NVM Express® in the Linux Stack

May 12, 2016, 11 AM Pacific Time

Presenters: Keith Busch, Intel and Matias Bjørling, CNEX Labs

Moderator: Stephen Bates, Microsemi

Some Logistics

- This webinar is being recorded and will be available on the BrightTalk site after the event. The slides will be available on the same site as a PDF after this talk.

- Feel free to ask questions via the question submission tool. We will respond to the best of them in the discussion session at the end of the talk.

- We have provided some further reading links at the end of this deck. If you want to learn more, feel free to dig into those.

Outline

- Introduction – **Stephen**
- The NVM Express Driver and Tools – **Keith**
- The Linux Block Layer – **Matias**
- Conclusions – **Stephen**
- Q & A Session

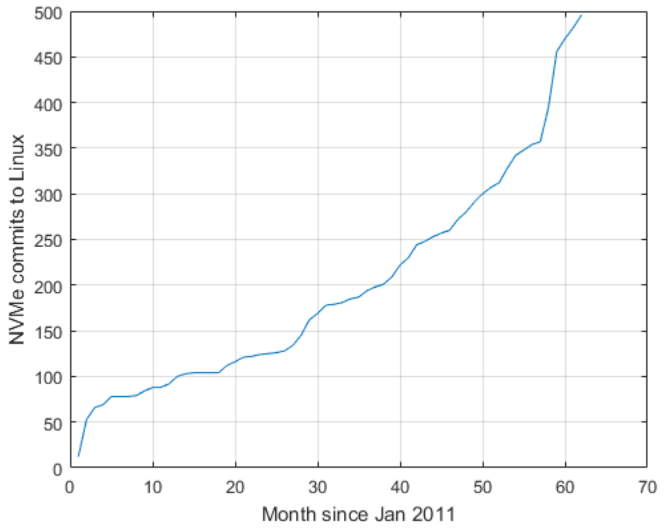
Fun “NVMe in Linux” Facts!

