SDC

STORAGE DEVELOPER CONFERENCE

SNIA SANTA CLARA, 2017

Linux Optimizations for Low-Latency Block Devices

Stephen Bates, PhD
Raithlin Consulting

Nomenclature: A Reminder

Low-Latency Block Devices are NOT blucky!!

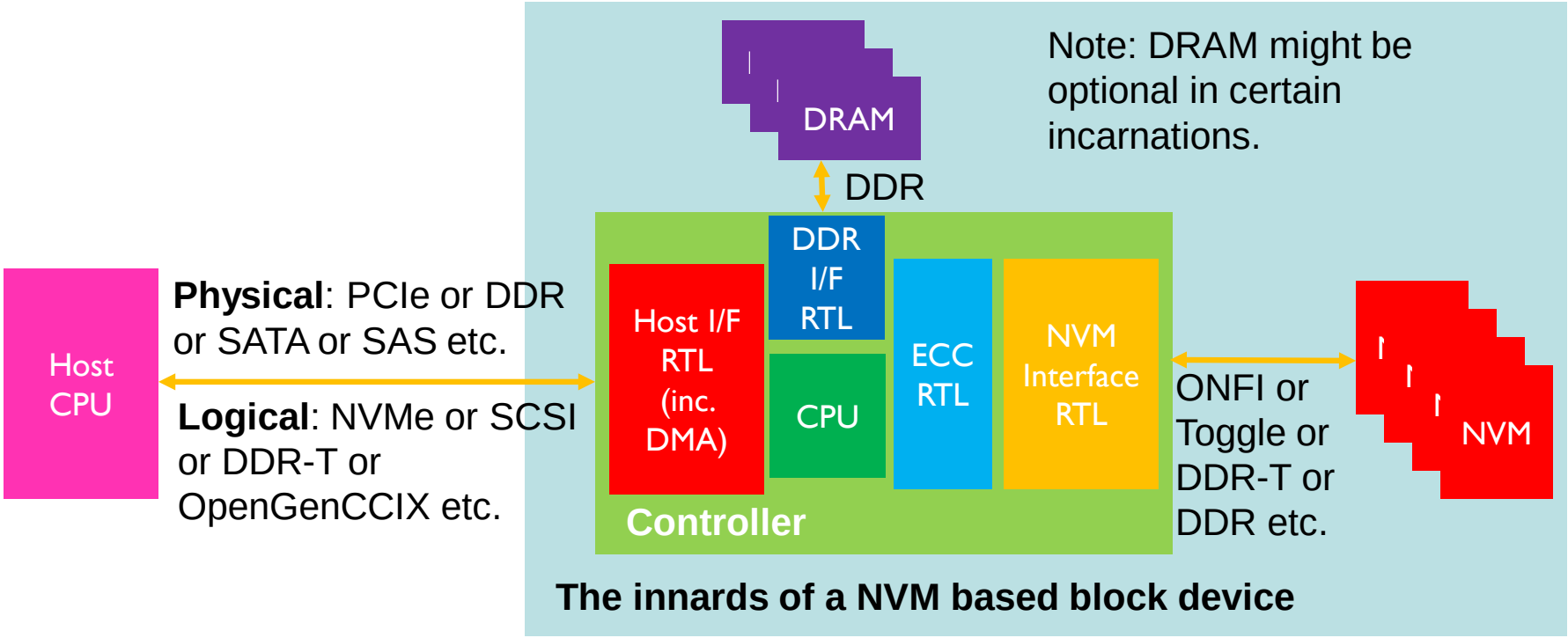
Linux Block Devices: A Reminder

- A Linux block device is a software construct that may be backed by a real device:
  - /dev/nullb0 – backed by nothing!
  - /dev/pmem0 – backed by Persistent Memory
  - /dev/nvme0n1 – backed by NVMe attached stuff.
  - /dev/sda1 – backed by SCSI attached stuff
  - /dev/nbd0 – backed by network attached stuff
  - /dev/md0 – backed by multiple block devices

Block Devices: A Reminder

- A physical block device has some important attributes:
  - Can be accessed randomly.
  - Is sector/block based (e.g. 512B or 4KB etc).
  - Sector/block operations are atomic (i.e. they either happen in their entirety or not at all).
  - Often involve DMA engines (the Jeeves of the CPU world).

