SDC 18
September 24-27, 2018
Santa Clara, CA
www.storagedeveloper.org

Achieving 10-Million IOPS from a single VM on Windows Hyper-V

Liang Yang & Danyu Zhu
Microsoft

Motivation: Why 10-Million IOPS?

- Strong customer demands
- Cloud market growth and competition
  - Higher density and capacity
  - Higher performance in terms of throughput and latency
- Hardware technologies advancing rapidly
  - Faster storage and advanced processor
- Storage optimized SKUs offered in Cloud
  - Handling intensive I/O workloads at millions IOPS level

Storage Technology Advancement Enables Higher IOPS

- Datacenter SSD storage bus interface
  - SAS/SATA →PCIe
- Protocol bandwidth and actual IOPS comparison

SSD	SATA	SAS	PCIe NVMe(x4)
Gen. I	150MBps	300MBps	1000MBps
Gen. II	300MBps	600MBps	2000MBps
Gen. III	600MBps (~100K IOPS)	1200MBps (~250K IOPS)	3940MBps (~1M IOPS)

Note: IOPS numbers are measured by 4KiB size

NVMe Enables Greater Scalability

- 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

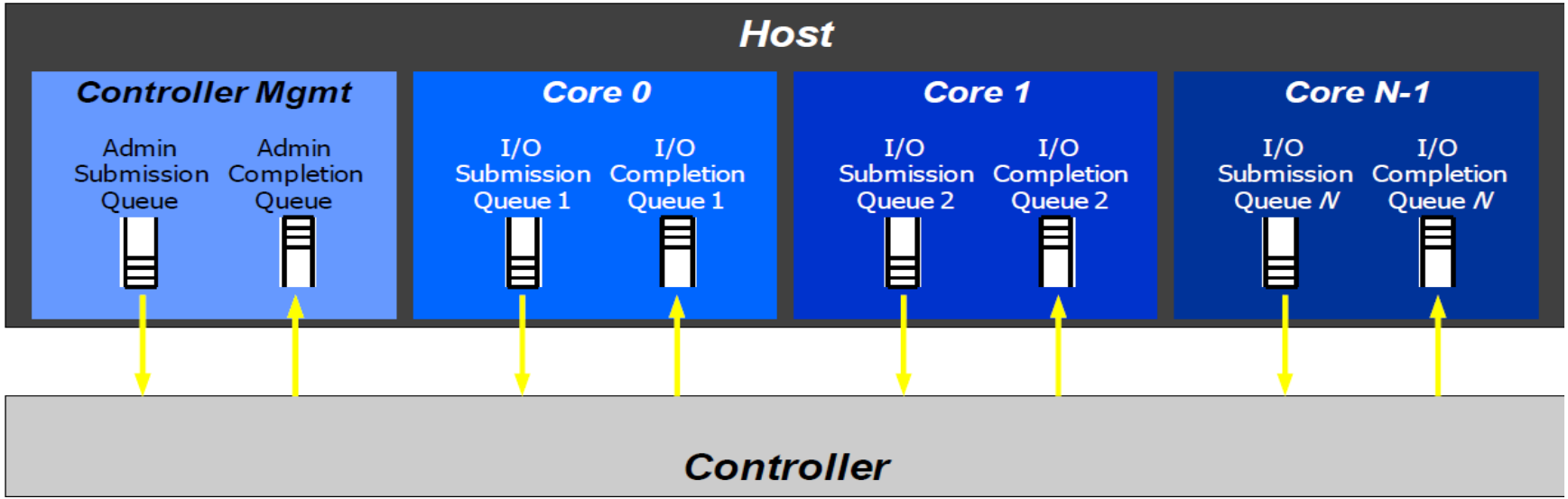


Image source: NVMe Spec

Windows Hyper-V Storage I/O Virtualization Path

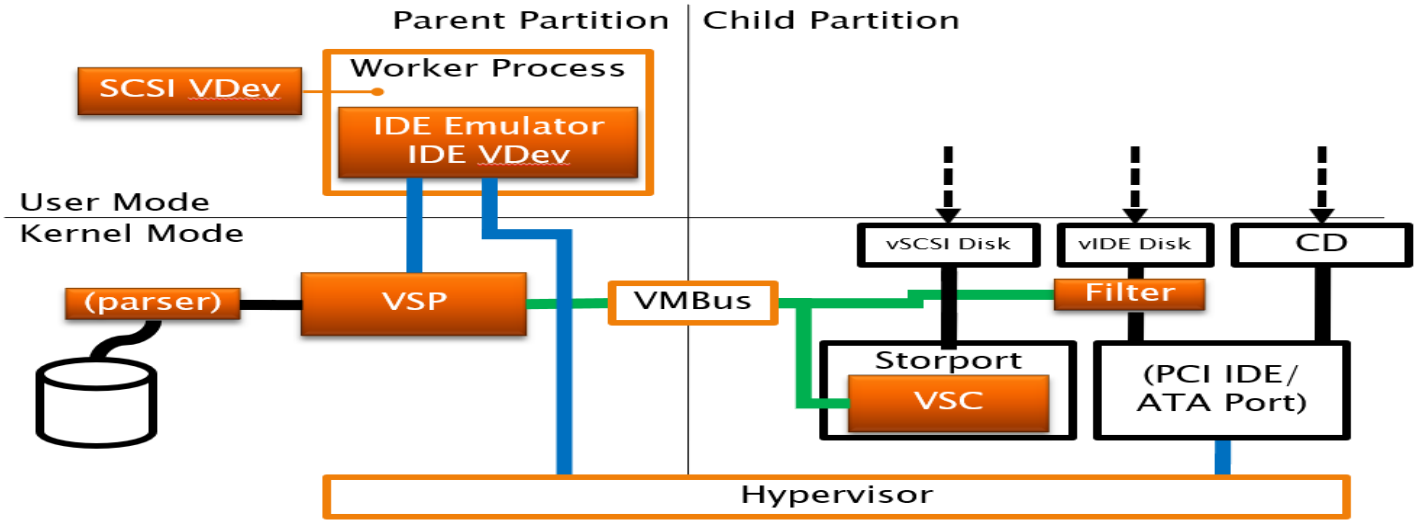
- VM Storage Access Mode
  - Emulation path: slower
  - Para-virtualized(synthetic) path: faster
  - Direct hardware assignment: fastest
- VM Storage Backing Mode
  - File based Virtual Hard Disk Image(VHD/X) : slower