59 contributors from over 20 different organizations

Over 500 git commits to NVMe driver since added in January 2011

First commit added by Matt Wilcox of Intel

Rate of commits is increasing as support for NVMe grows

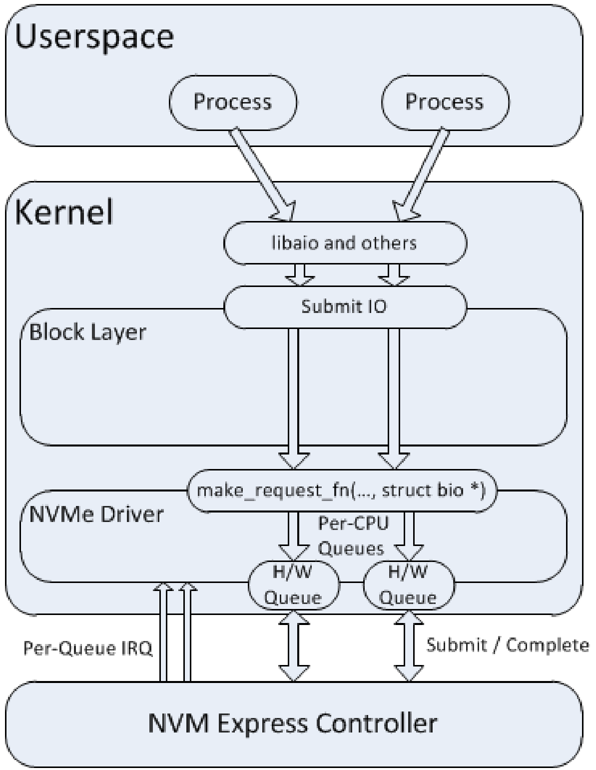


NVM Express® Linux Overview

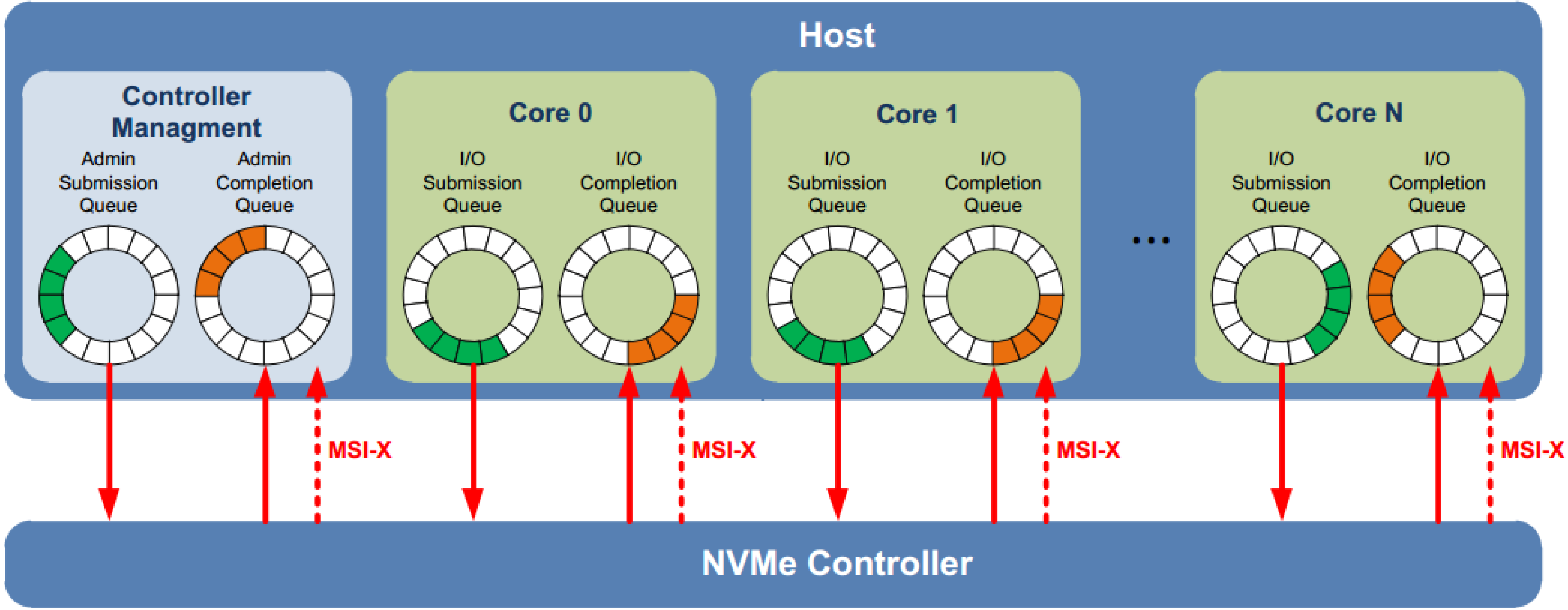
Keith Busch – Intel

NVMe Driver: High Level Implementation

- Bio-based for performance: lockless block layer
- Driver burdened to manage common boilerplate block driver issues:
  - Timeouts, command tag management, queue selection, IO splitting, DMA mapping, trace points, IO statistics
- Does not stack with the request-based device mappers (dm-multipath)

