Software and Management for NVMe Session A12 Part B

3:40 to 4:45

An overview and new features targeting NVMe-MI 1.1	Austin Bolen Myron Loewen	Senior Principal Engineer, Dell EMC Platform Architect in NVM Solutions Group , Intel
New features in NVMe drivers Linux, Windows, and VMware	Uma Parepalli, Lee Prewitt Suds Jain, VMware Parag Maharana,	Senior Director, Stealth Mode Startup Principal Program Manager, Microsoft Vmware SSD Architect, Seagate
Storage Performance Development Kit and NVM Express	Jim Harris	Principal Engineer, Intel

NVMe-MI Enhancements

Austin Bolen
Dell EMC

Myron Loewen
Intel

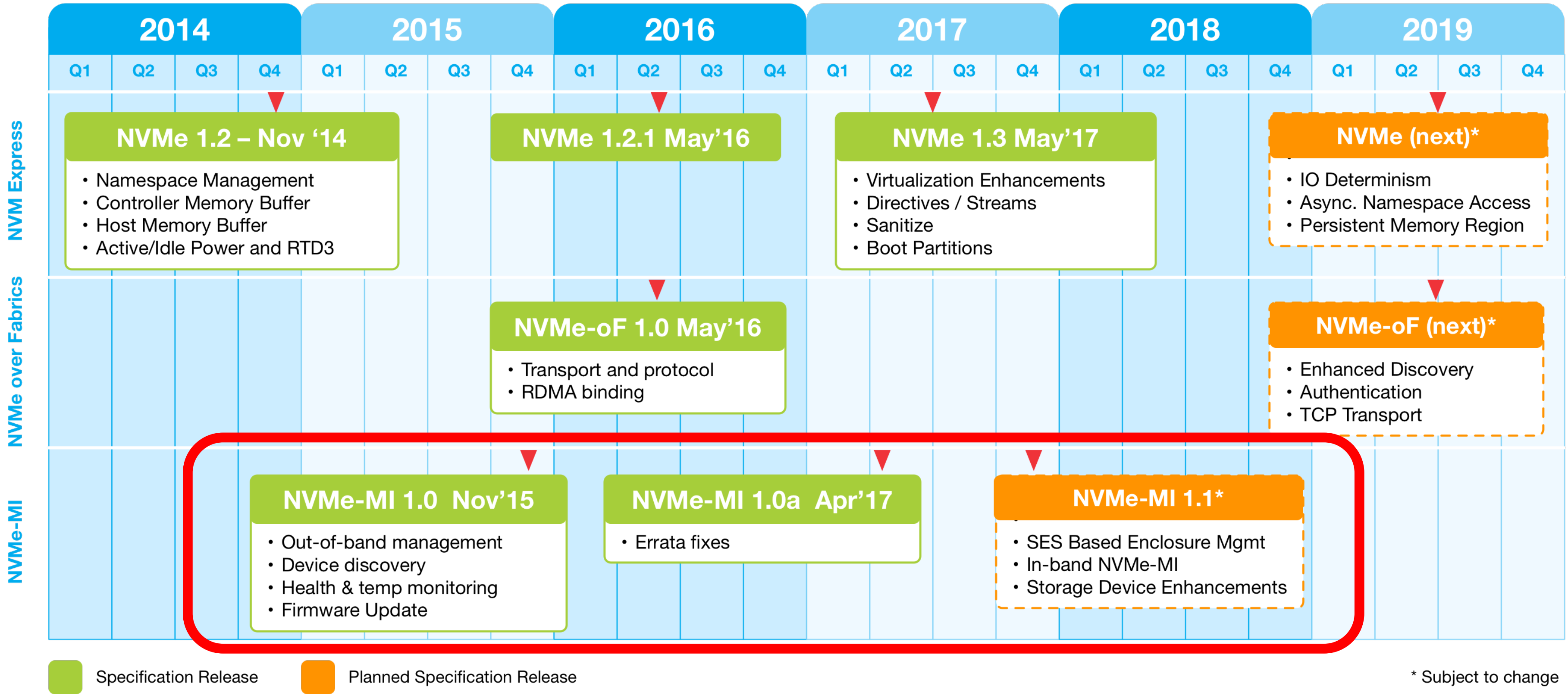
Peter Onufryk
Microsemi

Agenda

- NVMe-MI Workgroup Update
- NVMe-MI 1.0a Overview
- Proposed NVMe-MI 1.1 Major Features
  - In-Band NVMe-MI
  - NVMe-MI Enclosure Management
  - NVMe Storage Device Extension
- Summary

Flash Memory Summit

NVMe-MI Workgroup Update



NVMe 1.2 – Nov '14
- Namespace Management
- Controller Memory Buffer
- Host Memory Buffer
- Active/Idle Power and RTD3

NVMe 1.2.1 May'16

NVMe 1.3 May'17
- Virtualization Enhancements
- Directives / Streams
- Sanitize
- Boot Partitions

NVMe (next)*
- IO Determinism
- Async. Namespace Access
- Persistent Memory Region

NVMe-oF 1.0 May'16
- Transport and protocol
- RDMA binding

NVMe-oF (next)*
- Enhanced Discovery
- Authentication
- TCP Transport

NVMe-MI 1.0 Nov'15
- Out-of-band management
- Device discovery
- Health & temp monitoring
- Firmware Update

NVMe-MI 1.0a Apr'17
- Errata fixes

NVMe-MI 1.1*
- SES Based Enclosure Mgmt
- In-band NVMe-MI
- Storage Device Enhancements

🟩 Specification Release 🟧 Planned Specification Release

* Subject to change

NVMe-MI 1.0a Overview

NVMe Management Interface 1.0a

What is the NVMe Management Interface 1.0a?

- A programming interface that allows *out-of-band* *management* of an NVMe *Field Replaceable Unit* (FRU) or an embedded NVMe NVM Subsystem

Management Fundamentals

What is meant by “management”?

Four pillars of systems management:

- Inventory
- Configuration
- Monitoring
- Change Management

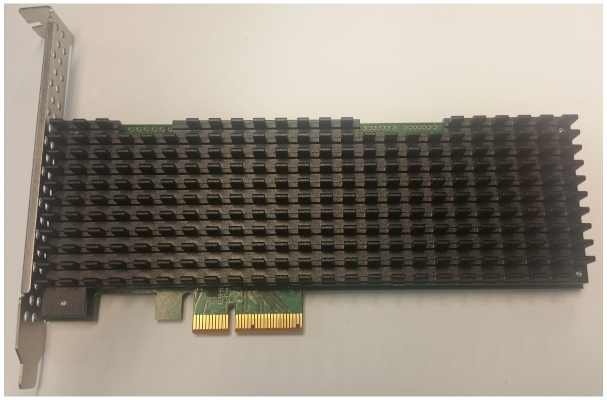
Management operational times:

- Deployment (No OS)
- Pre-OS (e.g. UEFI/BIOS)
- Runtime
- Auxiliary Power
- Decommissioning

Field Replaceable Unit (FRU)

FRU definition (Wikipedia):

- A circuit board, part or assembly that can be quickly and easily removed from a computer or other piece of electronic equipment, and replaced by the user or a technician without having to send the entire product or system to a repair facility.



Out-of-Band Definition

- Per MCTP Overview White Paper (DSP2016), version 1.0.0:
  - **Out-of-band**

    Management that operates with hardware resources and components that are ***independent of the operating systems control***.

- In NVMe-MI:
  - **Out-of-band**

    The out-of-band communication path for NVMe-MI is from a Management Controller (BMC) to a Management Endpoint (NVMe storage device) via:

    1. MCTP over SMBus/I2C
    2. MCTP over PCIe VDM
    3. IPMI FRU Data (VPD) access over SMBus/I2C per IPMI Platform Management FRU Information Storage Definition

In-Band Definition

- Per MCTP Overview White Paper (DSP2016), version 1.0.0:
  - **In-band**

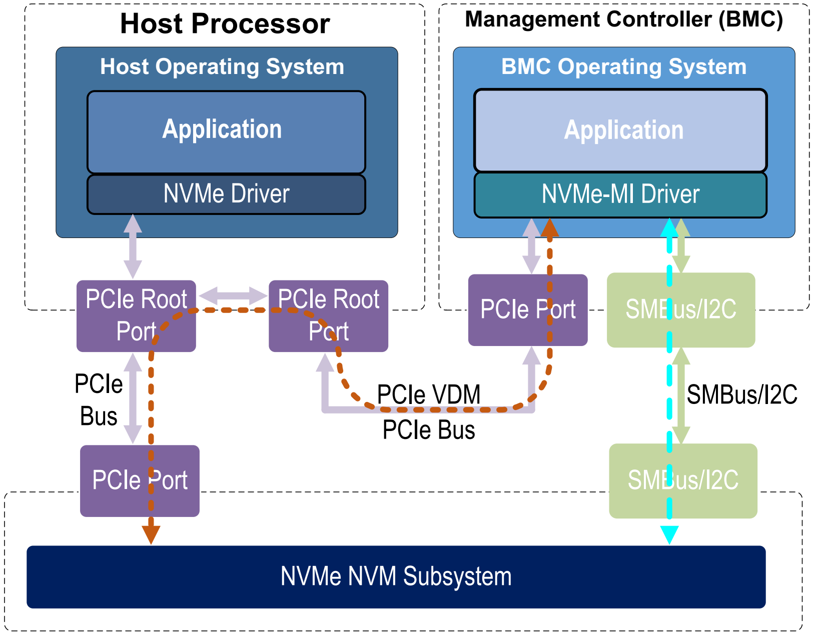
    Management that operates with the support of hardware components that are critical to and ***used by the operating system***.

    Note: The operating system reference here is the “host” operating system, not the BMC operating system.

- In NVMe-MI:
  - **In-band**

    The in-band communication path for NVMe-MI is from host software to an NVMe Controller via the NVMe Admin Queue using the NVMe-MI Send and NVMe-MI Receive commands.

NVMe-MI 1.0a (out-of-band)



- NVMe driver communicates to NVMe controllers over PCIe per NVMe Spec
- MC runs on its own OS on it own processor independent from host OS and driver
- Two OOB paths: PCIe VDM and SMBus
- PCIe VDMs are completely separate from in-band PCIe traffic though they share the same physical connection

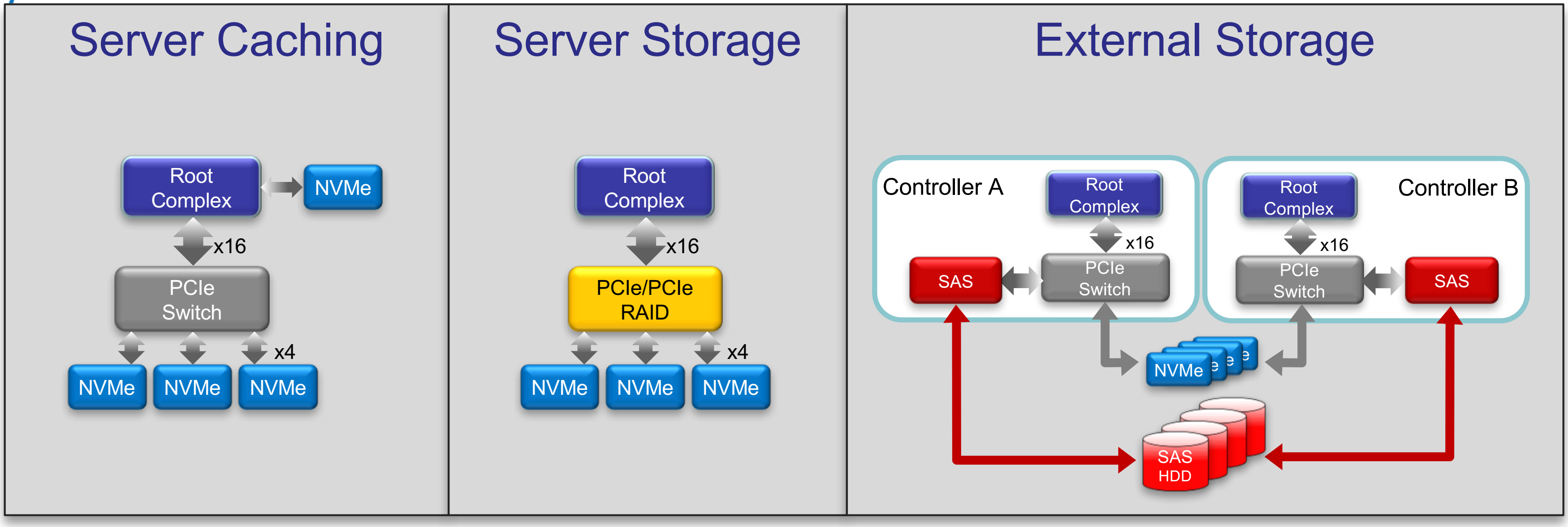
Out-of-Band Data Flow

⟷ **Out-of-Band: NVMe-MI over MCTP over PCIe VDM**

⟷ **Out-of-Band: NVMe-MI over MCTP over SMBus/I2C and VPD**

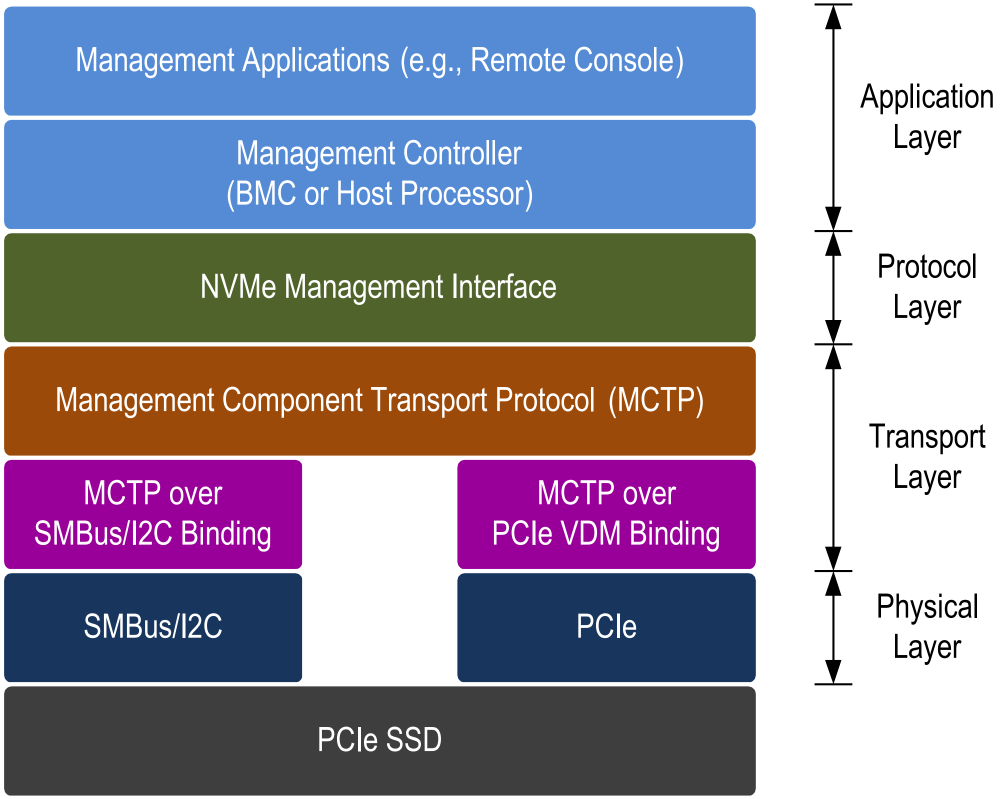
NVMe-MI 1.0a is out-of-band only

NVMe Storage Device Management



- Example Pre-boot Management
  - Inventory, Power Budgeting, Configuration, Firmware Update
- Example Out-of-Band Management During System Operation
  - Health Monitoring, Power/Thermal Management, Firmware Update, Configuration

