SDC
STORAGE DEVELOPER CONFERENCE
SNIA BANGALORE, 2016

MAY 26-27, 2016
BANGALORE, INDIA

NVMe Virtualization

Comparative study of NVMe Implementation & Performance in Virtualization models.

Nitin Kumar
Mallesh Lepakshaiah

Power Virtualization Development, IBM Systems

Agenda

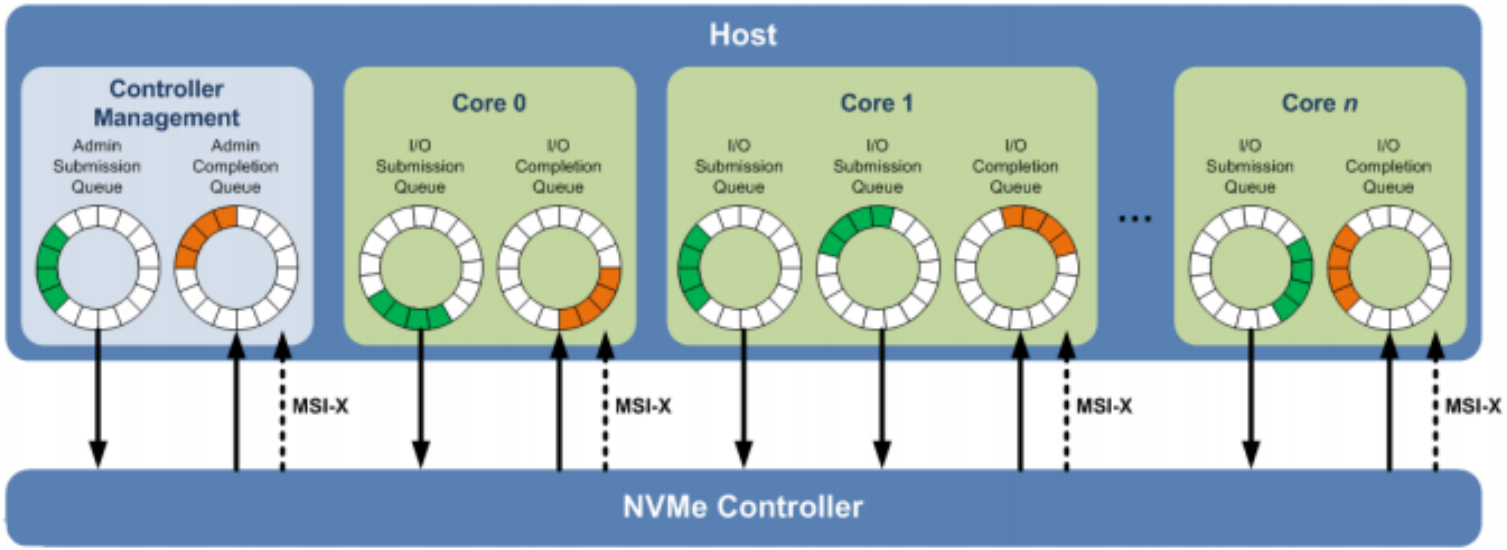
- What is NVMe
- NVMe Technical Overview
- NVMe Virtualization – Why?
- NVMe Current Limitations in Virtualization
- Virtualization Models
- NVMe Virtualization
- NVMe Virtualization Performance

What is NVMe

- NVM Express is a standardized high performance software interface for PCIe SSDs
- It's a scalable host controller interface designed to address the needs of Enterprise, Datacenter, and Client
- Standardizes register set, feature set, and command set to deliver performance
- Architected from ground up for NVM to be more efficient, scalable and manageable
- 13 Promoter companies
- Intel, Micron, LSI, Marvell, Cisco, EMC, Dell, Oracle, NetApp, sTec, Samsung, SanDisk, PMC Sierra
- Over 90 NVMe member companies