Block Devices: A Reminder



Latency: A reminder

Throughput easy; latency hard

SNIA PERSISTENT MEMORY SUMMIT
JANUARY 18, 2017 | SAN JOSE, CA

Throughput is easy

Latency is hard

Throughput is an engineering problem; latency is a physics problem!

Persistent Memory: A reminder

Persistent Memory (PM)

SNIA PERSISTENT MEMORY PM SUMMIT
JANUARY 18, 2017 | SAN JOSE, CA

Low Latency

Memory Semantics

Storage Features

Persistent Memory: A reminder

Persistent Memory (PM)

SNIA PERSISTENT MEMORY PM SUMMIT
JANUARY 18, 2017 | SAN JOSE, CA

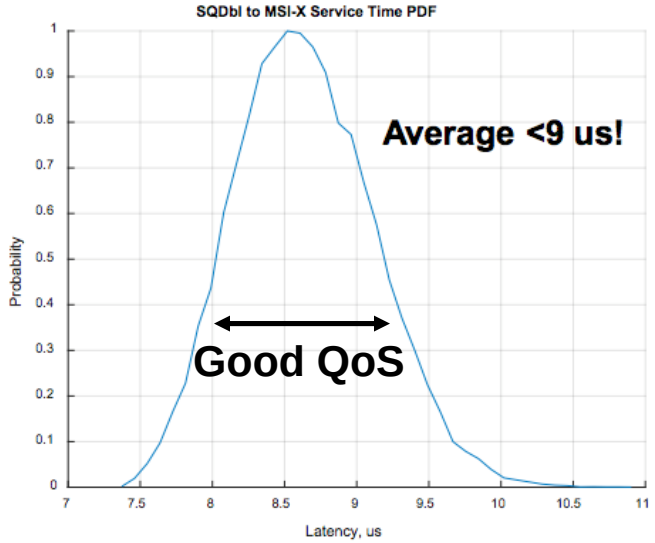
Low Latency

Memory Semantics

Storage Features

So Why Block?

NVMe Latency



NVMe is fast but not PM fast (nor byte addressable, nor coherent).

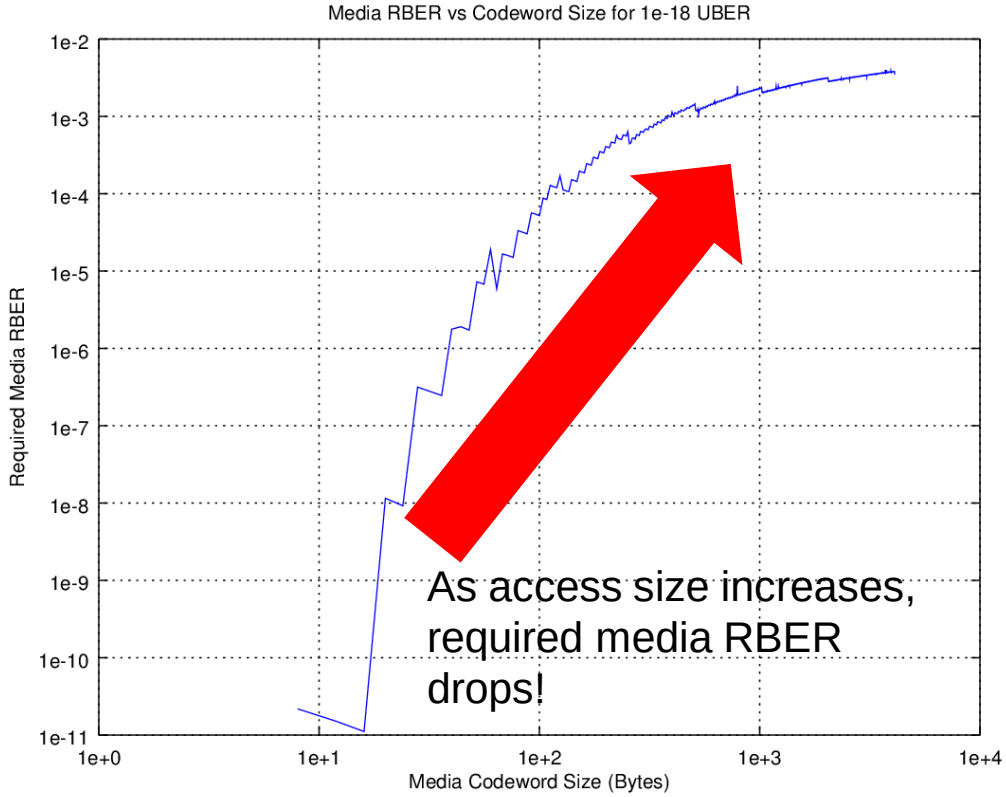
NVMe QoS is pretty good in the system we tested.