NVMe-MI Protocol Layering



NVMe-MI 1.0a Command Set Overview

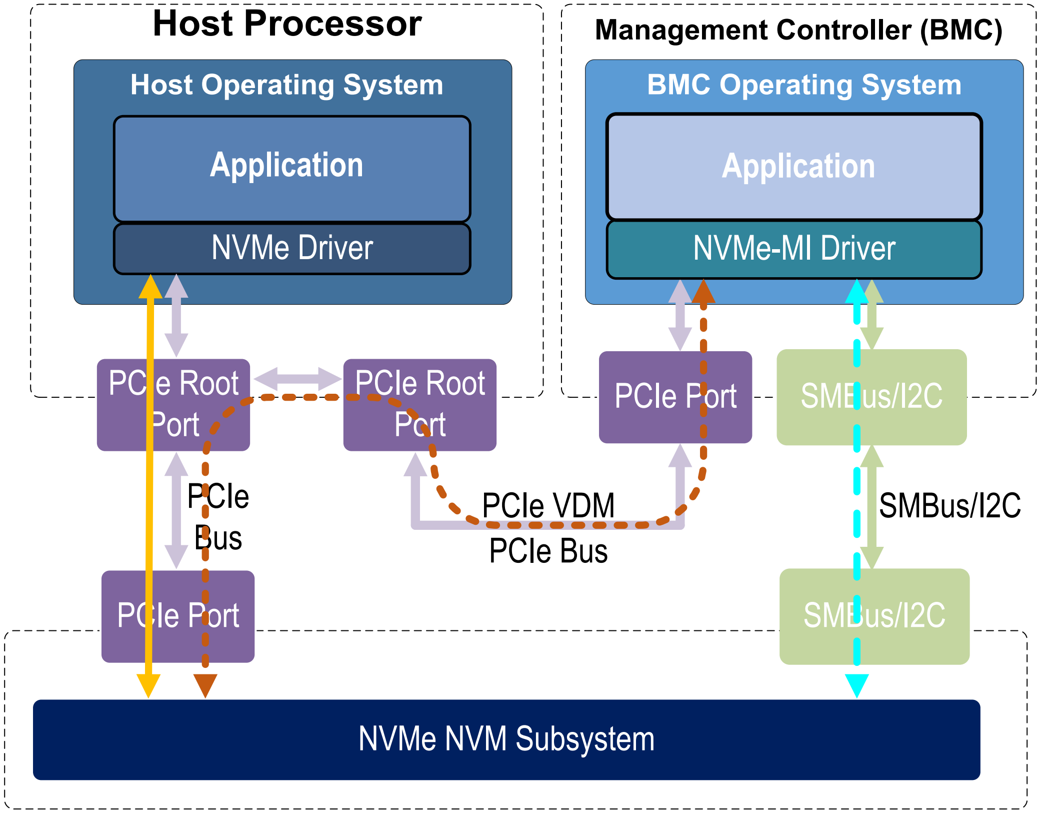
Command Type	Command
NVMe Management Interface Specific Commands	Read NVMe-MI Data Structure
	NVM Subsystem Health Status Poll
	Controller Health Status Poll
	Configuration Get
	Configuration Set
	VPD Read
	VPD Write
	Reset
	…
PCIe Command	PCIe Configuration Read
	PCIe Configuration write
	PCIe I/O Read
	PCIe I/O Write
	PCIe Memory Read
	PCIe Memory Write
	…

Command Type	Command
NVMe Commands	Firmware Activate/Commit
	Firmware Image Download
	Format NVM
	Get Features
	Get Log Page
	Identify
	Namespace Management
	Namespace Attachment
	Security Send
	Security Receive
	Set Features
	…

In-band NVMe-MI

NVMe-MI 1.1 (out-of-band and in-band)

- NVMe-MI 1.1 adds in-band NVMe-MI tunnel
- NVMe-MI command tunneled using two new NVMe Admin Commands
  - NVMe-MI Send
  - NVMe-MI Receive



Out-of-Band and In-band Data Flow

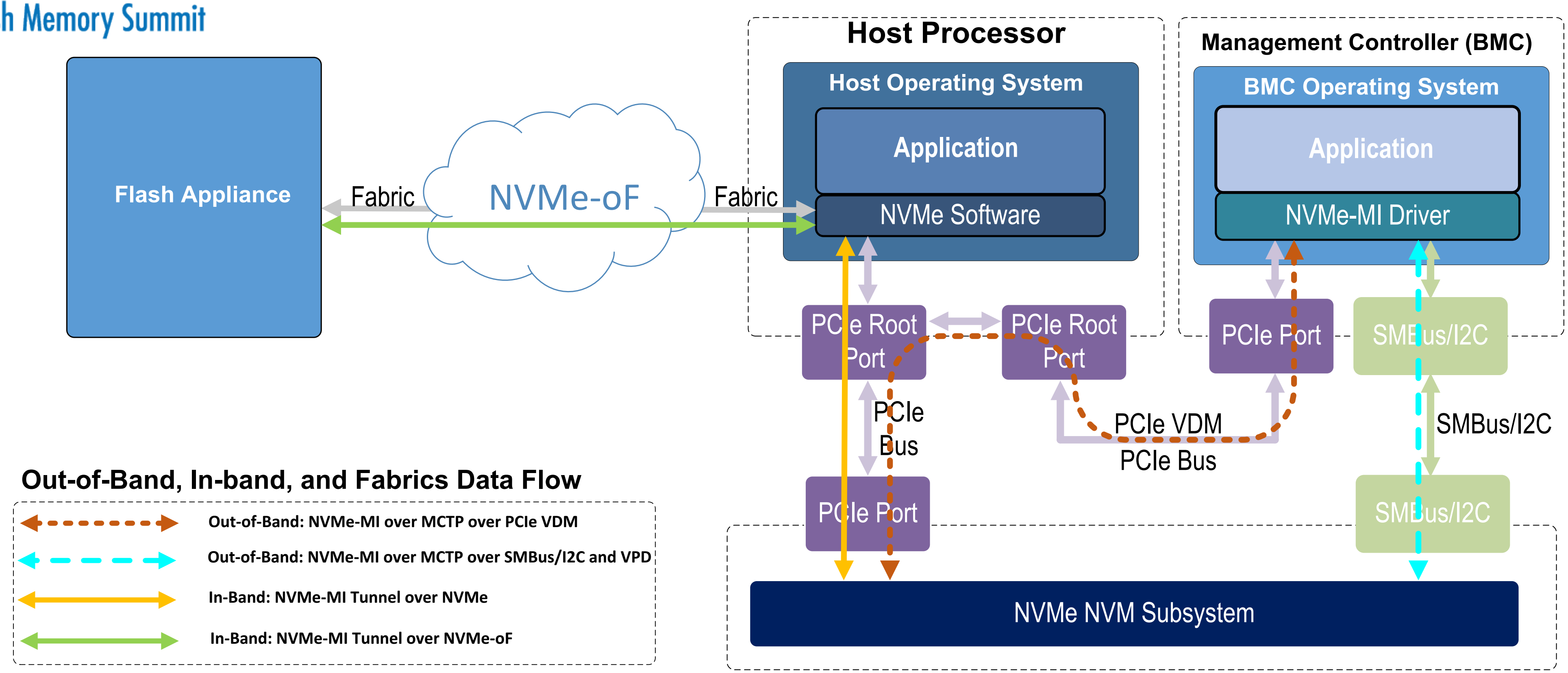
Out-of-Band: NVMe-MI over MCTP over PCIe VDM

Out-of-Band: NVMe-MI over MCTP over SMBus/I2C and VPD

In-Band: NVMe-MI Tunnel over NVMe

NVMe-MI 1.1 adds in-band NVMe-MI Tunnel

NVMe-MI over NVMe-oF



Out-of-Band, In-band, and Fabrics Data Flow

- Out-of-Band: NVMe-MI over MCTP over PCIe VDM
- Out-of-Band: NVMe-MI over MCTP over SMBus/I2C and VPD
- In-Band: NVMe-MI Tunnel over NVMe
- In-Band: NVMe-MI Tunnel over NVMe-oF

Plumbing in place for NVMe-MI over NVMe-oF

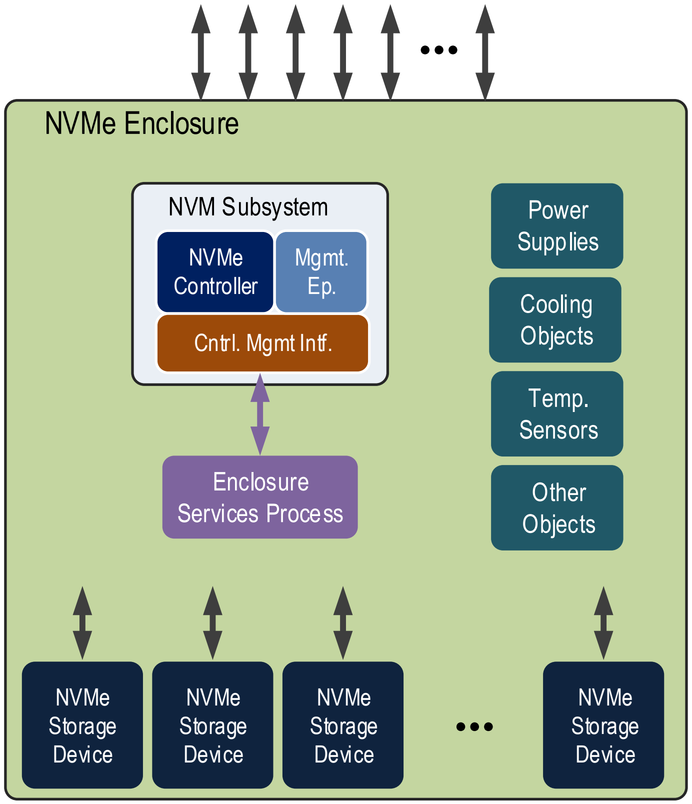
Benefits

- NVMe-MI offers features not available in-band via NVMe. For example:
  - Ability to manage NVMe at the FRU level
  - Vital Product Data (VPD) Access
  - Enclosure Management
- NVMe-MI in-band tunnel allows defining commands once in NVMe-MI and utilizing them out-of-band, in-band, and over fabrics.
- Allows NVMe Technical Workgroup to focus on non-management related work

Flash Memory Summit

Enclosure Management

Example Enclosure



Flash Memory Summit

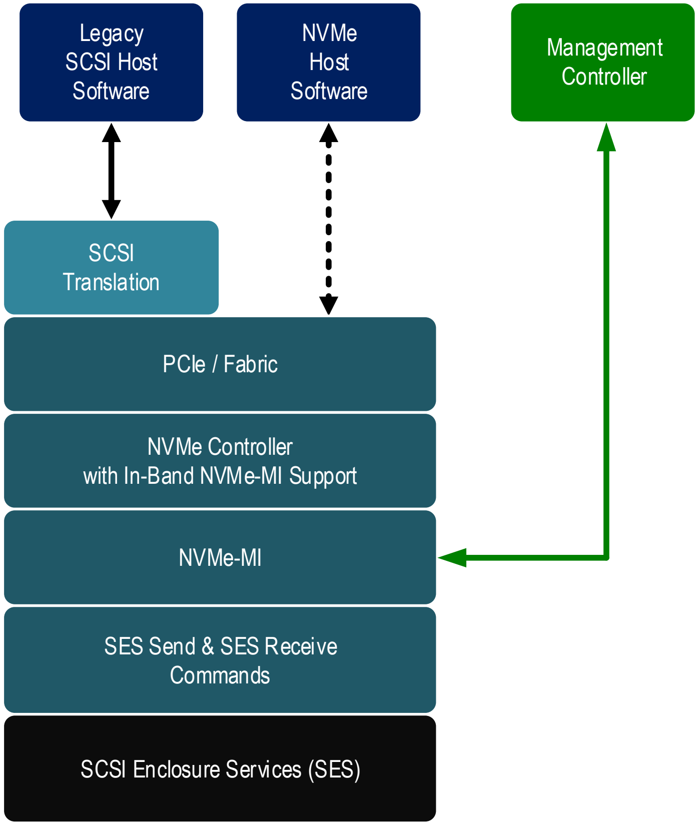
Enclosure Management

- Native PCIe Enclosure Management (NPEM)
  - Submission to PCI-SIG Protocol Workgroup (PWG) on behalf of the NVMe Management Interface Workgroup (NVMe-MI)
  - Transport specific basic management that is outside the scope of the NVMe-MI workgroup
- SES Based Enclosure Management
  - Technical proposal being developed in NVMe-MI workgroup
  - Comprehensive enclosure management

SES Based Enclosure Management

- Reuse NVMe drivers
- Reuse SCSI Enclosure Services (SES) developed by T10 for management of enclosures using the SCSI architecture
- While the NVMe and SCSI architectures differ, the elements of an enclosure and the capabilities required to manage these elements are the same
  - Example enclosure elements: power supplies, fans, display or indicators, locks, temperature sensors, current sensors, and voltage sensors
- NVMe-MI leverages SES for enclosure management
  - SES manages the elements of an enclosure using control and status diagnostic pages transferred using SCSI commands (SCSI SEND DIAGNOSTIC & SCSI RECEIVE DIAGNOSTIC RESULTS)
  - NVMe-MI uses these same control and status diagnostic pages, but transfers them using the SES Send and SES Receive commands.

