facebook

What’s new in the world of storage for Linux

Jens Axboe

Software Engineer

Kernel Recipes 2017, Sep 27th 2017

blk-mq status update

- Conversions: stec, nbd, MMC (almost)
- scsi-mq now the default
  - Well, almost
- cciss now under SCSI
- Stragglers
  - About ~15 left
  - It ain’t over until floppy.c is converted

blk-mq scheduling

- Main missing feature of the new framework
- Various blk-mq design decisions made this hard
  - Fixed tags
  - flush handling
  - Scalability
- 4.11 added blk-mq-sched
  - **none** and **mq-deadline**
- 4.12 added **BFQ** and **Kyber**

Writeback throttling

- Periodic background writeback behavior

Unregistered HyperCam 2
imgflip.com

Writeback throttling

- Periodic background writeback behavior
    - Sucks for both background and other IO
- Attempt to balance both performance and latency
- Similar, in spirit, to CoDel

“When applied to network routers, RED probabilistically either marks packets with ECN or drops them, depending on the configuration. When dealing with disk I/O, POSIX does not have any mechanism with which to notify the caller that the disk is congested, so we instead only provide the latter strategy.”

Omar’s April 1st IO scheduler posting

Writeback throttling

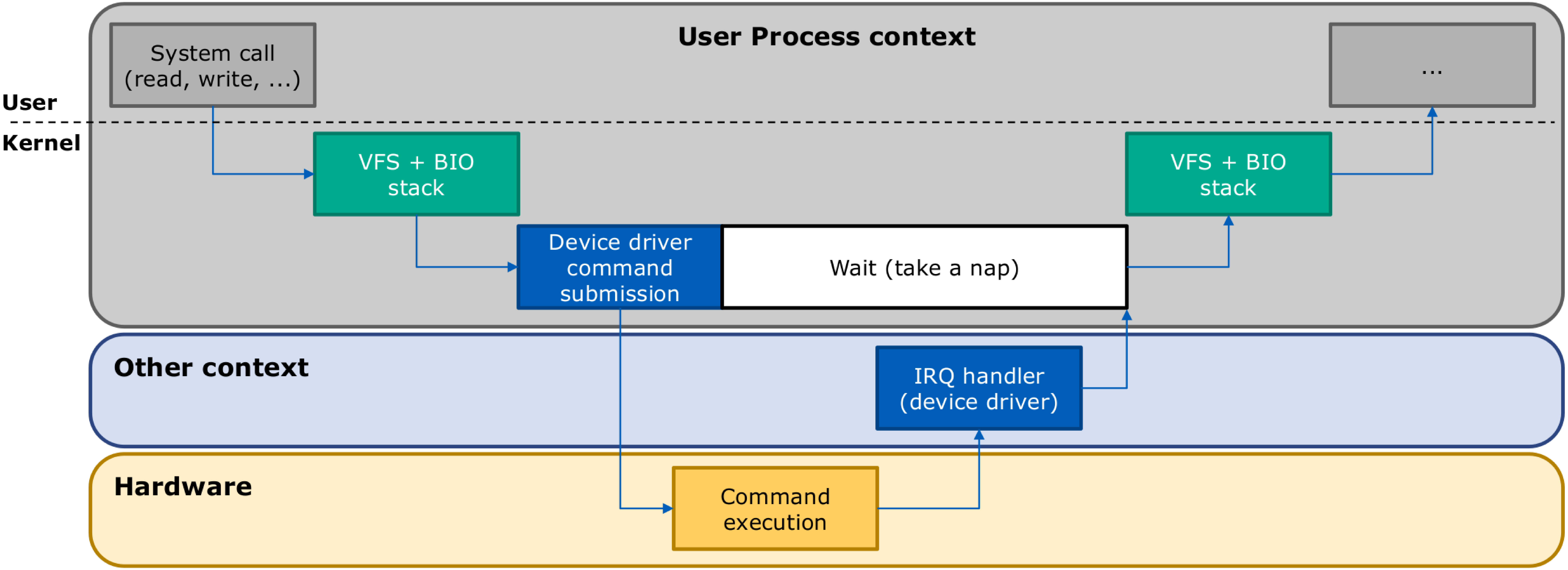
- Periodic background writeback behavior
  - Sucks for both background and other IO
- Attempt to balance both performance and latency
- Similar, in spirit, to CoDel
- Monitor read latencies in the presence of writes
  - *wbt_lat_usec*
- Splits writes into categories
- Scales up or down depending on behavior
- Added in 4.10