Device	Average	P99
/dev/nullb0	3.9us	5.3us
/dev/pmem0	3.31us	6.2us
/dev/nvme0n1	12us	18.5us

11

So Why Block?

Bates-Conjecture: For any new NVM media, block will come to market first!



The RBER needed to hit 1e-18 UBER is 8 orders of magnitude less for block than for byte access.

Easier to make materials work at 1e-3 than 1e-11!

Low-Latency Block Devices Are Here…

EVERSPIN TECHNOLOGIES

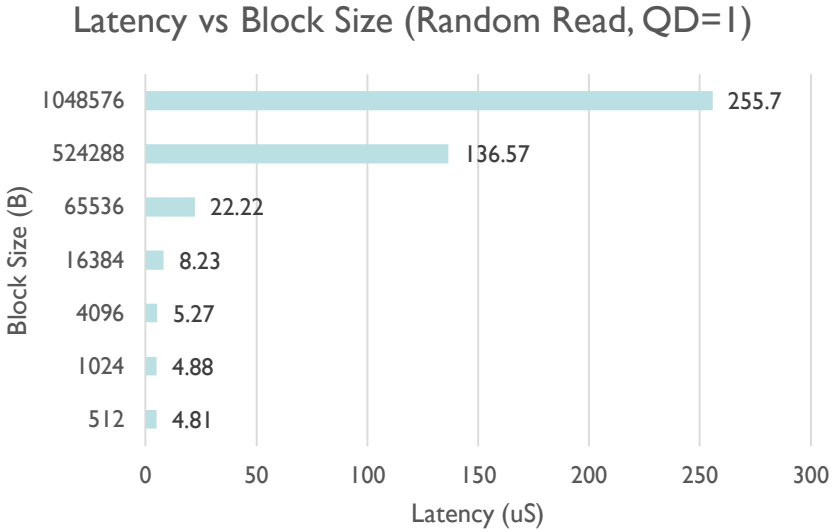
nvm EXPRESS

intel

Microsemi

And They're Pretty Frickin' Fast!

- Sub 5us latency for 512B at QD=1.
- Measured via FIO on a 4.12 based Linux kernel.