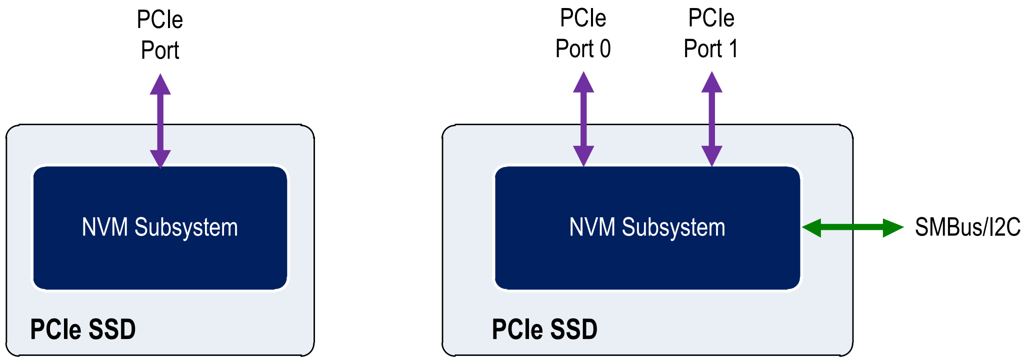
NVMe-MI SES Layering


Legacy SCSI Host Software
NVMe Host Software
Management Controller
SCSI Translation
PCIe / Fabric
NVMe Controller with In-Band NVMe-MI Support
NVMe-MI
SES Send & SES Receive Commands
SCSI Enclosure Services (SES)

Flash Memory Summit

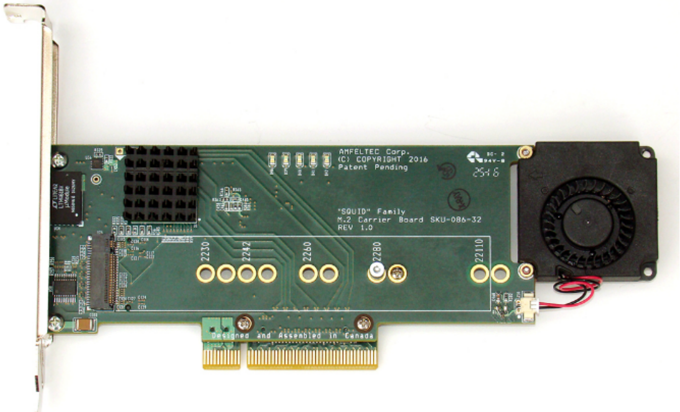
NVM Storage Device Enhancement

Original NVMe Storage Devices

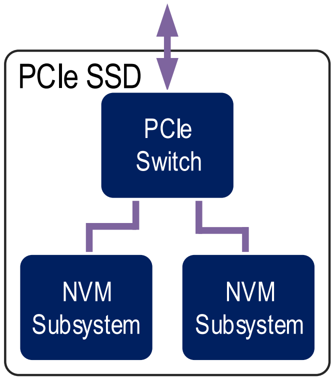


- An NVMe Storage Device consists of one NVM Subsystem with
  - One or more PCIe ports
  - An optional SMBus/I2C interface

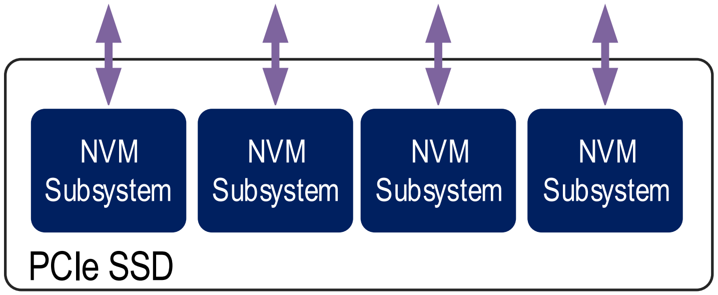
New Multi Sub System NVMe Devices



M.2 Carrier Board from Amfeltec

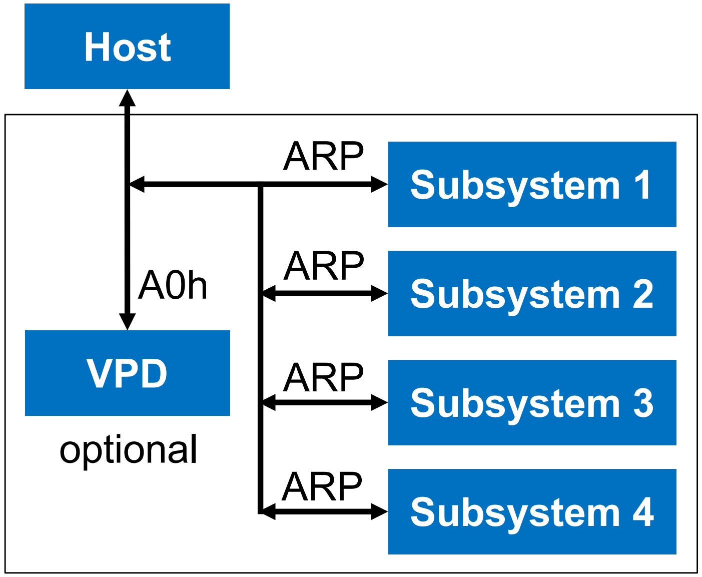


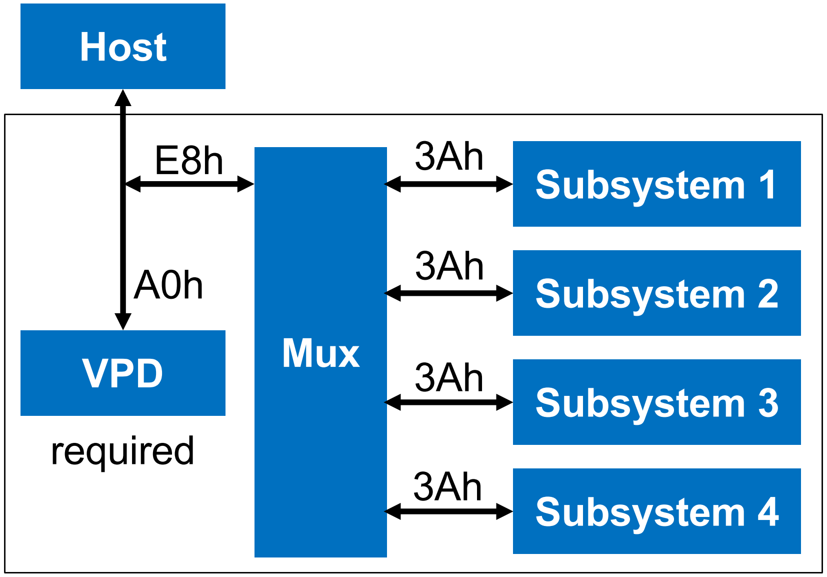
ANA Carrier Board from Facebook



SMBus Topologies

- Multiple subsystems on a single SMBus path
- ARP and Mux supported
- Scalable from 2 to 8 or more subsystems





Related Changes

- ARP changes
  - NVMe-MI specification to enable additional devices
  - PMBus/SMBus specification to add new default slave type
- VPD Updates in NVMe-MI
  - Indicate topology and details for mux
  - Optional temperature sensor on carrier board
  - Update for multiple ports

Flash Memory Summit

Summary

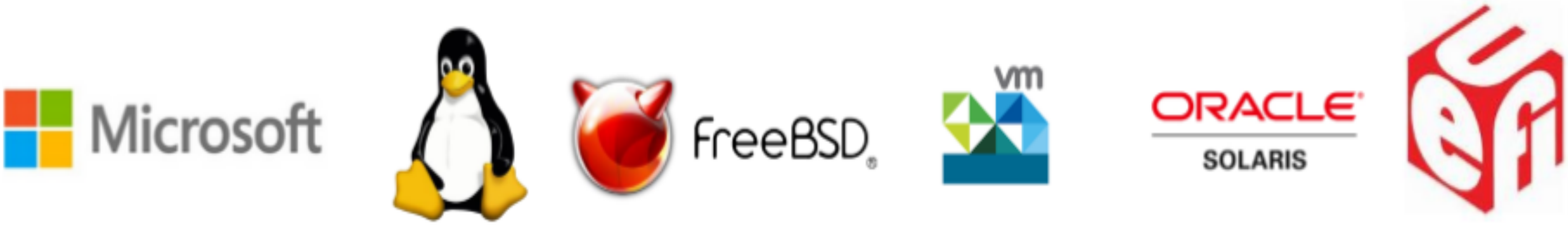
- NVMe-MI 1.0a has been released
  - Multiple NVMe devices passed the UNH-IOL NVMe-MI Compliance Tests and are shipping
  - Systems that support NVMe-MI 1.0a devices are shipping
- NVMe-MI 1.1 targeting release by end of 2017
  - In-band NVMe-MI
  - Enclosure Management
  - NVMe storage device enhancements

NVMe Device Drivers

Update and New Features

- Uma Parepalli, Lee Prewitt, Parag Maharana, Suds Jain

NVMe Ecosystem



http://www.nvmexpress.org/resources/drivers/

New Features in NVMe Drivers

- UEFI & Windows Community Drivers – Uma Parepalli
- Microsoft Windows – Lee Prewitt, Microsoft
- VMWare – Suds Jain, VMWare
- Linux – Parag Maharana, Seagate

NVMe UEFI Drivers

Uma Parepalli

NVMe UEFI Drivers

- Non-blocking I/O - Added NVMe Read/Write Block I/O protocols to issue command and poll/signal when transfer is complete.
- ARM platforms support NVMe at UEFI level.
- Namespace support at UEFI level is available through some implementations.
- NVMe UEFI diagnostics drivers available.

NVMe Windows Community Driver

Uma Parepalli

NVMe Windows Community Driver

- Latest Rev.1.5.0.0 released in in Dec 2016.
- Separate from Microsoft inbox drivers.
- Maintained by dedicated engineers.
- Hosted on OFA site.
- Heavily used by some OEMs and IHVs for custom test & debug.

NVMe Windows Community Driver

- Namespace Management (Create, Delete, Attach, Detach)
- EOL Read Only Support
- Win 8.1 Timers
- Surprise Removal Support in IOCTL Path
- Disk Initialization Performance Optimization

NVMe Windows Community Driver

- Storage Request Block Support
- StorPort Performance Options
- StorPort DPC Redirection
- Security Send/Receive with Zero Data Length
- SNTI updates for SCSI to NVMe Translation

NVMe Windows Community Driver

- Includes additional bug fixes
- Performance improvement & robustness
- NVMe Spec rev 1.2 feature compliance
- Support for MSFT Windows 10, 8.1, 7, Server 2012 R2, 2012 and 2008 R2
- Support for both 32-bit & 64-bit

Flash Memory Summit

Windows Inbox NVMe Driver

Lee Prewitt

Microsoft

Agenda

- New Additions for Windows Creators Addition (RS2)
- New Additions for Fall Update (RS3)
- Futures

NVMe Additions for Windows Creators Addition (RS2)

- Host Memory Buffer enabled by default
- Firmware Update and Activate
- Performance tuning
- Power tuning

- New Additions for Fall Update (RS3)

- Timestamp (v 1.3)
- Firmware Update Granularity (v1.3)
- Namespace Optimal IO Boundary (v1.3)
- Asynchronous Events for Namespace Addition
- Pass-through support of vendor unique log pages, Device Self-test, Compare commands
- Support for Controller Fatal Status Flag
- Streams (for Azure)

Futures*

- NVMe SCSI Translation Reference
- IO Determinism

*Not plan of record

NVM Express in vSphere Environment

Sudhanshu (Suds) Jain

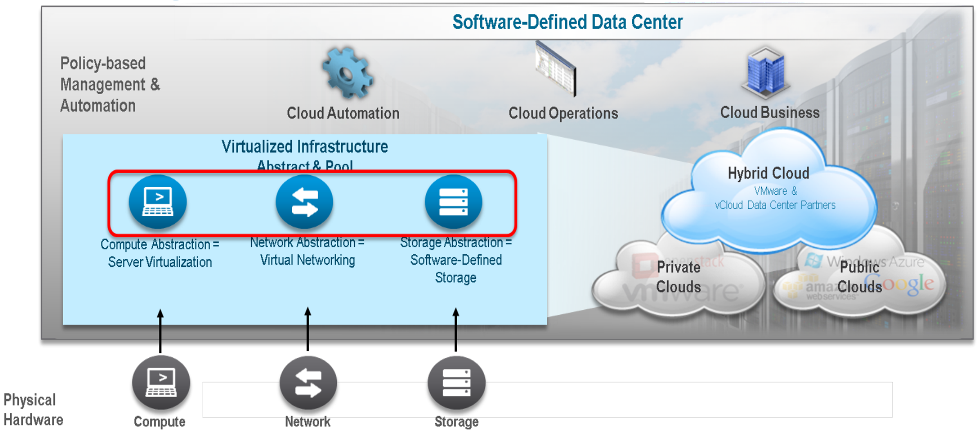
VMware

Agenda

- vSphere NVMe Driver EcoSystem

Flash Memory Summit

Software-Defined Infrastructure



Software-Defined Data Center
Policy-based Management & Automation
Cloud Automation
Cloud Operations
Cloud Business
Virtualized Infrastructure Abstract & Pool
Compute Abstraction = Server Virtualization
Network Abstraction = Virtual Networking
Storage Abstraction = Software-Defined Storage
Hybrid Cloud
VMware & vCloud Data Center Partners
Private Clouds
Public Clouds
Physical Hardware
Compute
Network
Storage