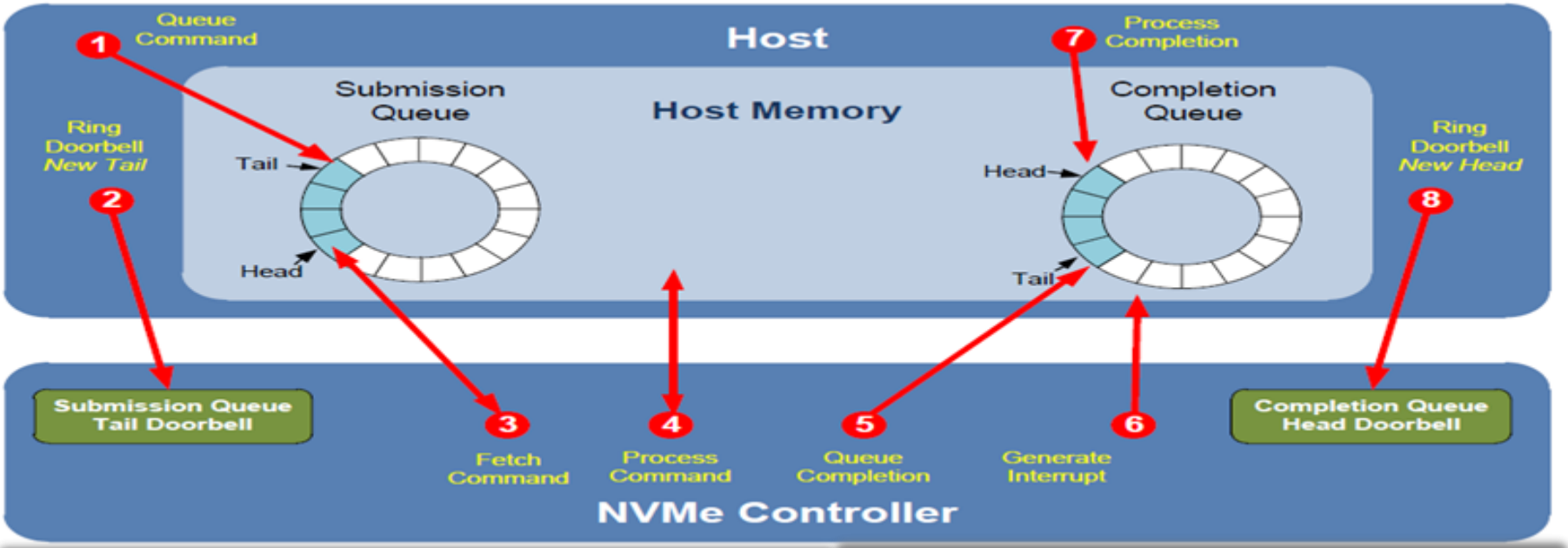
NVMe Technical Overview

- Supports deep queues (64K commands per queue, up to 64K queues)
- Supports MSI-X / Flexible interrupt aggregation
- Streamlined & simple command set – Admin and I/O Command set
  - 13 required commands – 10 Admin Commands and 3 I/O commands
  - 25 Available I/O commands
- Speed: 1GB/s per line – Scalable up to 16 Lines
- Scalable Queuing Interface with High Performance (1M IOPS)
  - Per core submission and completion queues
  - High Performance and Low Latency command issue
  - No Locking between Cores



Queuing

- Circular Queues to pass messages - Submission and Completion Queue pairs
- Queues are typically located in host memory – 1 per NVMe controller and 64K for I/O
- A Queue consists of set of fixed sized elements – 4K for admin and 64K for I/O
- Minimum of two queues - Multiple submission queues can be associated with a completion queue.



Command Submission

1. Host writes command to Submission Queue
2. Host writes updated Submission Queue tail pointer to doorbell

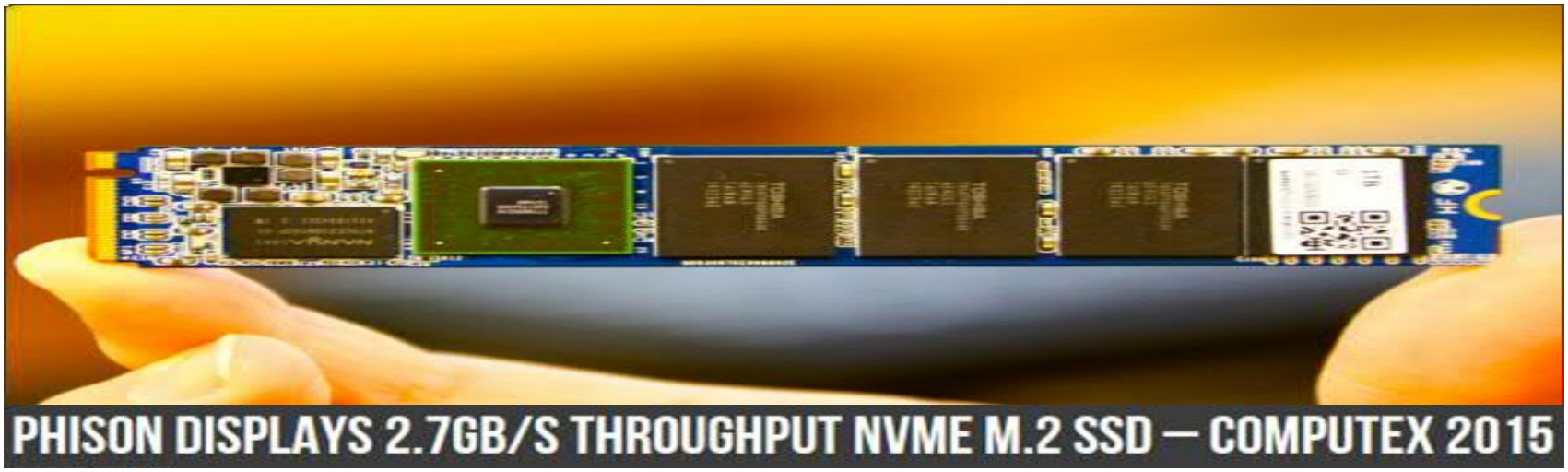
Command Processing

3. Controller fetches command
4. Controller processes command

Command Completion

5. Controller writes completion to Completion Queue
6. Controller generates MSI-X interrupt
7. Host processes completion
8. Host writes updated Completion Queue head pointer to doorbell