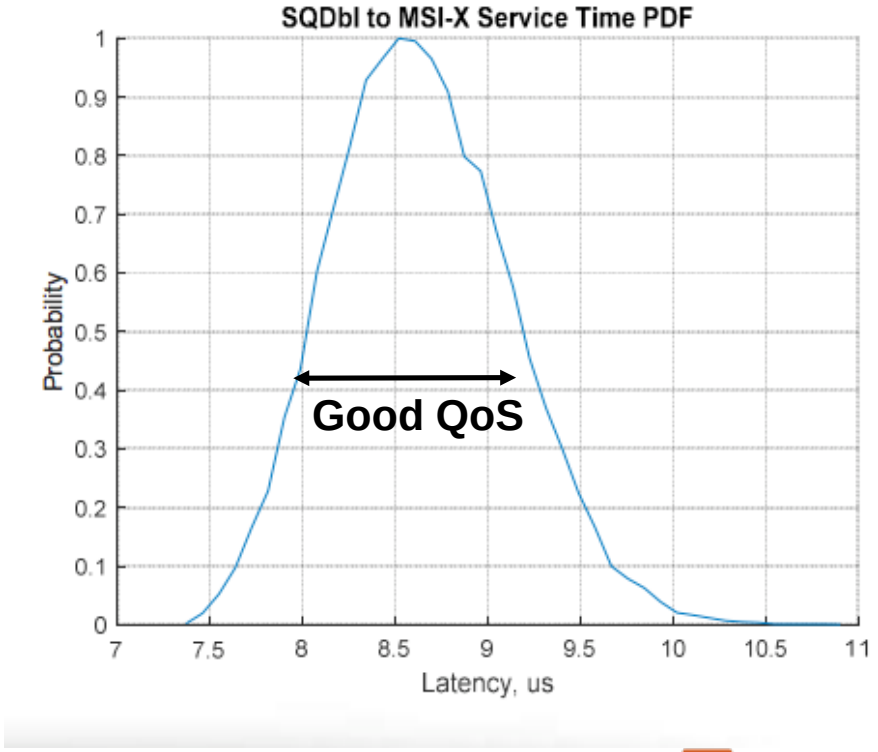


EVERSPIN TECHNOLOGIES

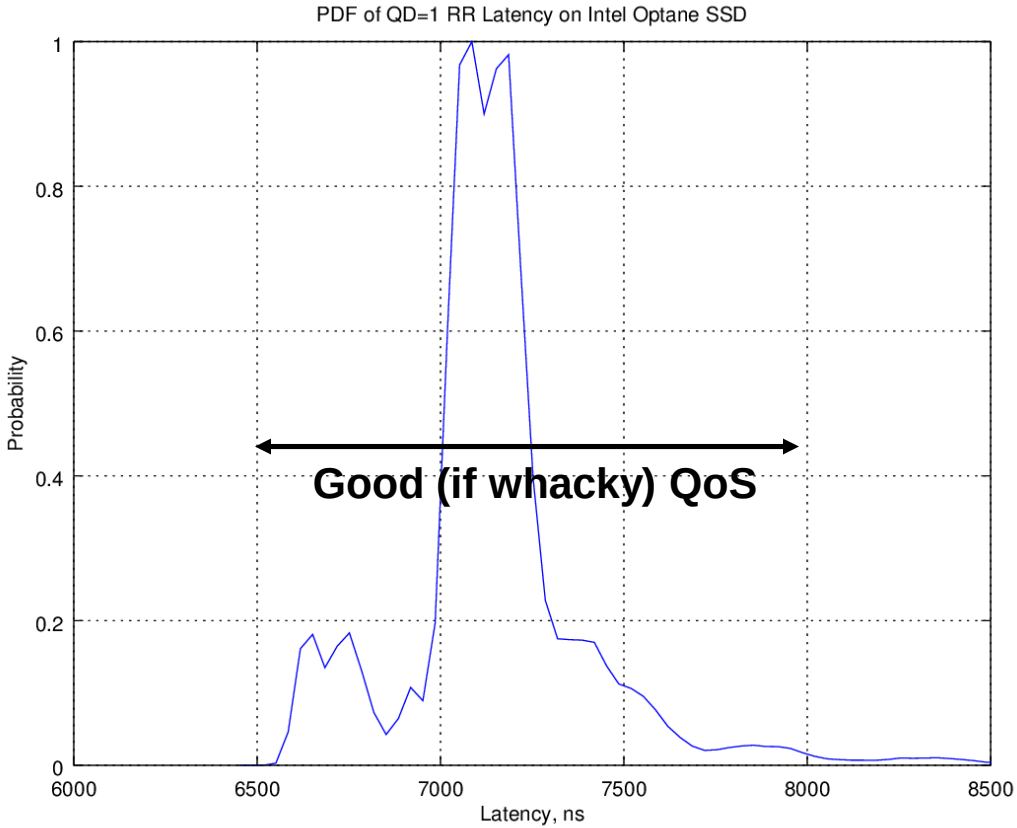
nvm EXPRESS

Oh and the QoS is *really* good

Microsemi



Oh and the QoS is *really* good

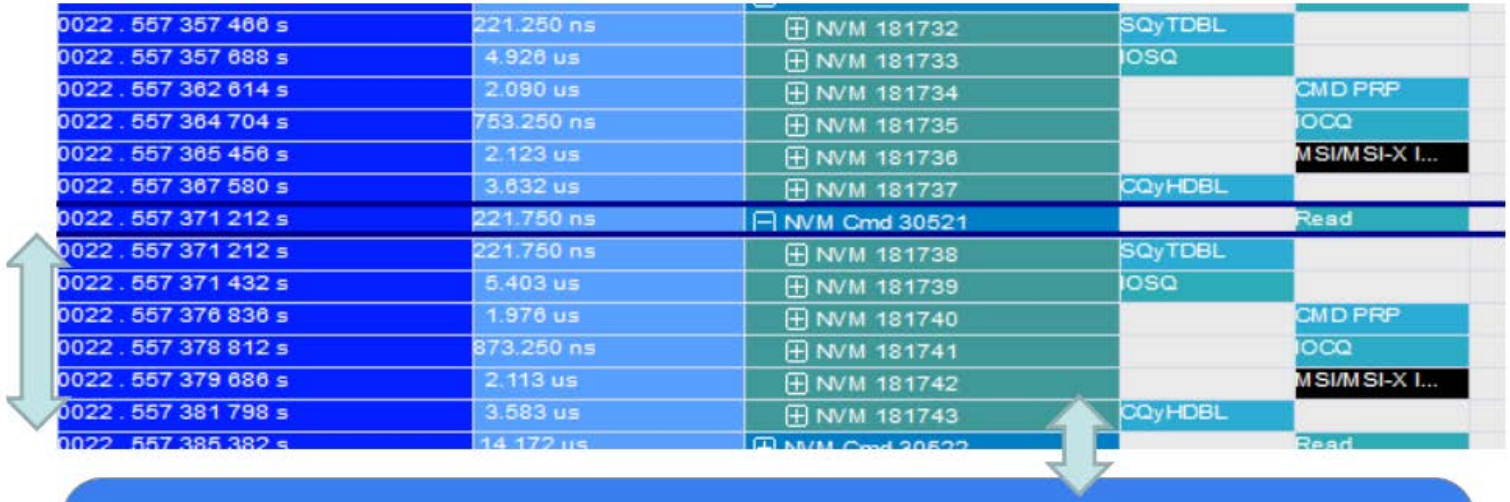


Reminder: An NVMe Read Command

1. Host[1] puts 64B NVMe command on a submission queue located in either main or io memory (e.g. CMB).
2. Host[1] rings doorbell (PCIe MMIO register) associated with the queue in step 1.
3. SSD pulls in 64B command, it will include information on LBAs to be read from NVM and location in memory to place resultant data.
4. SSD pulls relevant LBAs from NVM and DMAs result to desired location (optionally via SGL).
5. SSD places 16B NVMe completion entry on relevant completion queue.
6. (Optional) SSD asserts an interrupt to inform system the IO is done.

[1] OK, technically the host does not have to do this. Another IO device could do this (e.g. Mellanox CX5 NVMe offload engine)