  - Physical disk based(bypassing host file system) including SCSI Passthru or PCIe Passthru(e.g. Discrete Device Assignment in Windows Server 2016 or newer): faster

Review of Windows Hyper-V VM Storage Performance

- Our journey to million-IOPS VM
  - Announced achieving 1-Million IOPS from single VM in Windows Server 2012 in Aug. 2012: Presented at SDC 2012
  - Announced achieving 2-Million IOPS from single VM in Windows Server 2012 R2 in Oct. 2013: Presented at TechEd Europe 2013
  - Configurations used
    - 64 SCSI Passthru Disks backed by SATA SSDs
    - Para-virtualized path based
- Experiments showed throughput within VM is going to be capped at around 3-million IOPS with traditional para-virtualized path
  - A single PCIe Gen3.0 x4 NVMe SSD delivers close to 1-Million IOPS
  - High performance VM heavily uses local storage

Review of Windows Hyper-V Para-virtualized Storage Path

- Storage I/O goes through I/O initiation and completion path twice: within VM and again on host
- Contention between root virtual processors and VM virtual processors becomes a bottleneck capping VM performance(existing mitigations like Hyper-V minroot and CPU Groups)



Windows Hyper-V Storage Path CPU Overhead Breakdown

Component	Sub-component	Overhead
Guest (~20%)	HyperV guest enlightenment (storvsc)	5%
	Guest 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%

Note: Components CPU cost breakdown was measured by sampling under 1-Million IOPS workload on Windows and they could vary depending on the traffic and OS.

