



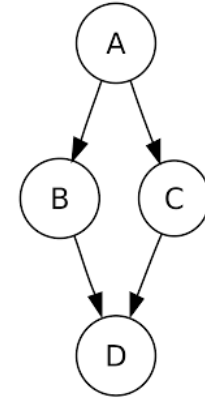
Multipathing PCI-Express Storage

Keith Busch
Linux Vault
12 March 2015

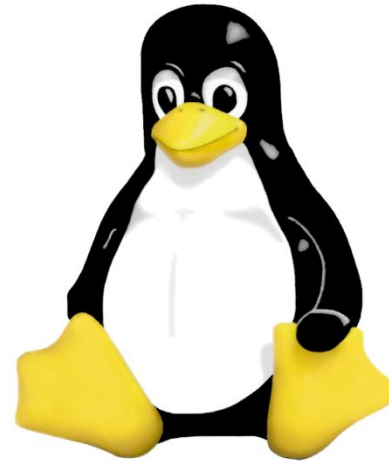
Agenda

(in no particular order)

- Why we care
- PCI-e Storage Standardization
- Storage stacking details
- Results and future work



PCI 
EXPRESS[®]



PCI-e Storage Standard:

Non-Volatile Memory Express



NVMe: Who



ORACLE



EMC²



SanDisk

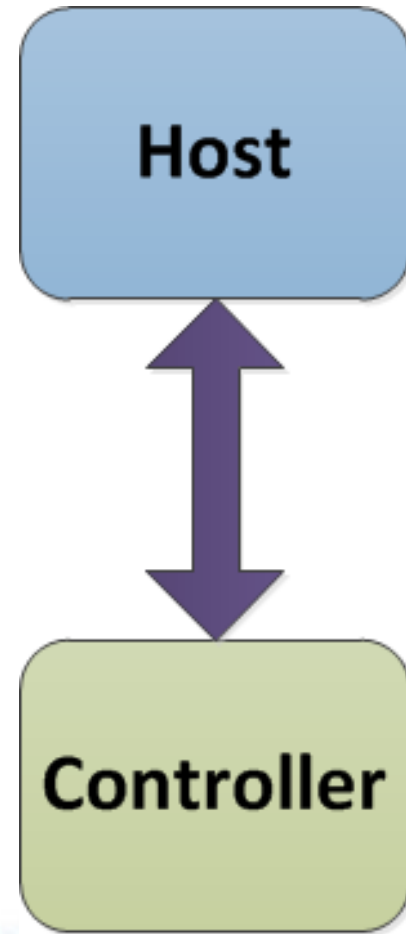


NVMe: What

Storage standard defining:

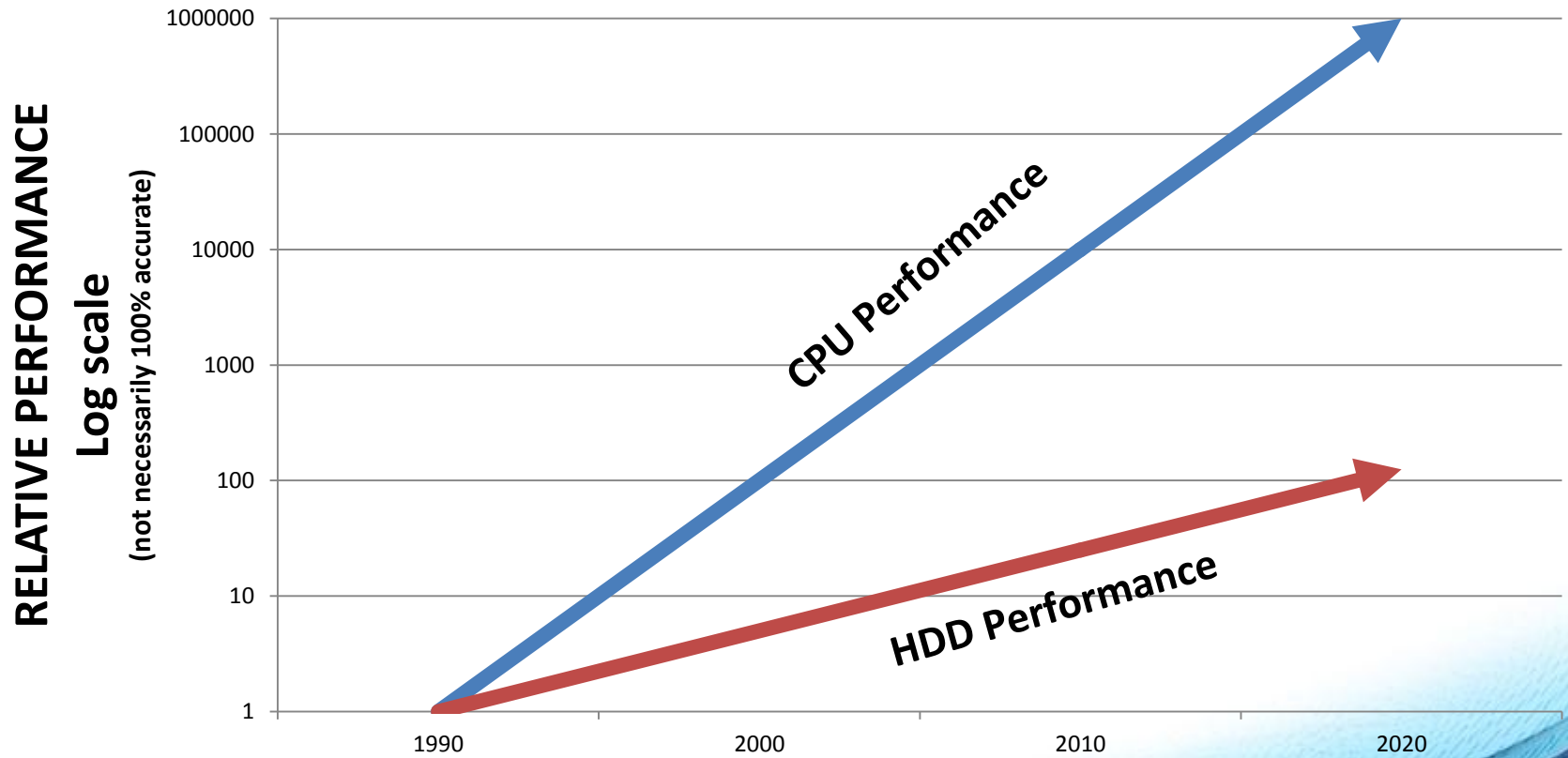
- Host controller interface
- Queueing model
- Command set

Designed for performance and scalability in mind



NVMe: Why

CPU vs. Storage Performance Gap



NVMe: Why

... but PCI-e storage predates NVMe, right?



Virident



Fusion-io



OCZ



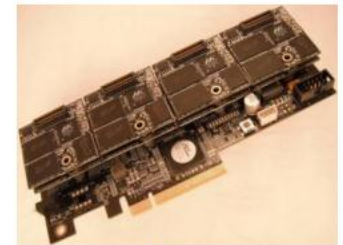
Micron



Intel



LSI



Marvell

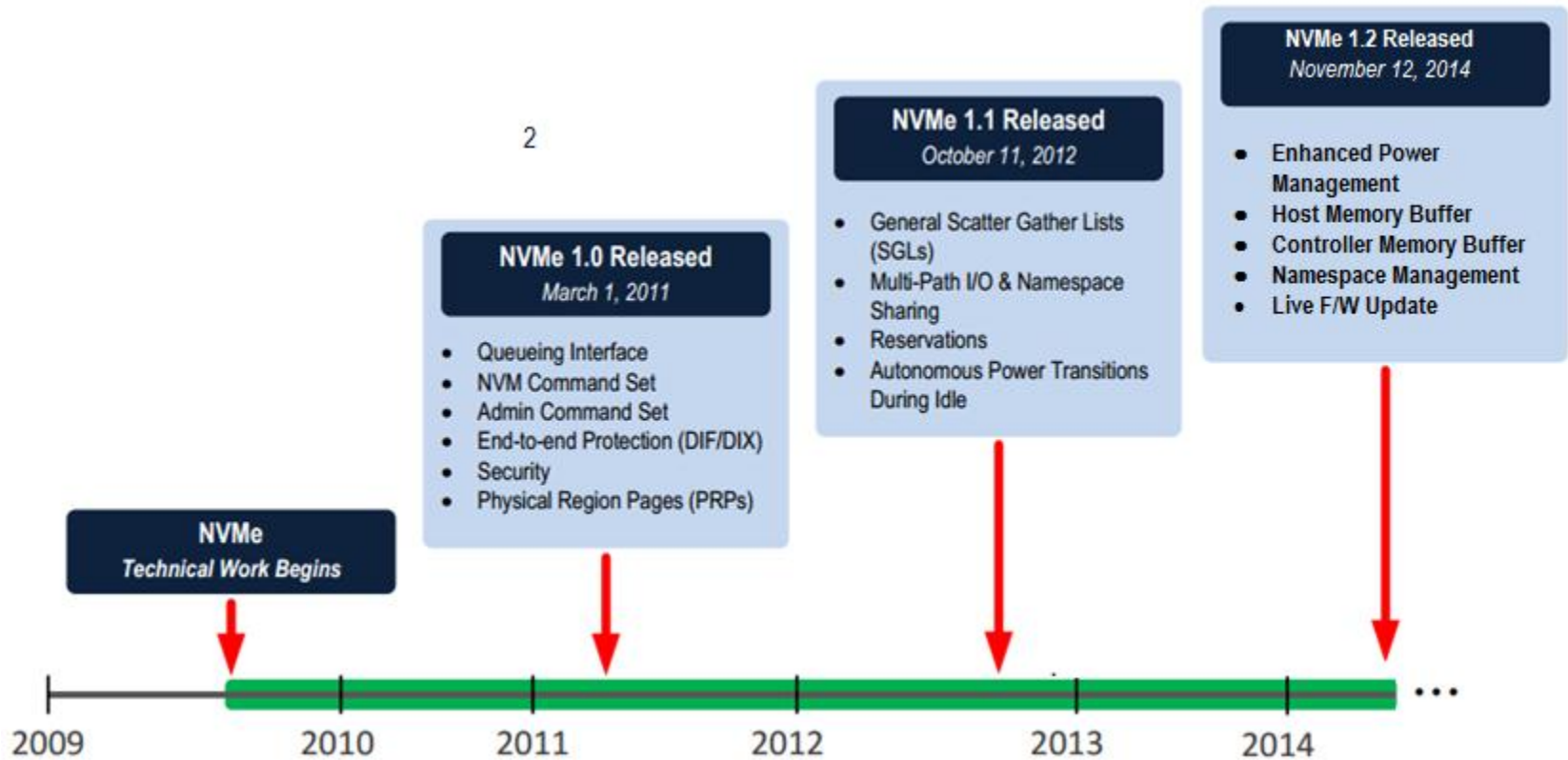
NVMe: Why

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)