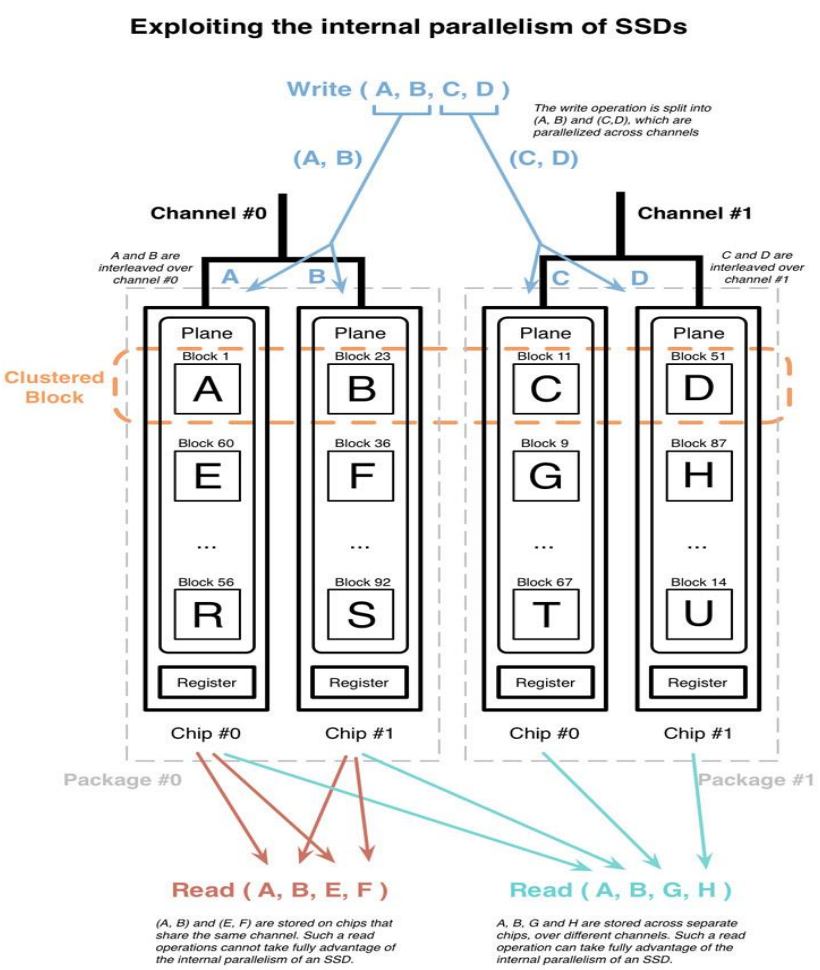
NVMe Virtualization - Why



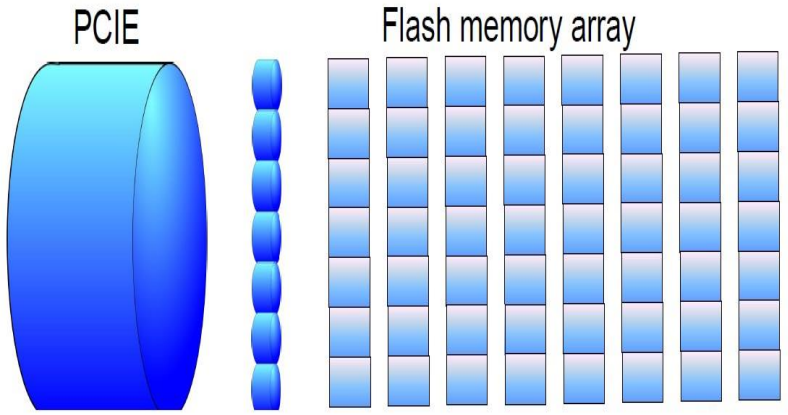
- Huge Capacity and it’s growing
- Supports deep queues (64K commands per queue, up to 64K queues)
- Hi Speed - Supports Parallel access
- Cost Effective
- Predictable Reliability

NVMe Virtualization – Why?

- SSD's are essentially parallel
- several levels of parallelism can be exploited

