Linux Block IO: Introducing Multiqueue SSD Access on Multicore Systems

Matias Bjørling[1,2], Jens Axboe[2], David Nellans[2], Philippe Bonnet[1]

2013/03/05

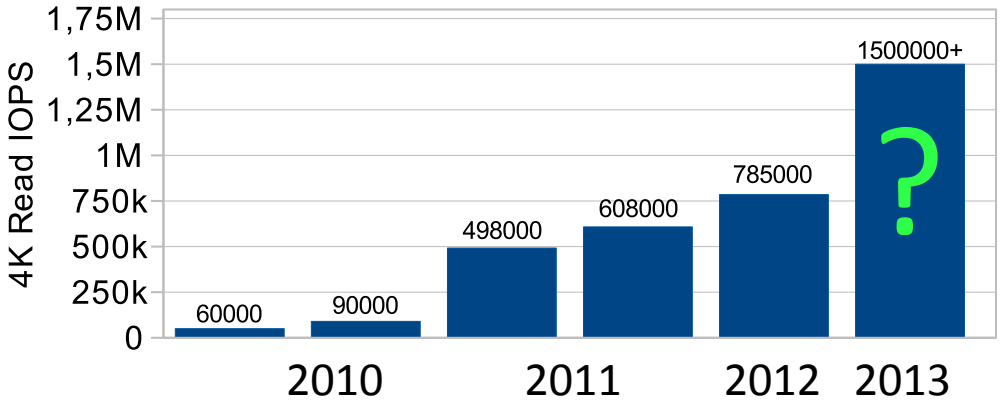
[1]IT University of Copenhagen - Fusion-io[2]

IT University of Copenhagen

FUSION-IO

Non-Volatile Memory Devices

- Have higher IOPS and lower latency than magnetic disks
- Continues to improve
  - Parallel architecture
  - Future NVM technologies (PCM, MRAM, etc.)

