SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2014

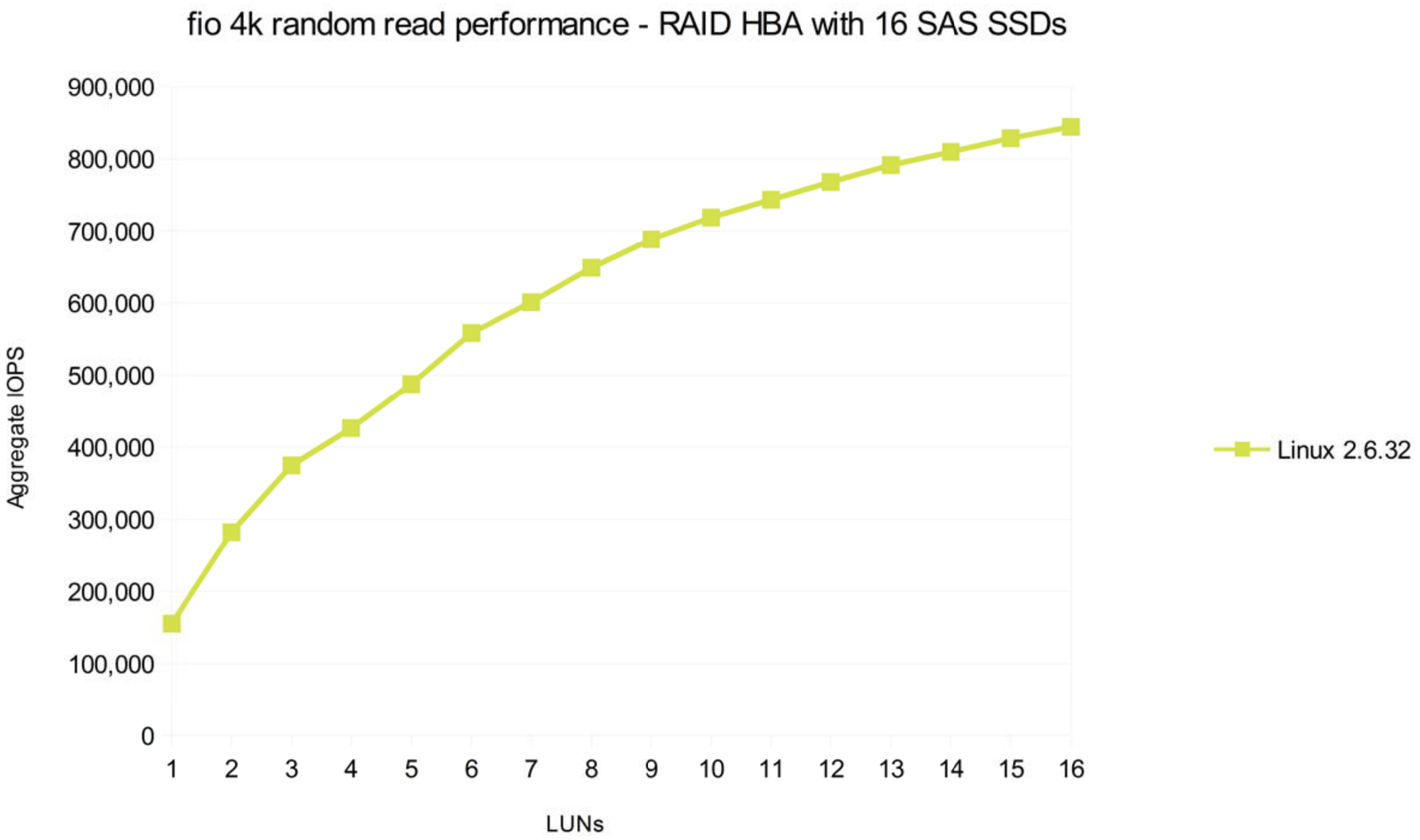
Multiqueue Block Storage in Linux

Christoph Hellwig

Problem Statement

- ❐ The Linux storage stack doesn't scale:
  - ~ 250,000 to 500.000 IOPS per LUN
  - ~ 1,000,000 IOPS per HBA
  - High completion latency
  - High lock contention and cache line bouncing
  - Bad NUMA scaling