- CPU cost is typically a bottleneck for high IOPS workloads
  - New storage virtualization technologies help mitigate host as well as VM to host overhead
  - New processor I/O virtualization technologies help mitigate hypervisor overhead

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 VPs and VM VPs
- Challenges: storage SR-IOV path remains unclear
  - NVMe Spec defines a general way to support SR-IOV but definition not sufficient(e.g. lack of resource control) for real use
  - Today different SSD vendors have to make up missing details for themselves to implement SR-IOV
  - Software stack support for storage SR-IOV is difficult without industry standard

Hyper-V Discrete Device Assignment(DDA)

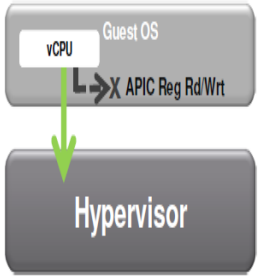
- DDA allows pass through PCIe devices directly to a Guest VM
- An experimental feature introduced in Windows Server 2016
- Pros: allowing VM user to access I/O queues of NVMe device directly brings significant performance gain compared with traditional para-virtualized path
- Cons: security concerns caused by exposing admin queue of NVMe device to (malicious)VM user

Secured Direct Storage HW Access in Cloud VM

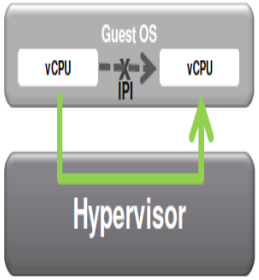
- PCIe NVMe device is best suited for this purpose
- Filtering out unsafe NVMe admin commands from VM while accessing I/O queues directly
  - Software solution: Hypervisor or dedicated host filter driver to intercept and filter out admin requests
  - Hardware solution: FPGA or customized ASIC for filtering purpose
  - Other solution: risky device management through BMC only

VM Interrupt Delivery w/o HW Acceleration: Slow

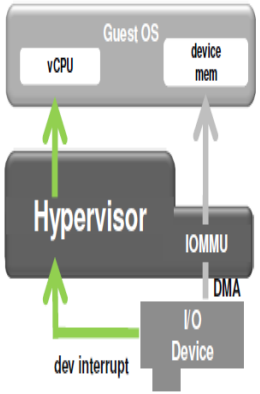
- VMEXIT overhead(in terms of hypervisor intercepts) during virtual interrupt delivery is primary hypervisor bottleneck capping high IOPS workload performance



APIC registers access(IRR, ICR, EOI)



Inter-Processor Interrupt(IPI)



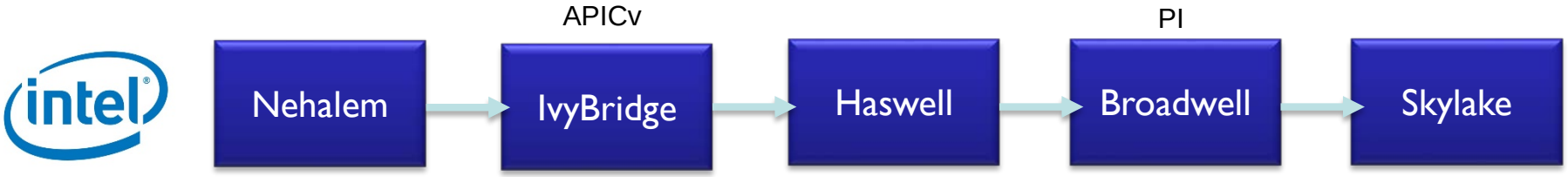
External Interrupt(I/O device)

Hyper-V Hypervisor Virtual Processor	_Total
% Guest Run Time	21.465
% Hypervisor Run Time	20.216
% Total Run Time	41.681
APIC EOI Accesses/sec	438,400.500
APIC IPIs Sent/sec	1,229,112.795
APIC MMIO Accesses/sec	0.000
APIC Self IPIs Sent/sec	74,545.745
APIC TPR Accesses/sec	0.000
External Interrupts/sec	4,084,892.024
Hardware Interrupts/sec	2,258,680.272
Hypercalls/sec	83.000
Other Intercepts/sec	2.000
Pending Interrupts/sec	2,735,141.642
Synthetic Interrupts/sec	68.000
Total Intercepts/sec	9,214,942.472
Virtual Interrupts/sec	2,745,270.607