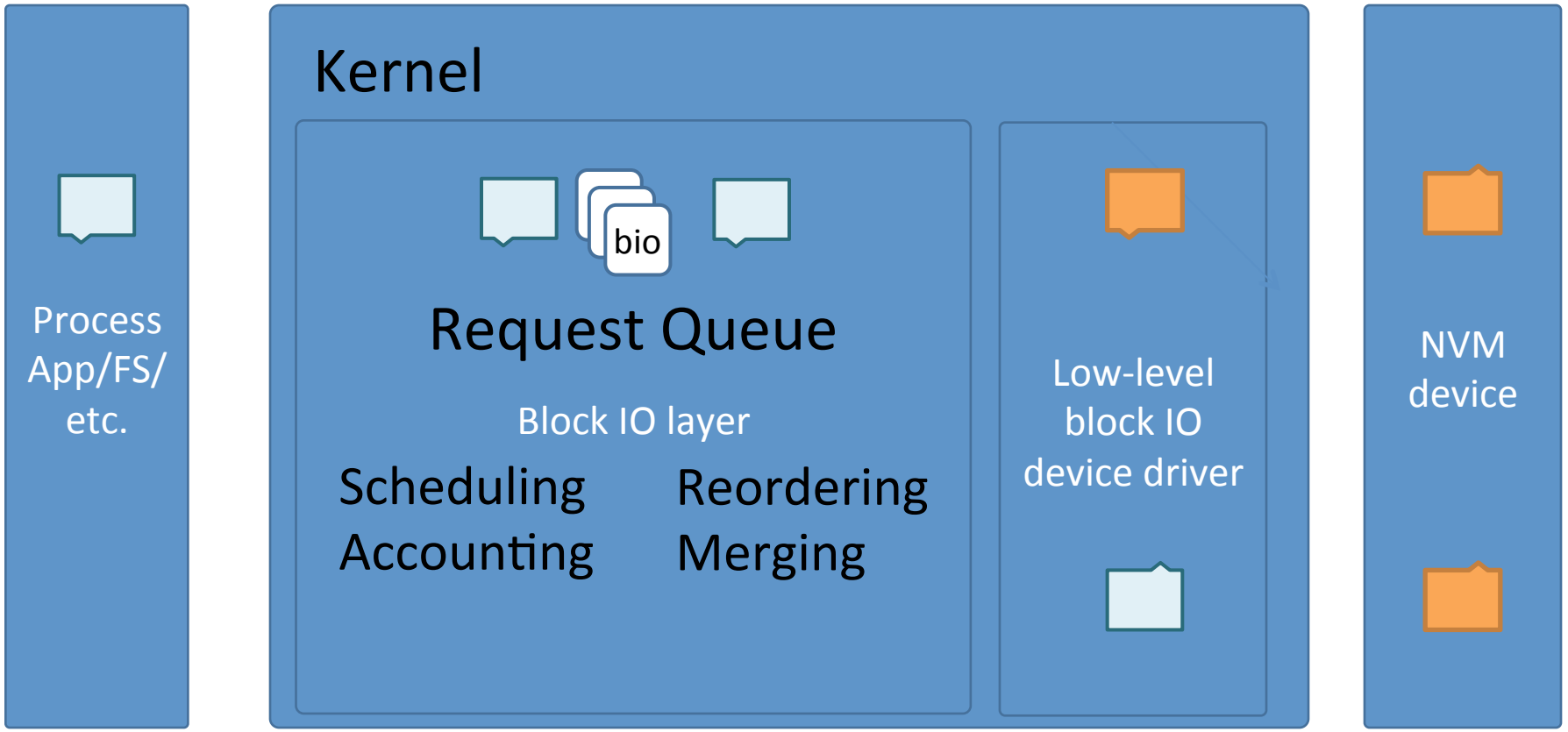


- Software design decisions
  - Fast sequential access
  - Slow random access

The Block Layer in the Linux Kernel

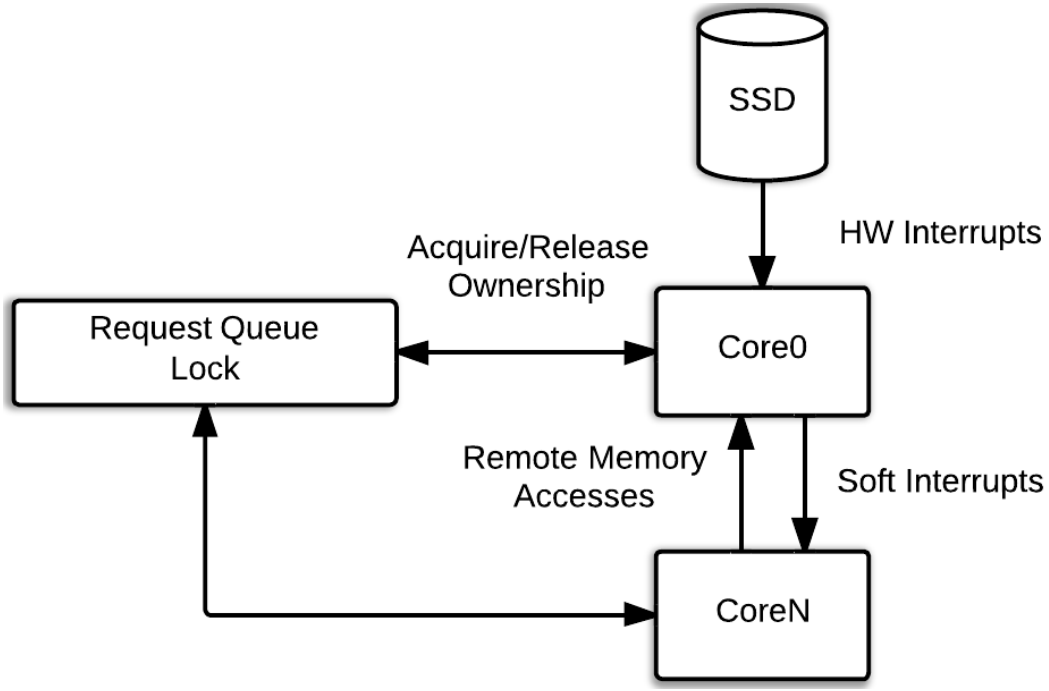
- Provides
  - Single point of access for block devices
  - IO scheduling
  - Merging & reordering
  - Accounting
- Optimized for traditional hard-drives
  - Trade CPU cycles for sequential disk accesses
  - Single queue -> Performance bottleneck

In-Kernel IO Submission / Completion



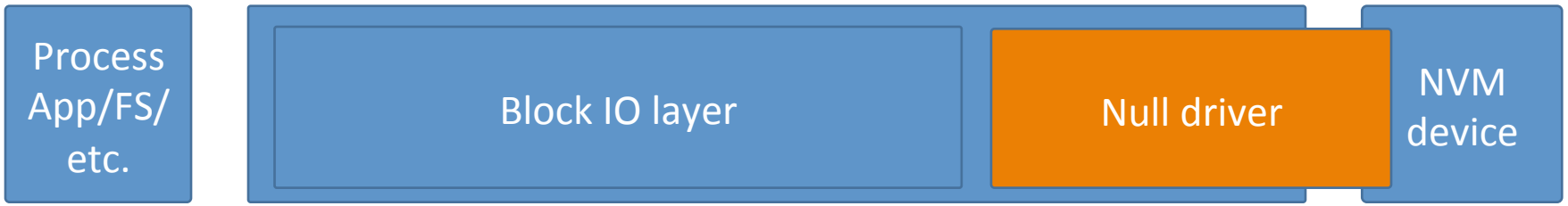
Block Layer IO for Disks

- Single request queue
  - Single lock data structure
  - Cache-coherence expensive on multi-CPU systems