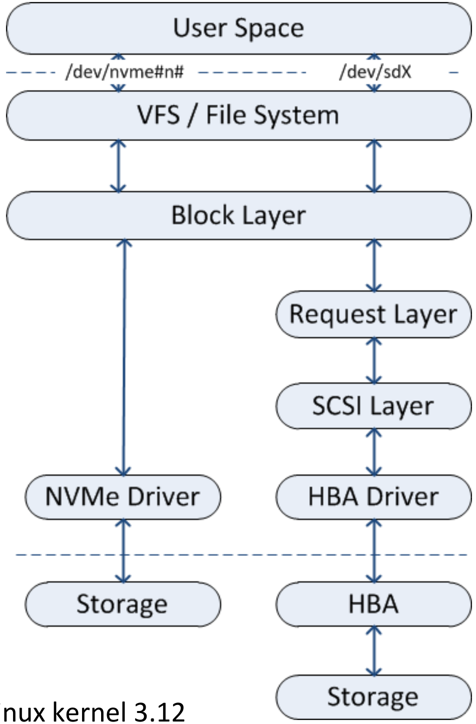


Per-CPU H/W Queues



Storage Stack Comparison

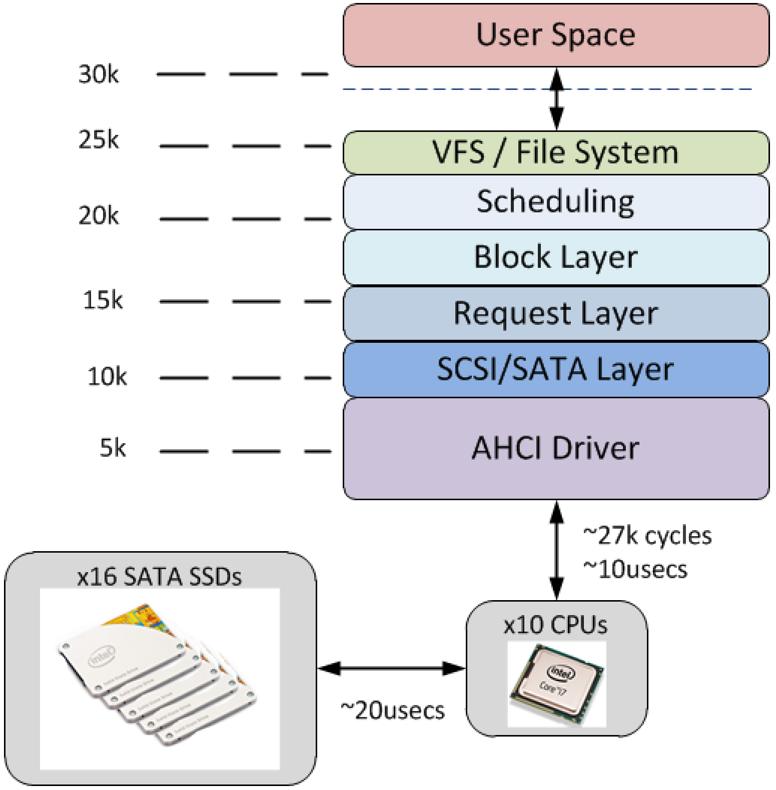
- Submission latency and CPU cycles reduced >50%*:
  - NVMe: 2.8us, 9,100 cycles
  - SAS: 6.0us, 19,500 cycles



* Measurement taken on Intel® Core™ i5-2500K 3.3GHz 6MB L3 Cache Quad-Core Desktop Processor using Linux kernel 3.12