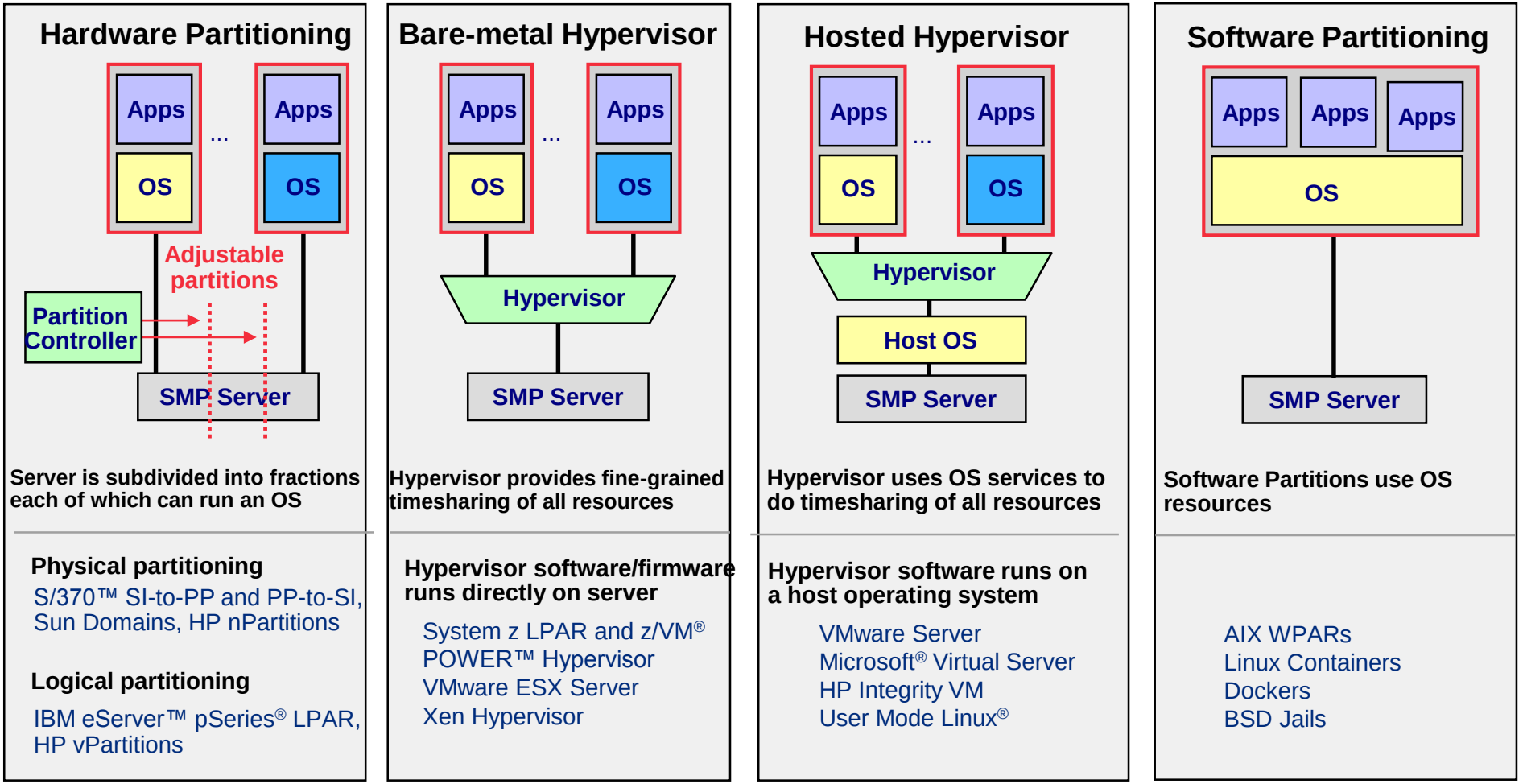


- **Channel-level parallelism**
- **Package-level parallelism**.
- **Chip-level parallelism**.
- **Plane-level parallelism**

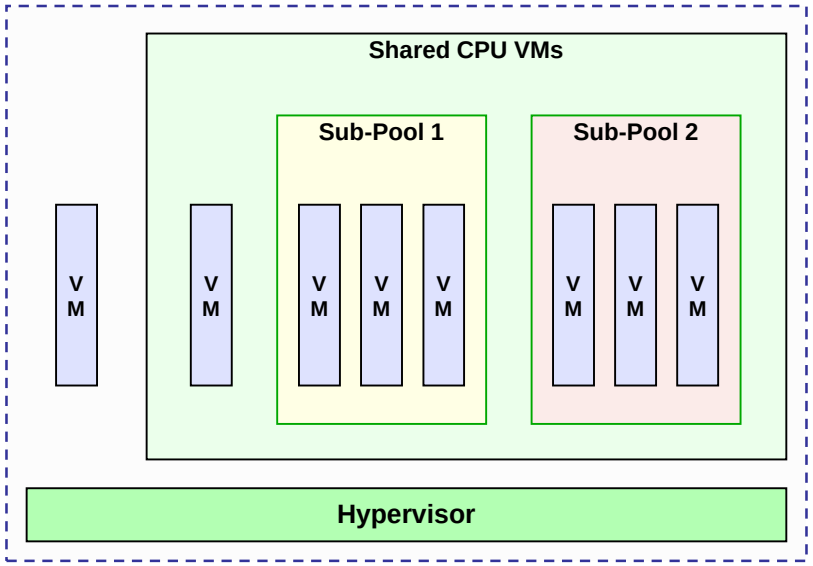


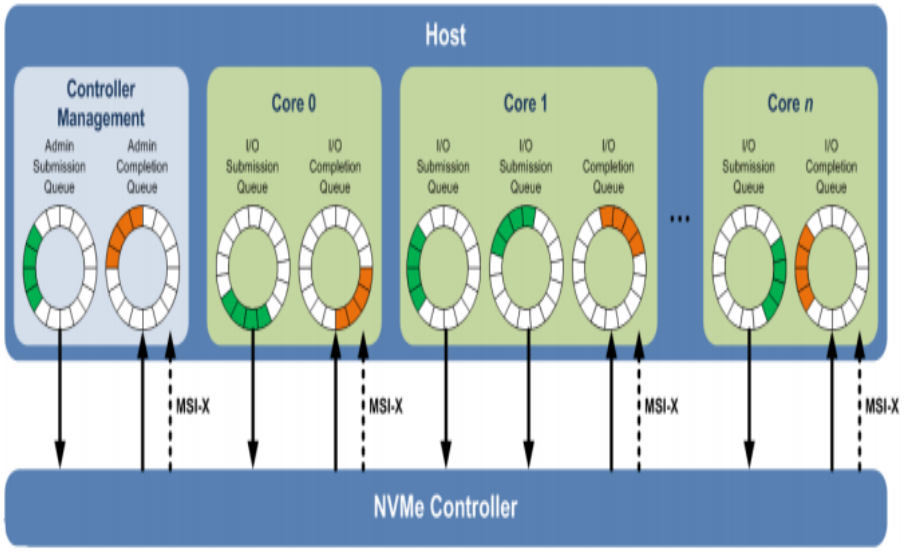
Different Virtualization Models

- Hardware Partitioning
- Bare Metal Hypervisor
- Hosted Hypervisor
- Software Partitioning