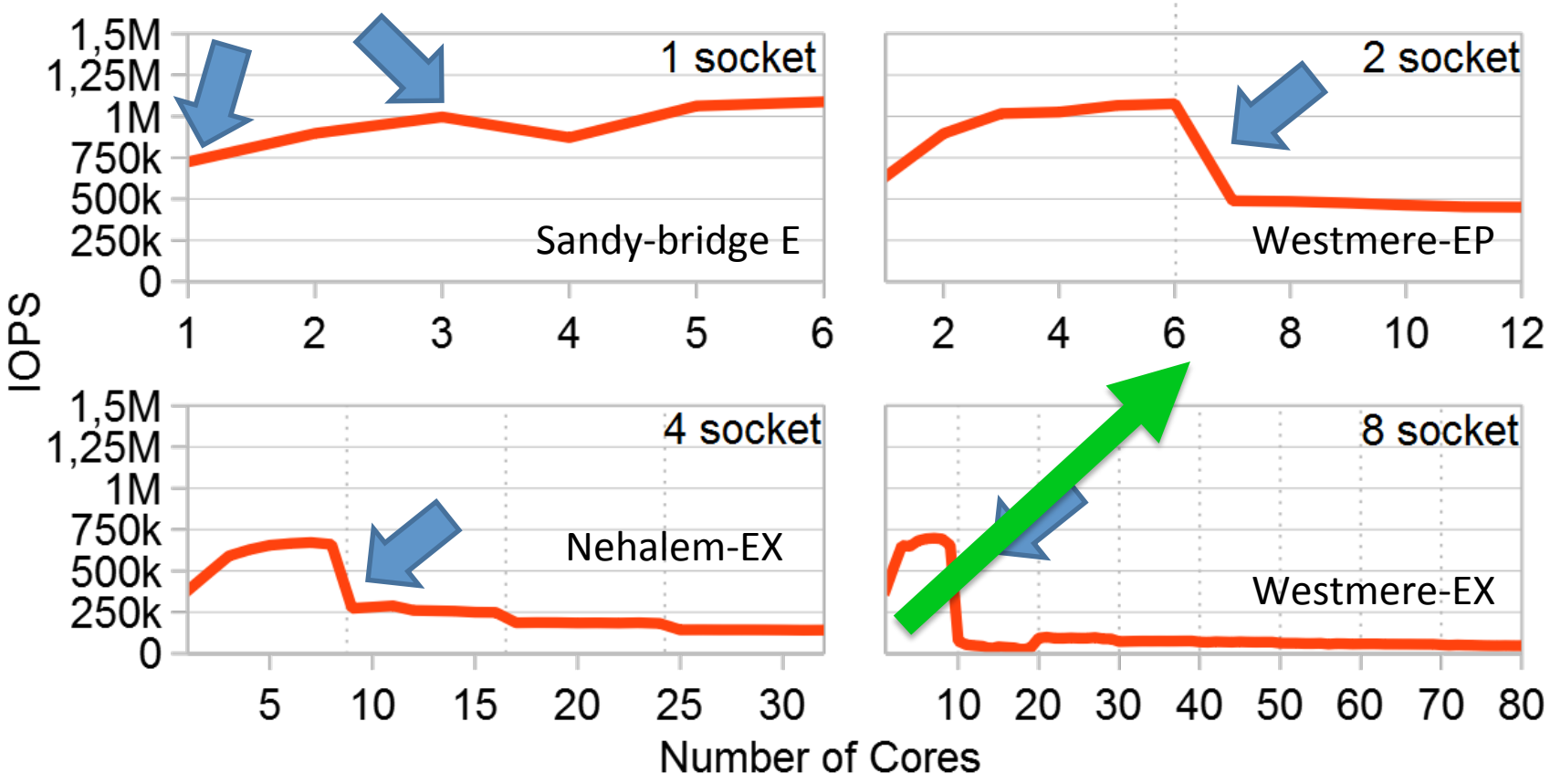


Methodology

- Null block device driver
  - In kernel dummy device
  - Acts as proxy for a no latency device
  - Avoids the DMA overhead
  - Removes driver effects from block layer
  - ”Device” completions can be in-line or soft irq



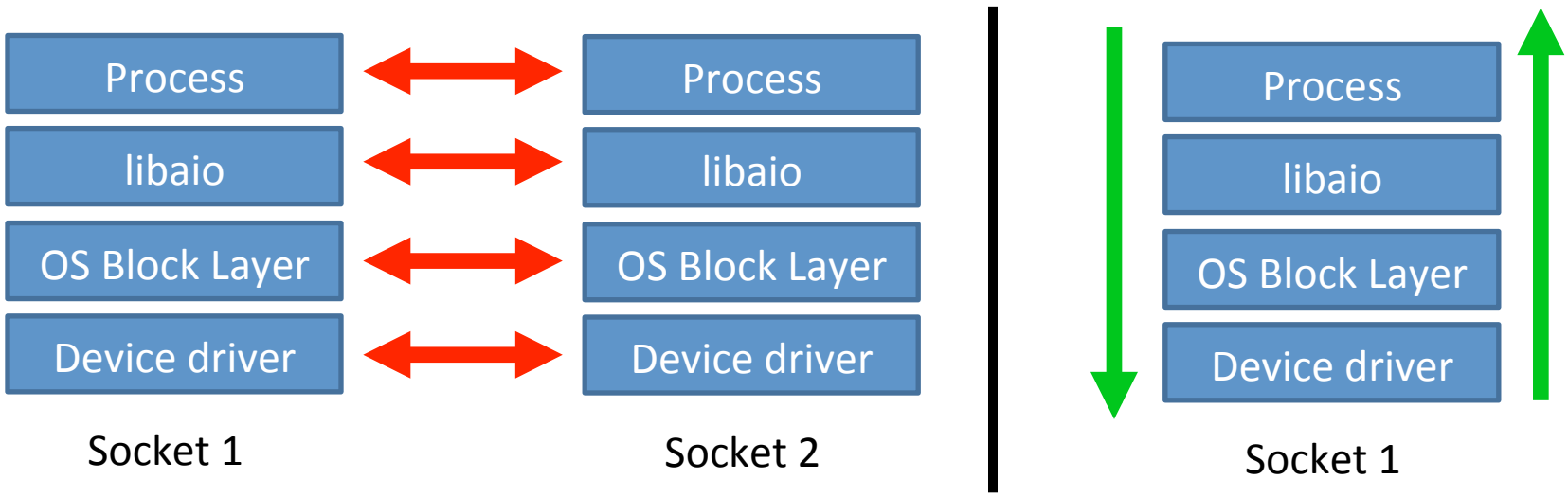
Current Block Layer (IOPS)