Flash and NVMe

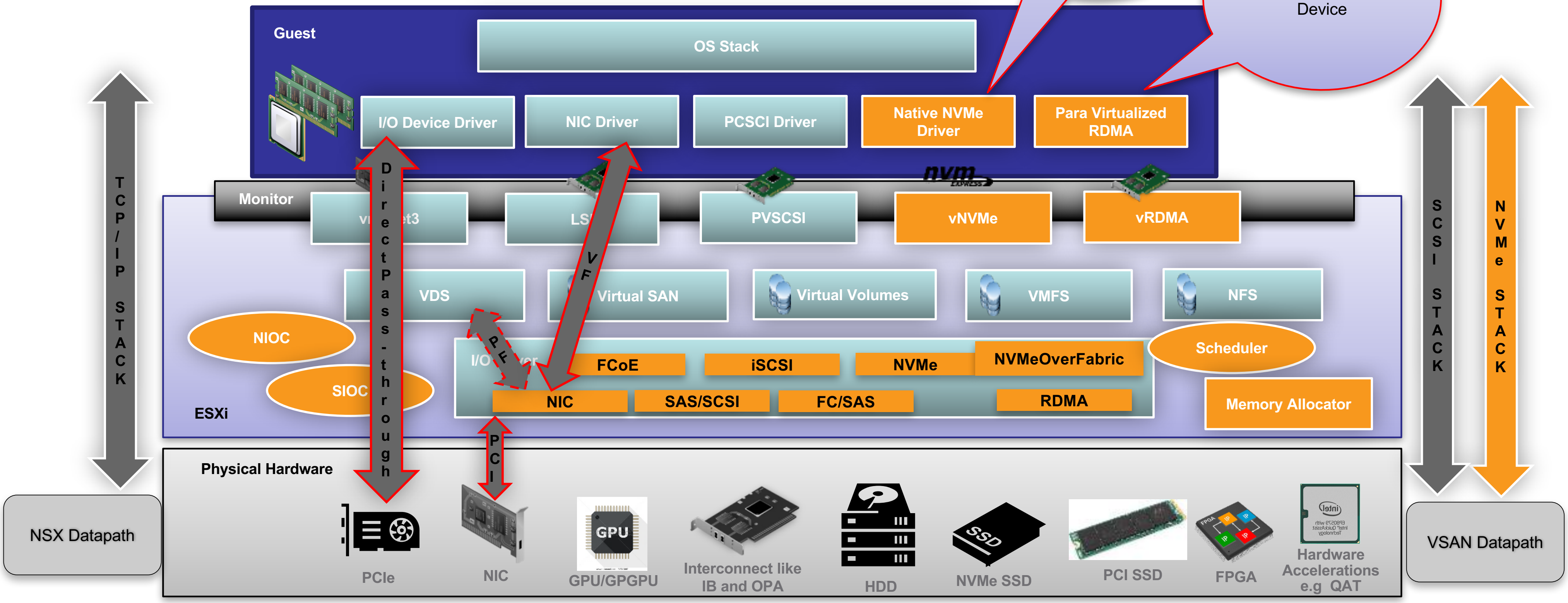
- A major focus area (moving forward)
- vSphere Flash Use-Cases: (KB 2145210)
  - Host swap cache
  - Regular Datastore
  - vSphere Flash Read Cache (aka Virtual Flash)
  - VSphere ESXi Boot Disk
  - VSphere ESXi Coredump device
  - VSphere ESXi Logging device
  - Virtual SAN (VSAN)

vmware

Flash Memory Summit

vSphere I/O Stack

Powering Your Applications & Infrastructure



Device Emulation

Para-Virtualized Device

Guest

OS Stack

I/O Device Driver

NIC Driver

PCSCI Driver

Native NVMe Driver

Para Virtualized RDMA

Monitor

vmxnet3

LSI

PVSCSI

vNVMe

vRDMA

VDS

Virtual SAN

Virtual Volumes

VMFS

NFS

NIOC

SIOC

I/O Driver

FCoE

iSCSI

NVMe

NVMeOverFabric

Scheduler

NIC

SAS/SCSI

FC/SAS

RDMA

Memory Allocator

ESXi

Direct Pass-through

VF

PF

PCI

TCP/IP STACK

SCSI STACK

NVMe STACK

NSX Datapath

VSAN Datapath

Physical Hardware

PCIe

NIC

GPU/GPGPU

Interconnect like IB and OPA

HDD

NVMe SSD

PCI SSD

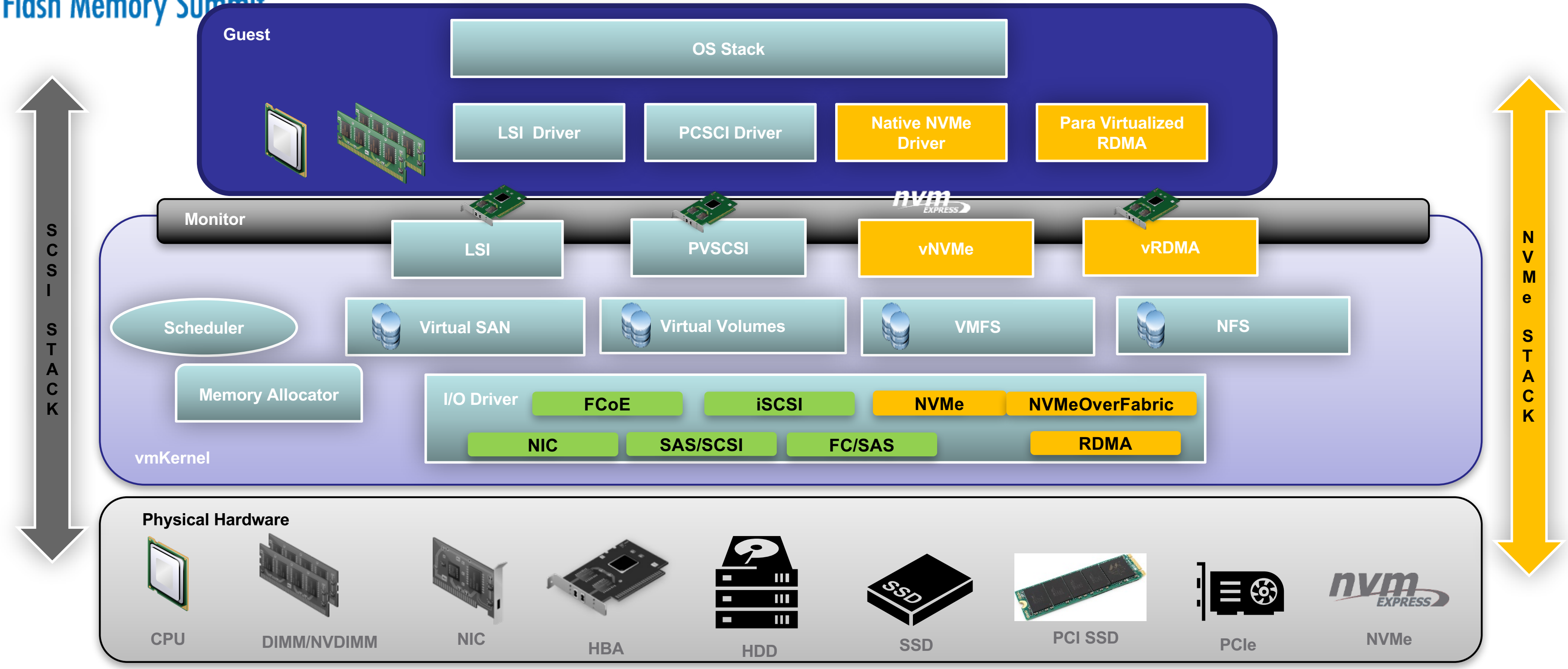
FPGA

Hardware Accelerations e.g QAT

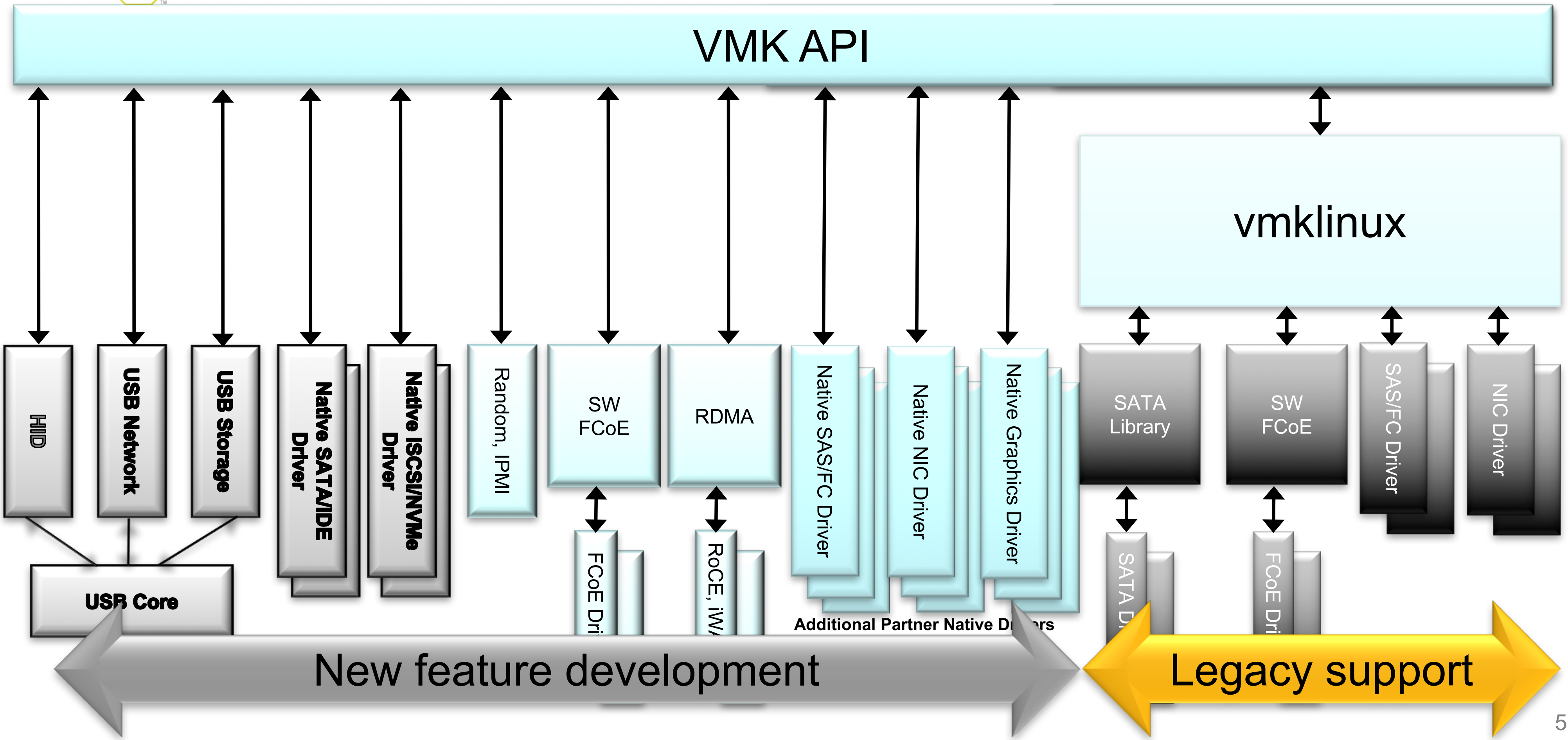
Flash Memory Summit

vmware

vSphere NVMe Native Driver Stack

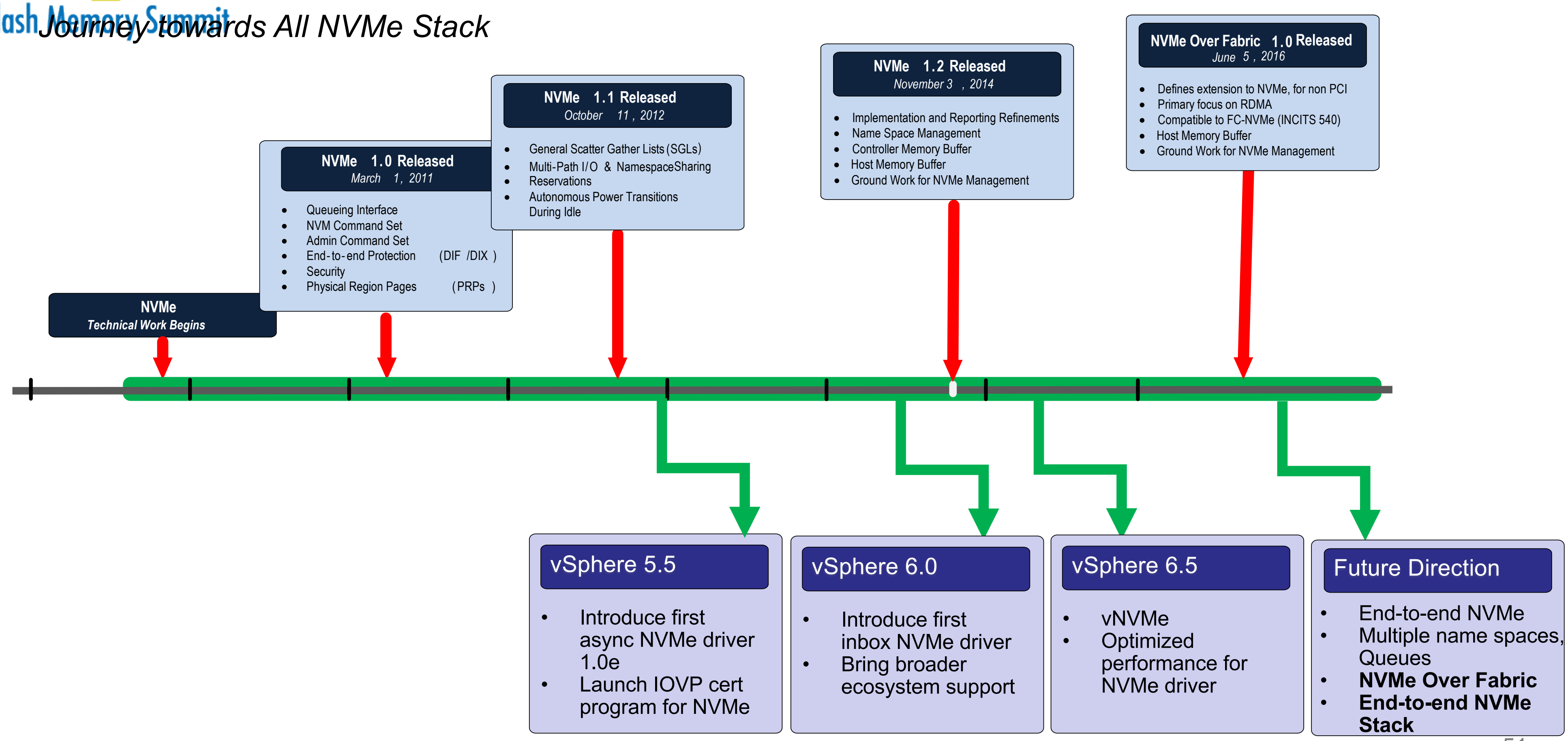


vSphere Driver Architecture Evolution



NVM Express Evolution and vSphere

Journey towards All NVMe Stack



NVMe
Technical Work Begins

NVMe 1.0 Released
March 1, 2011

- Queueing Interface
- NVM Command Set
- Admin Command Set
- End-to-end Protection (DIF /DIX)
- Security
- Physical Region Pages (PRPs)

NVMe 1.1 Released
October 11 , 2012

- General Scatter Gather Lists (SGLs)
- Multi-Path I/O & NamespaceSharing
- Reservations
- Autonomous Power Transitions During Idle

NVMe 1.2 Released
November 3 , 2014

- Implementation and Reporting Refinements
- Name Space Management
- Controller Memory Buffer
- Host Memory Buffer
- Ground Work for NVMe Management