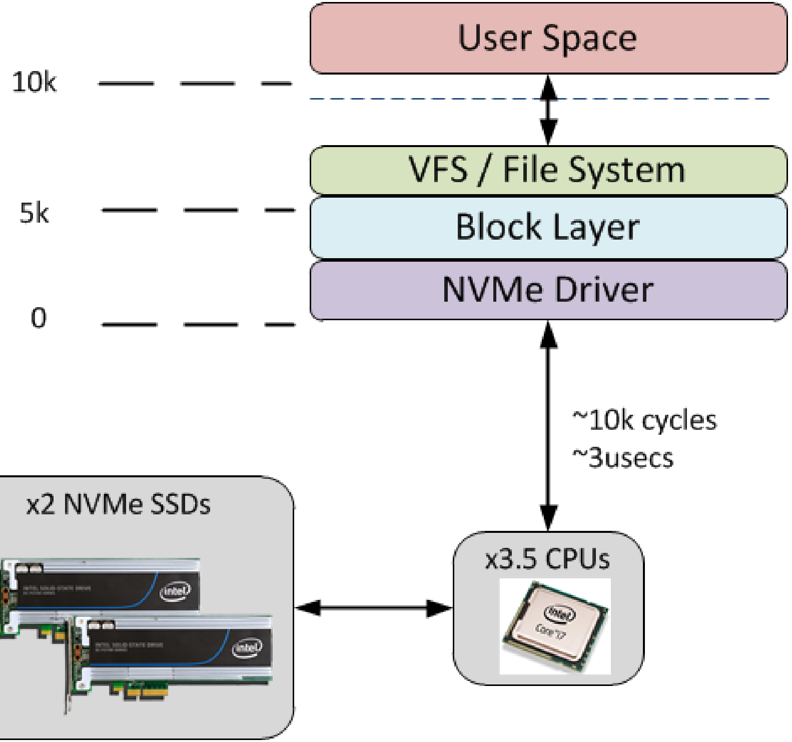
Getting 1M IOPs: SATA

- Resource intensive: software and transport protocol overhead



Getting 1M IOPs: NVMe

- More efficient h/w utilization scales IOPs!

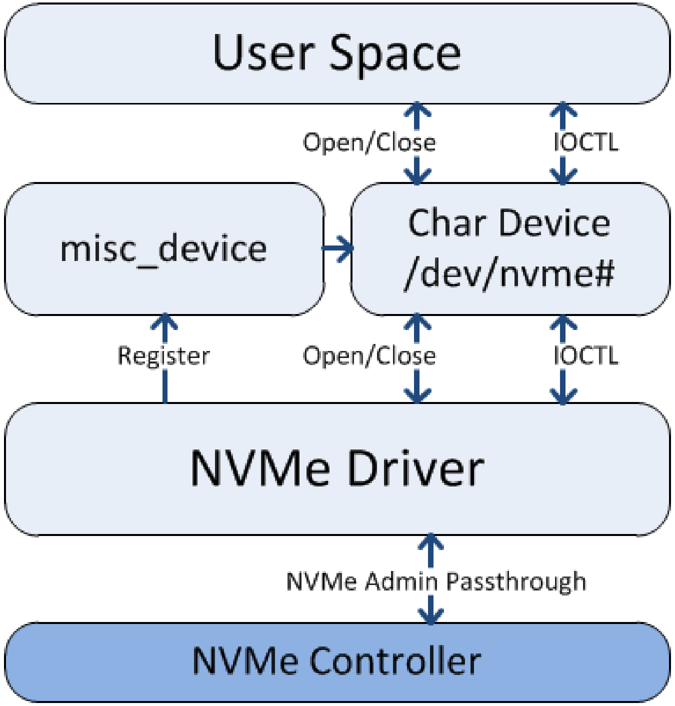


Hot Plug

- Most frequently broken feature during merges
- Dependencies on platform, pci kernel subsystem, and pciehp module.
- Surprise hot removal mostly working as of 3.16 with inflight IO
- Simultaneous hot-plug events still not handled well

Device Management: IOCTL

- Special character device handle created for each controller
- Accepts various IOCTLs for management:
  - Admin and IO Command Passthrough
    - STOP USING SG_IO!
  - Namespace Identification
  - Controller Reset
  - Subsystem Reset



Device Management: sysfs

- NVMe provides its own class for controllers
  - /sys/class/nvme/
- Individual controller handles (nvme<#>/) provide device information and control
  - Model, F/W revision, Serial, Controller ID
  - Controller reset
- Child links to each Namespace block handle include namespace identification
  - NGUID, EUI-64, or driver constructed unique identifier accessible through “wwid”
- Child link to PCI device for access to PCIe resources
  - SR=IOV, PCI capabilities, topology information

Open Source Tooling: nvme-cli

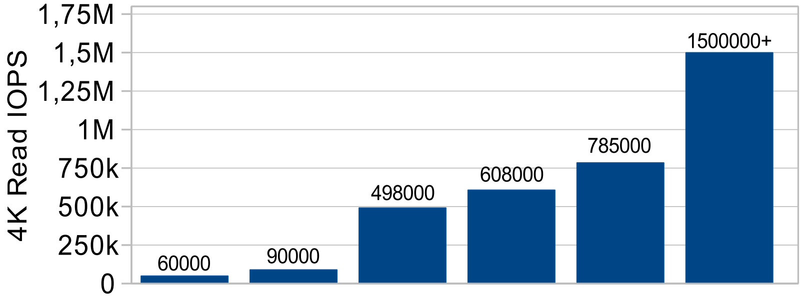
- General purpose NVMe tool for Linux, available on github
- Utilizes the IOCTL interface for submitted arbitrary admin and io commands
- Provides options and structure decoding for human readability for all NVMe commands and structures defined in 1.2 specification
- Can map controller register set for debugging