Null block device, noop IO scheduler, 512B random reads

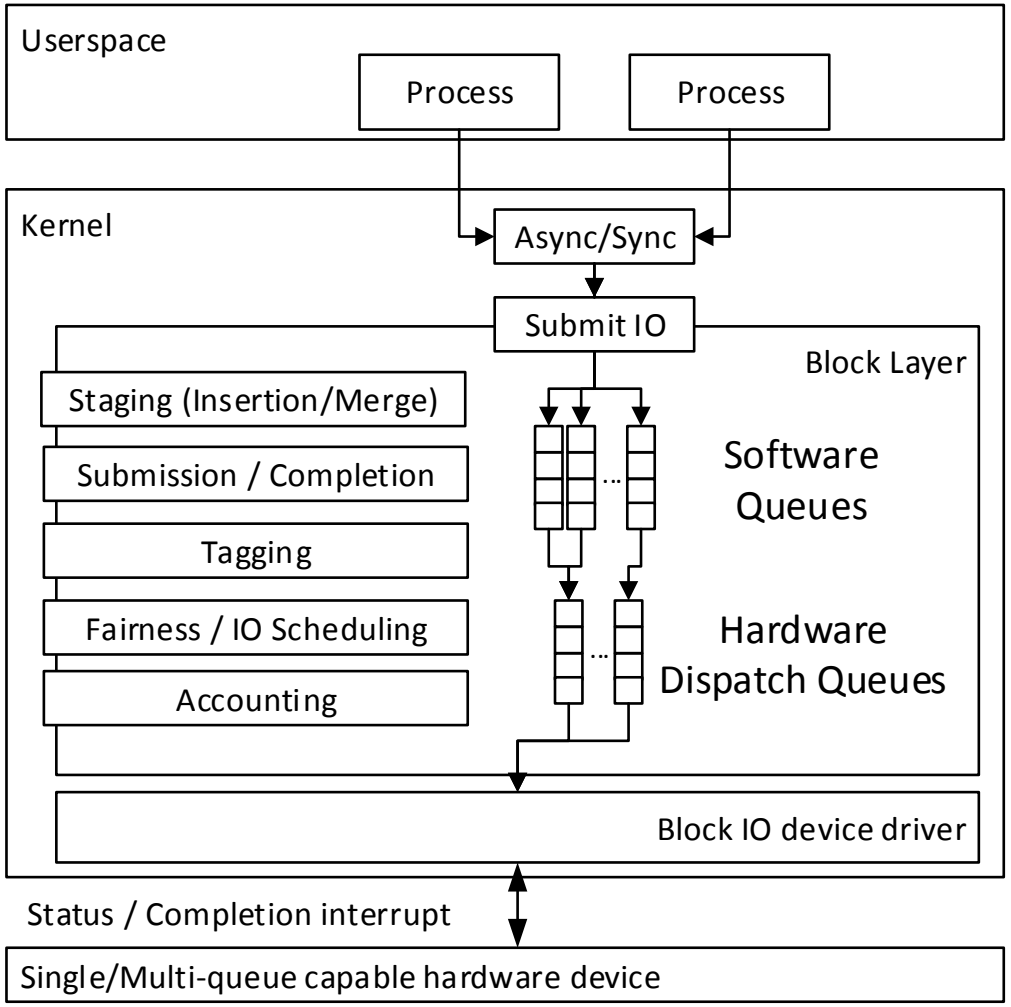
Multi-Queue Block Layer

- Balance IO workload across cores
- Reduce cache-line sharing
- Similar functionality as SQ
- Allow multiple hardware queues

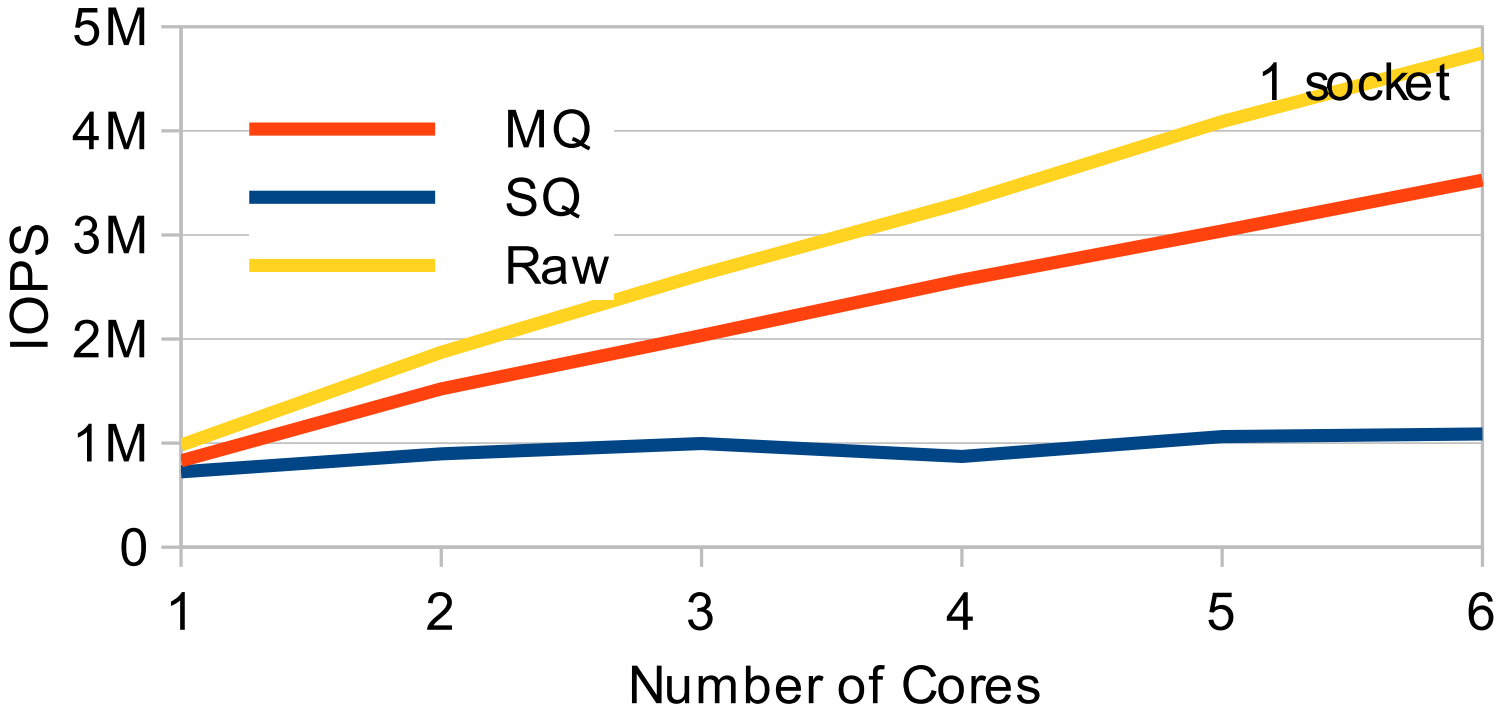


Multi-Queue Block Layer

- Two levels of queues
  - Software Queues
  - Hardware Dispatch Queues
- Per CPU accounting
- Tag-based completions