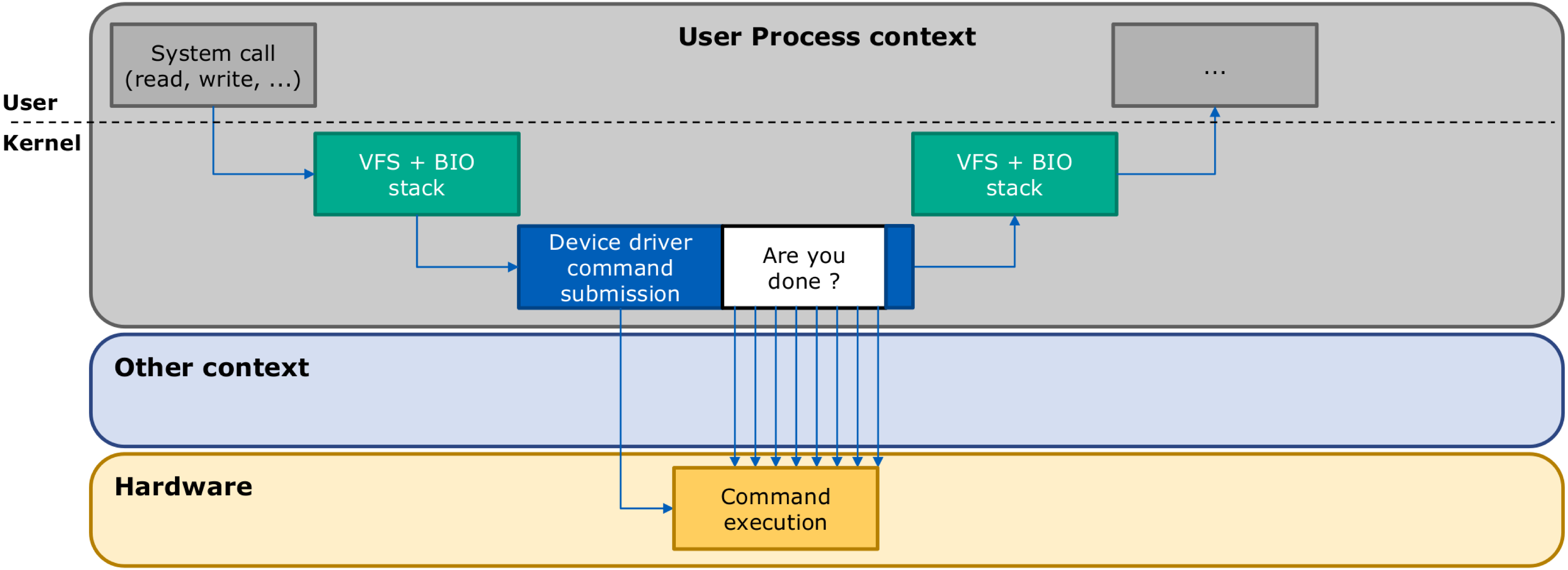
wbt production

- Monitor service QoS, while:
  - **small-files-1.0-1.x86_64.rpm**, 128^2 files, 2-64k
  - **big-files-1.0-1.x86_64.rpm**, 4 files, 3-400MB
- **io.go** test app
- NVMe (> 10msec)
  - Off: 4.8 violations, avg 79msec, max 139msec
  - On: 3.0 violations, avg 17msec, max 17msec
- Hard drive (> 100msec)
  - Off: 18.4 violations, avg 1633msec, max **6.5s**
  - On: 16.4 violations, avg 209msec, max 478msec

IO Polling

- Faster completion times


User Process context
System call
(read, write, ...)
...
User
Kernel
VFS + BIO
stack
VFS + BIO
stack
Device driver
command
submission
Wait (take a nap)
Other context
IRQ handler
(device driver)
Hardware
Command
execution