Reminder: An NVMe Read Command



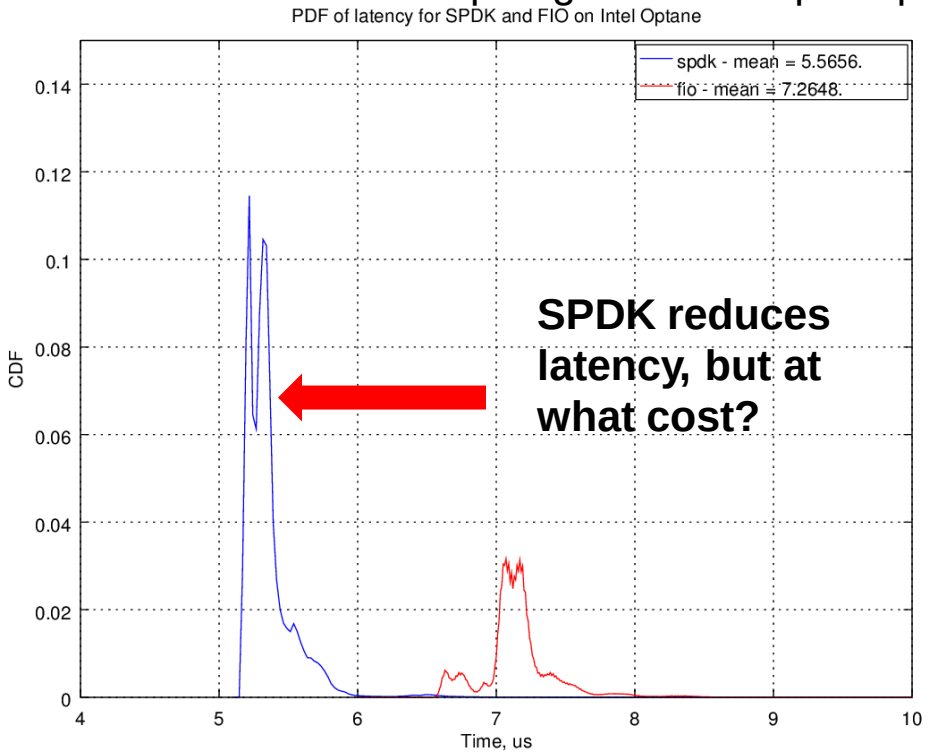
The anatomy of a single NVMe Read command.
~10us total time for QD=1 NVMe Read

OK, Got to Mention SPD ”F%^King” K ;-)

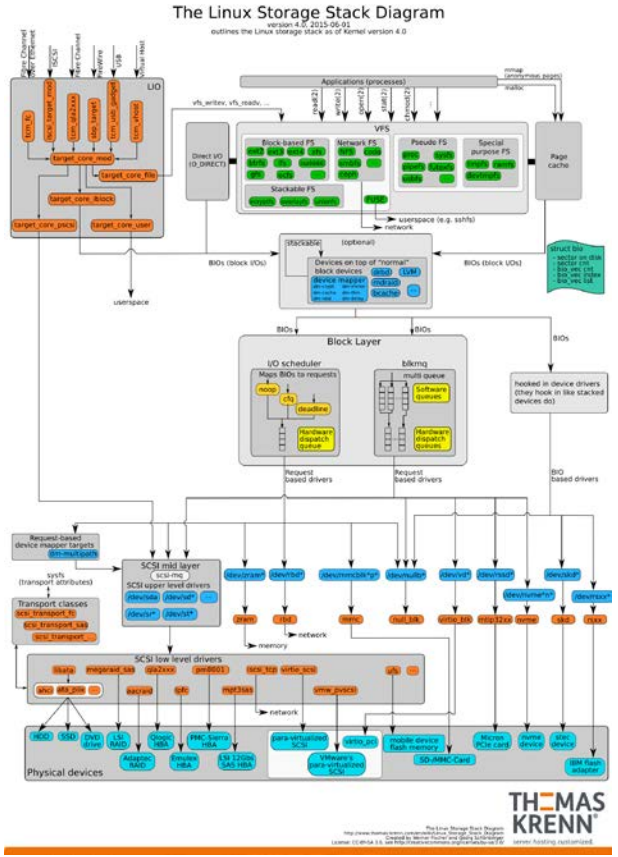
https://github.com/spdk/spdk

- OK, ok, SPDK will beat the kernel for latency
- However it comes at a cost (no FS, no blktrace, no iostat etc)
- So, how well can the kernel do?

Same SSD and IO pattern. How applications access device alters mean and PDF of latency!



- The Linux block layer must be all things to all people.
- Not manically focused on latency and performance.
- However it does evolve!



Polling Baby!

Testing done on Intel® Optane™ SSDs[1] using this script[2].

- The ability for the block layer to poll was added in v4.4.
- Support for NVMe polling was also added in v4.4.
- Trades CPU cycles for latency.

Mode	Avg.	99	CPU
No Poll	9.1u	17.5u	28.7%
Poll	7.4u	14.3u	100%

[1] 4.12 kernel, Intel® SSDPED1K375GAQ 375GB Optane™ SSD, fio, 512B randread.