Virtualization and NVMe

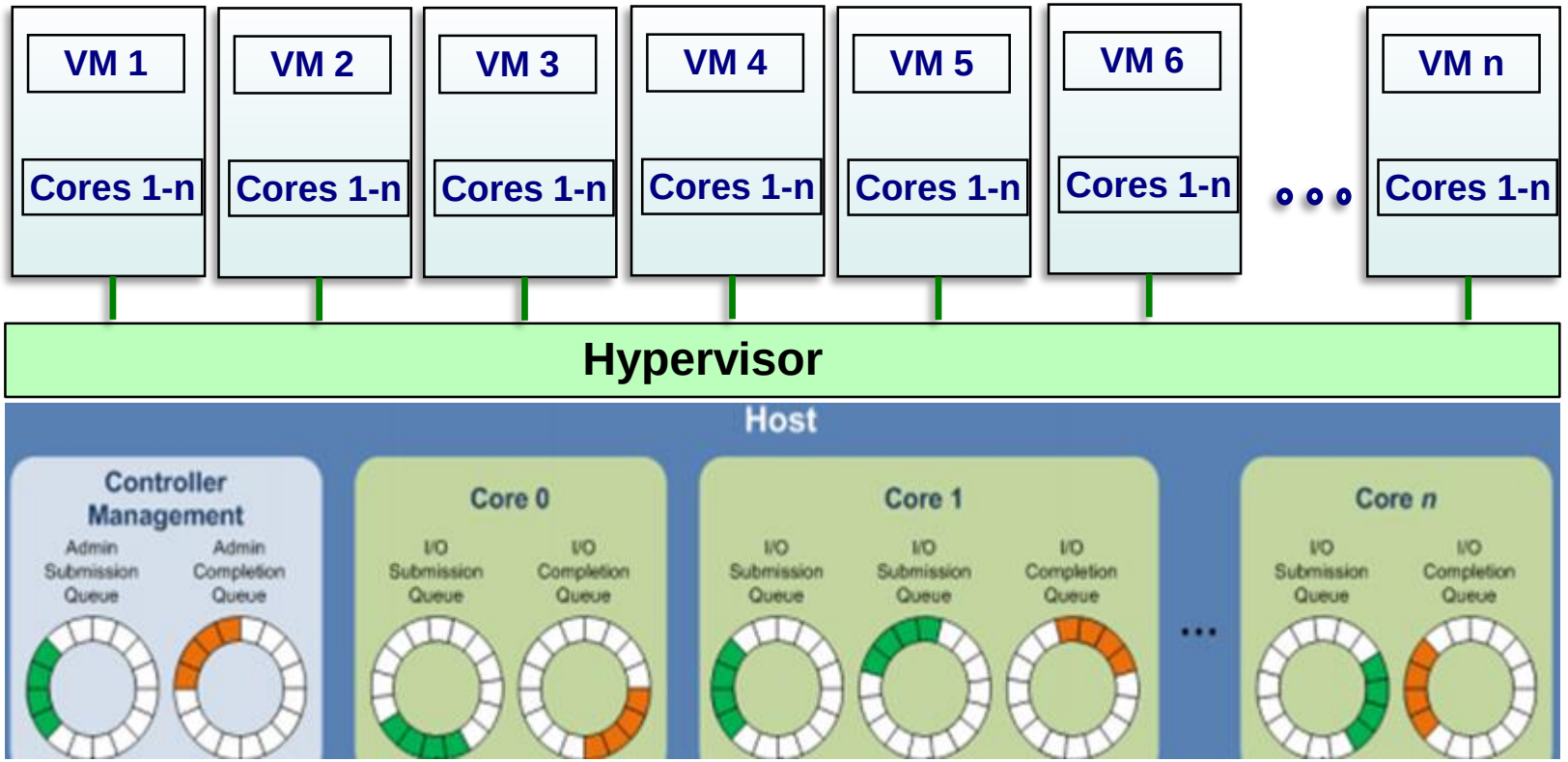




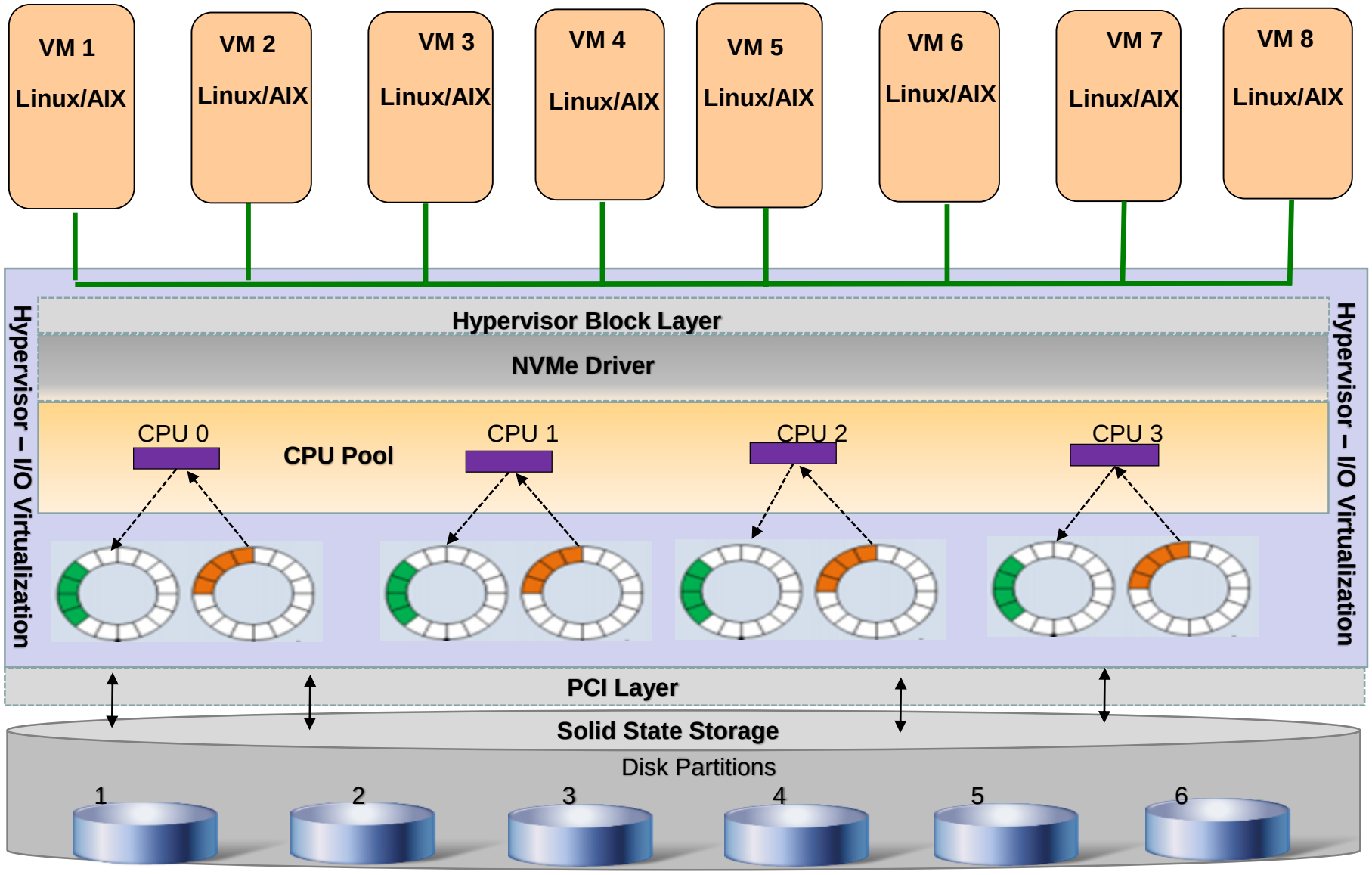
- CPU is a dispensable asset in Virtualization
- Dynamic Partition operations can migrate CPUs from one VM to another
- CPUs can be migrated across sub-pools
- Current NVMe implementations are tightly coupled to CPUs
- Goal of Virtualization is to have loosely coupled resources which could be shared across VMs