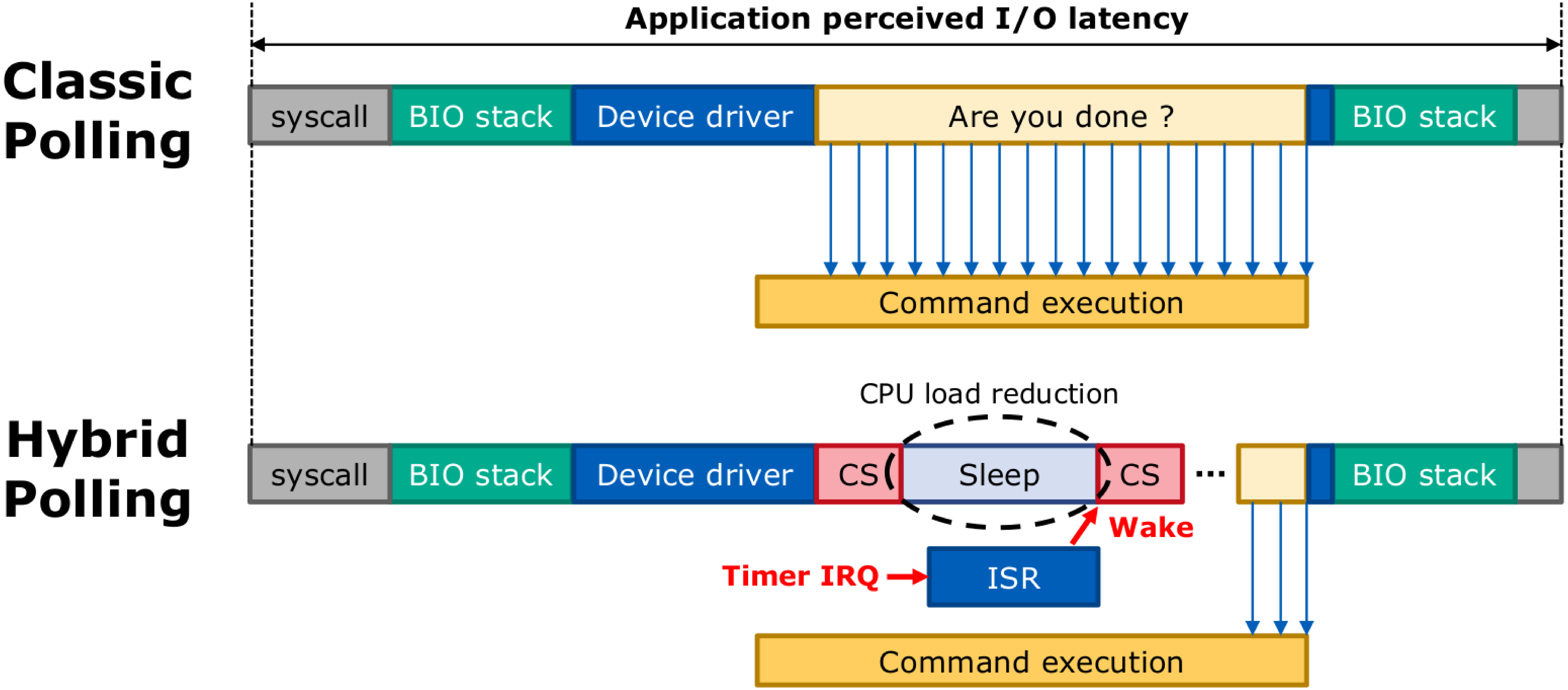

User Process context
System call (read, write, ...)
...
User
Kernel
VFS + BIO stack
VFS + BIO stack
Device driver command submission
Are you done ?
Other context
Hardware
Command execution

IO Polling

- Faster completion times
- Extra CPU cost due to spinning
  - But smaller sleep+wakeup cost