Multi-Queue Block Layer Integration

Matias Bjørling – CNEX Labs

Storage Evolution

I/Os Per Second



2009 - 2015

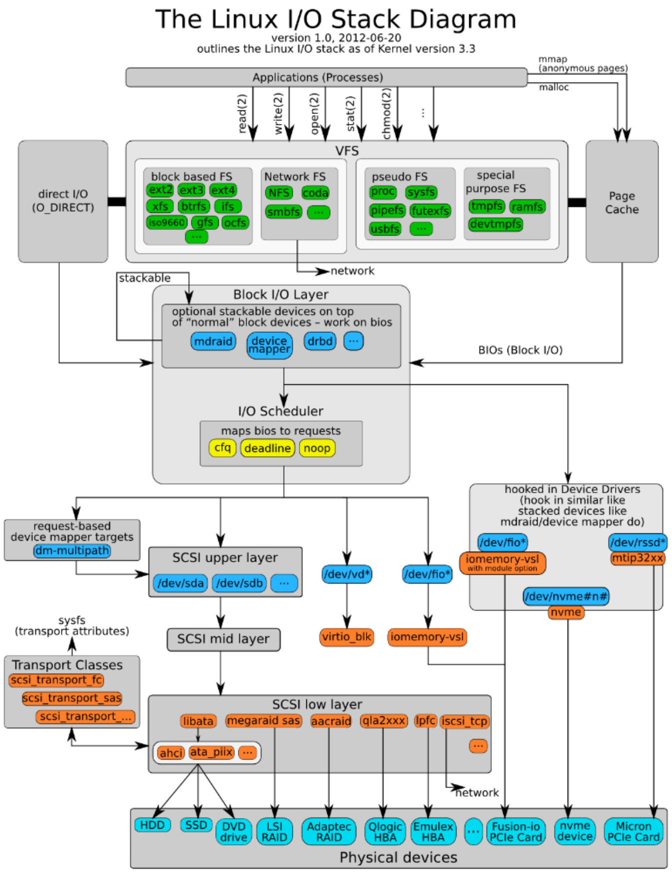
Storage devices are getting faster and faster and processors scale with additional cores

Access Latencies

Device	Read	Write
Millisecond Scale		
10G Intercontinental RPC	100 ms	100 ms
10G Intracontinental RPC	20 ms	20 ms
Hard Disk	10 ms	10 ms
10G Interregional RPC	1 ms	1 ms
Microsecond Scale		
10G Intraregional RPC	300 μs	300 μs
SATA NAND SSD	200 μs	50 μs
PCIe/NVMe NAND SSD	60 μs	15 μs
10Ge Inter-Datacenter RPC	10 μs	10 μs
40Ge Inter-Datacenter RPC	5 μs	5 μs
PCM SSD	5 μs	5 μs
Nanosecond Scale		
40 Gb Intra-Rack RPC	100 ns	100 ns
DRAM	10 ns	10 ns
STT-RAM	$<$10 ns	$<$10 ns