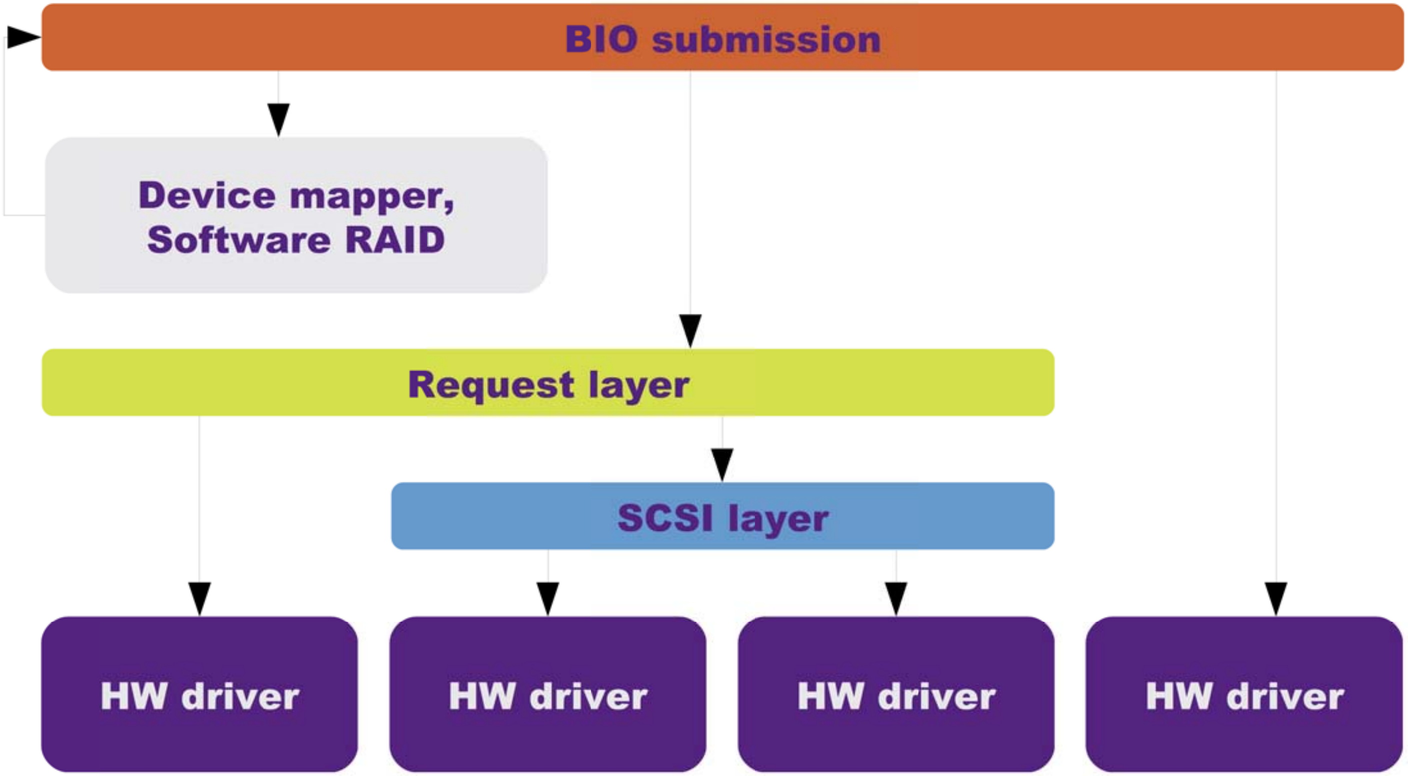
Linux SCSI Performance



Linux Storage Stack - Issues

- The Linux block layer can't handle high IOP or low latency devices
  - *All the block layer?*

Linux Storage Stack



Linux Storage Stack – Issues (2)

- The ***request layer*** can't handle high IOPS or low latency devices
- Vendors work around by implementing make_request based drivers
  - Lots of code duplication
  - Missing features
- SCSI drivers are tied into the request framework