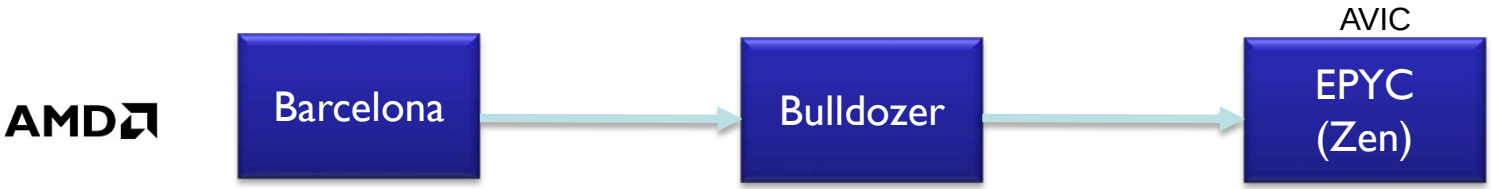
Windows VM VP Perfmon Counter under 4.5Million IOPS

Image source: Introduction of AMD AVIC Xen Summit 2012 by Wei Huang

Server Processor Virtualization Technology Advancement Enables Higher IOPS



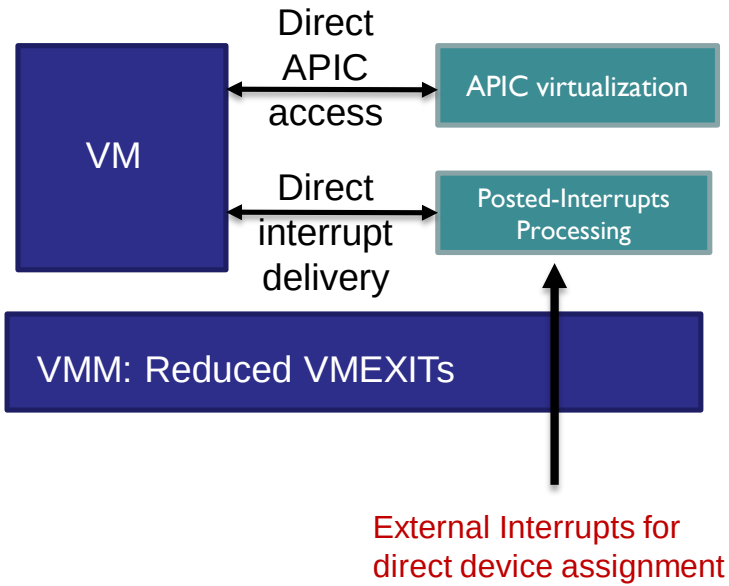
- APIC virtualization(APICv) support starts from Ivy Bridge
- Posted Interrupts(PI) support starts from Broadwell



- AVIC(Advanced Virtual Interrupt Controller) support is available on EPYC server processors

Intel Virtualization Technologies Advancement for I/O

- APIC virtualization (APICv) allows guest to directly access APIC registers from virtual APIC page
- Posted-Interrupt (PI) enables direct external interrupts delivery to VM virtual processors with reduced hypervisor involvement
- Intel Posted Interrupts(PI) and APICv support are enabled by default on Windows Server 2019



AMD Virtualization Technologies Advancement for I/O

- AVIC is Advanced Virtual Interrupt Controller(A virtual APIC to guest OS with hardware acceleration)
- AVIC allow most APIC access and interrupt delivery into guest directly with reduced VMEXITs
- IOMMU AVIC support
  - Doorbell signal to direct interrupts to guest for a running VP
  - Guest Virtual APIC (GA) logging is used to record pending virtual interrupts targeted for non-running VP
- Potential performance implications on GALog entry updates and overflow handling with high IOPS traffic

Building a 10-Million Storage IOPS Platform

- Physical machine configurations

System	Commodity HPE DL560 Gen10 4-Socket 192 logical processor with Hyper-Threading on
Processor	Intel Xeon Scalable Skylake Platinum 8168 Processor 2.70GHZ
Memory	512G 2666MHZ DDR4
Storage	8x Intel PCIe Gen 3.0 x8 HHHL AIC P4608 6.4TB NVMe SSD (~1.4M IOPS/SSD)
Host OS	Windows Server 2019