[2] https://github.com/sbates130272/fio-stuff/blob/master/misc/iopoll-test.sh

Hybrid Polling Baby!

- Why poll from time 0?
- Wait for a while, then poll.
- Right now start polling at half average completion time (or set your own time).
- Added in v4.10

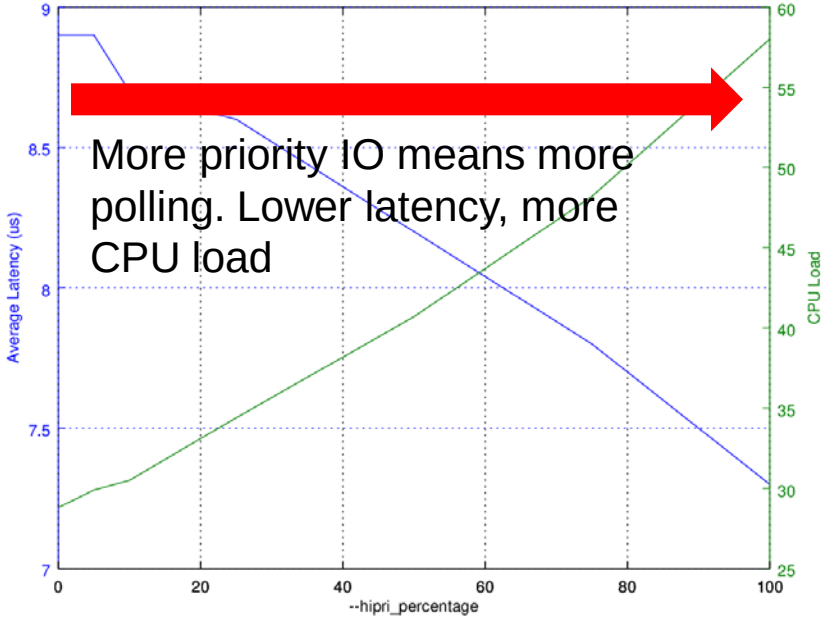
Mode	Avg.	99	CPU
No Poll	9.1u	17.5u	28.7%
Poll	7.4u	14.3u	100%
Hybrid	7.3u	14.7u	58%

Hybrid almost as good as polling but saves ~40% CPU load!

More Hybrid Polling Baby!

Connects Applications to NVMe SSDs!

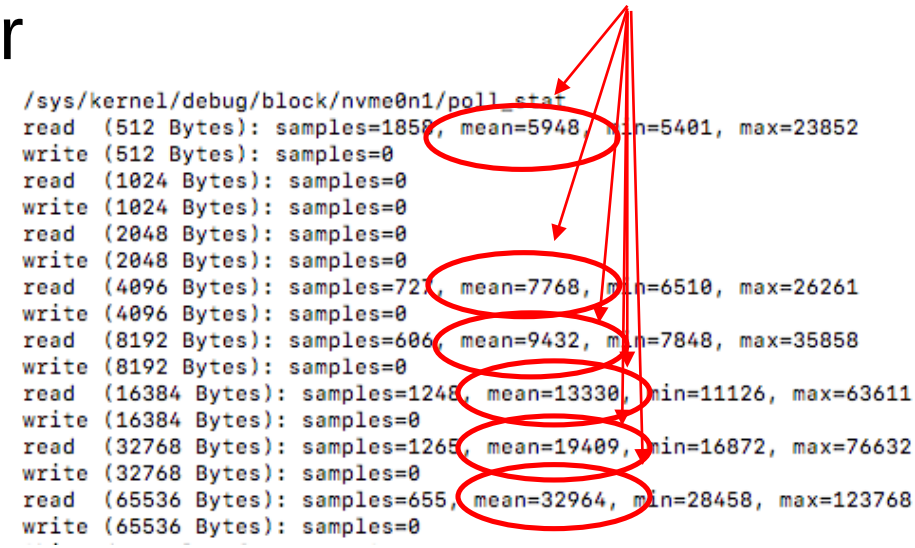
- Block layer only polls on direct IO issued by the preadv2 and pwritev2 system calls
- Still being tied into glibc
- FIO directly makes syscall for now.
- We can alter what percentage of IO are hipri and see what happens



(Better) Hybrid Polling Baby!