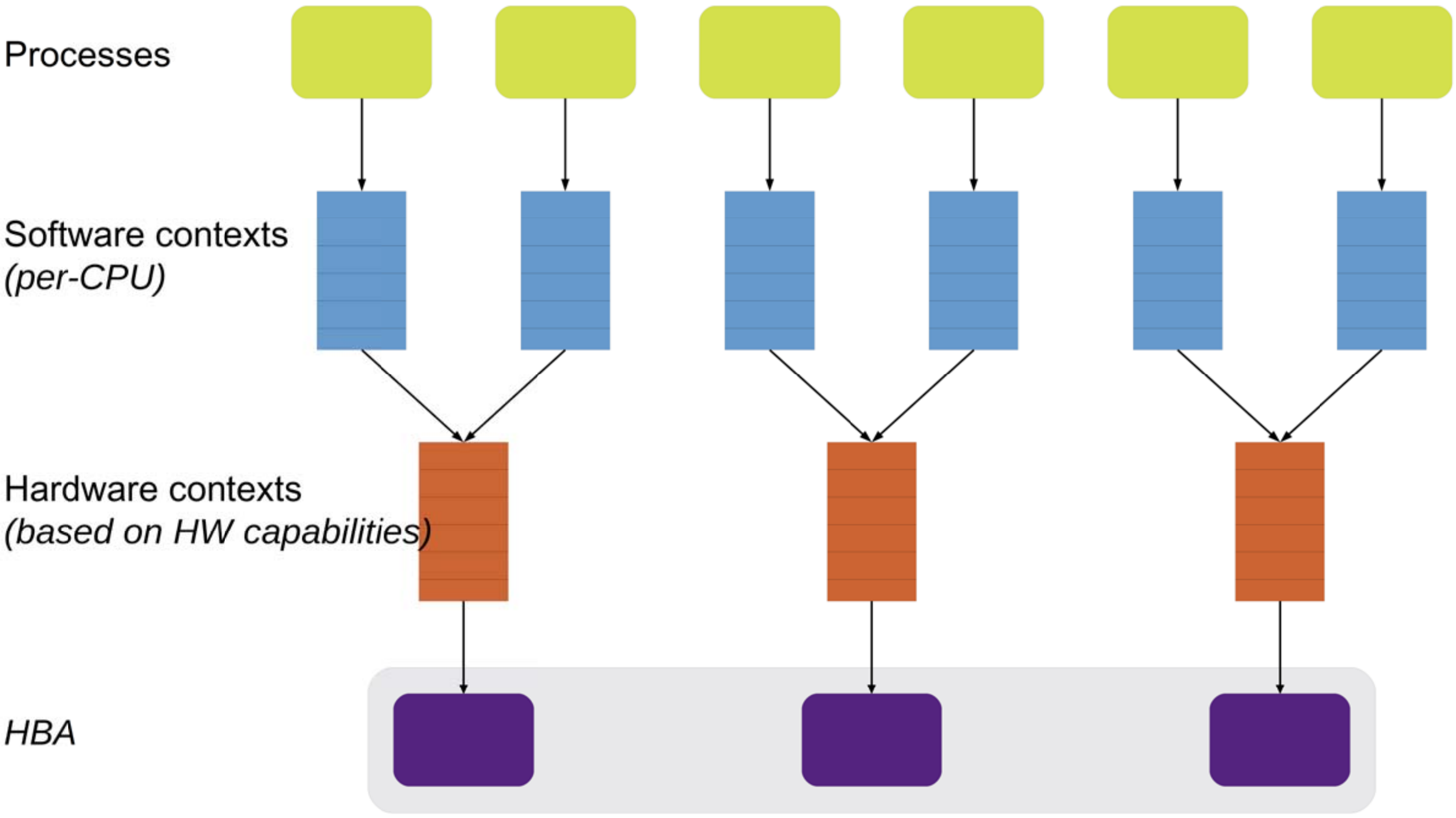
Linux Storage Stack – blk-mq

- A replacement for the request layer
  - First prototyped in 2011
  - Merged in Linux 3.13 (2014)
- Not a drop-in replacement
  - Different driver API
  - Different queuing model (push vs pull)

Blk-mq – architecture

- Processes dispatch into per-cpu software queues
- Software queues map to hardware issue queues
  - In the optimal case:
    - **N(**hardware queues**)** = **N(**CPU cores**)**
  - For now the most common case is::
    - **N(**hardware queues**)** = **1**

Blk-mq I/O submission path



Processes

Software contexts
(per-CPU)

Hardware contexts
(based on HW capabilities)

HBA

Blk-mq – request allocation and tagging

- Provides combined request allocation and tagging
  - Requests are allocated at initialization
  - Requests are indexed by the tag
  - Tag and request allocation are combined
- Avoids per-request allocations in the driver
  - Driver data in “slack” space behind request
  - S/G list is part of driver data