NVMe current Limitations in Virtualization

- Currently queuing is based on Host Partition Cores
- Separate cores are assigned to Host OS and Guest OS (Virtual Machines)
- Not able to utilize available NVMe bandwidth
- Unable to extend NVMe advantage to Virtual machines
  - Number of queues are limited by number of cores



NVMe Current Model in Virtualization Area



CPU Usage during Sequential Reads/Writes

CPU usage during sequential writes

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6242	root	20	0	0	0	0	D	8.0	0.0	0:32.41	kworker/u16:2
7531	root	20	0	8264	2304	1252	D	8.0	0.1	0:03.68	dd
7534	root	20	0	8264	2260	1216	D	7.6	0.1	0:03.72	dd
7535	root	20	0	8264	2184	1124	D	7.6	0.1	0:03.70	dd
7538	root	20	0	8264	2364	1312	D	7.6	0.1	0:03.74	dd
7530	root	20	0	8264	2308	1244	D	7.3	0.1	0:03.67	dd
7533	root	20	0	8264	2372	1316	D	7.3	0.1	0:03.70	dd
7537	root	20	0	8264	2308	1248	D	7.3	0.1	0:03.75	dd
7532	root	20	0	8264	2368	1316	D	7.0	0.1	0:03.62	dd

CPU usage during sequential reads

7664	root	20	0	8264	1384	328	D	45.9	0.0	0:32.03	dd
7660	**root**	**20**	**0**	**8264**	**1376**	**332**	**R**	**45.2**	**0.0**	**0:31.65**	**dd**
7663	root	20	0	8264	1400	328	D	45.2	0.0	0:32.24	dd
7665	root	20	0	8264	1388	320	D	45.2	0.0	0:31.87	dd
7661	root	20	0	8264	1392	328	D	44.9	0.0	0:31.41	dd
7666	root	20	0	8264	1400	328	D	44.9	0.0	0:31.88	dd
7667	root	20	0	8264	1388	336	D	44.9	0.0	0:31.54	dd
7662	root	20	0	8264	1388	336	D	44.5	0.0	0:31.89	dd

CPU lead or saturated queues?

- Context switch at IO itself can be CPU intensive now.