NVMe Over Fabric 1.0 Released
June 5 , 2016

- Defines extension to NVMe, for non PCI
- Primary focus on RDMA
- Compatible to FC-NVMe (INCITS 540)
- Host Memory Buffer
- Ground Work for NVMe Management

vSphere 5.5

- Introduce first async NVMe driver 1.0e
- Launch IOVP cert program for NVMe

vSphere 6.0

- Introduce first inbox NVMe driver
- Bring broader ecosystem support

vSphere 6.5

- vNVMe
- Optimized performance for NVMe driver

Future Direction

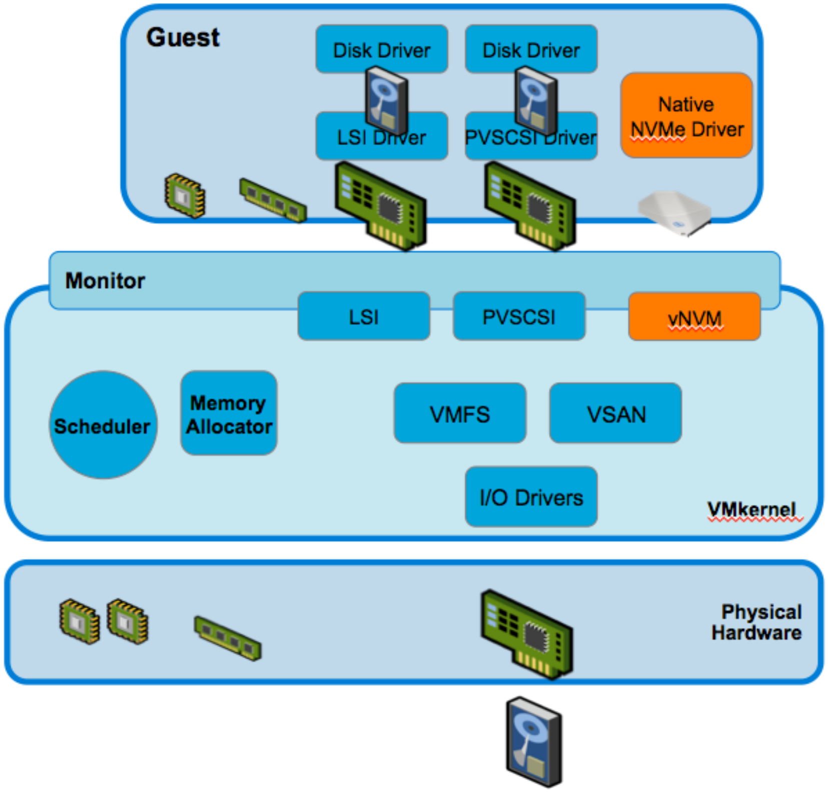
- End-to-end NVMe
- Multiple name spaces, Queues
- **NVMe Over Fabric**
- **End-to-end NVMe Stack**

NVMe Driver Ecosystem

- Available as part of base ESXi image from vSphere 6.0 onwards
  - Faster innovation with async release of VMware NVMe driver
- VMware led vSphere NVMe Open Source Driver project to encourage ecosystem to innovate
  - https://github.com/vmware/nvme
- Broad NVMe Ecosystem on VMware NVMe Driver https://www.vmware.com/resources/compatibility/search.php?deviceCategory=io
  - Close to 300 third party NVMe devices certified on VMware NVMe driver
- Also available for download (async) VMware ESXi 5.5 nvme 1.2.0.27-4vmw NVMe Driver for PCI Express based Solid-State Drives

Introducing Virtual NVMe

High Performance Guest Block I/O



New in vSphere 6.5

Feature:

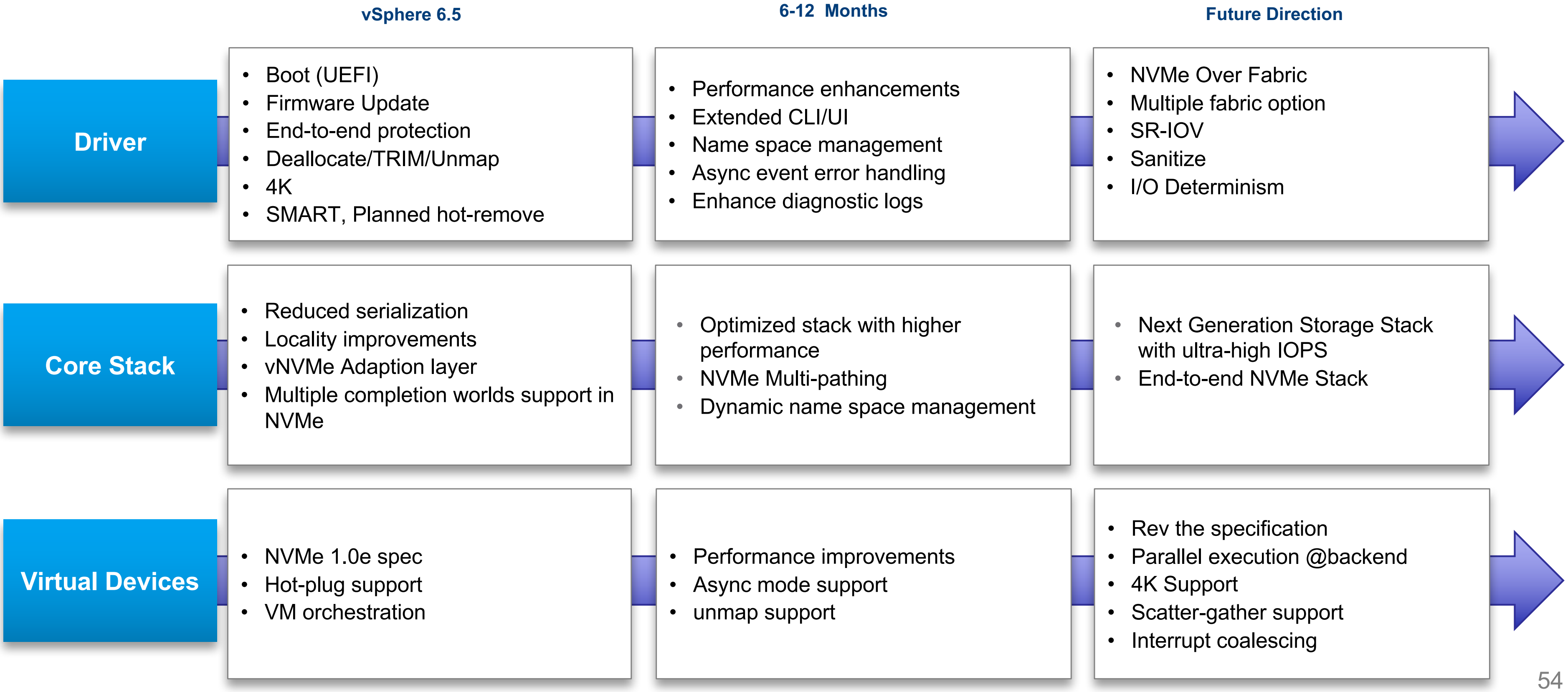
- ❖ NVMe 1.0e Device Emulation
- ❖ Works with inbox NVMe driver is various OS
- ❖ Hot add/remove support

Benefits:

- ❖ Improved application performance, better IOPS and latency numbers
- ❖ Leverage Native Stack from Guest OS (Linux, Windows…)

NVMe Focus @VMware

Summary



	vSphere 6.5	6-12 Months	Future Direction
Driver	• Boot (UEFI) • Firmware Update • End-to-end protection • Deallocate/TRIM/Unmap • 4K • SMART, Planned hot-remove	• Performance enhancements • Extended CLI/UI • Name space management • Async event error handling • Enhance diagnostic logs	• NVMe Over Fabric • Multiple fabric option • SR-IOV • Sanitize • I/O Determinism
Core Stack	• Reduced serialization • Locality improvements • vNVMe Adaption layer • Multiple completion worlds support in NVMe	• Optimized stack with higher performance • NVMe Multi-pathing • Dynamic name space management	• Next Generation Storage Stack with ultra-high IOPS • End-to-end NVMe Stack
Virtual Devices	• NVMe 1.0e spec • Hot-plug support • VM orchestration	• Performance improvements • Async mode support • unmap support	• Rev the specification • Parallel execution @backend • 4K Support • Scatter-gather support • Interrupt coalescing

Flash Memory Summit

NVMe Core and Fabrics Linux Drivers Update

Parag Maharana
SSD Architect
Seagate

NVMe Linux Drivers Overview

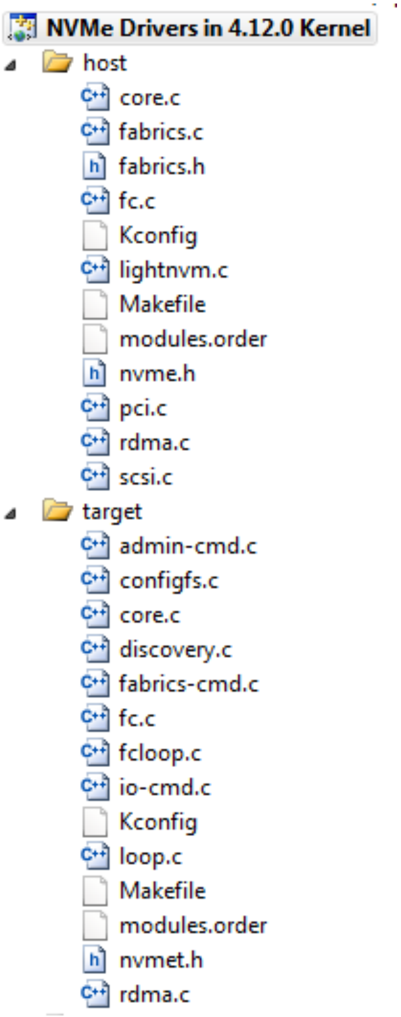
- Linux Core and Fabrics Drivers are based on Fabrics Spec 1.0 and Core Spec 1.2.1
- Linux Host driver is re-architected to support multiple transports (PCIe and Fabrics)
- Linux Fabrics Driver has Host and Target components:
  - Host has Core, PCIe and Fabric modules
  - Target components has Core and Fabric modules
  - Target side required new configuration tool (nvmetcli)
- Linux Fabrics Driver is part of Linux Kernel 4.8 from June'16

Implemented Features Previously

- NVMe Host Driver
  - Support for RDMA transport (Infiniband™/RoCE™/iWARP™/Intel OmniPath®)
  - Connect/Disconnect to multiple controllers
  - Transport of NVMe commands/data generated by NVMe core
  - Initial Discovery service implementation
  - Multi-Path
  - Keep Alive
- NVMe Target Driver
  - Support for mandatory NVMe and Fabrics commands
  - Support for multiple hosts/subsystems/controls/namespaces
  - Namespaces backed by <any> Linux block devices
  - Initial Discovery service; Discovery Subsystem/Controller(s)
  - Target Configuration interface using Linux configfs
    - Create NVM and Discovery Subsystems

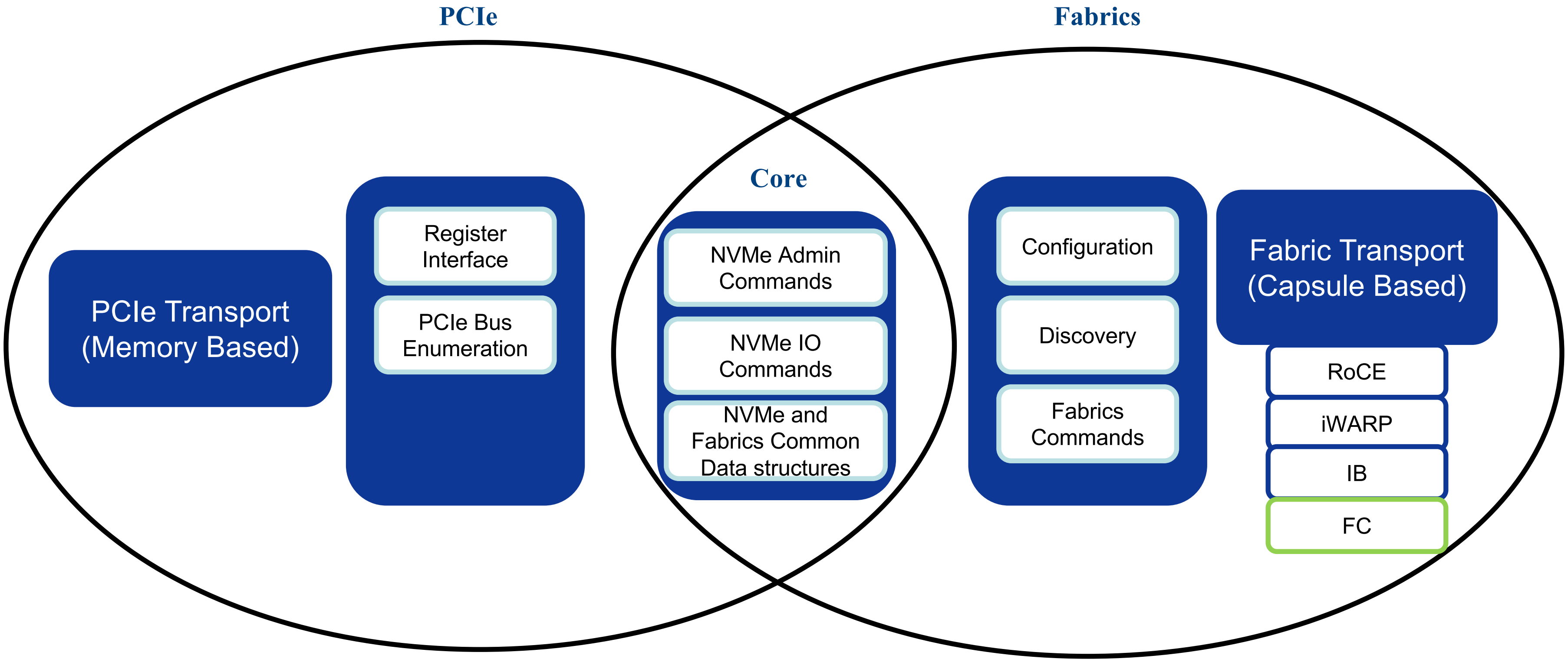
New Features

- NVMe Host Driver
  - Support for transport (FC)
  - Automated host multi-path (work in progress)
- NVMe Target Driver
  - Support FC Fabric transport
  - Log page support (smart log pages, error log pages, …)