Block Storage Stack

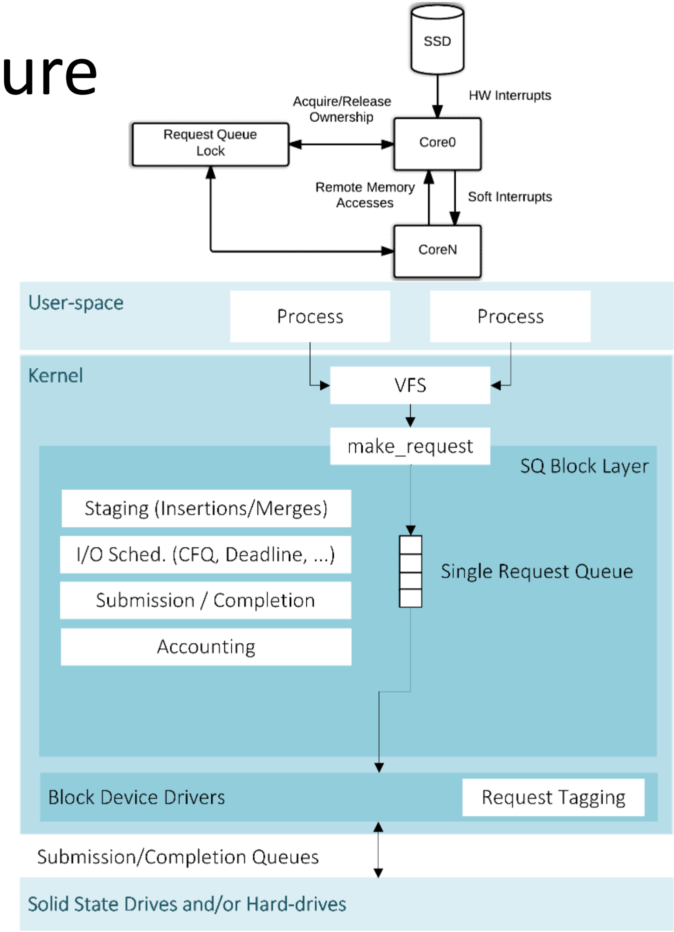
- Applications
- File Systems (Ext4, btrfs, XFS, ...)
- Block Layer
- Device drivers (SCSI/ATA)
- Hardware communication (NCQ/TCQ, Interrupt-driven, ...)
- All Layers assume to some degree
  - Fast sequential access
  - Slow random access



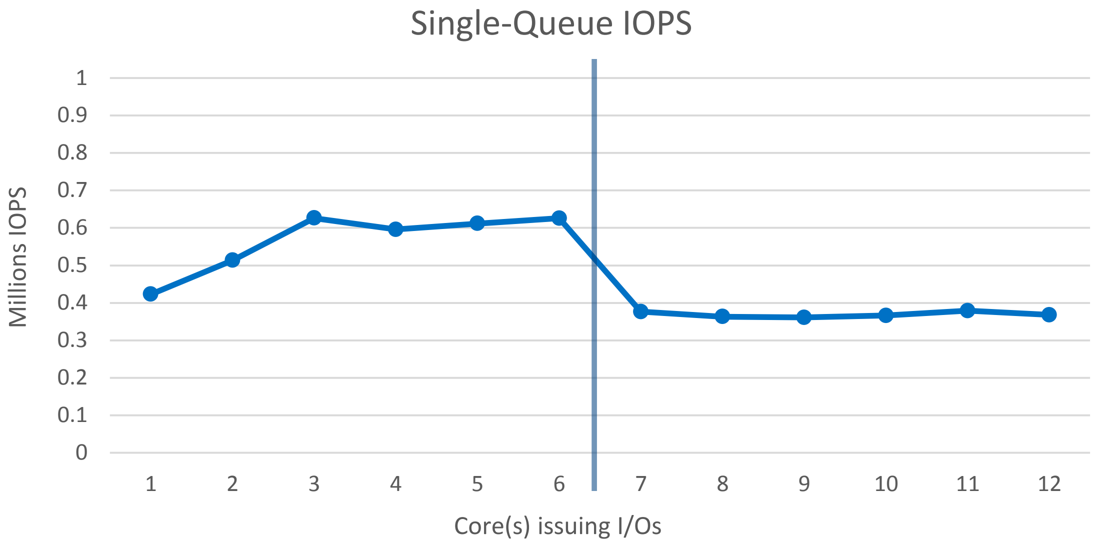
Single-Queue Block Layer Architecture

Common library for block storage device drivers and entry point for applications

- I/O Scheduling
- I/O Merging and Reordering
- I/O Accounting and Statistics
- Single request queue
  - Single lock, single device dispatch queue
  - Cache-coherence is expensive
- Does not take advantage of multi-core systems to scale performance