<http://xkcd.com/927/>

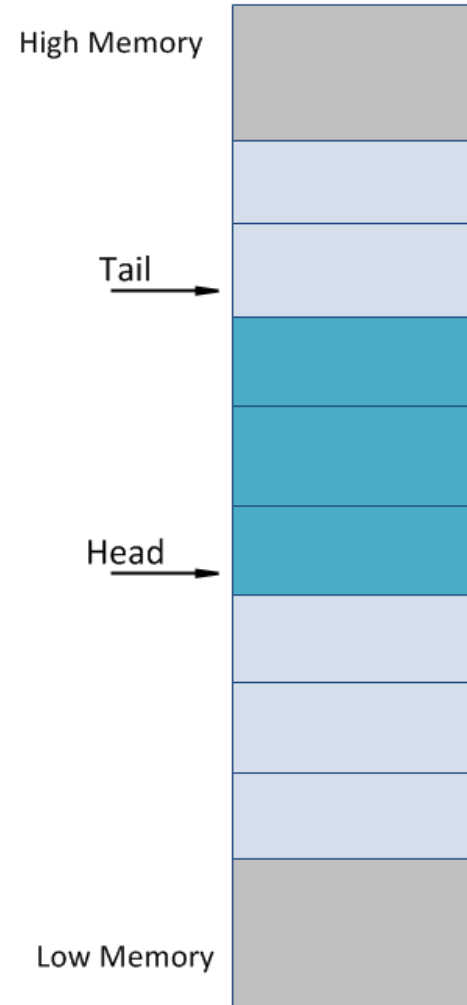
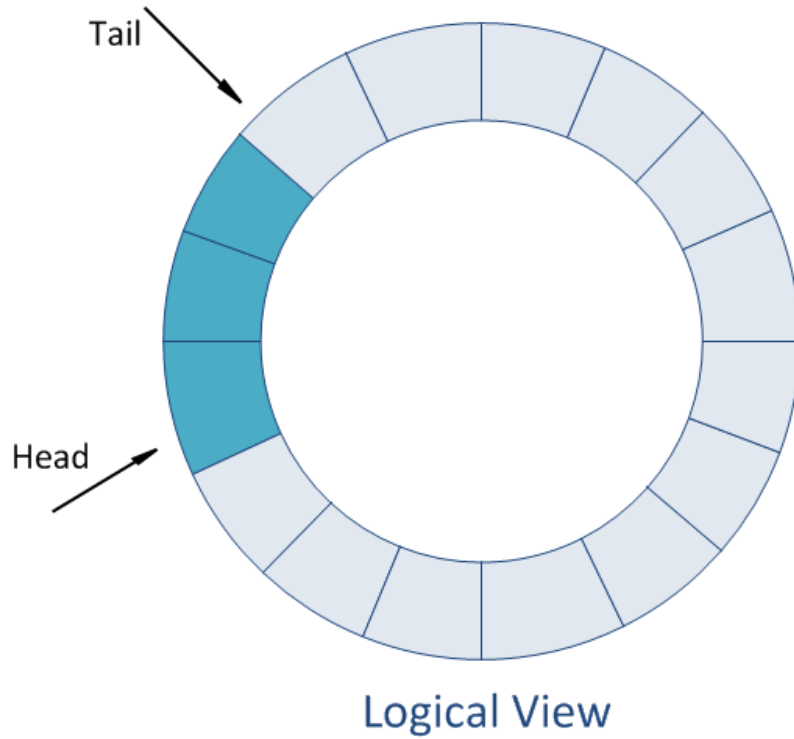
NVMe: When



NVMe: How

Offset	Symbol	Description
00h	CAP	Controller Capabilities
08h	VS	Version
0Ch	INTMS	Interrupt Mask Set
10h	INTMC	Interrupt Mask Clear
14h	CC	Controller Configuration
18h	Rsvd	Reserved
1Ch	CSTS	Controller Status
20h	NSSR	NVM Subsystem Reset
24h	AQA	Admin Queue Attributes
28h	ASQ	Admin Submission Queue Base Address
30h	ACQ	Admin Completion Queue Base Address
38h	Rsvd	Reserved
F00h	Rsvd	Command set Specific
1000h	SQ 0 TDBL	Admin Submission Queue Tail Doorbell
1000h + (1 * 4 << CAP.DSTRD)	CQ 0 HDBL	Admin Completion Queue Head Doorbell
1000h + (2 * 4 << CAP.DSTRD)	SQ 1 TDBL	IO Submission Queue 1 Tail Doorbell
1000h + (3 * 4 << CAP.DSTRD)	CQ 1 TDBL	IO Completion Queue 1 Head Doorbell

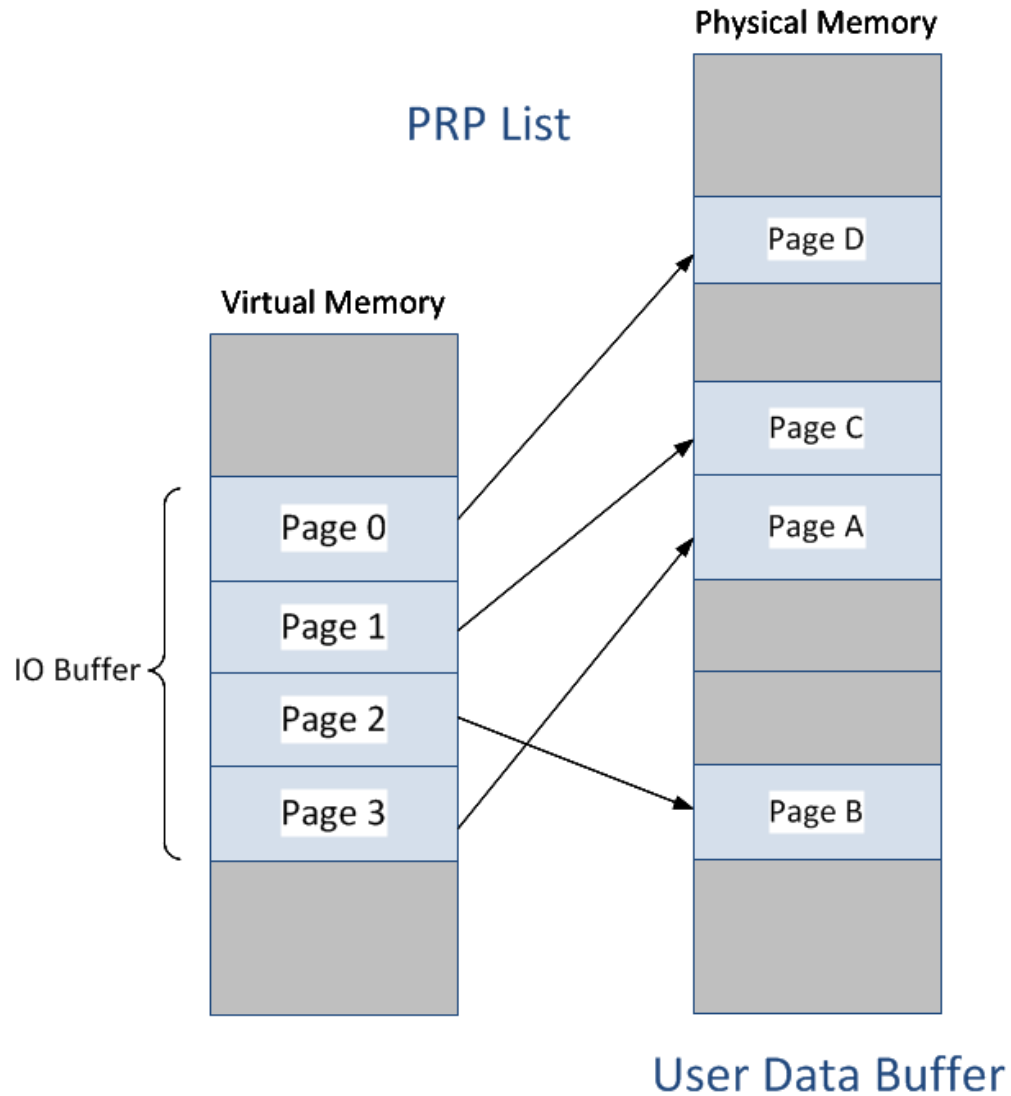
NVMe: How



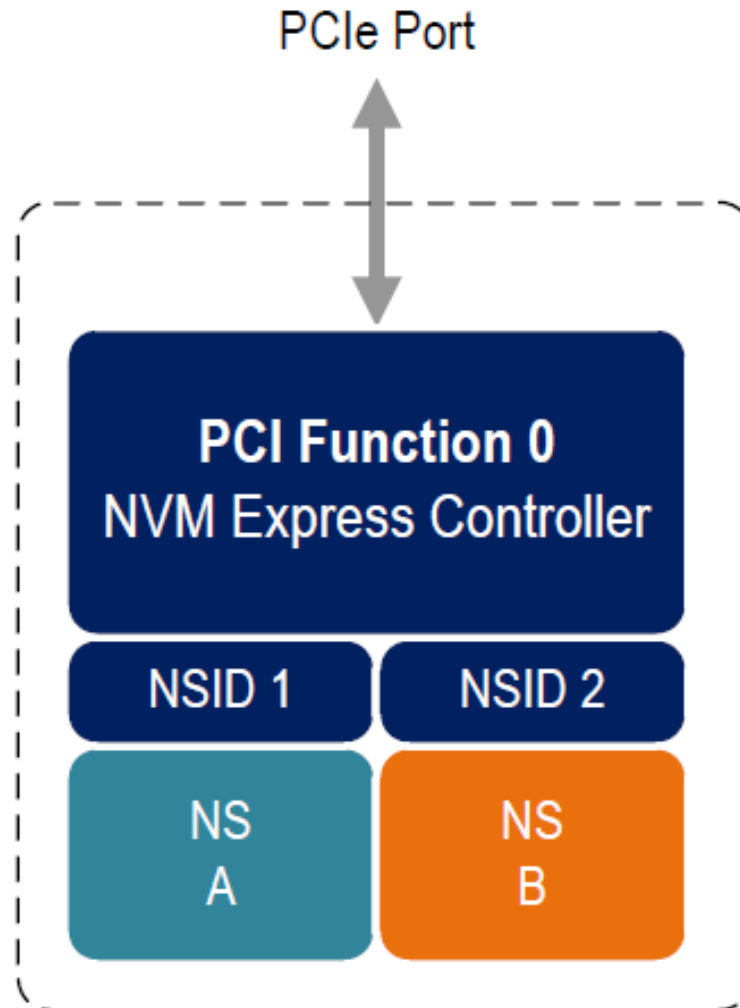
NVMe: How

Byte Word	3	2	1	0
0	Command ID		Flags	Opcode
1	Namespace ID			
2	Reserved			
3				
4	Metadata Pointer			
5				
6	PRP 1			
7				
8	PRP 2			
9				
10	Command DWORDS 10 - 15			
11				
12				
13				
14				
15				