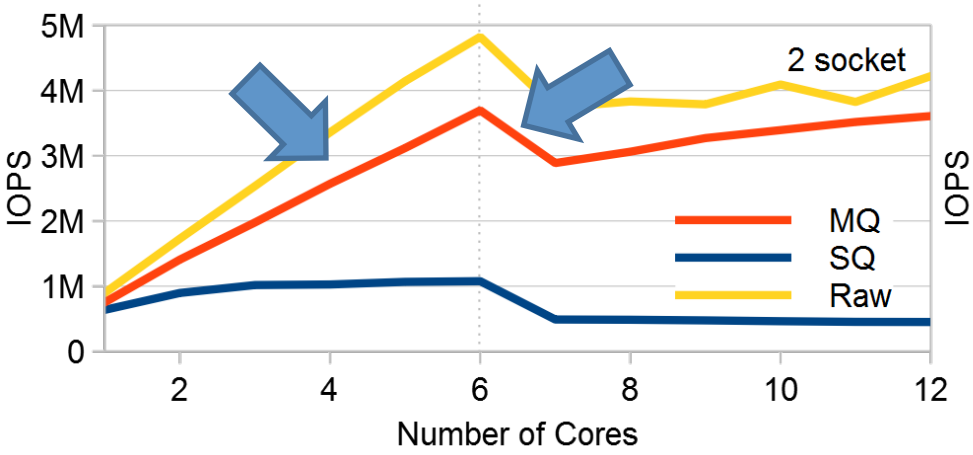


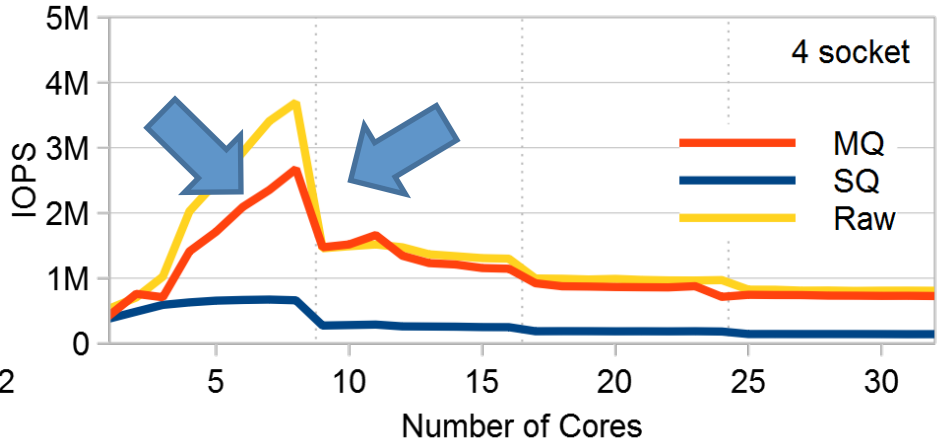
Multi-Queue IOPS

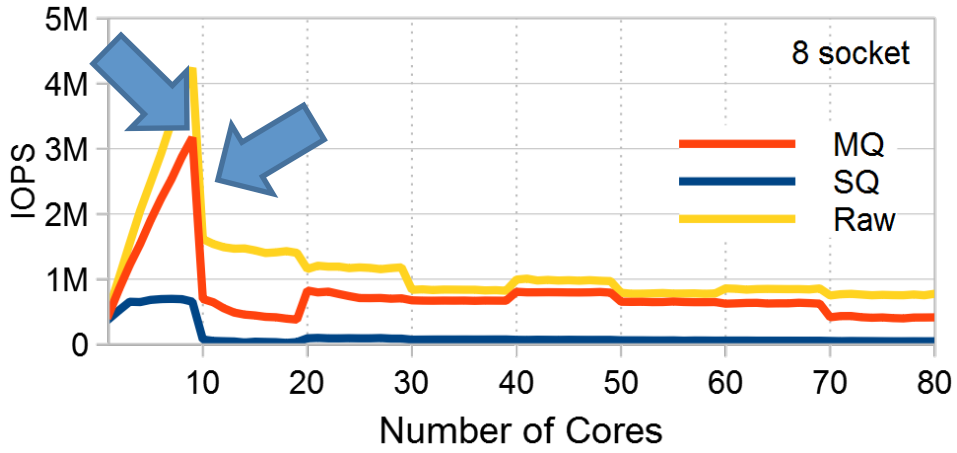


SQ – Current block layer, Raw – Block layer bypass, MQ – Proposed design

Multi-socket Scaling

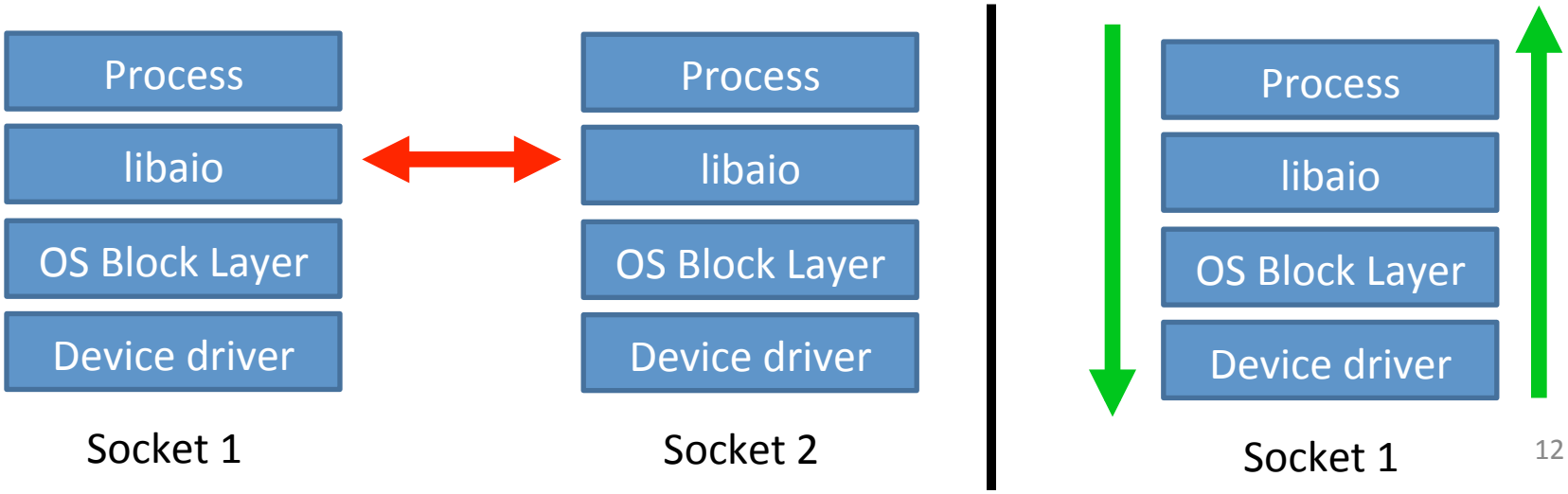