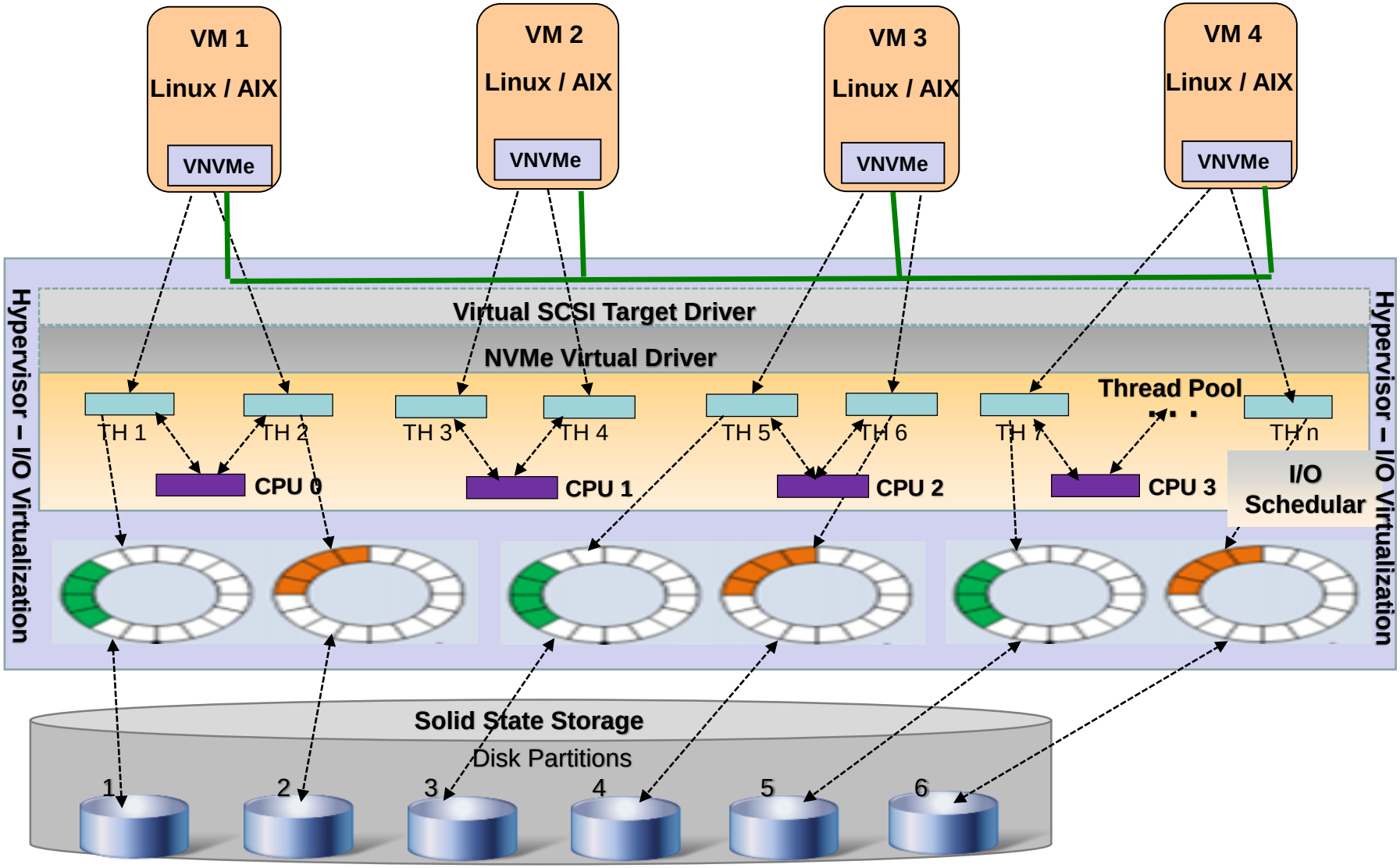
```
[51498.097744] NVME: submit sync
[51498.097748] NVME: START IO:
[51498.097927] NVME : IO COMPLETE
[51498.098889] NVME: submit sync
[51498.098895] NVME: START IO:
[51498.099074] NVME : IO COMPLETE
[51498.099123] NVME: submit sync
[51498.099127] NVME: START IO:
[51498.099293] NVME : IO COMPLETE
[51498.099478] NVME: submit sync
[51498.099484] NVME: START IO:
[51498.099654] NVME : IO COMPLETE
[51498.099729] NVME: START IO:
[51498.099734] NVME: SUBMIT IO
[51498.099818] NVME : IO COMPLETE
[51498.099853] NVME: START IO:
[51498.099856] NVME: SUBMIT IO
[51498.099938] NVME : IO COMPLETE
[51498.100108] NVME: START IO:
[51498.100112] NVME: SUBMIT IO
[51498.100250] NVME : IO COMPLETE
[51498.880144] NVME: Submit command
```

Time for this io .000089

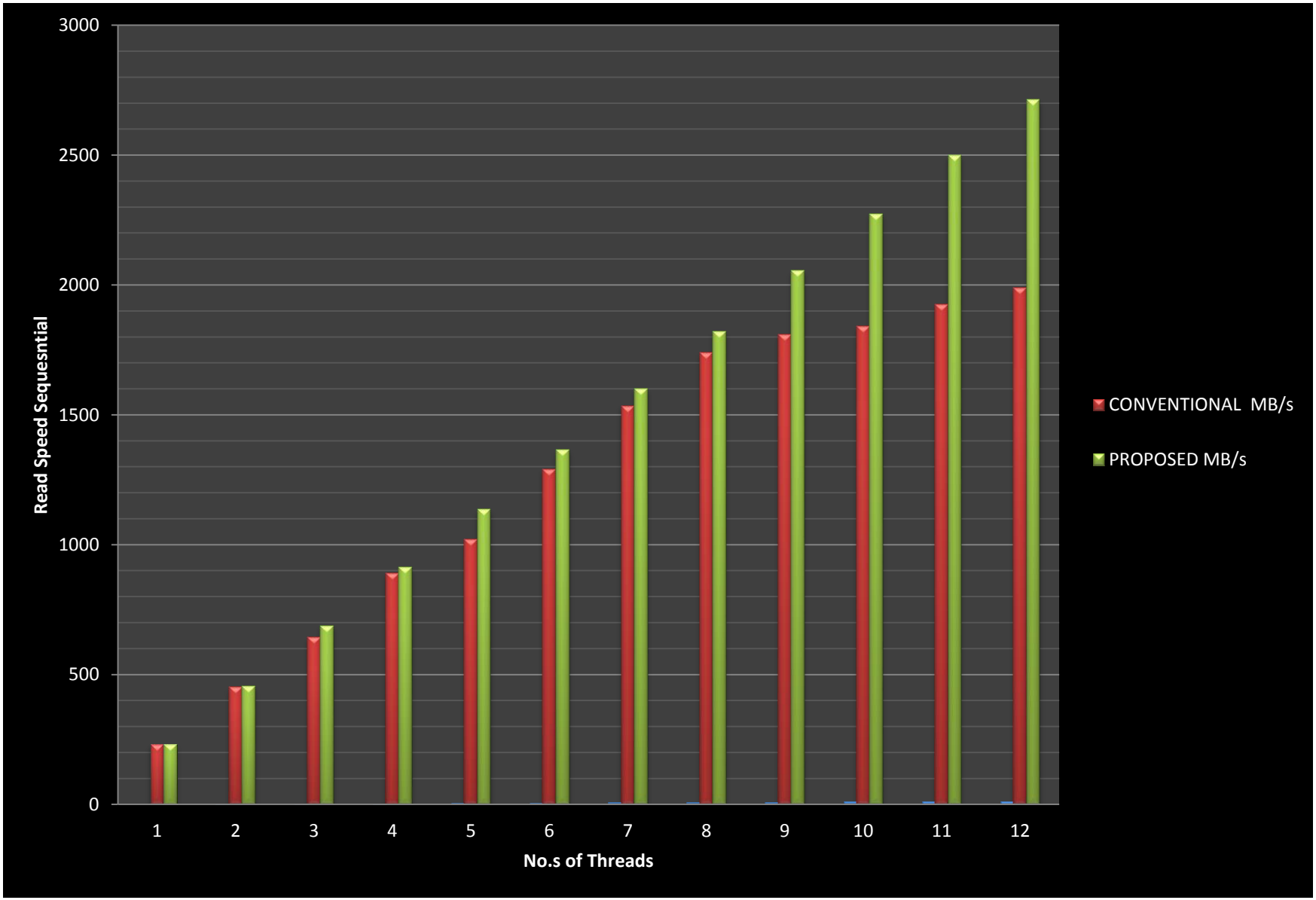
IO time .000082

- Delta time for IO's average 0.000085 seconds
- Continuous IO delta (IO breakup & Issue time) 0.000070 seconds
- Wise to release CPU amymore?