NVMe: How



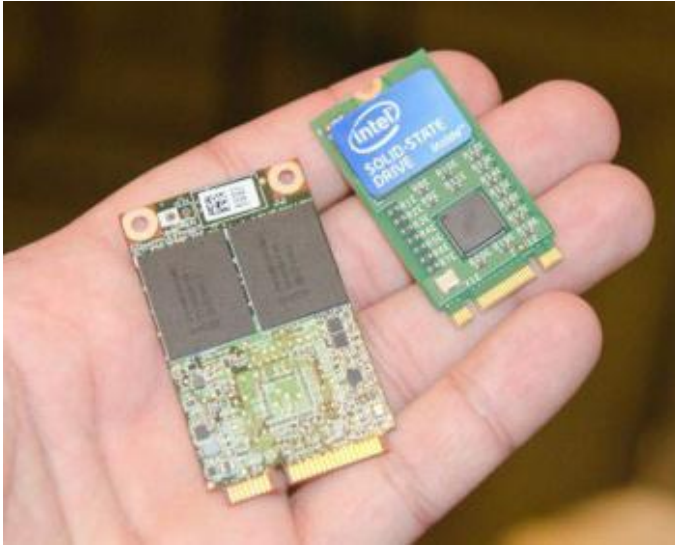
NVMe: How



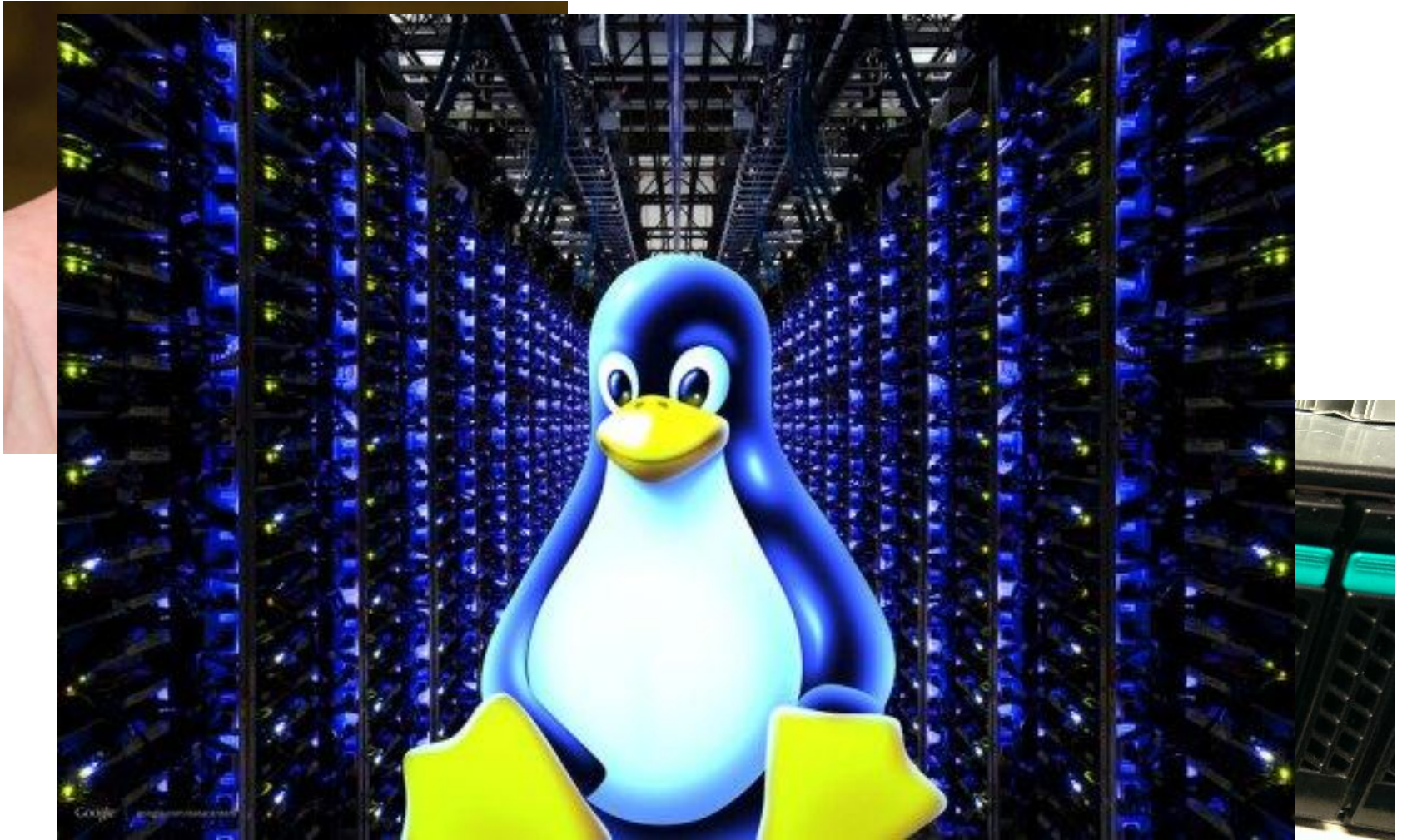
NVMe: Where



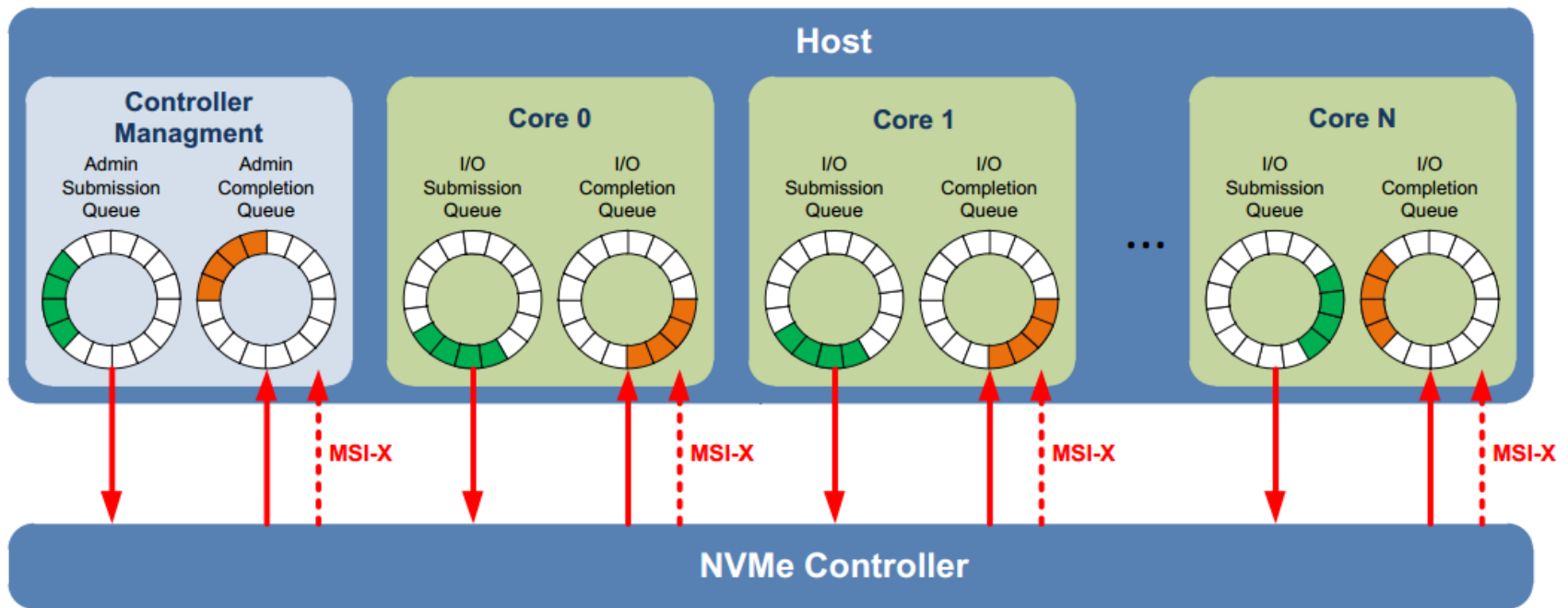
NVMe: Where



NVMe: Where

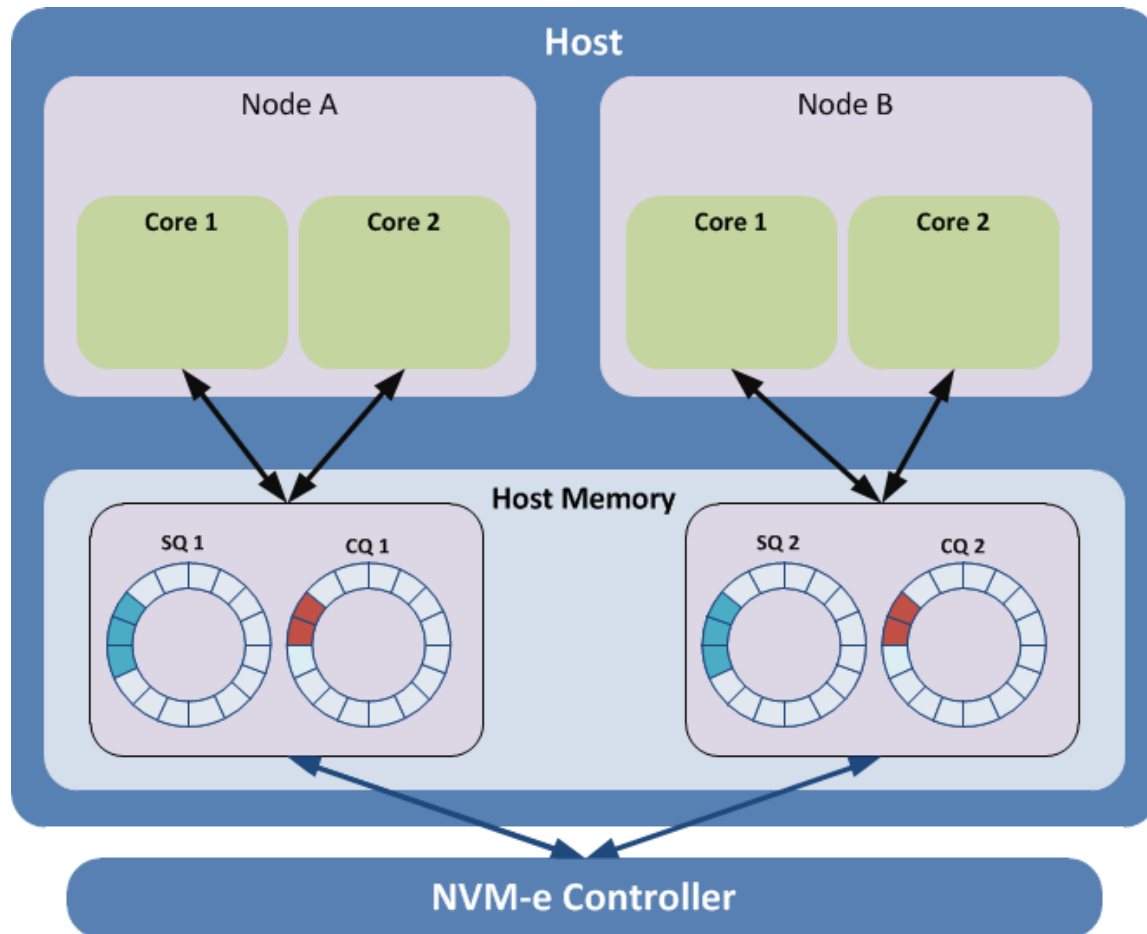


NVMe: per-cpu h/w queues



When CPUs exceed available h/w queues

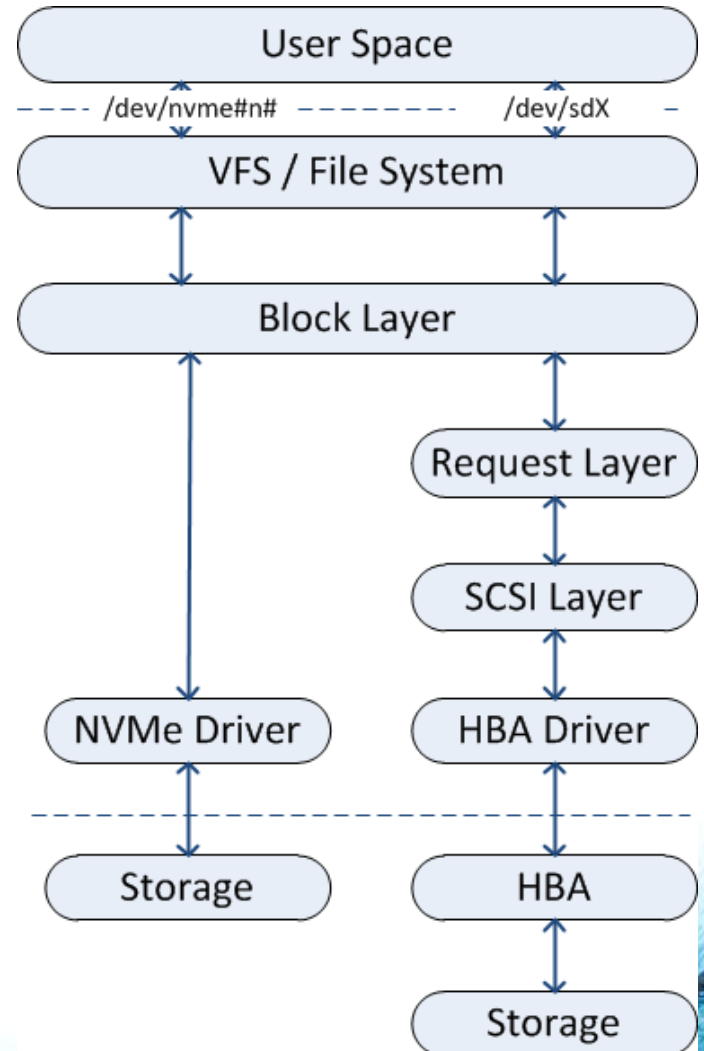
Share with your neighbors



NVMe: CPU Efficient

Submission latency and CPU cycles reduced >50%*:

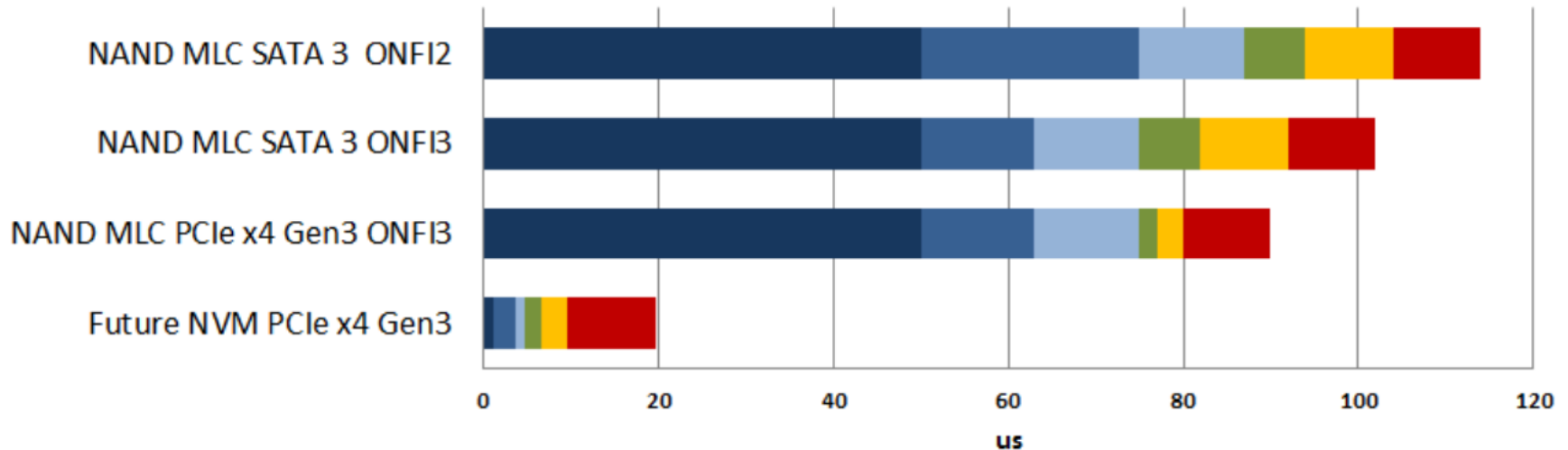
- NVMe: 2.8us, 9,100 cycles
- SAS: 6.0us, 19,500 cycles



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