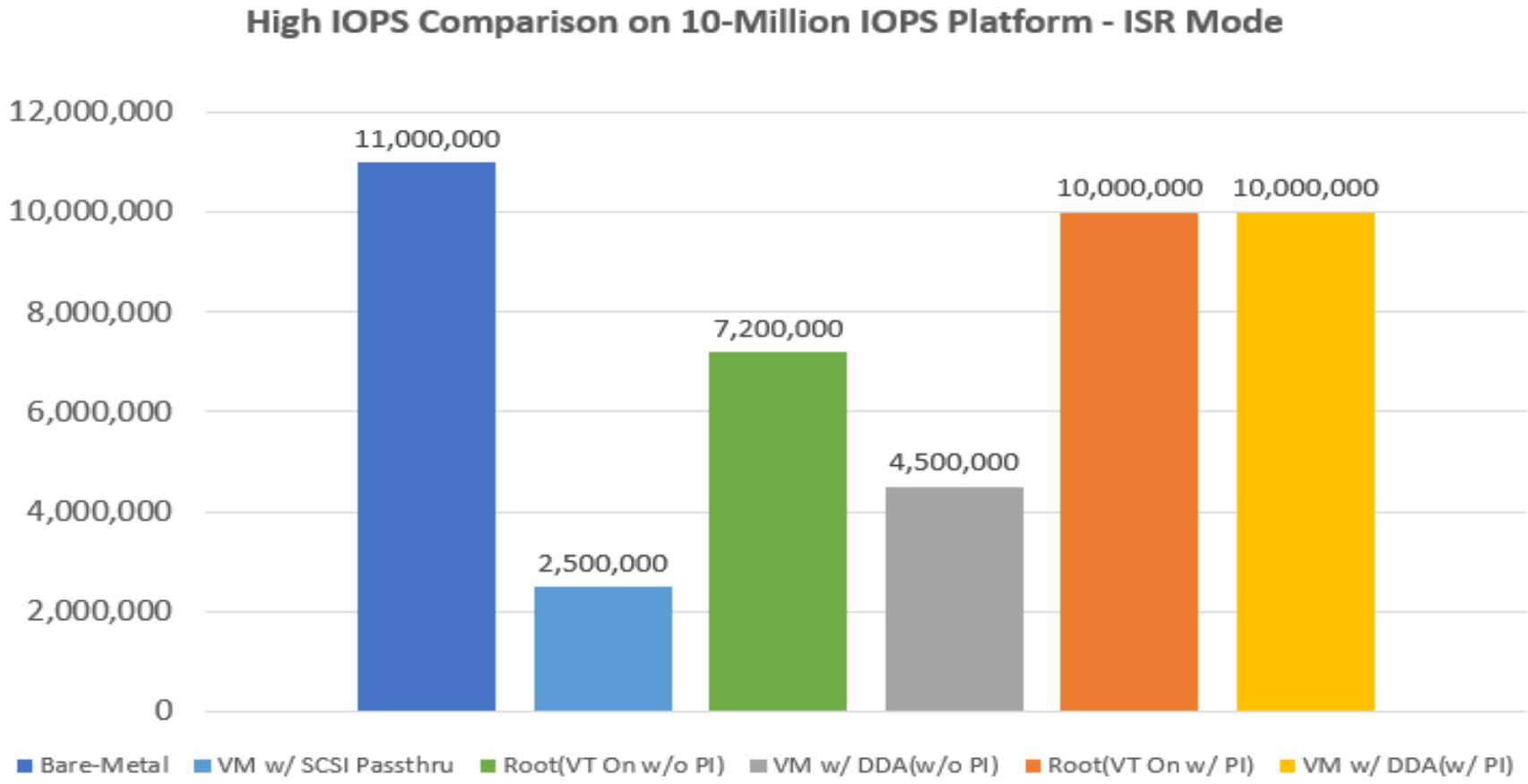
- Virtual machine configurations

Root Virtual Processors	192 Root VP
VM Virtual Processors	192 VM VP (for fair comparison with bare-metal and root)
Memory	64G RAM
Virtual NUMA	Enabled and matching physical NUMA(48 VP per vNUMA)
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 8 I/O threads affinitized to 8 VM VPs
  - 128 Queue Depth per thread and 1024 queue depths per SSD.
  - 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
  - *Command line for first NVMe device: diskspd -ag0,0,1,2,3,4,5,6,7 -b4k -r4k -t8 -o128 -Su -L -W30 –d240 -C30 #1*
- Performance comparison
  - Bare Metal vs. Root vs. VM