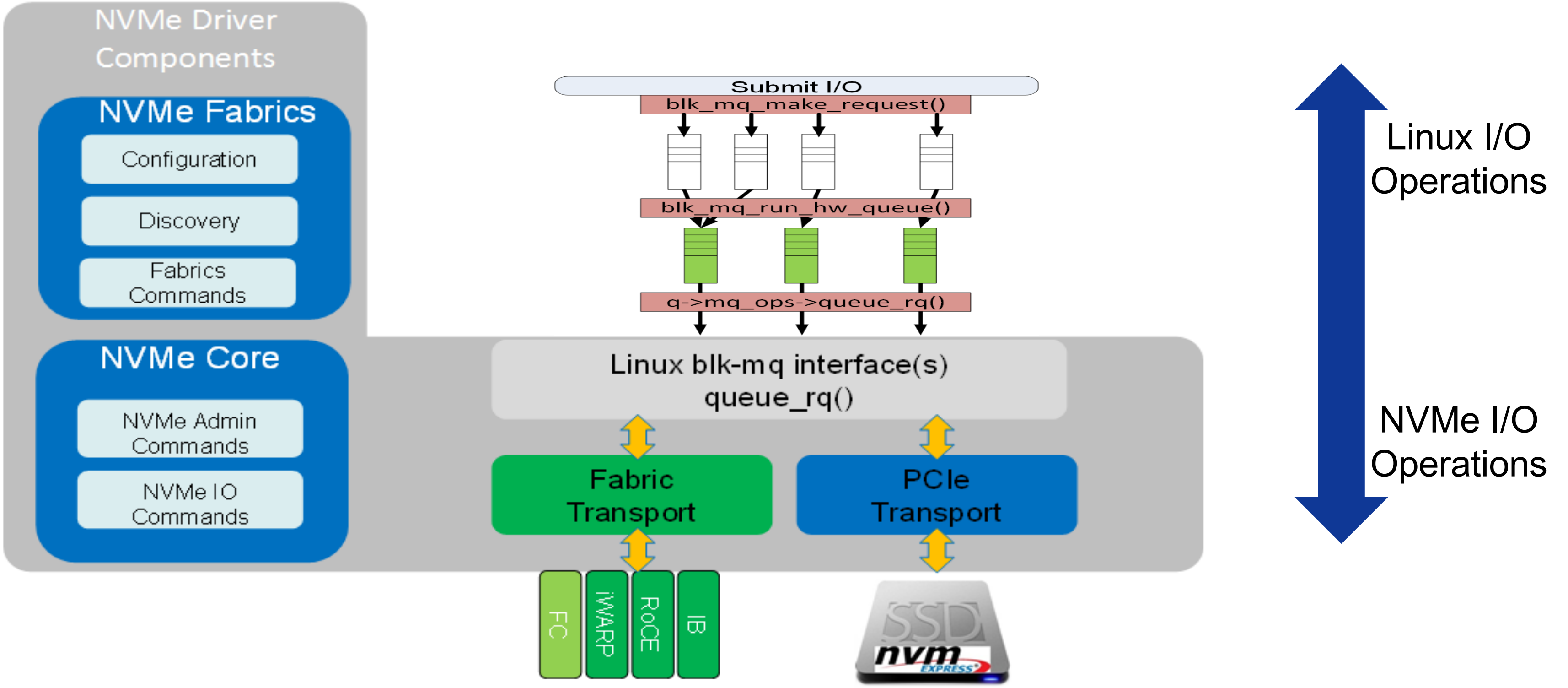


```
NVMe Drivers in 4.12.0 Kernel
◢ host
      core.c
      fabrics.c
      fabrics.h
      fc.c
      Kconfig
      lightnvm.c
      Makefile
      modules.order
      nvme.h
      pci.c
      rdma.c
      scsi.c
◢ target
      admin-cmd.c
      configfs.c
      core.c
      discovery.c
      fabrics-cmd.c
      fc.c
      fcloop.c
      io-cmd.c
      Kconfig
      loop.c
      Makefile
      modules.order
      nvmet.h
      rdma.c
```

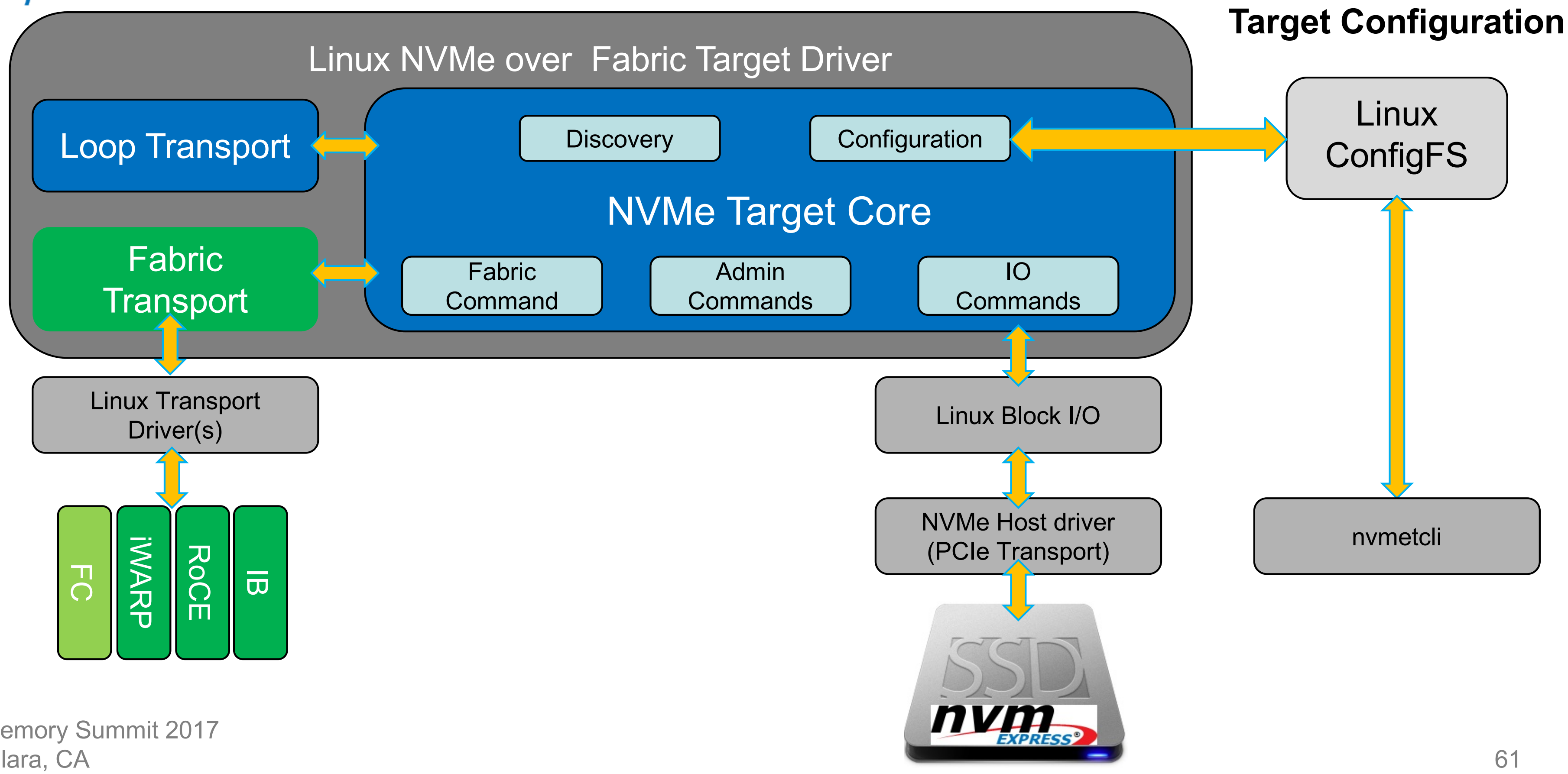
NVMe Over Fabrics Host and Target Driver Components



Host Driver Components



Target Driver Components



Linux Driver WG Next Steps

- Next steps
  - Fabric
    - Authentication features
    - Controller Memory Buffer
  - NVMe 1.3 complainant and New features
    - Directive Stream Support
    - Virtualization Support
    - Sanitize
  - IO Determinism
- Call for Action:
  - Download driver and try it out
  - Provide suggestion/comment/feedback
  - Suggest any future enhancement

Linux Driver Reference

- Linux Fabrics drivers
  - NVMe Specification
    - http://www.nvmexpress.org/specifications/
  - NVMe Fabric Driver Resource
    - http://www.nvmexpress.org/resources/nvme-over-fabrics-drivers/
  - NVMe Linux Fabric Drivers Source
    - www.kernel.org
  - NVMe-Cli (nvme) Source
    - http://github.com/linux-nvme/nvme-cli/
  - NVMe-Target-Cli (nvmetcli) Source
    - http://git.infradead.org/users/hch/nvmetcli.git
  - NVMe Linux Fabric Mailing List
    - linux-nvme@lists.infradead.org

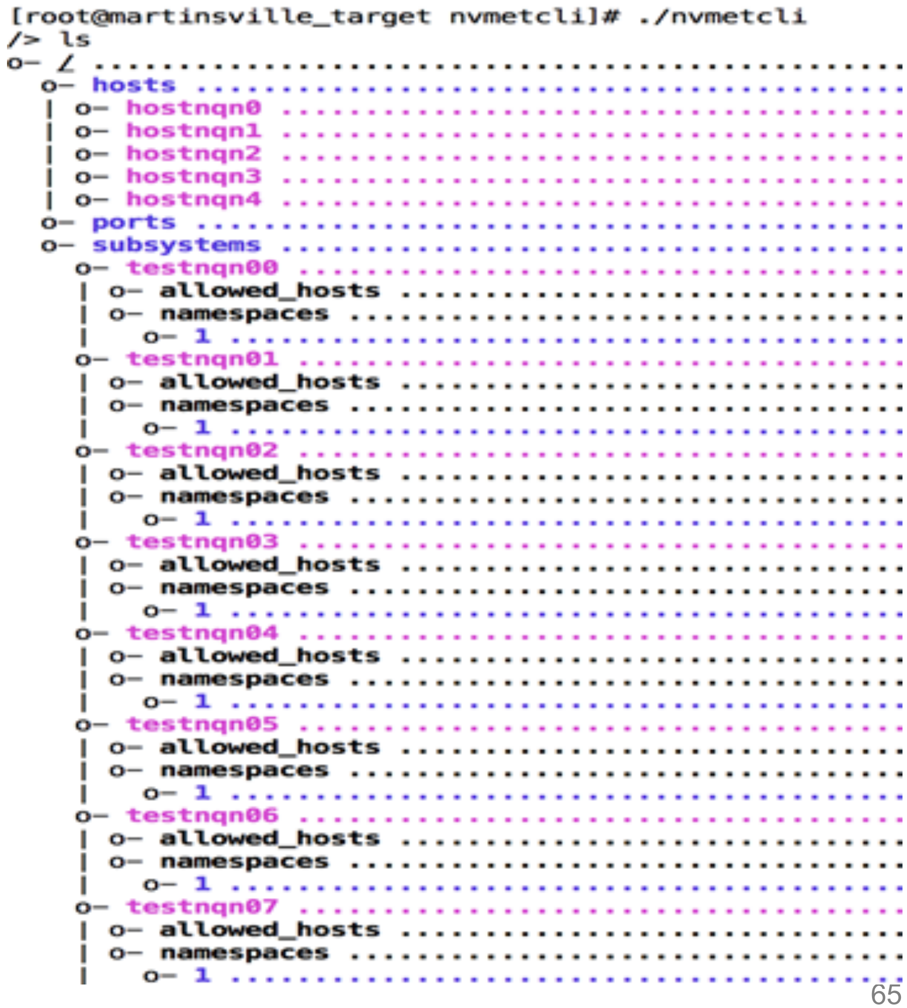
Thank You!

Architected for Performance

www.nvmexpress.org

NVMETCLI

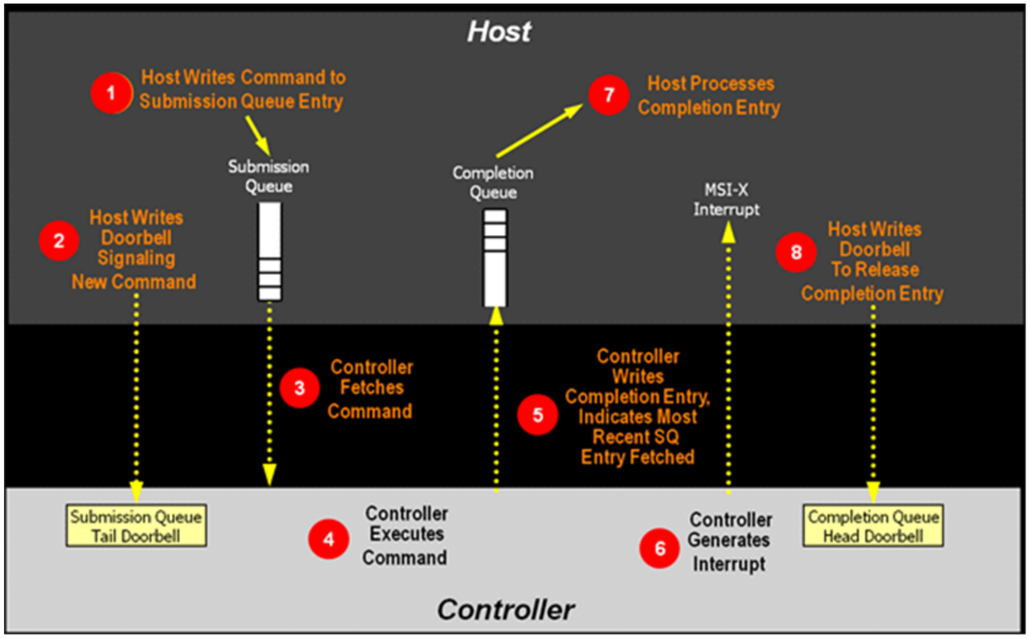
- Example nvmetcli
  - Target NVMe controllers are exposed as hostnqn’N’
  - Target NVMe SSDs are exposed as testnqn’MM’



```
[root@martinsville_target nvmetcli]# ./nvmetcli
/> ls
o- / ................................................
  o- hosts ..........................................
  | o- hostnqn0 .....................................
  | o- hostnqn1 .....................................
  | o- hostnqn2 .....................................
  | o- hostnqn3 .....................................
  | o- hostnqn4 .....................................
  o- ports ..........................................
  o- subsystems .....................................
    o- testnqn00 ....................................
    | o- allowed_hosts ..............................
    | o- namespaces .................................
    |   o- 1 ........................................
    o- testnqn01 ....................................
    | o- allowed_hosts ..............................
    | o- namespaces .................................
    |   o- 1 ........................................
    o- testnqn02 ....................................
    | o- allowed_hosts ..............................
    | o- namespaces .................................
    |   o- 1 ........................................
    o- testnqn03 ....................................
    | o- allowed_hosts ..............................
    | o- namespaces .................................
    |   o- 1 ........................................
    o- testnqn04 ....................................
    | o- allowed_hosts ..............................
    | o- namespaces .................................
    |   o- 1 ........................................
    o- testnqn05 ....................................
    | o- allowed_hosts ..............................
    | o- namespaces .................................
    |   o- 1 ........................................
    o- testnqn06 ....................................
    | o- allowed_hosts ..............................
    | o- namespaces .................................
    |   o- 1 ........................................
    o- testnqn07 ....................................
    | o- allowed_hosts ..............................
    | o- namespaces .................................
    |   o- 1 ........................................
```