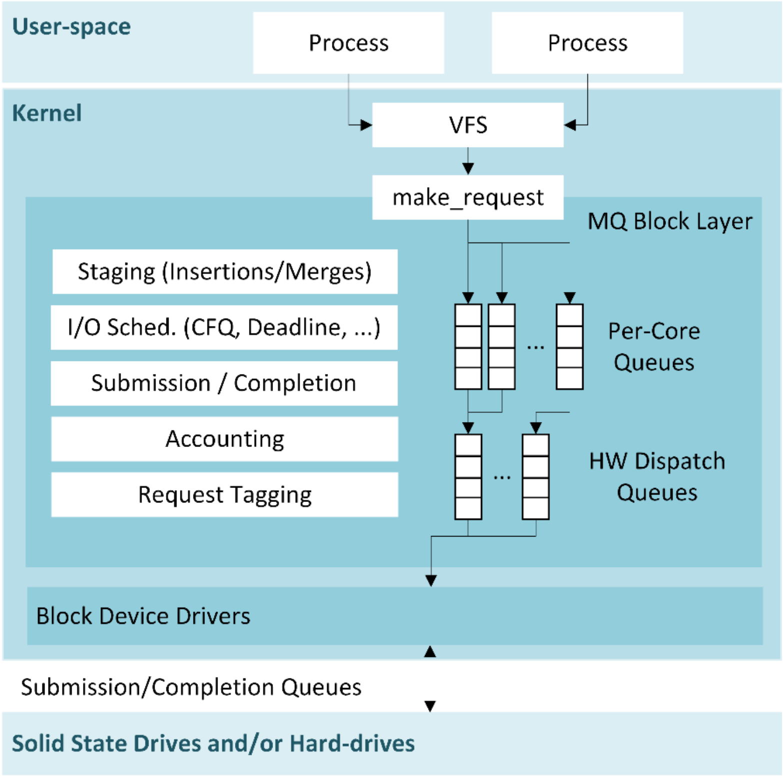
Performance of Single-Queue Block Layer



Null block device (nullblk), 512B random reads, 64 queue depth
System: 2x Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz

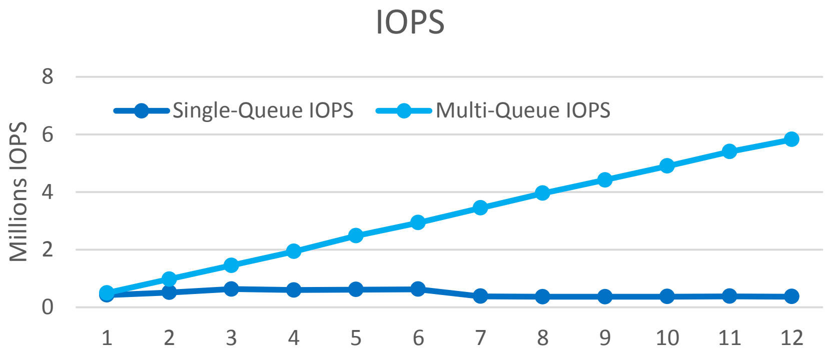
Multi-Queue Block Layer Architecture

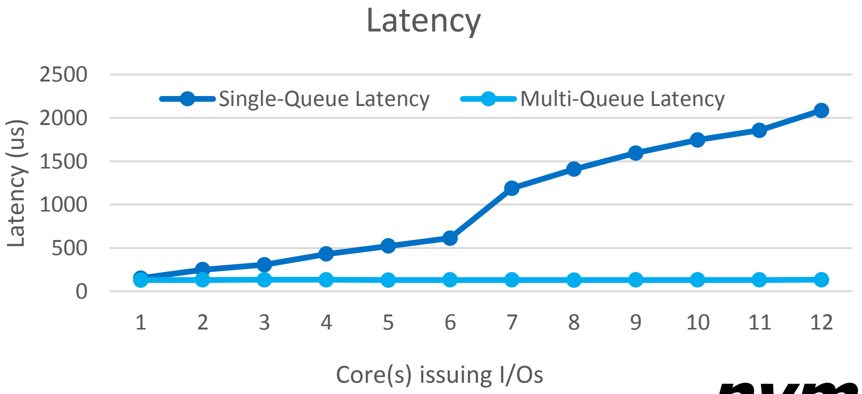
- Balance I/O workload accross cores
- Reduce cacheline sharing (per-core queues)
- Maintain functionality to existing block layer
    - Queue mgmt., timeouts, bio splits, accounting, staging
- Implement multiple hardware queues
- Scalable command tagging
    - Per-core command tag pool
- Merged in Linux Kernel 3.13



