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.
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

![Diagram showing parallelism at various levels in SSDs](image)
Different Virtualization Models

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

**Hardware Partitioning**

- Server is subdivided into fractions each of which can run an OS
- Physical partitioning:
  - S/370™ SI-to-PP and PP-to-SI, Sun Domains, HP nPartitions
- Logical partitioning:
  - IBM eServer™ pSeries® LPAR, HP vPartitions

**Bare-metal Hypervisor**

- Hypervisor provides fine-grained timesharing of all resources
- Hypervisor software/firmware runs directly on server
- System z LPAR and z/VM®
  - POWER™ Hypervisor
  - VMware ESX Server
  - Xen Hypervisor

**Hosted Hypervisor**

- Hypervisor uses OS services to do timesharing of all resources
- Hypervisor software runs on a host operating system
- VMware Server
  - Microsoft® Virtual Server
  - HP Integrity VM
  - User Mode Linux®

**Software Partitioning**

- Software Partitions use OS resources
  - AIX WPARs
  - Linux Containers
  - Dockers
  - BSD Jails
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

```
VM 1
Cores 1-n

VM 2
Cores 1-n

VM 3
Cores 1-n

VM 4
Cores 1-n

VM 5
Cores 1-n

VM 6
Cores 1-n

VM n
Cores 1-n
```

Hypervisor
# CPU Usage during Sequential Reads/Writes

## CPU usage during sequential writes

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

## CPU usage during sequential reads

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>7664</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>8264</td>
<td>1384</td>
<td>328</td>
<td>D</td>
<td>45.9</td>
<td>0.0</td>
<td>0:32.03</td>
<td>dd</td>
</tr>
<tr>
<td>7660</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>8264</td>
<td>1376</td>
<td>332</td>
<td>R</td>
<td>45.2</td>
<td>0.0</td>
<td>0:31.65</td>
<td>dd</td>
</tr>
<tr>
<td>7663</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>8264</td>
<td>1400</td>
<td>328</td>
<td>D</td>
<td>45.2</td>
<td>0.0</td>
<td>0:32.24</td>
<td>dd</td>
</tr>
<tr>
<td>7665</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>8264</td>
<td>1388</td>
<td>320</td>
<td>D</td>
<td>45.2</td>
<td>0.0</td>
<td>0:31.87</td>
<td>dd</td>
</tr>
<tr>
<td>7661</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>8264</td>
<td>1392</td>
<td>328</td>
<td>D</td>
<td>44.9</td>
<td>0.0</td>
<td>0:31.41</td>
<td>dd</td>
</tr>
<tr>
<td>7666</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>8264</td>
<td>1400</td>
<td>328</td>
<td>D</td>
<td>44.9</td>
<td>0.0</td>
<td>0:31.88</td>
<td>dd</td>
</tr>
<tr>
<td>7667</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>8264</td>
<td>1388</td>
<td>336</td>
<td>D</td>
<td>44.9</td>
<td>0.0</td>
<td>0:31.54</td>
<td>dd</td>
</tr>
<tr>
<td>7662</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>8264</td>
<td>1388</td>
<td>336</td>
<td>D</td>
<td>44.5</td>
<td>0.0</td>
<td>0:31.89</td>
<td>dd</td>
</tr>
</tbody>
</table>
CPU lead or saturated queues?

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

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>51498.097744</td>
<td>NVME: submit sync</td>
</tr>
<tr>
<td>51498.097748</td>
<td>NVME: START IO:</td>
</tr>
<tr>
<td>51498.097927</td>
<td>NVME : IO COMPLETE</td>
</tr>
<tr>
<td>51498.098889</td>
<td>NVME: submit sync</td>
</tr>
<tr>
<td>51498.098895</td>
<td>NVME: START IO:</td>
</tr>
<tr>
<td>51498.099074</td>
<td>NVME : IO COMPLETE</td>
</tr>
<tr>
<td>51498.099123</td>
<td>NVME: submit sync</td>
</tr>
<tr>
<td>51498.099127</td>
<td>NVME: START IO:</td>
</tr>
<tr>
<td>51498.099293</td>
<td>NVME : IO COMPLETE</td>
</tr>
<tr>
<td>51498.099478</td>
<td>NVME: submit sync</td>
</tr>
<tr>
<td>51498.099484</td>
<td>NVME: START IO:</td>
</tr>
<tr>
<td>51498.099654</td>
<td>NVME : IO COMPLETE</td>
</tr>
<tr>
<td>51498.099729</td>
<td>NVME: START IO:</td>
</tr>
<tr>
<td>51498.099734</td>
<td>NVME: SUBMIT IO</td>
</tr>
<tr>
<td>51498.099818</td>
<td>NVME : IO COMPLETE</td>
</tr>
<tr>
<td>51498.099853</td>
<td>NVME: START IO:</td>
</tr>
<tr>
<td>51498.099856</td>
<td>NVME: SUBMIT IO</td>
</tr>
<tr>
<td>51498.099938</td>
<td>NVME : IO COMPLETE</td>
</tr>
<tr>
<td>51498.100108</td>
<td>NVME: START IO:</td>
</tr>
<tr>
<td>51498.100112</td>
<td>NVME: SUBMIT IO</td>
</tr>
<tr>
<td>51498.100250</td>
<td>NVME : IO COMPLETE</td>
</tr>
<tr>
<td>51498.880144</td>
<td>NVME: Submit command</td>
</tr>
</tbody>
</table>

- Delta time for IO’s average 0.000085 seconds
- Continuous IO delta (IO breakup & Issue time) 0.000070 seconds
- Wise to release CPU anymore?
NVMe Virtualization Performance - Writes

![Bar Chart]

- **Write Speed Sequential**
- **No.s of Threads**
- **CONVENTIONAL MB/s**
- **PROPOSED MB/s**
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