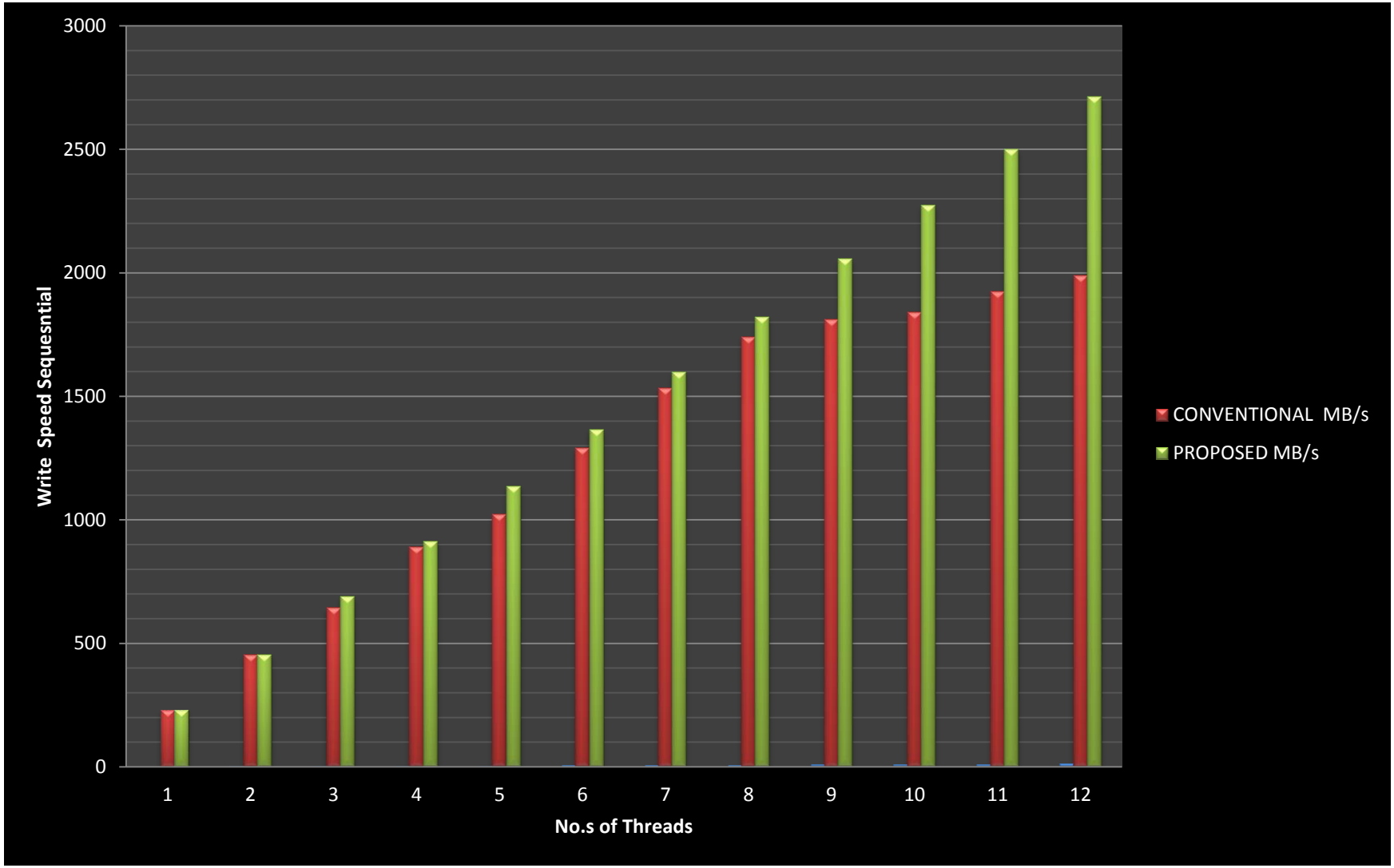
NVMe – Proposed Virtualization Model



NVMe Virtualization Performance - Reads



NVMe Virtualization Performance - Writes



Backup

NVMe Benefits

- **Lower latency:** Direct connection to CPU
- **Scalable performance:** 1 GB/s per lane – 4 GB/s, 8 GB/s, ...in one SSD
- **Industry standards:** NVM Express and PCI Express (PCIe) 3.0
- **Increased I/O:** Up to 40 PCIe lanes per CPU socket
- **Security protocols:** Trusted Computing Group Opal
- **Low Power features:** Low power link (L1.2), NVMe power states
- **Form factors:** SFF-8639, SATA Express* , M.2, Add in card, Future: BGA (PCI SIG)
- NVMe reduces latency overhead by more than 50%
  - SCSI/SAS: 6.0 µs 19,500 cycles
  - NVMe: 2.8 µs 9,100 cycles
- SATA supports 1 command with 32 Queues whereas NVMe supports 64000 commands with 64000 queues at a time

Thanks