Multi-Queue Block Layer Performance Throughput

- Single-Queue Block Layer
  - =< 1M IOPS
  - Low throughput with multiple sockets
  - Increasing Latency
- Multi-Queue Block Layer
  - >6M IOPS
  - Scales with cores
  - Improved Submission and Completion latency





NVMe with the Multi-Queue Block Layer

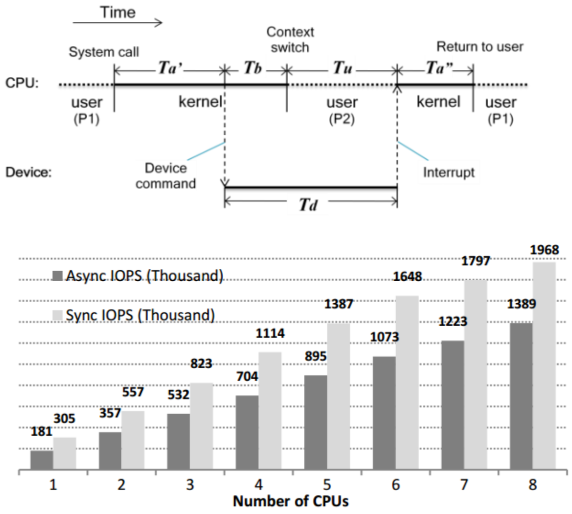
Greatly simplifies storage stack and improves performance with both latency and throughput. Block Layer now handles:

- Scalable per-command data tagging
- Timeout logic
- Improved queue suspension logic
- Removal of RCU usage for unsuspected unplugging
- Assignment of NVMe submission and completion queues

Merged in Linux kernel 3.19

New Features

- I/O Polling
  - Synchronous data access without context switch
- LightNVM -- Open-Channel SSDs
  - Physical Page Addressing
- Streams
  - Patches available
- NVMe over Fabrics
  - NVMe driver being factored for support



*Yang et. al. When Poll is Better than Interrupt, FAST 2012

Conclusions

Stephen Bates – Microsemi

Conclusions

- NVM Express and blk-mq play well together to enable very high-throughput/low latency NVM storage devices within Linux
- Tons of code and tools there to help you tune performance, debug errors, withstand failure events
- The Linux kernel is a living thing. Get involved, track the codebase, help make things better!
- Some very exciting things coming down the pipe:
  - Polling completions for lower latency (not-NAND SSDs ;-))
  - NVMe over Fabrics for NVMe with distance and scale

Further Reading

- "Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems" – http://kernel.dk/blk-mq.pdf
- The Linux Kernel Source Code – https://www.kernel.org/
- Block-Layer IO Polling – https://lwn.net/Articles/663879/
- NVM Express Standard – http://www.nvmexpress.org/specifications/
- Contributing code to the Linux Kernel – https://www.kernel.org/doc/Documentation/development-process/1.Intro
- nvme-cli source – https://github.com/linux-nvme/nvme-cli

Thank you for attending our NVM Express® webcast!

Some resources for additional information:

- View NVM Express® webcasts in our BrightTalk channel – https://www.brighttalk.com/channel/12367/nvm-express
- Follow NVM Express, Inc. on Twitter @nvmexpress – https://twitter.com/NVMexpress
- Visit us on LinkedIn – https://www.linkedin.com/groups/4307826
- Find us on the web at http://www.nvmexpress.org

nvm EXPRESS®

Architected for Performance