Blk-mq – I/O completions

- Uses IPIs to complete on the submitting node and avoid false cache line sharing
  - Can be disabled, or forced to the submitting core
- Old request code provided similar functionality
  - Non-integrated additional functionality
  - Uses software interrupts instead of IPIs

Prototype for blk-mq usage in SCSI

- First “scsi-mq” prototype from Nic Bellinger
  - Published in late 2012
  - Used early blk-mq to drive SCSI
  - Demonstrated millions of IOPS
  - Required (small) changes to drivers
  - Only using a single hardware queue
  - Did not support various existing SCSI stack features

Production design for blk-mq in SCSI

- Should be a drop in replacement
  - Must support full SCSI stack functionality
  - Must not require driver API changes
  - Driver should not be tied to blk-mq
- Should avoid code duplication
  - Push as much as possible work to blk-mq
  - Refactor SCSI code to avoid separate code paths as much as possible

Production design for blk-mq in SCSI - Request allocation and tagging

- ❐ Considerations for request and tag allocation:
  - – Allocating a request for each per-LUN tag would inflate memory usage
  - – Various hardware requires per-host tags anyway
- ❐ Thus went with blk-mq changes to allow per-host tag sets

Production design for blk-mq in SCSI - S/G lists

- Modern SCSI HBAs allow for huge S/G lists
  - Linux supports up to 2048 S/G list entries, which require 56 KiB of S/G list structures
  - We don't want to preallocate that much
- Preallocate a single 128 entry chunk
  - Enough for most latency sensitive small I/O
  - The rest is dynamically allocated as needed

Blk-mq work driven by SCSI

- ❒ Transparent pre/post-flush request handling
- ❒ Head of queue request insertion
- ❒ Partial completion support
- ❒ BIDI request support
- ❒ Shared tag space between multiple request_queues
- ❒ Better support for requeuing from IRQ context
- ❒ Lots of bugfixes and small features / cleanups