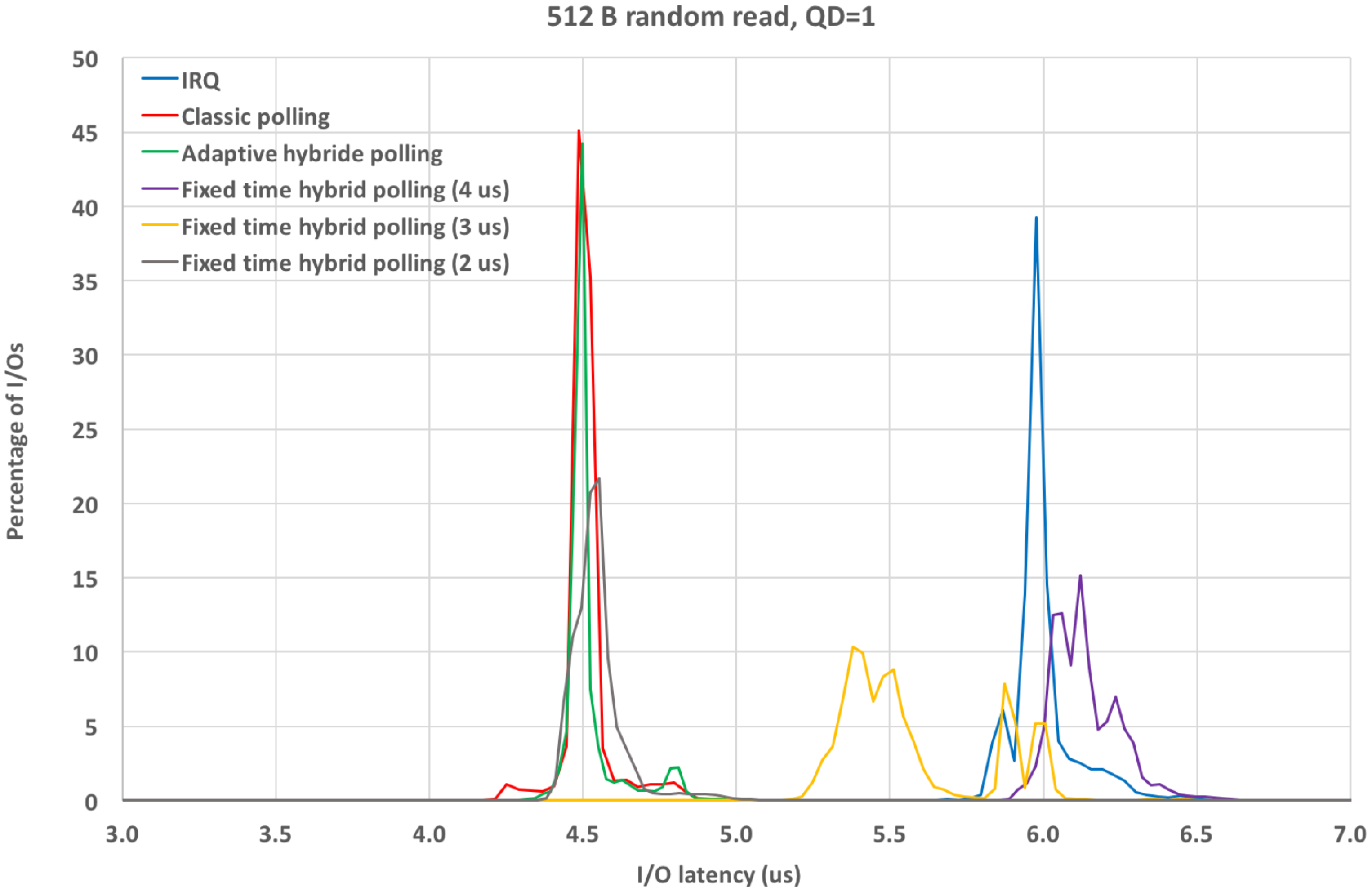
IO Polling

- Faster completion times
- Extra CPU cost due to spinning
  - But smaller sleep+wakeup cost
- Is there a more optimal solution?


Application perceived I/O latency
Classic Polling
syscall
BIO stack
Device driver
Are you done ?
BIO stack
Command execution
CPU load reduction
Hybrid Polling
syscall
BIO stack
Device driver
CS
Sleep
CS
...
BIO stack
Wake
Timer IRQ
ISR
Command execution

IO Polling

- Faster completion times
- Extra CPU cost due to spinning
  - But smaller sleep+wakeup cost
- Is there a more optimal solution?
- Hybrid polling
  - Sleeps for *mean/2*
  - Tracks completion times in incremental buckets
- preadv2/pwritev2
  - *RWF_HIPRI*
- sysfs: *io_poll* and *io_poll_delay*


512 B random read, QD=1
IRQ
Classic polling
Adaptive hybride polling
Fixed time hybrid polling (4 us)
Fixed time hybrid polling (3 us)
Fixed time hybrid polling (2 us)
Percentage of I/Os
50
45
40
35
30
25
20
15
10
5
0
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
I/O latency (us)

Faster O_DIRECT

- New devices and polling expose overhead
  - Existing implementation is a pig
- Basically two paths
  - Small O_DIRECT
  - Large O_DIRECT
- fs/iomap.c improves file system side
- Shaves 6-7% of IO time (~6.4 usec → 6.0 usec)
- Merged in 4.10

Faster IO accounting

- IO accounting tracking needed overhaul
  - Easily 1-2% of CPU usage in testing
  - Heaviest part of the stack
  - Synthetic null_blk tests, 20M → 2M IOPS
- Rewritten to not have any per-dev shared data
  - “Free” by utilizing blk-mq tagging
  - No more per-io inc/dec of inflight count
- Merged in 4.14

Write lifetime hints

- Allows an application to signal expected write lifetime
- *fcntl*(2) based
  - *F_{GET,SET}_RW_HINT* for inodes
  - *F_{GET,SET}_FILE_RW_HINT* for files
- **Short**, **medium**, **long**, and **extreme**
- Initial application is for flash based storage
  - Allows device to write more intelligently
  - Garbage collection, erase blocks are huge
- NVMe supports it (1.3)
- 25-30% reduction in writes RocksDB/MyRocks

IO throttling support

- cgroup tied to CFQ
  - Not ideal for moving to blk-mq
- Scales better and functions better on SSDs
- Supports cgroup2
- Merged for 4.12
- Still experimental
  - Interface concerns
  - IO cost estimation is not easy

2015 KR: Future work

- An IO scheduler
- Better helpers for IRQ affinity mappings
- IO accounting
- IO polling
- More conversions
  - Long term goal remains killing off request_fn

NAILED IT!

Future

- IO Determinism
- Continued efficiency improvements

facebook