The importance of reducing software latency

App to SSD IO Read Latency (QD=1, 4KB)




■ NVM Tread ■ NVM xfer ■ Misc SSD ■ Link Xfer ■ Platform + adapter ■ Software

NVMe: When we know we succeeded



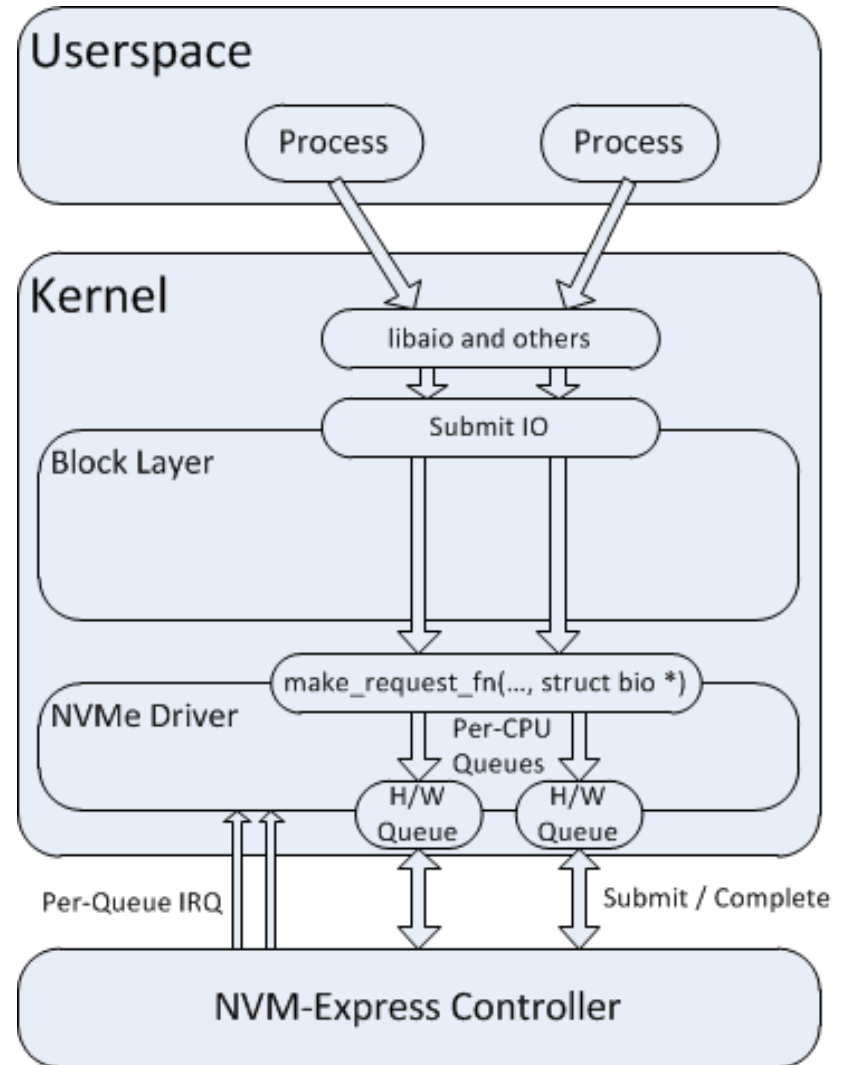
NVMe: When we know we succeeded

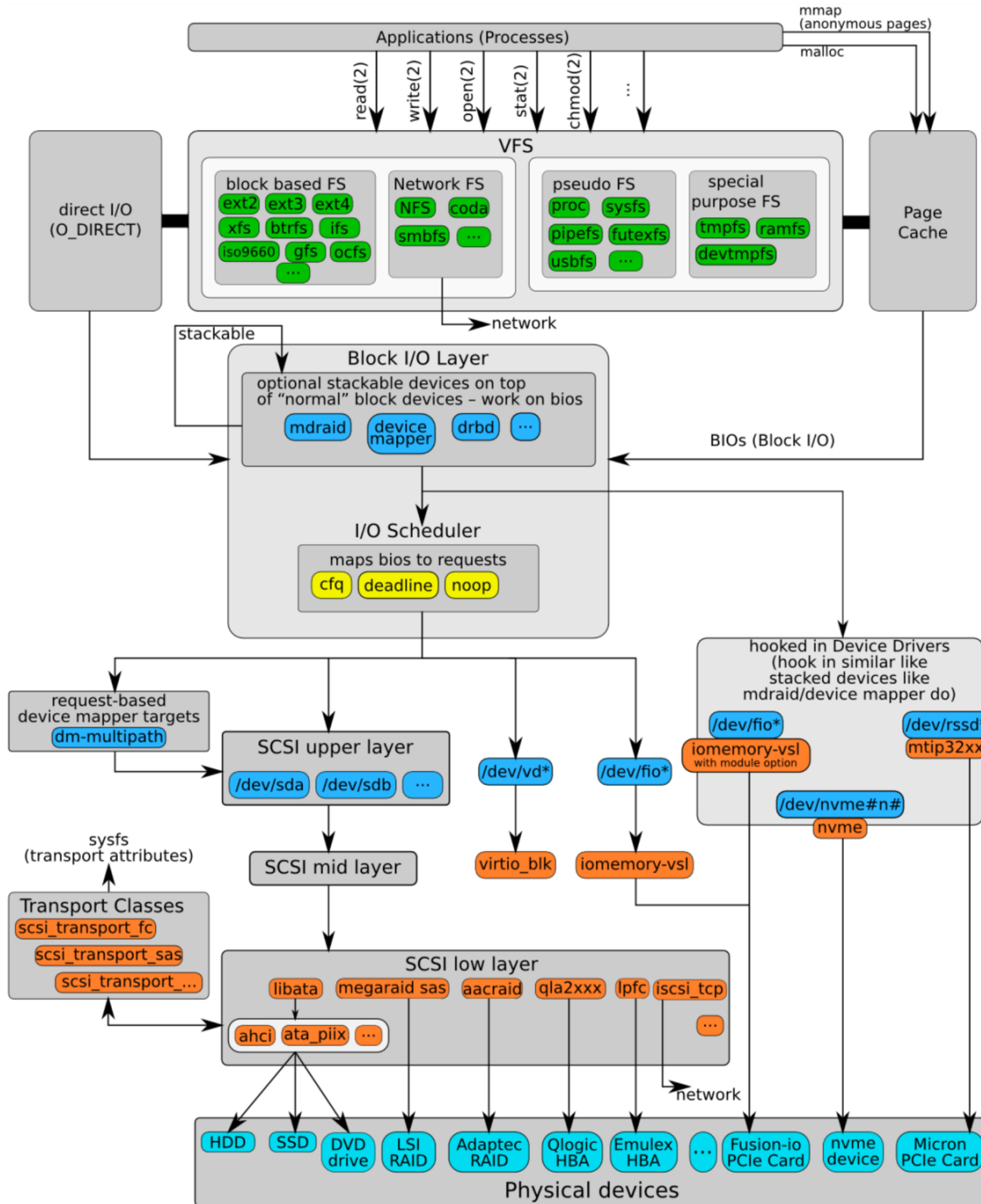
```
root@pc# grep "SCSI\|NVM" .config  
CONFIG_BLK_DEV_NVM=y  
# CONFIG_SCSI is not set
```

A decorative graphic at the bottom of the slide showing a stylized, colorful circuit board or microchip design, transitioning from blue and white at the top to yellow and red at the bottom.