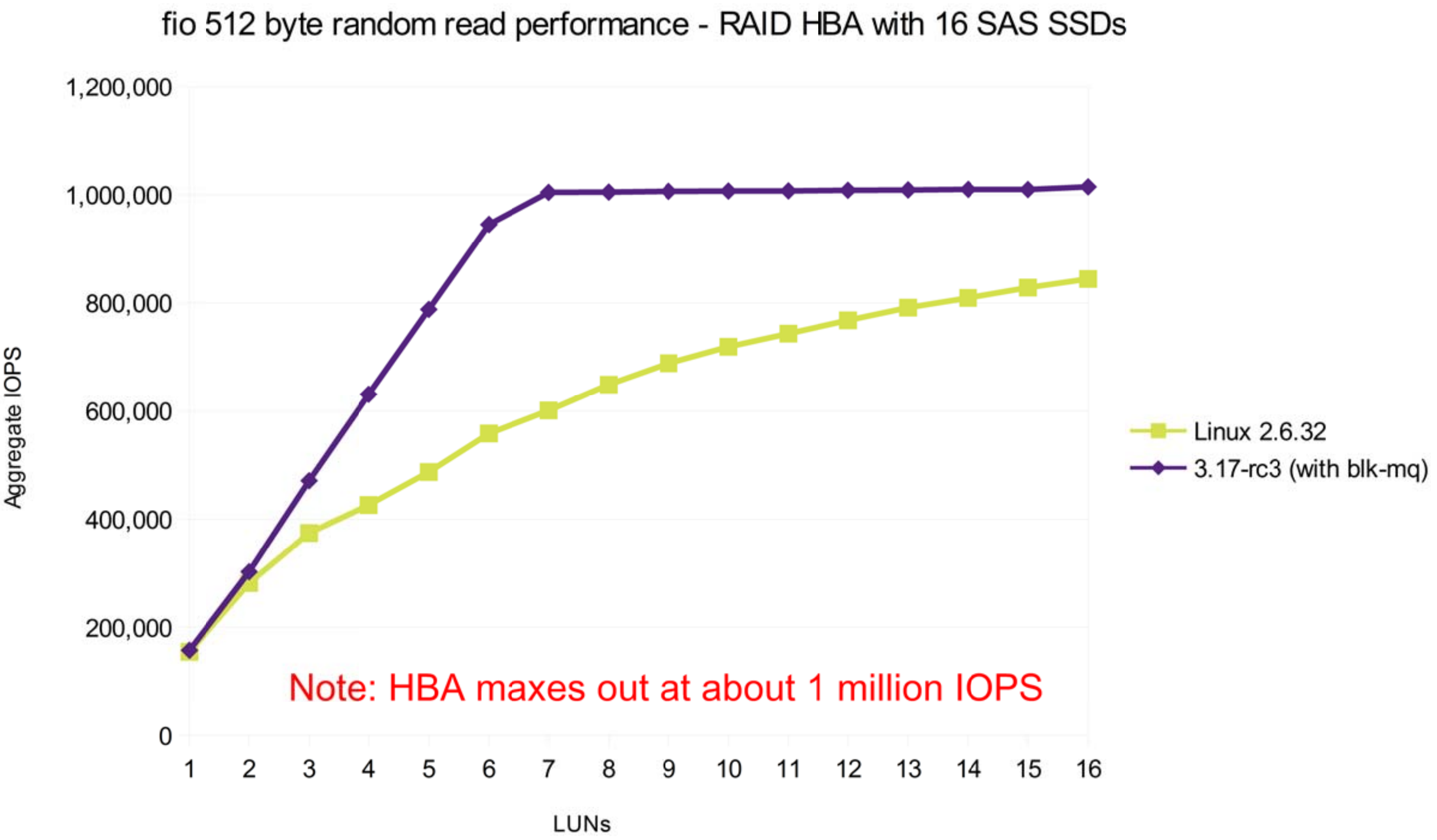
SCSI preparation for blk-mq

- ❐ New cmd_size field in host template
  - – Allows to allocate per-driver command data
- ❐ Host-lock reductions
  - – Elimination of host-wide spinlocks in I/O submission and completion
- ❐ Upper level driver refactoring
  - – Avoids legacy request layer interaction
  - – Provides a cleaner drivers abstraction

SCSI blk-mq status

- Required blk-mq features included in Linux 3.16
- Preparatory SCSI work merged in Linux 3.16
- Blk-mq support for SCSI merged in Linux 3.17-rc1
  - Must be enabled by scsi_mod.use_blk_mq=Y boot option
  - Does not work with dm-multipath
- Big distributions include preparatory patches