Reference: “What's New with DiskSpd, Microsoft's Storage Performance Tool for Windows” by Daniel Pearson from Microsoft at SDC2018

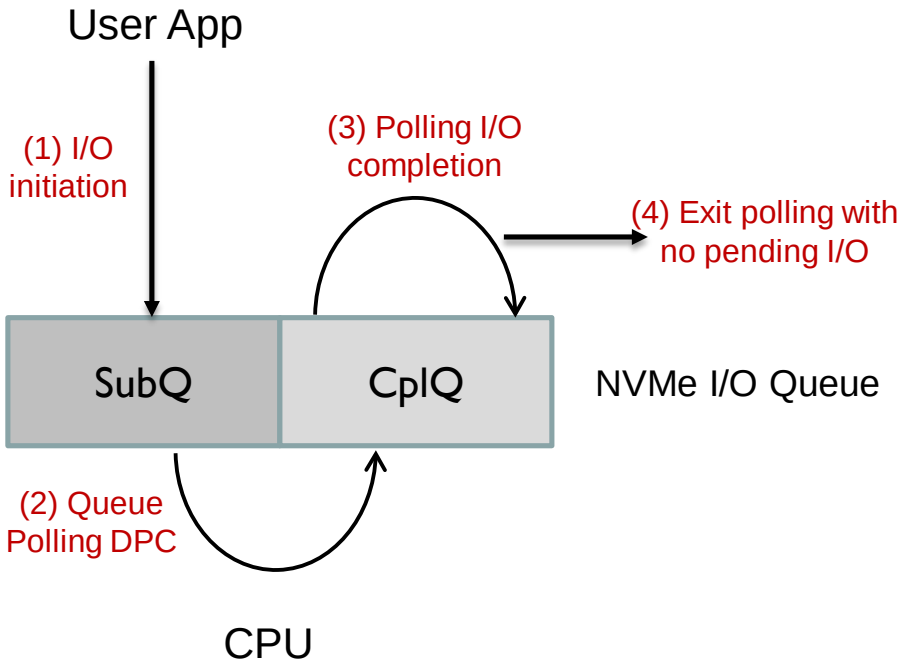


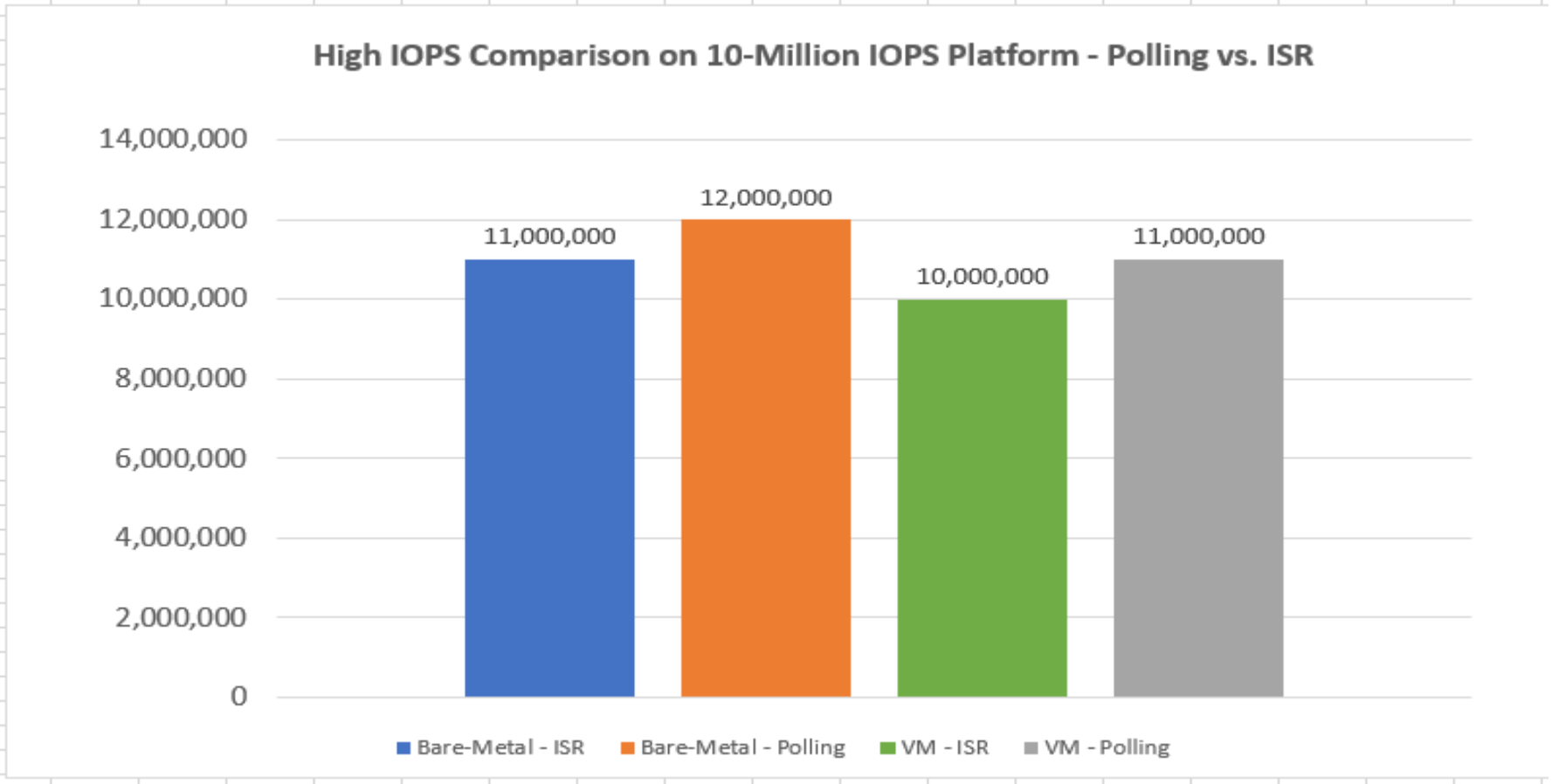
Experimental NVMe Polling Mode

- Why Polling? : Help VM achieve close to bare-metal performance for systems lack of efficient PI support
  - Eliminate interrupt delivery overhead
  - Bypass uneven MSI-X interrupt to target processors mapping issue (causing big performance concerns in virtualization environment)
- Polling vs. ISR performance
  - Polling provides lower and more consistent I/O tail latency
  - Polling incurs higher CPU cost for low queue depth workloads
- Linux kernel 4.4 introduced NVMe polling support, hybrid polling was added to 4.10
- None of existing Windows NVMe drivers support polling mode
  - Experimenting polling with modified StorNVMe driver on Windows Server 2019

A Highly Efficient NVMe Polling Implementation

- Interrupt is disabled for every polling mode queue
- Support mixed of polling queues and ISR queue(s)
- Use either ordinary DPC (optional) or threaded DPC to poll I/O completion for more responsive system under intensive traffic
- DPC scheduled on I/O initiating processor to balance CPU usage
- No dedicated polling threads are used.
- Only polling with outstanding I/Os





Conclusions

- I/O optimized SKU is critical offering for Cloud providers
- Traditional Para-virtualized path already hits performance limit as storage technologies advancing
- Direct device assignment and I/O virtualization enhancement are keys to achieve near-native performance from within VM
- Windows Hyper-V stack can provide 10 million IOPS from a single VM on modern commodity hardware

Acknowledgement

- Intel and HPE: for providing us hardware used to build this 10-million IOPS platform

Q/A