NVMe: Original Driver Implementation

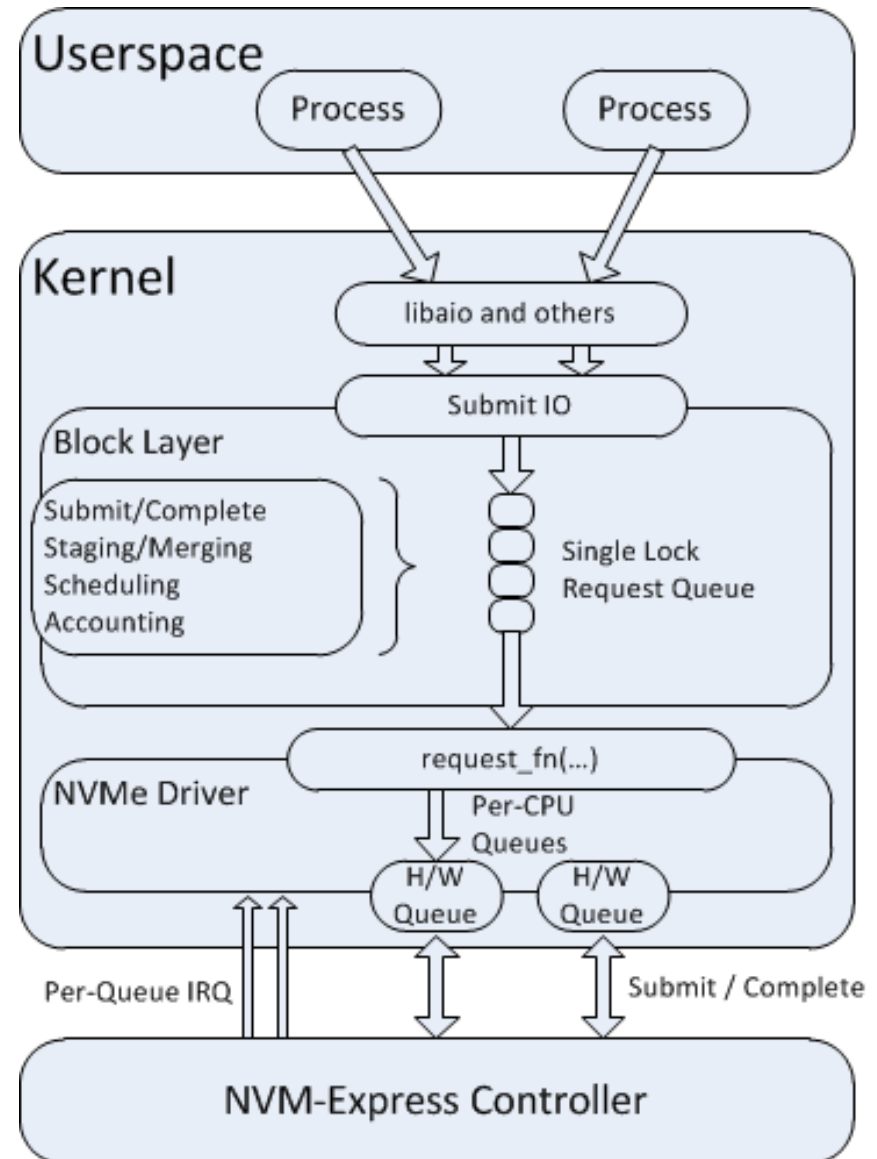
- bio-based for performance: lockless block layer
- Driver burdened to manage:
 - timeouts, io statistics, h/w access, tagging, SGL mapping, trace points, queue-to-cpu binding, splitting





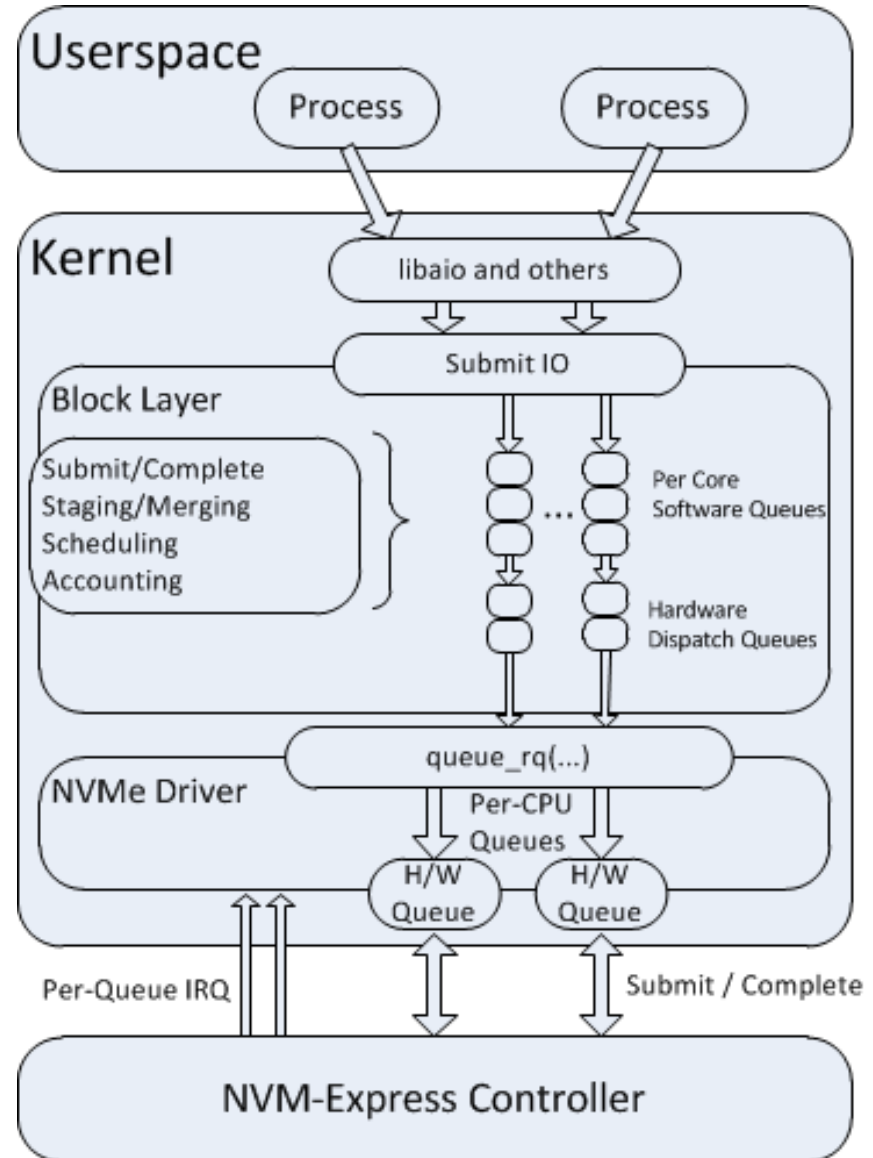
Converting NVMe to Request Based

- High IOPS devices cannot reach their potential under single lock

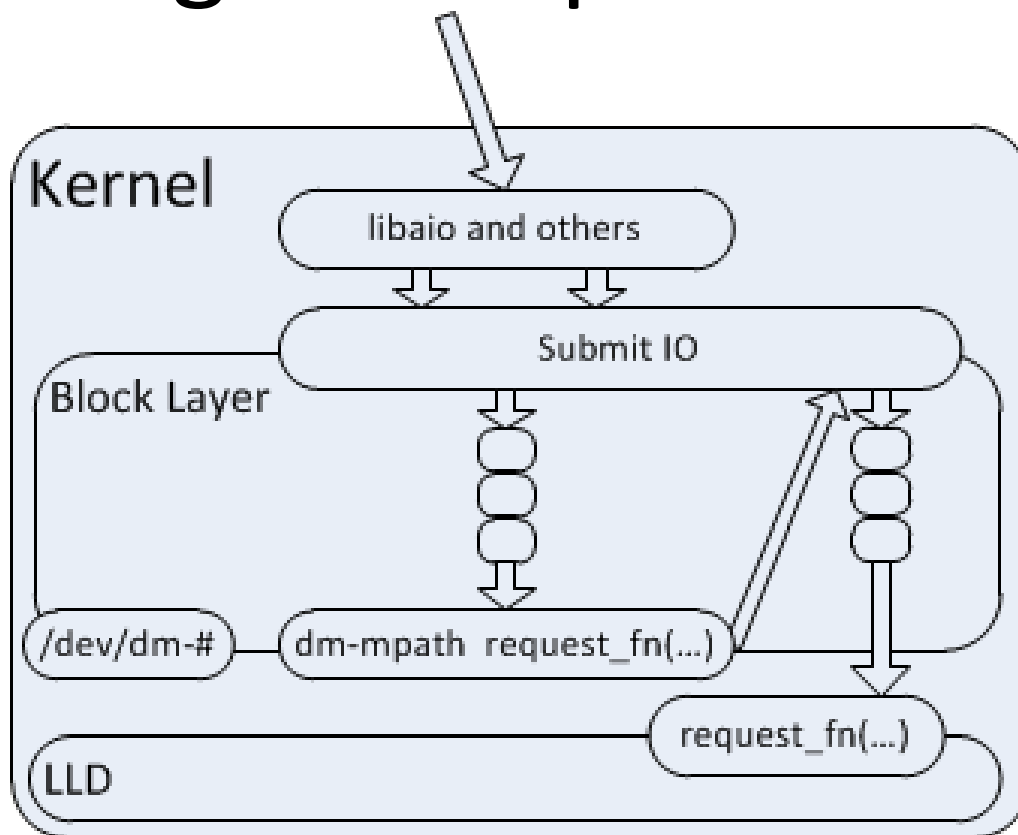


Converting NVMe to Request Based

- High IOPS devices cannot reach their potential under single lock
- But blk-mq can be multithreaded all the way to the h/w