Linux SCSI Performance



Note: HBA maxes out at about 1 million IOPS

SCSI profiling data

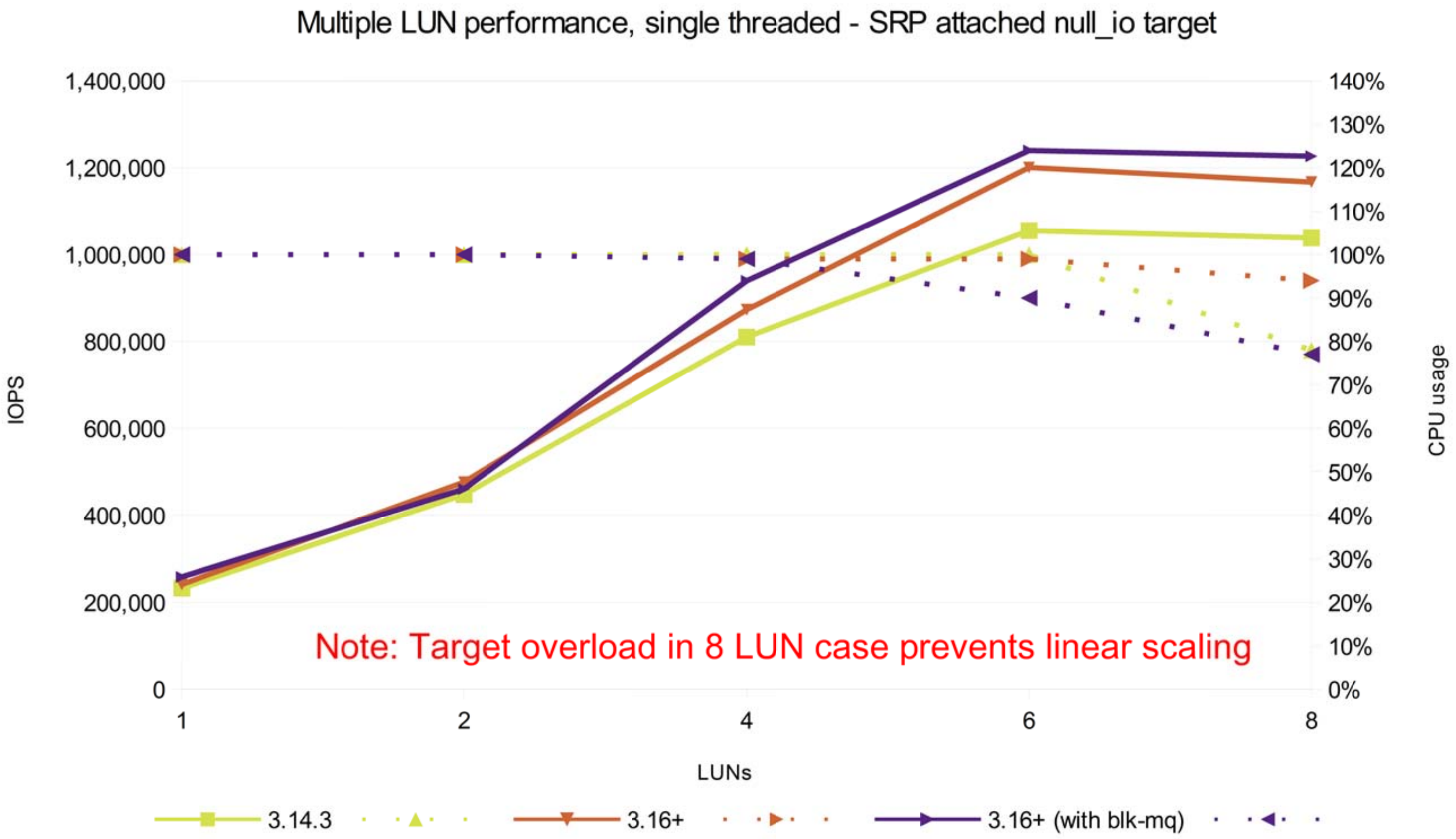
Linux 2.6.32

```
46.13%  [kernel]                [k] _spin_lock_irq
26.92%  [kernel]                [k] _spin_lock_irqsave
 9.32%  [kernel]                [k] _spin_lock
 0.47%  [kernel]                [k] kmem_cache_alloc
 0.45%  [kernel]                [k] scsi_request_fn
 0.39%  [kernel]                [k] _spin_unlock_irqrestore
 0.33%  [kernel]                [k] kref_get
 0.32%  [kernel]                [k] __blockdev_direct_IO_newtrunc
 0.32%  [kernel]                [k] kmem_cache_free
 0.30%  [kernel]                [k] native_write_msr_safe
```

Linux 3.17-rc3
(with blk-mq)