- Why use same delay for all IO sizes?
- Calculate sleep IO size for each IO size (within reason)
- Added in v4.12.

Use mean/2 for the relevant IO size.



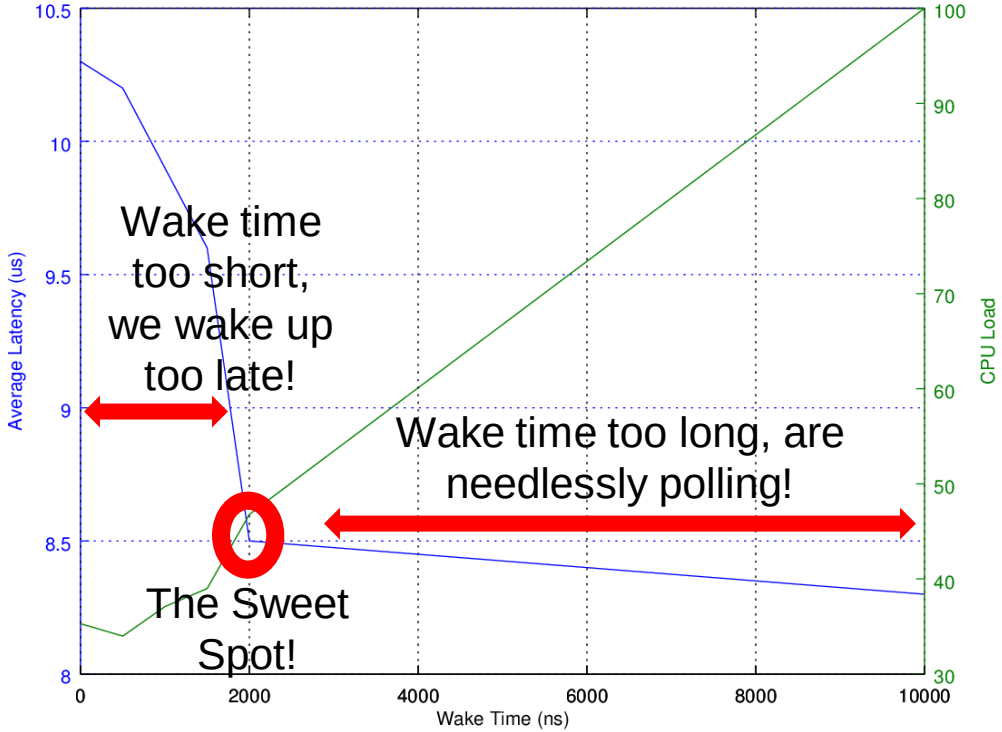
Also see great Vault paper by Damien Le Moal from WD - https://vault2017.sched.com/event/9WQX/io-latency-optimization-with-polling-damien-le-moal-western-digital

(Even Better) Hybrid Polling Baby!

- Why use mean/2?
- Ideally we want to poll after the minimum response time minus some wakeup time.
- So let's try that!

Ideal Sleep Time = Minimum Response Time – Maximum Wake Time

(Even Better) Hybrid Polling Baby!



Altering the waketime allows for a trade-off between average latency and CPU usage.

The extremes represent legacy hybrid polling (0) and legacy polling (10000).

In this system a 2us sleep time is the sweet spot!

Submitted this code for consideration for Linux kernel[1].

1 https://lkml.org/lkml/2017/8/21/486

What's Next?

- Industry is (manically) focused on QoS.
- RWF_HIPRI first of many flags to help place data on NVMe SSDs
- SSDs getting better at QoS and data placement:
  - Streams – added in 4.13 (tied into IO lifetime)
  - Directives and IO determinism
  - IO priority
  - IO expected lifetime
  - OpenChannel
- The Linux kernel will add support for these features

Thanks!

A big Thank You to Intel® for providing access to their NVMe Optane™ SSDs for this work.