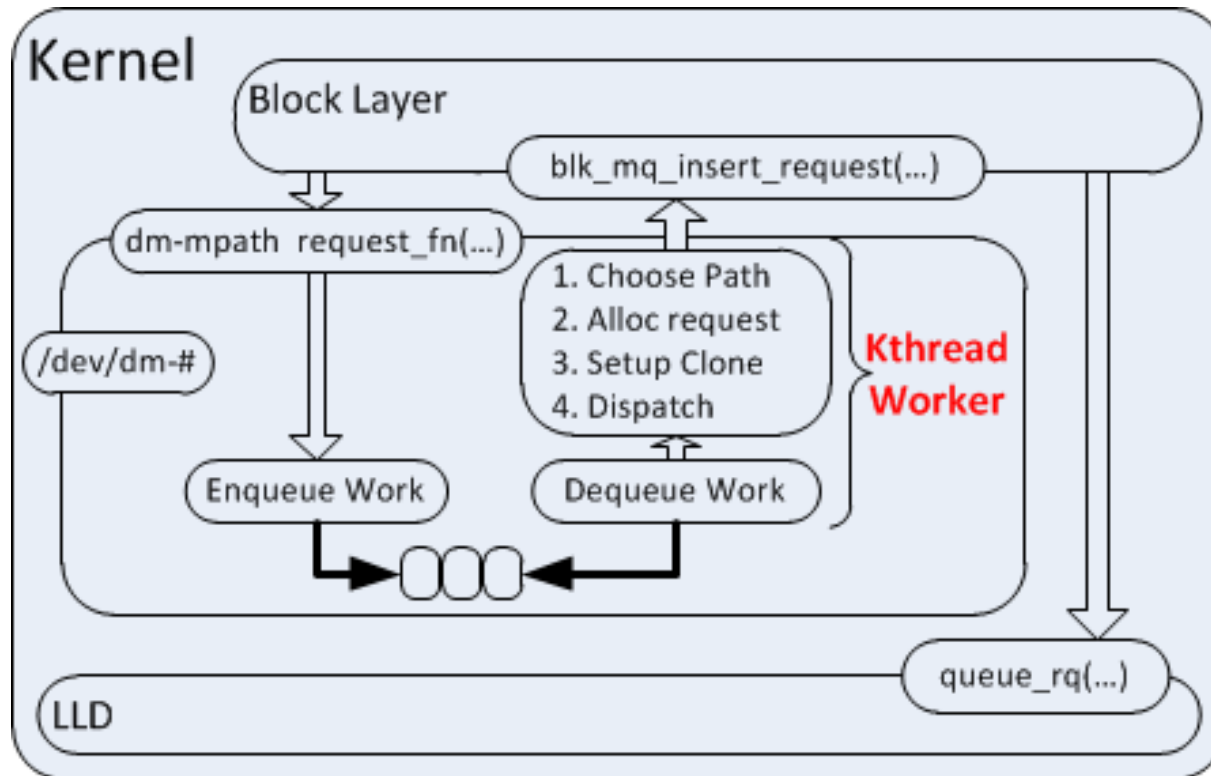


Supporting blk-mq from dm-mpath



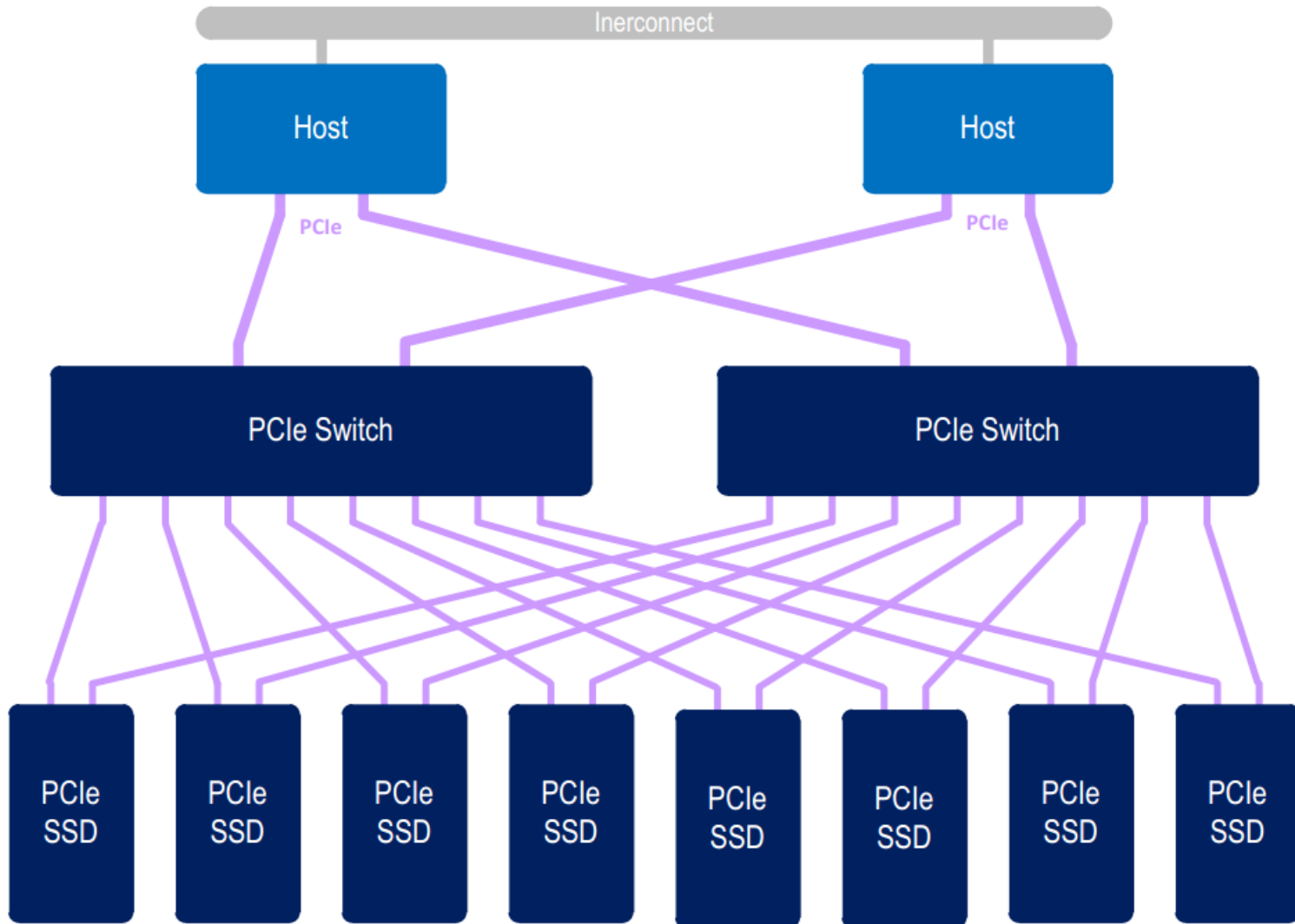
- Problems:
 - Required clone before path chosen
 - Submission occurs in atomic context

Supporting blk-mq from dm-mpath

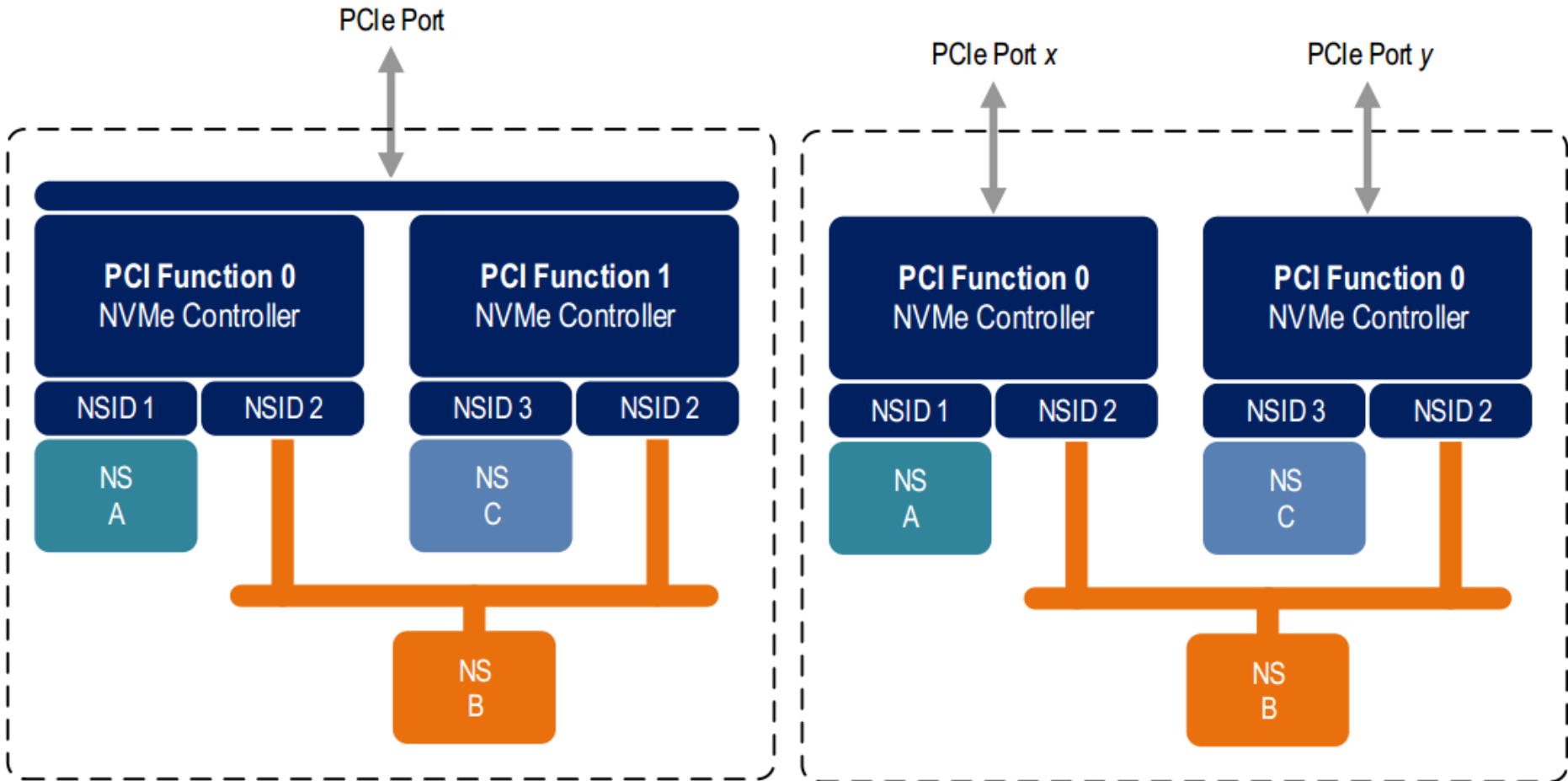


- Request handling deferred to worker thread
- Clone allocated from path's request_queue

