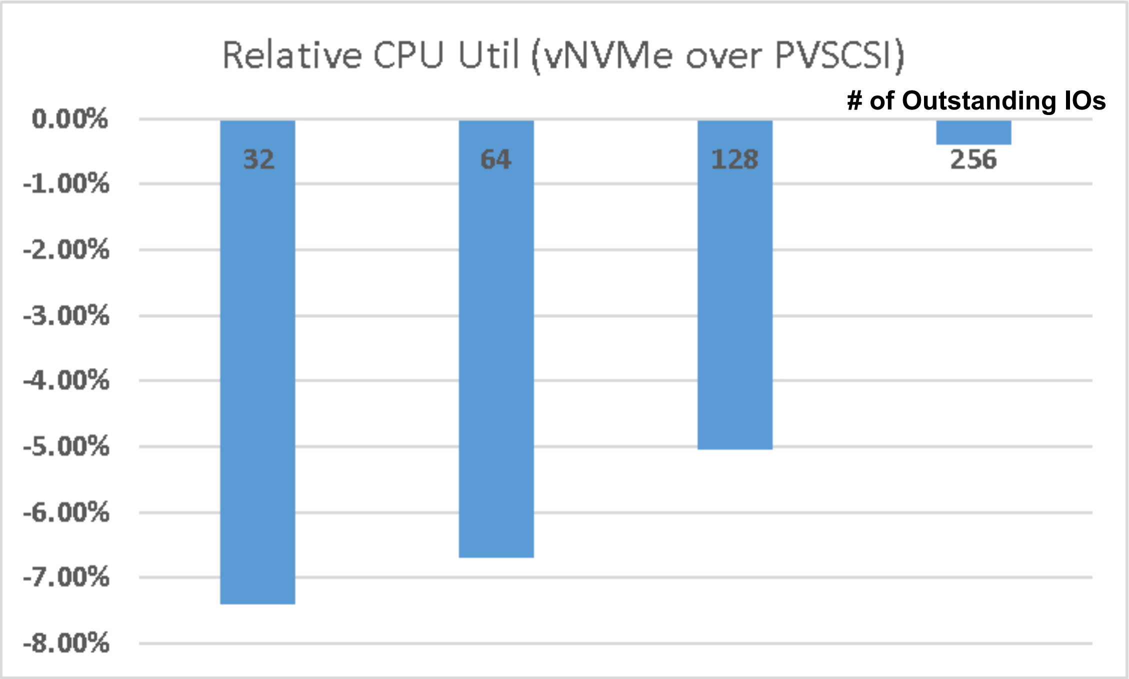


- FIO (1 OIO, 4K 100% read)
- 8 vCPU Ubuntu 16.04 VM
- 8 NVMe backed virtual disk
- Intel Broadwell @2.6 GHz
- 1 Samsung PM1725 NVMe

- **IOPS scales well in both virtual HBAs up to device's advertised max IOPS.**
- **vNVMe outperforms PVSCSI in nominal # of outstanding IOs.**

* Subject to change depending on measurement setup

Relative CPU Util (NVMe SSD block storage)*



- **vNVMe consumes 0.5 ~ 7.5% less CPU cycles compared to most CPU efficient para-virtual SCSI HBA (PVSCSI).**

*** Subject to change depending on measurement setup**

Future*: feature rich virtual HBA

- **Namespace management support**
  - **Virtual disk or raw mapped physical NVMe namespace is good fit for in-band storage provisioning (create, delete, resize and so on).**
- **Emulated controller support (specified in NVMe v1.3)**
  - **Further reduce VM-exit overhead with guest driver support**
- **Advanced feature support**
  - **Directive: Streams, IO determinism and so on**

*** May or may not be implemented in the future**

Virtual NVMe: Summary

- **Virtual NVMe device is high performance virtual device for NVMe SSD and NVDIMM storage.**
  - **We showed promising results: ~7 usec end-to-end latency and ~1M IOPS for single VM.**
- **vNVMe provides good performance without losing compatibility with existing great features on virtualization platform (vMotion, FT and so on).**
- **vNVMe will evolve with NVMe specification and new features.**