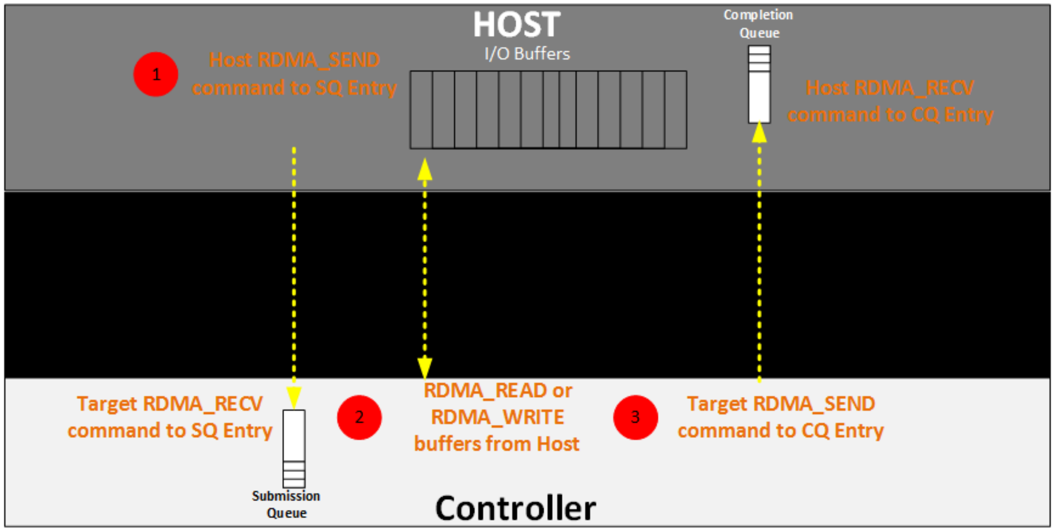
PCIe Memory Queuing Model

1. Host writes command to SQ
2. Host writes SQ tail pointer for doorbell
3. Controller fetches command
4. Controller processes command
5. Controller writes completion to CQ
6. Controller generates MSI-X interrupt
7. Host processes completion
8. Host writes to CQ head pointer for doorbell


Host
1 Host Writes Command to Submission Queue Entry
7 Host Processes Completion Entry
Submission Queue
Completion Queue
MSI-X Interrupt
2 Host Writes Doorbell Signaling New Command
8 Host Writes Doorbell To Release Completion Entry
3 Controller Fetches Command
5 Controller Writes Completion Entry, Indicates Most Recent SQ Entry Fetched
Submission Queue Tail Doorbell
4 Controller Executes Command
6 Controller Generates Interrupt
Completion Queue Head Doorbell
Controller

Fabrics Queuing Model

1. Host send RDMA_SEND that update in target as RDMA_RECV in target SQ
2. Target issue RDMA_READ or RDMA_WRITE to access data in host memory for Read or Write respectively
3. On completion target update in host CQ using RDMA_SEND that is received by host as RDMA_RECV
4. NVMe over Fabrics does not define an interrupt mechanism that allows a controller to generate a host interrupt. It is the responsibility of the host fabric interface (e.g., Host Bus Adapter) to generate host interrupts


HOST
I/O Buffers
Completion Queue
1
Host RDMA_SEND command to SQ Entry
Host RDMA_RECV command to CQ Entry
Target RDMA_RECV command to SQ Entry
2
RDMA_READ or RDMA_WRITE buffers from Host
3
Target RDMA_SEND command to CQ Entry
Submission Queue
Controller

Flash Memory Summit

Storage Performance Development Kit and NVM Express*

Jim Harris

Principal Software Engineer

Intel Data Center Group

NVMe* Software Overhead

- NVMe Specification enables highly optimized drivers
  - No register reads in I/O path
  - Multiple I/O queues allows lockless submission from multiple CPU cores in parallel
- But even best of class kernel mode drivers have non-trivial software overhead
  - 3-5us of software overhead per I/O
  - 500K+ IO/s per SSD, 4-24 SSDs per server
  - <10us latency with latest media (i.e. Intel Optane^{TM} SSD)
- Enter the Storage Performance Development Kit
  - Includes polled-mode and user-space drivers for NVMe

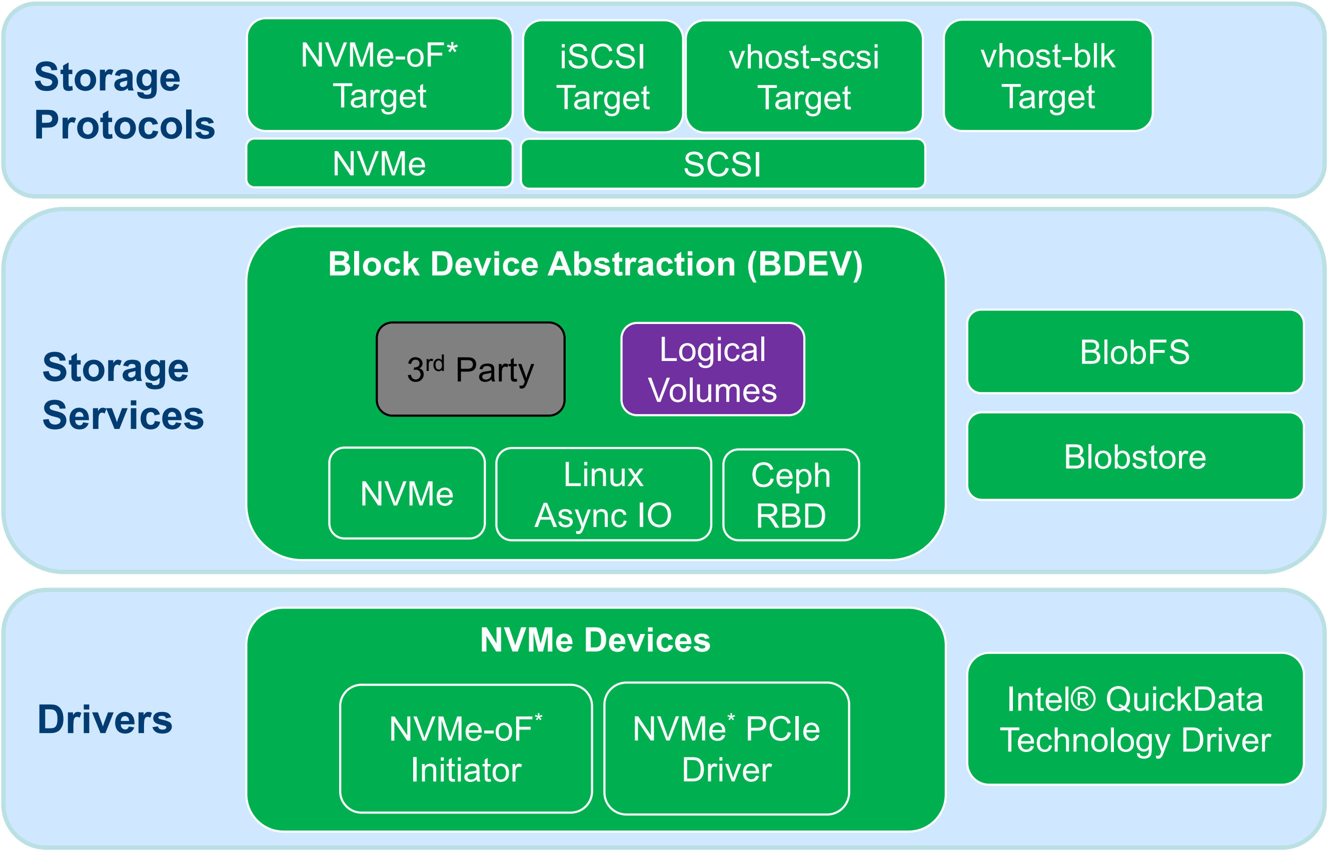
Storage Performance Development Kit (SPDK)

- Open Source Software Project
  - BSD licensed
  - Source code: http://github.com/spdk
  - Project website: http://spdk.io
- Set of software building blocks for scalable efficient storage applications
  - Polled-mode and user-space drivers and protocol libraries (including NVMe*)
- Designed for current and next generation NVM media latencies (i.e. Intel Optane^{TM})

Architecture

Released

Q4’17



Storage Protocols

NVMe-oF* Target

iSCSI Target

vhost-scsi Target

vhost-blk Target

NVMe

SCSI

Storage Services

Block Device Abstraction (BDEV)

3rd Party

Logical Volumes

NVMe

Linux Async IO

Ceph RBD

BlobFS

Blobstore

Drivers

NVMe Devices

NVMe-oF* Initiator

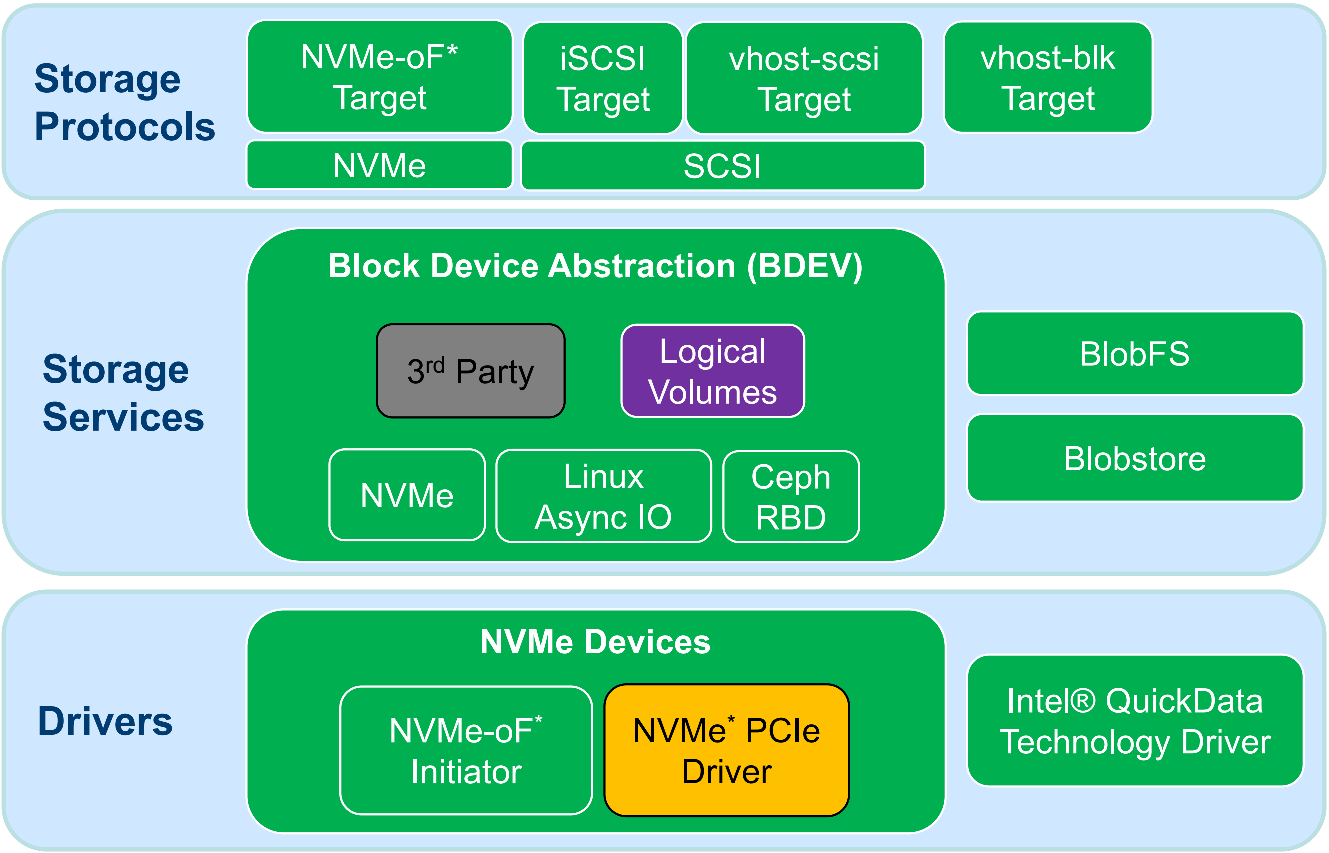
NVMe* PCIe Driver

Intel® QuickData Technology Driver

NVMe* Driver

🟩 Released

🟪 Q4'17



Storage Protocols

NVMe-oF* Target

iSCSI Target

vhost-scsi Target

vhost-blk Target

NVMe

SCSI

Storage Services

Block Device Abstraction (BDEV)

3rd Party

Logical Volumes

NVMe

Linux Async IO

Ceph RBD

BlobFS

Blobstore

Drivers

NVMe Devices

NVMe-oF* Initiator

NVMe* PCIe Driver

Intel® QuickData Technology Driver

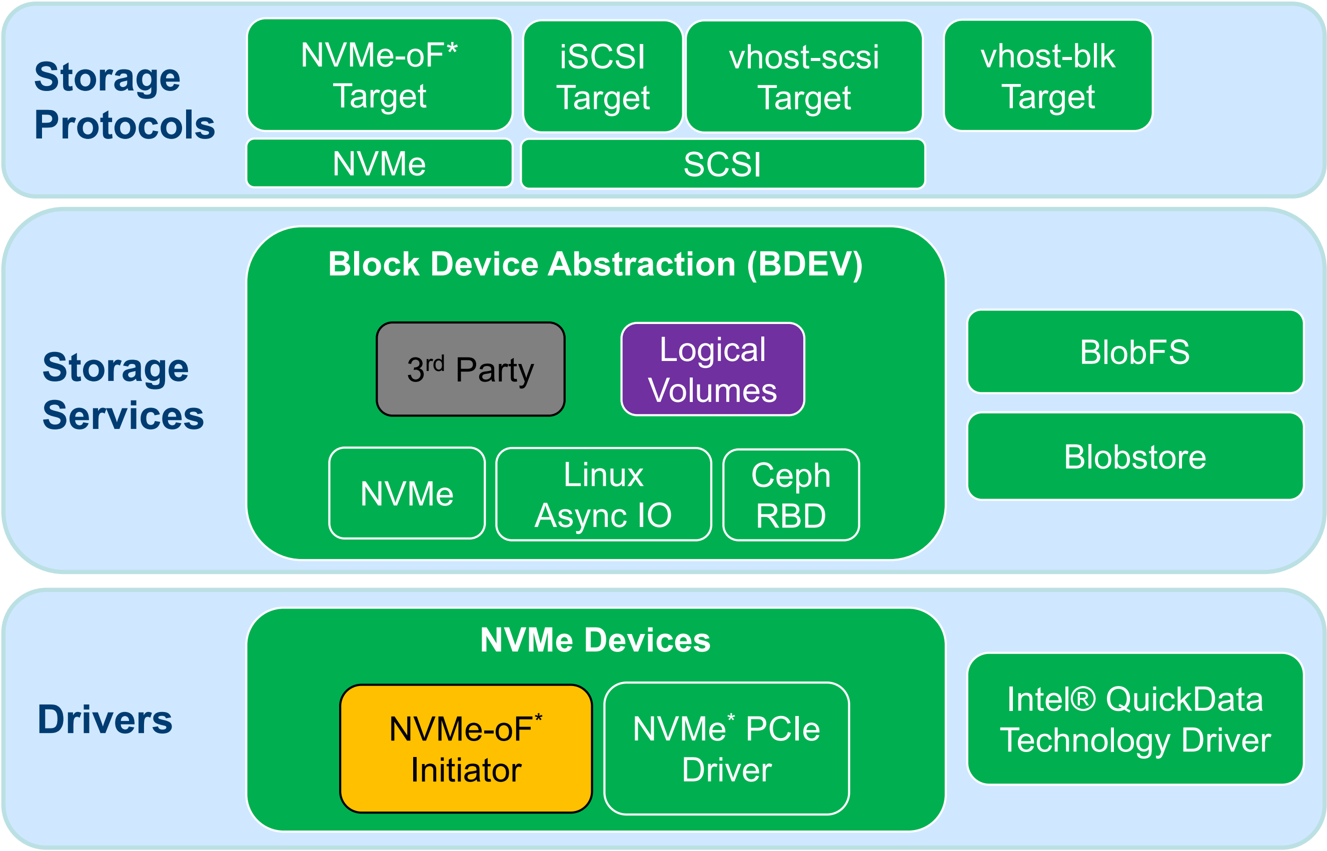
NVMe* Driver Key Characteristics

- Supports NVMe 1.0 to 1.3 spec-compliant devices
- Userspace Asynchronous Polled Mode operation
- Application owns I/O queue allocation and synchronization
- Features supported include:
  - End-to-end Data Protection
  - SGL
  - Reservations
  - Namespace Management
  - Weighted Round-Robin
  - Controller Memory Buffer
  - Firmware Update
  - Hotplug