PCI-e Storage Multipathing

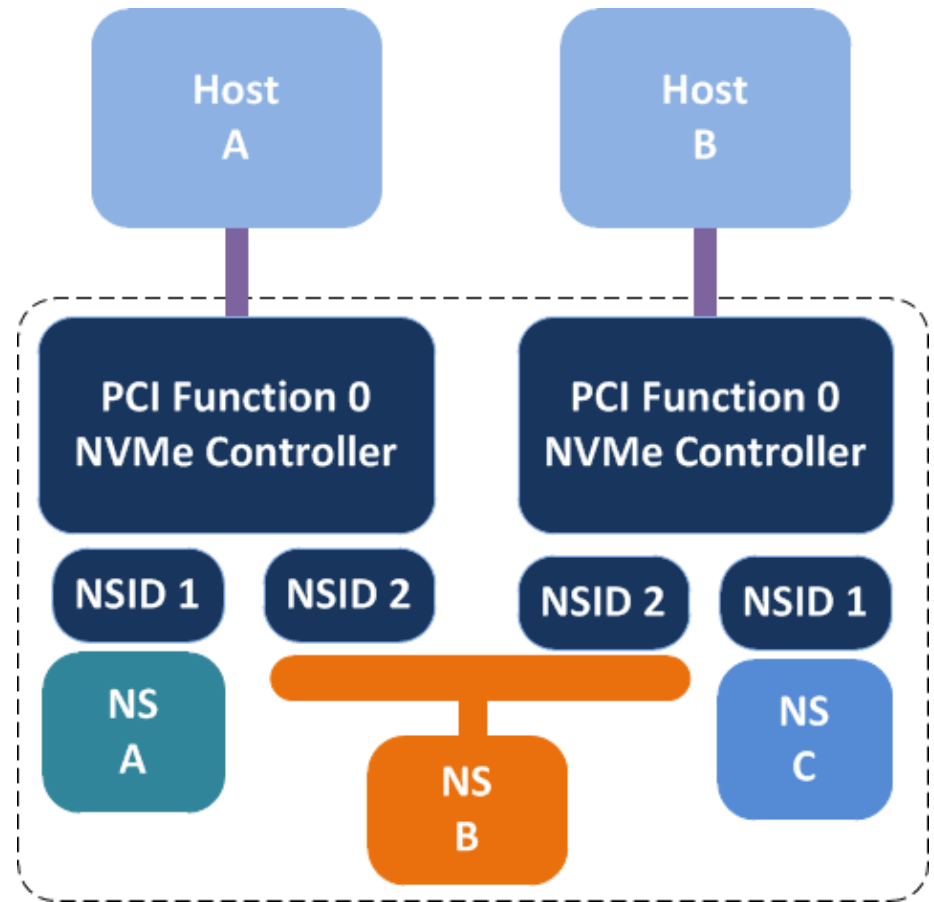


NVM-e: Subsystems

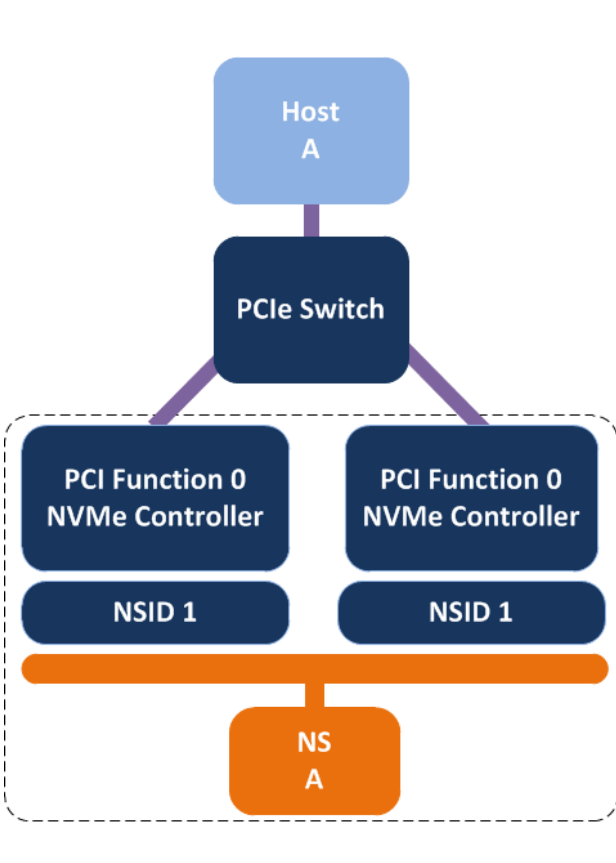


NVMe: Identifying Paths

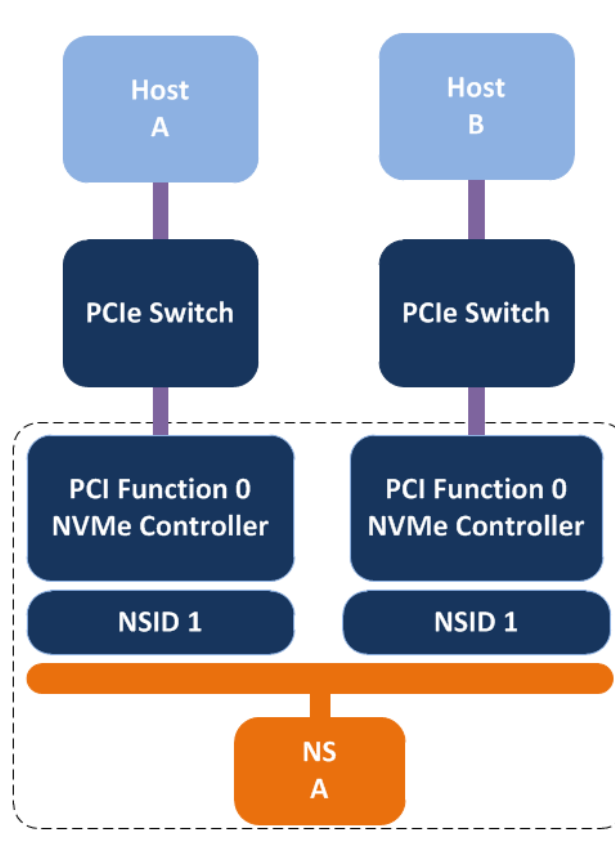
- IEEE EUI-64 and NGUID globally unique identifiers in a subsystem
- Linux tooling relies on SG_IO to inquire device identification and access restriction



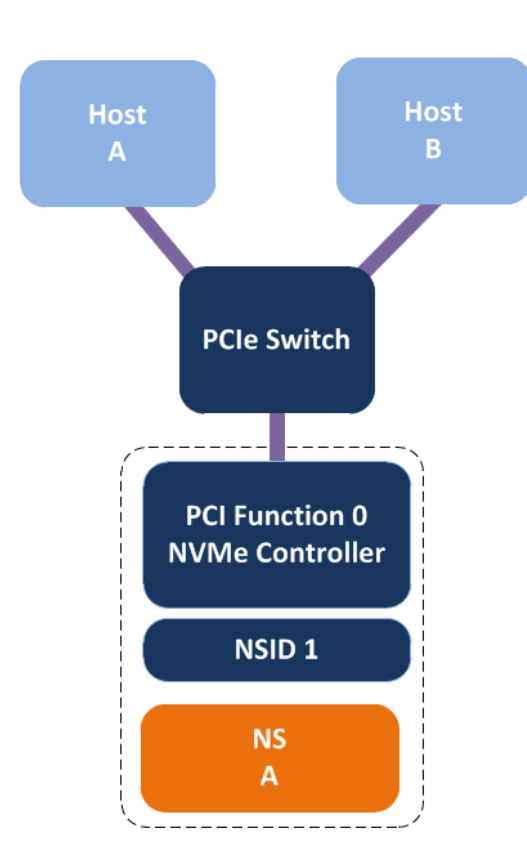
Know your PCI-e Topologies: Which is invalid?



A



B



C

blk-mq + dm-mpath in kernel 4.0-rc1

Credits:

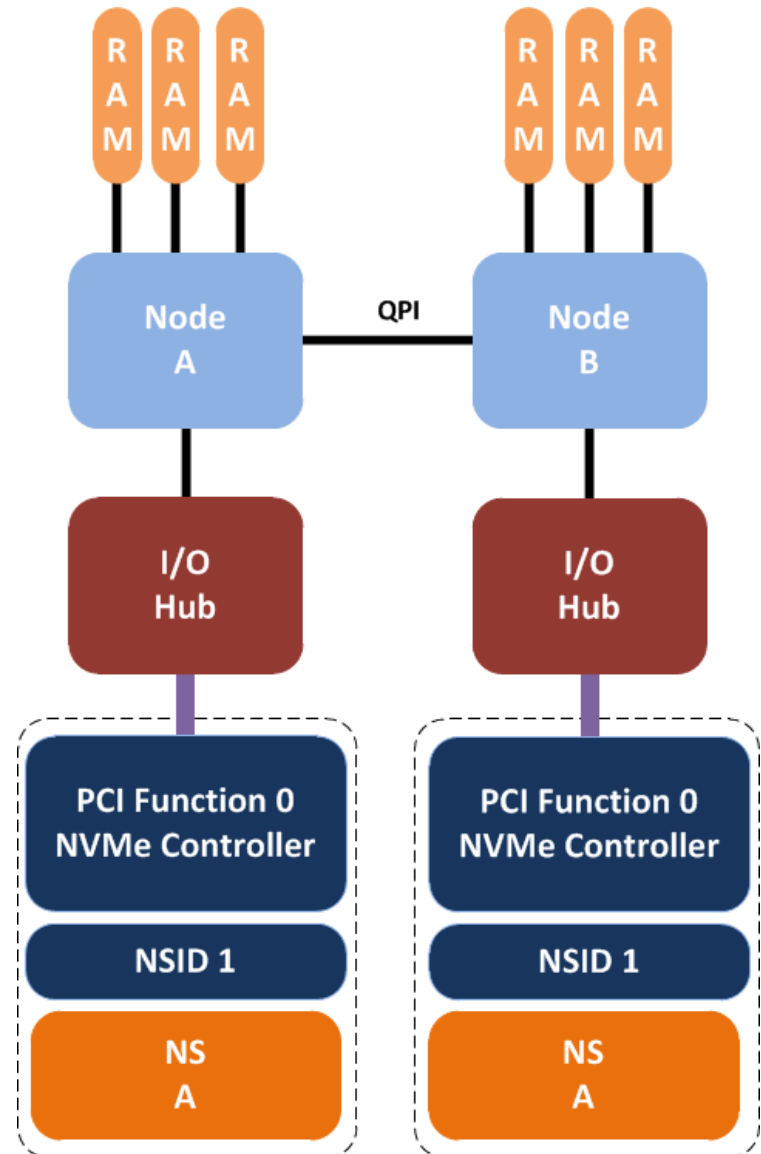
- Matias Bjørling: nvme conversion
- Mike Snitzer: multipath device-mapper
- Jens Axboe: block multiqueue
- Bart Van Assche: regression debugging
- Christoph Hellwig: moral support

Still more work to do!