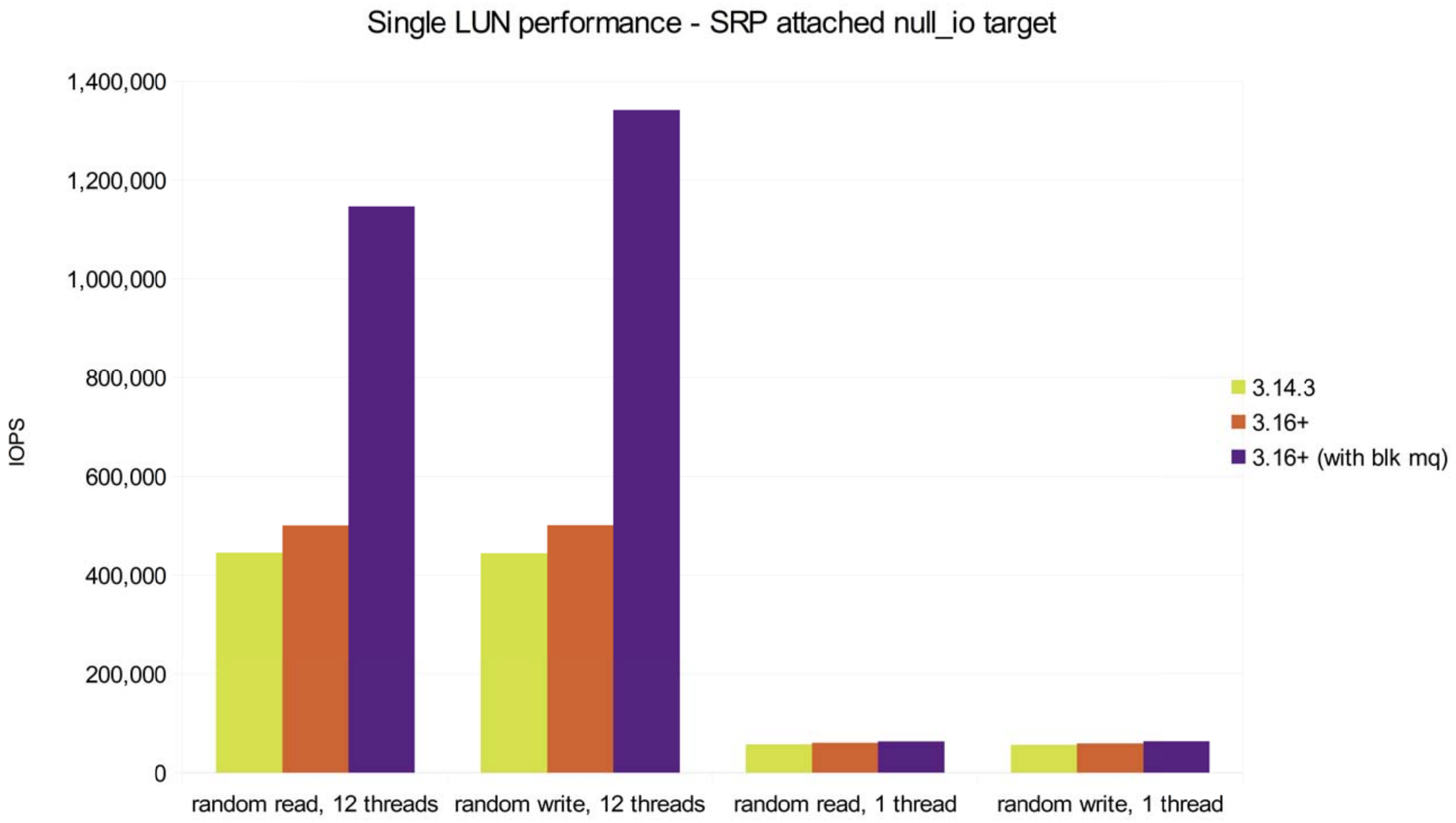
```
2.67%  [kernel]                     [k] do_blockdev_direct_IO
2.60%  [kernel]                     [k] __bt_get
2.43%  [kernel]                     [k] __blk_mq_run_hw_queue
2.07%  [kernel]                     [k] put_compound_page
1.87%  [kernel]                     [k] __blk_mq_alloc_request
1.60%  [kernel]                     [k] _raw_spin_lock
1.59%  [kernel]                     [k] kmem_cache_alloc
1.58%  [kernel]                     [k] scsi_queue_rq
1.44%  [kernel]                     [k] _raw_spin_lock_irqsave
```

Linux SCSI Performance



Note: Target overload in 8 LUN case prevents linear scaling

Linux SCSI Performance



SCSI blk-mq status - near term work

- ❒ Better way to select blk-mq vs legacy code path
- ❒ We would like to fully replace the old SCSI I/O path with the blk-mq one.
- ❒ Missing features:
  - – I/O scheduler support in blk-mq
  - – multipathing support

Future work – expose multiple HW queues

- ❐ SCSI core so far only exposes a single queue
  - Some drivers are ready for multiple queues
  - So far do internal queue mapping
- ❐ Needs a design for tag allocation
  - We want per-queue tag allocations for scalability reasons
  - But we need unique tags
  - Steal a byte per tag to indicate queue?

Future work – better integration

- ❒ Expose more blk-mq flags to SCSI
  - Request merge control
  - better command allocation/freeing hooks
  - Reserved tags for HBA use

Future work - longer term research

- Further reduction of shared cache lines:
  - let blk-mq handle per-host queuing limits
  - let hardware handle per-LUN or per-target queuing limits
- Map multiple LUNs (request_queues) to the same blk-mq contexts

References

- Benchmarks:
  - Bart van Assche (Fusion-io / Sandisk):
    - https://docs.google.com/file/d/0B1YQOreL3_FxWmZfbl8xSzRfdGM/edit?pli=1
  - Robert Elliot (HP):
    - http://marc.info/?l=linux-kernel&m=140313968523237&w=2

Thanks

- ❒ Fusion-io (now a Sandisk company)
  - For sponsoring the blk-mq in SCSI work
- ❒ Jens Axboe
  - For code and slide review, and blk-mq itself
- ❒ Bart van Assche, Robert Ellliot
  - For code and slide review as well as benchmark data

Questions?