NVMe-oF Initiator

🟩 Released

🟪 Q4’17



Storage Protocols

NVMe-oF* Target

iSCSI Target

vhost-scsi Target

vhost-blk Target

NVMe

SCSI

Storage Services

Block Device Abstraction (BDEV)

3rd Party

Logical Volumes

NVMe

Linux Async IO

Ceph RBD

BlobFS

Blobstore

Drivers

NVMe Devices

NVMe-oF* Initiator

NVMe* PCIe Driver

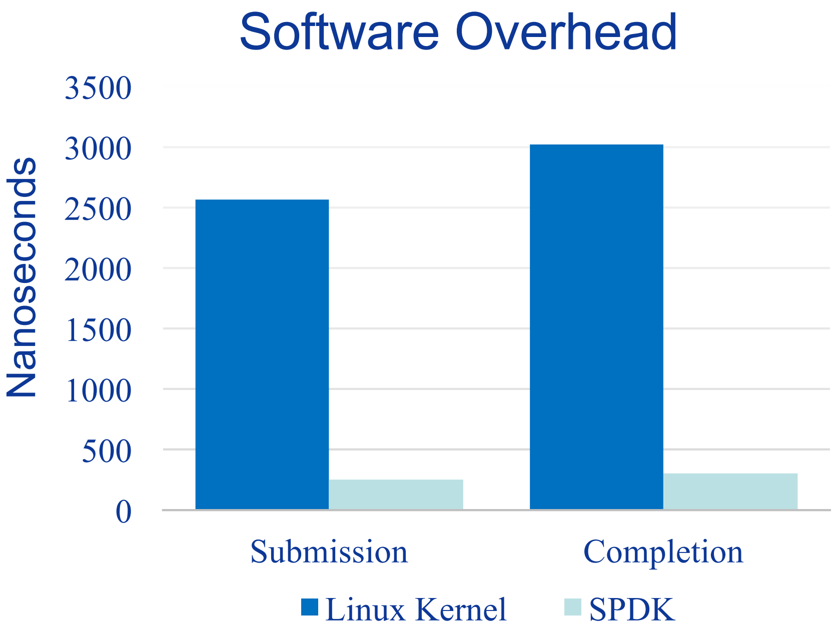
Intel® QuickData Technology Driver

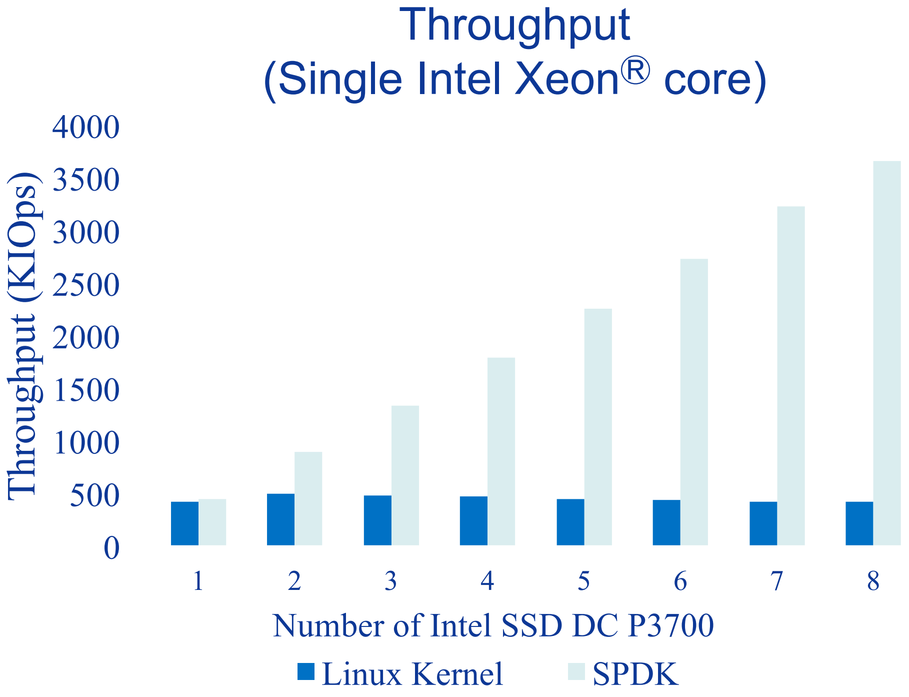
NVMe-oF Initiator

- Common API for local and remote access
  - Differentiated by probe parameters
- Pluggable fabric transport
  - RDMA supported currently (using libibverbs)
  - Allows for future transports (i.e. TCP)

Flash Memory Summit

NVMe* Driver Performance Comparison





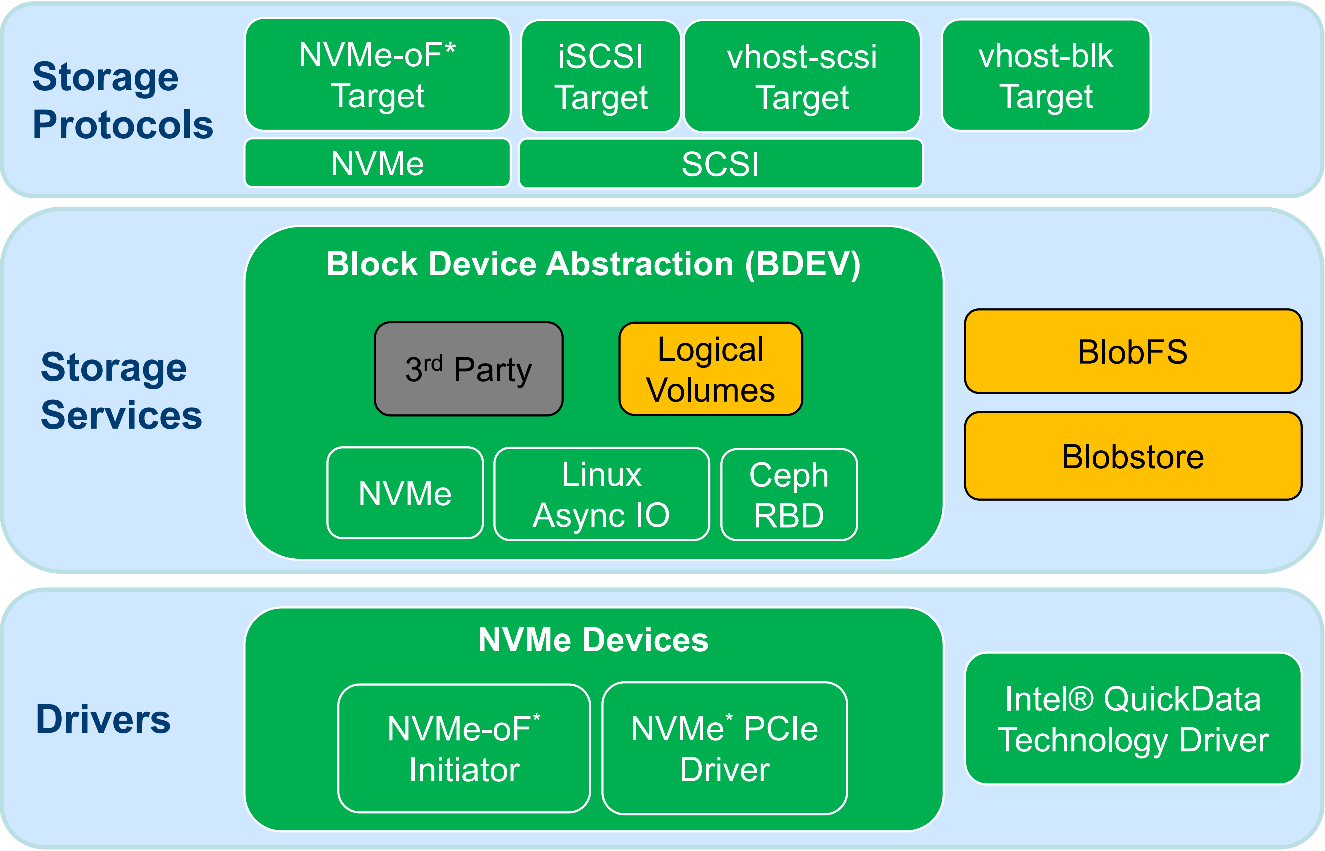
System Configuration: 2x Intel® Xeon® E5-2695v4 (HT off), Intel® Speed Step enabled, Intel® Turbo Boost Technology disabled, 8x 8GB DDR4 2133 MT/s, 1 DIMM per channel, CentOS* Linux* 7.2, Linux kernel 4.7.0-rc1, 1x Intel® P3700 NVMe SSD (800GB), 4x per CPU socket, FW 8DV10102, I/O workload 4KB random read, Queue Depth: 1 per SSD, Performance measured by Intel using SPDK overhead tool, Linux kernel data using Linux AIO

System Configuration: 2x Intel® Xeon® E5-2695v4 (HT off), Intel® Speed Step enabled, Intel® Turbo Boost Technology disabled, 8x 8GB DDR4 2133 MT/s, 1 DIMM per channel, CentOS* Linux* 7.2, Linux kernel 4.10.0, 8x Intel® P3700 NVMe SSD (800GB), 4x per CPU socket, FW 8DV101H0, I/O workload 4KB random read, Queue Depth: 128 per SSD, Performance measured by Intel using SPDK perf tool, Linux kernel data using Linux AIO

Blobstore

Released

Q4’17



Storage Protocols

NVMe-oF* Target

iSCSI Target

vhost-scsi Target

vhost-blk Target

NVMe

SCSI

Storage Services

Block Device Abstraction (BDEV)

3rd Party

Logical Volumes

NVMe

Linux Async IO

Ceph RBD

BlobFS

Blobstore

Drivers

NVMe Devices

NVMe-oF* Initiator

NVMe* PCIe Driver

Intel® QuickData Technology Driver

What about:

- filesystems?
- logical volumes?

SPDK Blobstore

- Userspace general purpose block allocator

SPDK Logical Volumes

- Enable dynamic partitioning

SPDK BlobFS

- Enables basic filesystem use cases
- Includes RocksDB integration

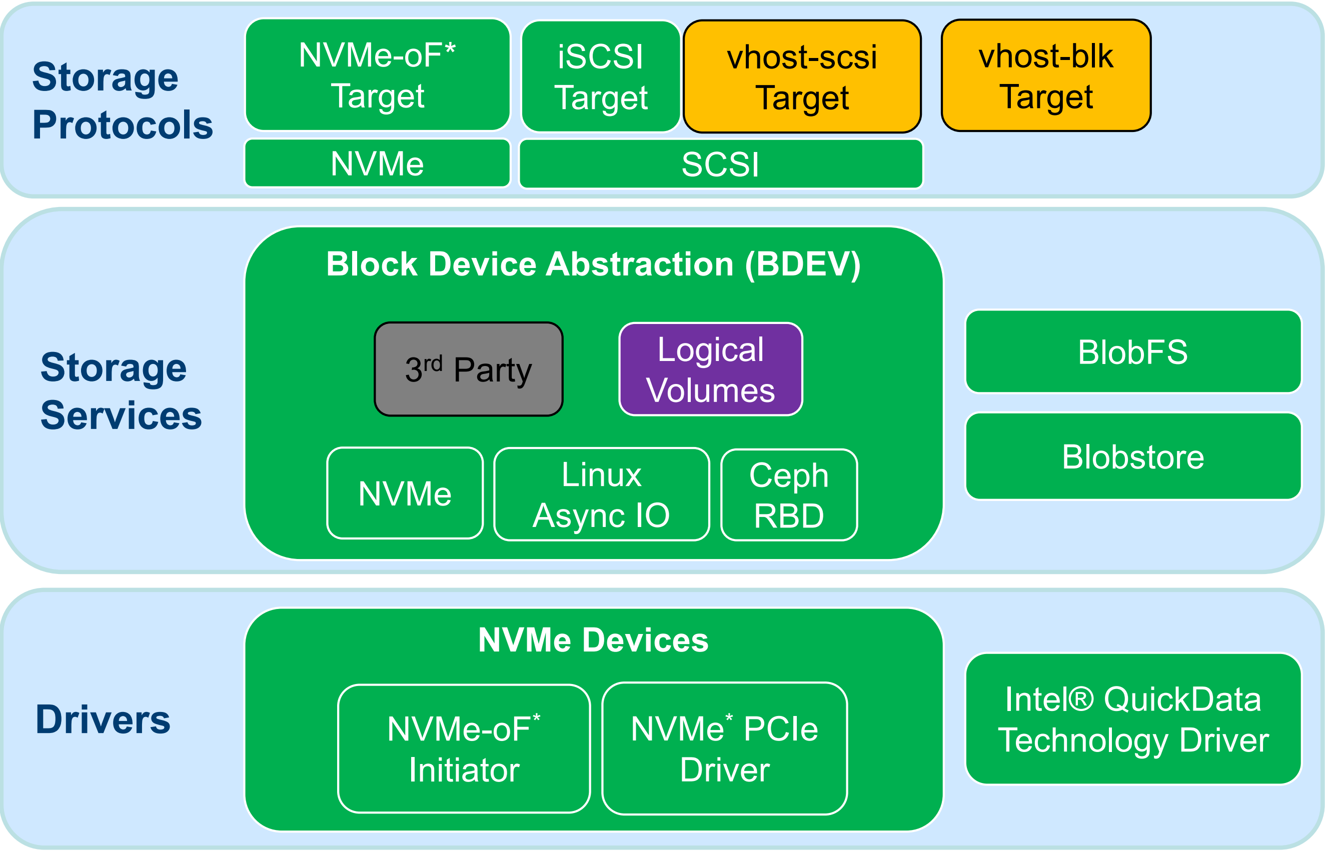
FMS Forum D-11

Flash Memory Summit

vhost

🟩 Released

🟪 Q4’17



Accelerated I/O Virtualization

- QEMU/KVM-based VMs

Reduces I/O overhead on I/O processing cores

- Leverage SPDK advantages for I/O on behalf of VMs

Reduces I/O overhead on VM cores

- Polling eliminates VMEXITs on submission

FMS Forum W-32

nvm EXPRESS®

Architected for Performance

Flash Memory Summit

Bios TBA

BACKUP