Multipathing for performance

- Submitting I/O to device on remote NUMA node incurs additional latency
- Worsens as node count increases



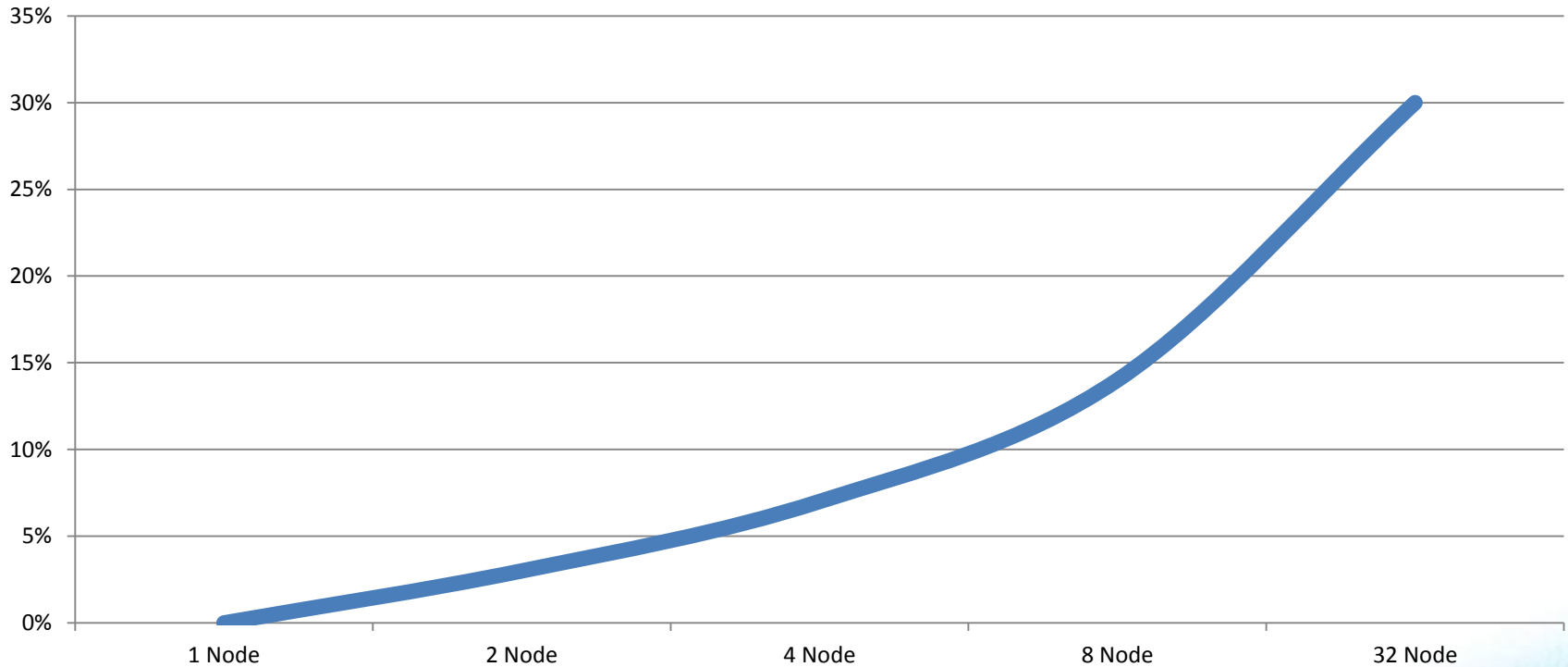
Case Study: 32 Sockets, 960 CPUs

- NUMA penalty: >30% performance lost
- NUMA “trickery” recovered:
 - irqbalance, numactl, libnuma, custom cpu-queue mapping
 - 30 **Million** IOPS (SC’14)



The cost of NUMA

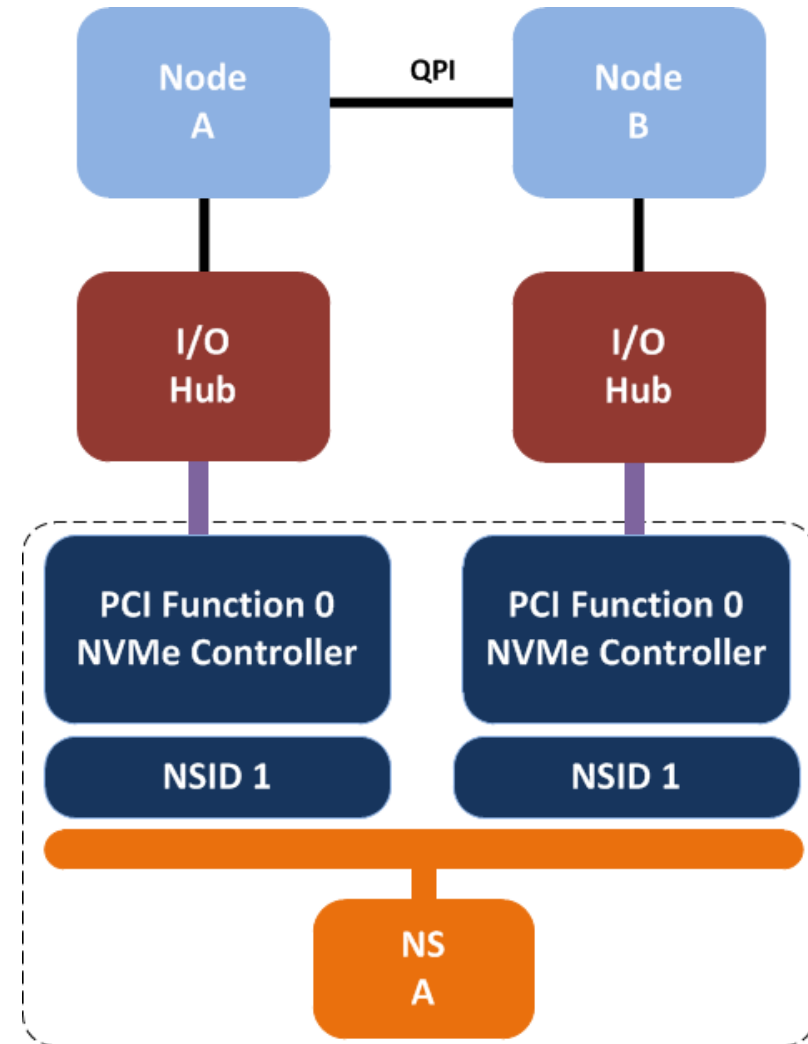
Observed Performance Loss on Randomly Scheduled Workloads



Number NVMe: 2x node count, 4k random read to all drives
No CPU pinning

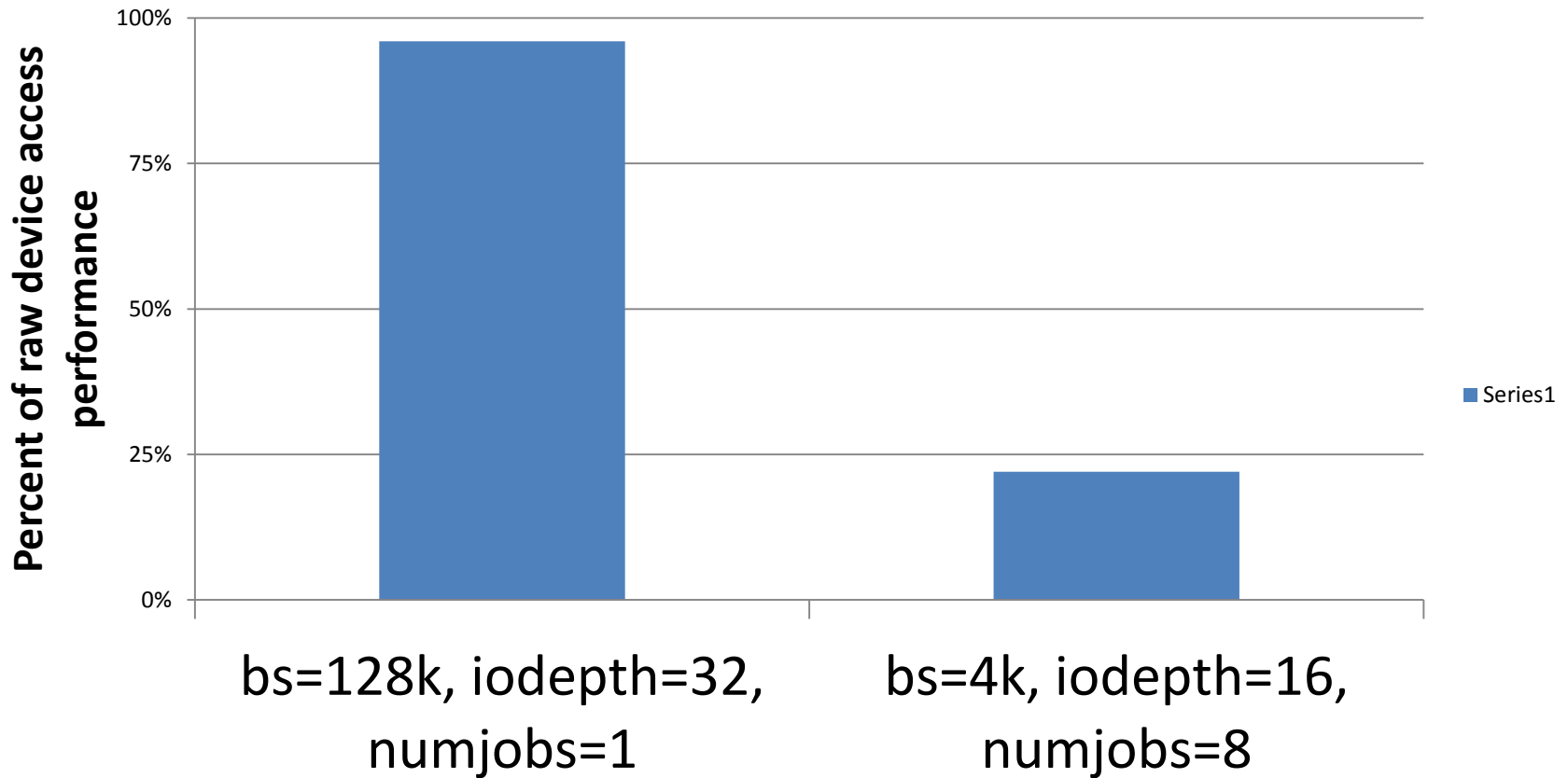
Locality based path selection proposal

- NUMA aware path selection: choose path closest to dispatching CPU
- Ineffective in 4.0: single threaded dispatch



NVMe mpath performance in 4.0

Request based DM performance comparison



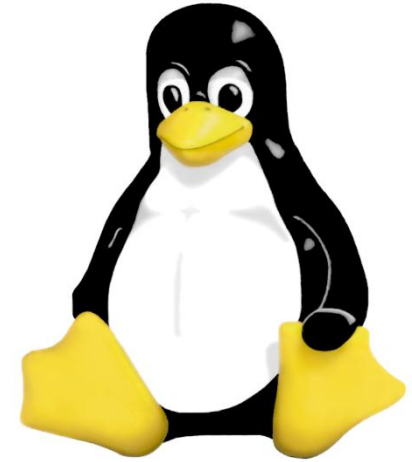
device-mapper blk-mq conversion

- Parallelizes entire stacking layer
 - Make blk-mq entry capable in preempt disabled context
- queue setup: how to determine number of “h/w” queues and tags per queue to allocate
 - Want enough to satisfy h/w, but not wasteful
- Developed by Mike Snitzer during LSFMM
 - performs at 99% raw speed in fio benchmarks on tested NVMe h/w
- Staged for Linux kernel 4.1 integration

Alternative multipath proposal

- Make blk-mq multipath aware
 - Removes stacking/re-entry requirement
 - More efficient use of resources
 - Tighter integration to h/w
- Ideas initiated by Hannes Reinecke and Christoph Hellwig
- Implementation and collaboration details ongoing
 - PoC expected within weeks

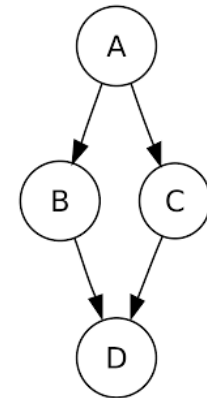




Thank you



keith.busch@intel.com



Backup

Path selection code snippet

```
static struct dm_path *numa_select_path(struct path_selector *ps,
                                       unsigned *repeat_count, size_t nr_bytes)
{
    struct selector *s = (struct selector *) ps->context;
    struct path_info *best = NULL, *pi;
    int cur = INT_MAX, node = cpu_to_node(smp_processor_id());

    list_for_each_entry(pi, &s->valid_paths, list) {
        int pnode = pi->path->dev->bdev->bd_queue->node;
        int val = node_distance(node, pnode);
        if (val < cur) {
            best = pi;
            cur = val;
        }
    }
    *repeat_count = 1;
    return best ? best->path : NULL;
}
```