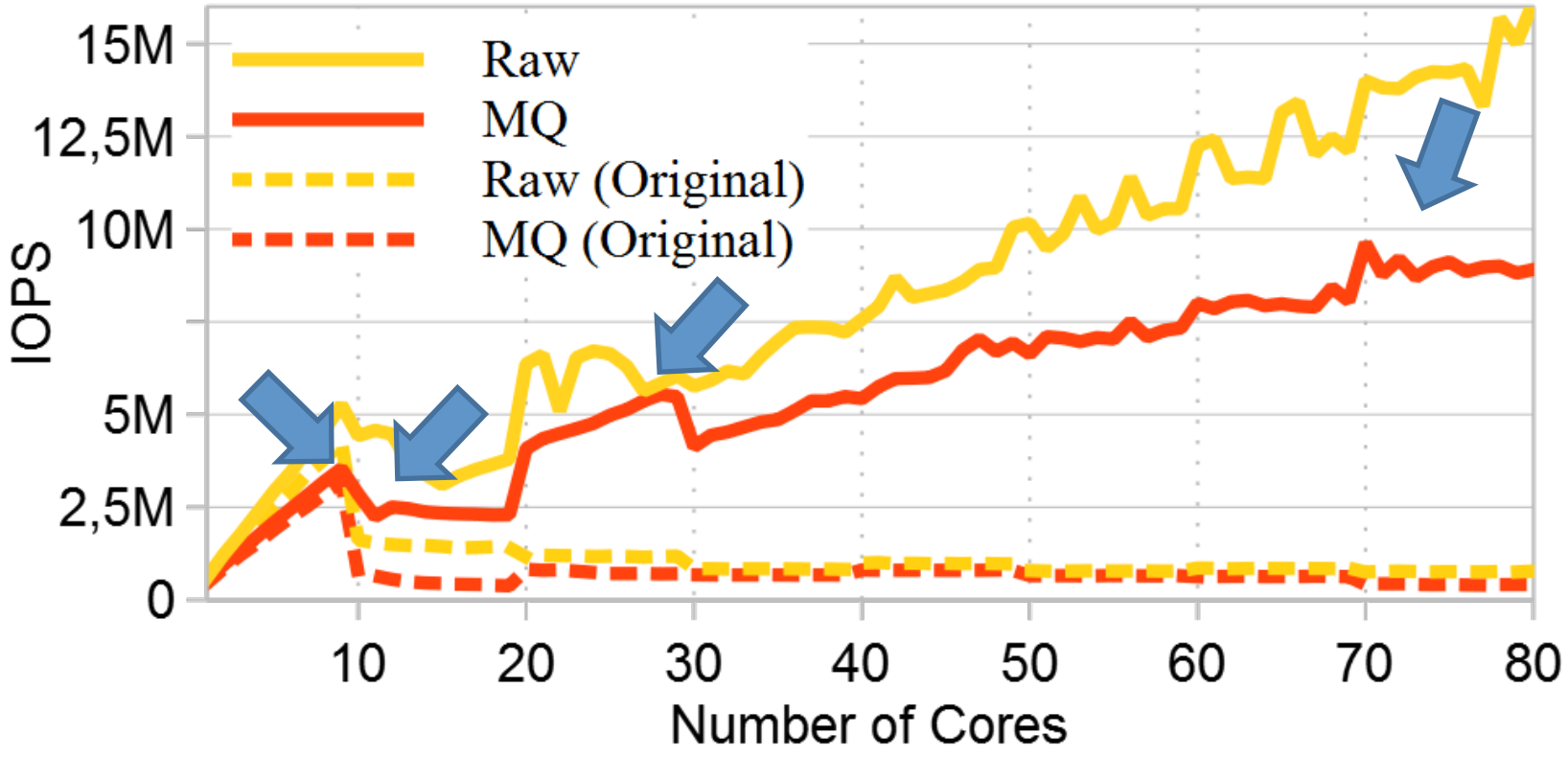


AIO interface

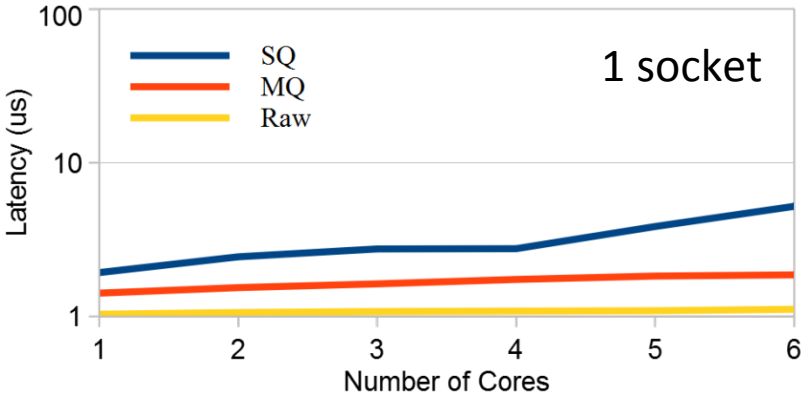
- Alternative interface to synchronous
- Prevents blocking of user threads
- Move bio list to a lockless list
- Remove legacy code and locking

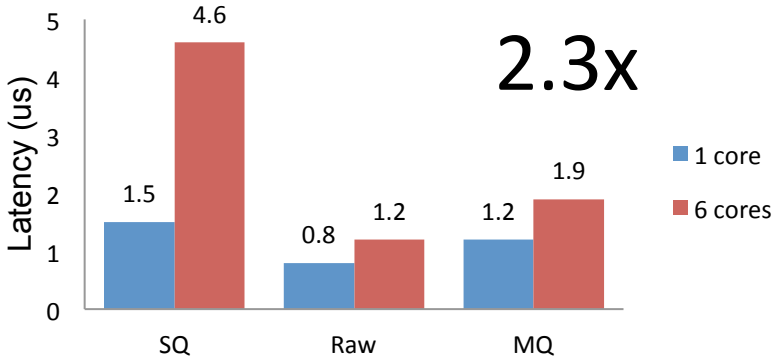


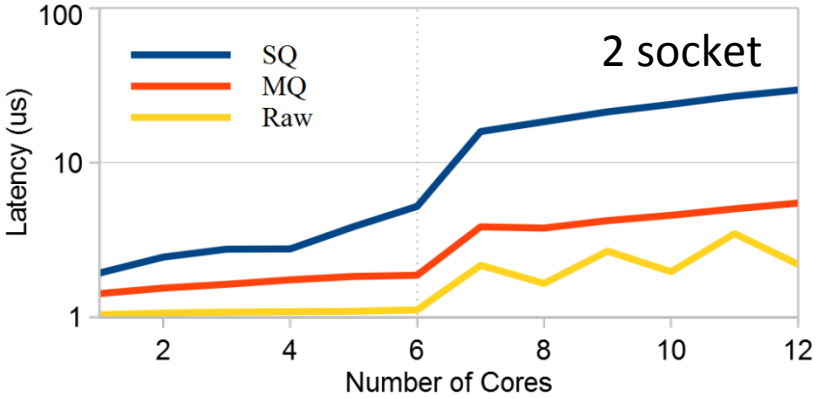
Multi-Queue, 8 socket

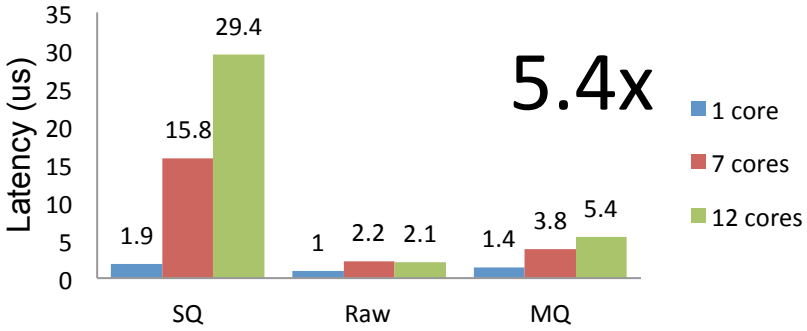


Latency Comparison



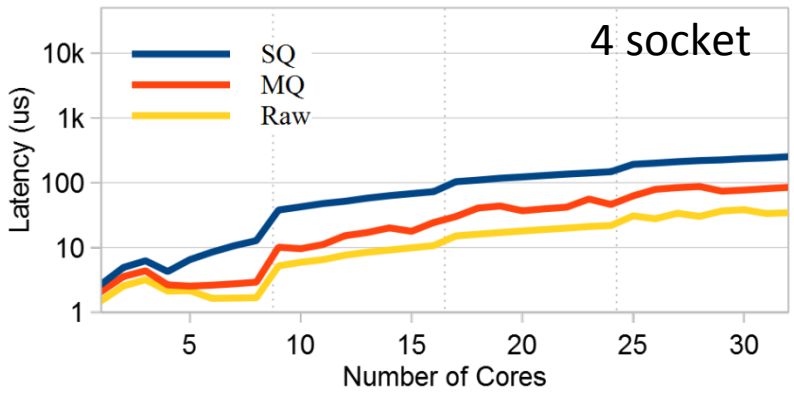


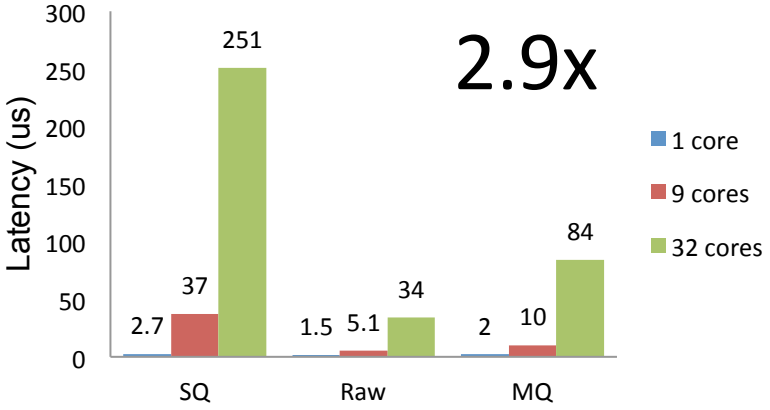


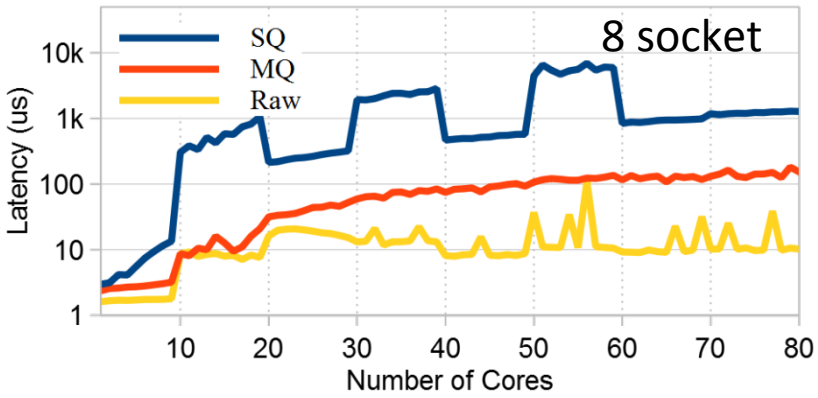


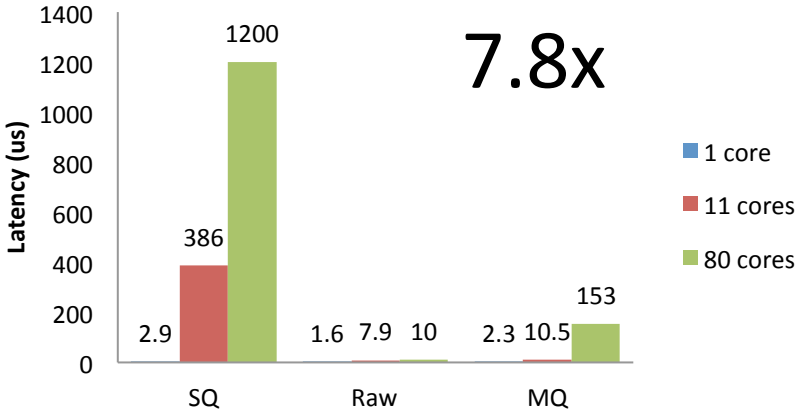
- Sync IO, 512B
- Queue depth grows with CPU count

Latency Comparison



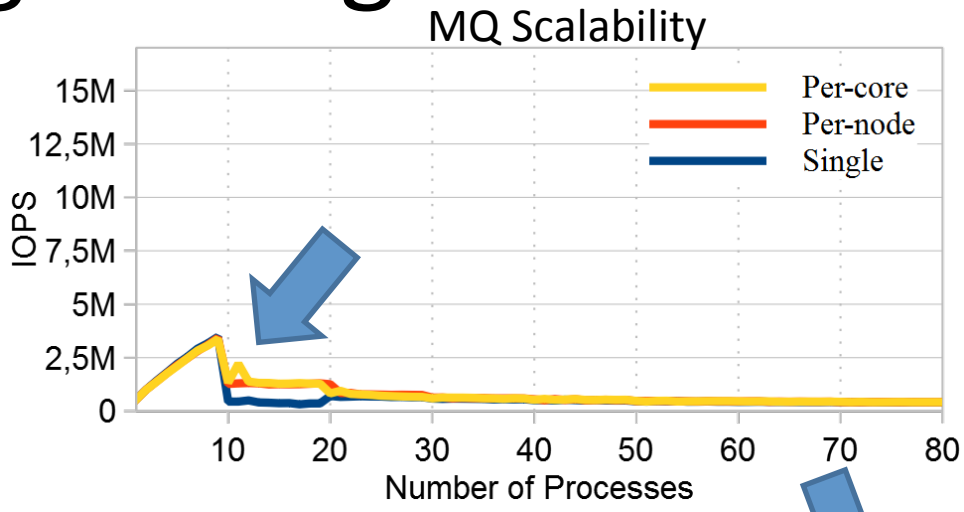




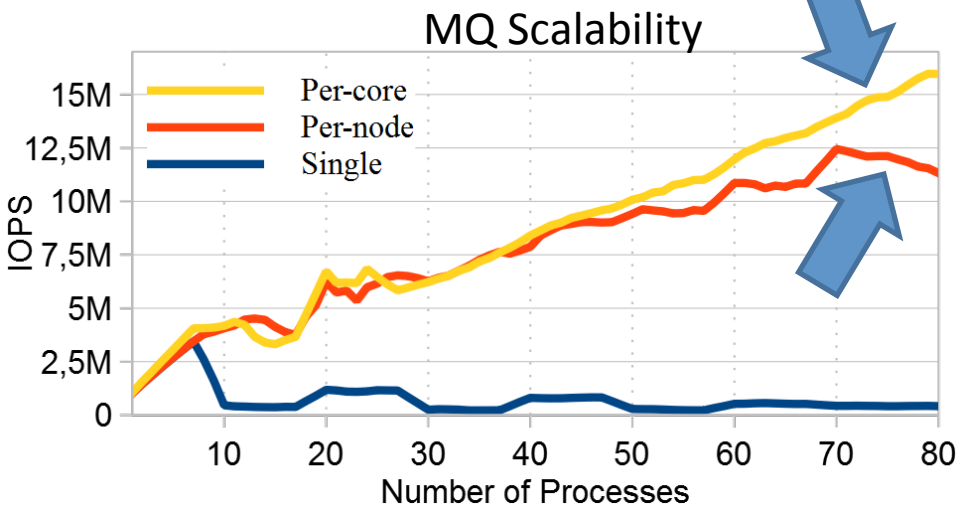


Mapping Strategies

- Single software queue
- Varying hardware dispatch queues



- Per-core software queues
- Varying hardware dispatch queues



Scalability = Multiple hardware dispatch queues

Conclusions

- Current block layer is inadequate for NVM
- Proposed Multi-Queue block layer
  - 3.5x-10x increase in IOPS
  - 10x-38x reduction in latency
  - Supports multiple hardware queues
  - Simplifies driver development
  - Maintains industry standard interfaces
- Enters the Linux kernel in the near future

Thank You & Questions

Implementation available at

git://git.kernel.dk/?p=linux-block.git

Systems

- 1 Socket
  - Sandy Bridge-E, i7-3930K, 3.2Ghz, 12MB L3
- 2 Socket
  - Westmere-EP, X5690, 3.46Ghz, 12MB L3
- 4 Socket
  - Nehalem-EX, X7560, 2.66Ghz, 24MB L3
- 8 Socket
  - Westmere-EX, E7-2870, 2.4Ghz, 30MB L3