SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

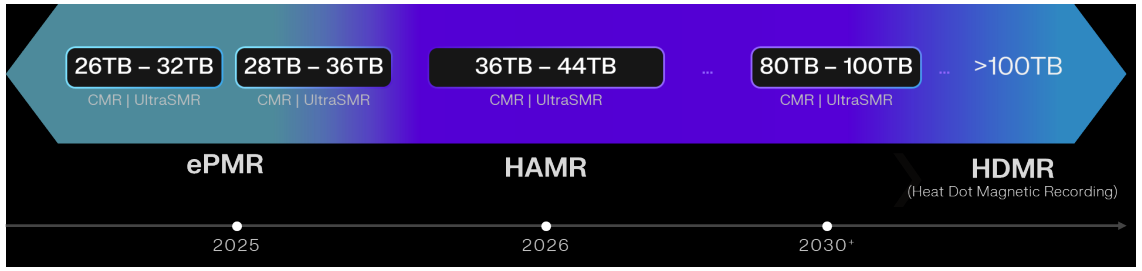
Command Duration Limits (CDL) Improving IOPS per TB in HDDs

David Landsman, Distinguished Engineer, Western Digital
Damien Le Moal, Distinguished Engineer, Western Digital

www.sniadeveloper.org

I/O Density (IOPS/TB, MB/s/TB) a Challenge as Capacity Grows

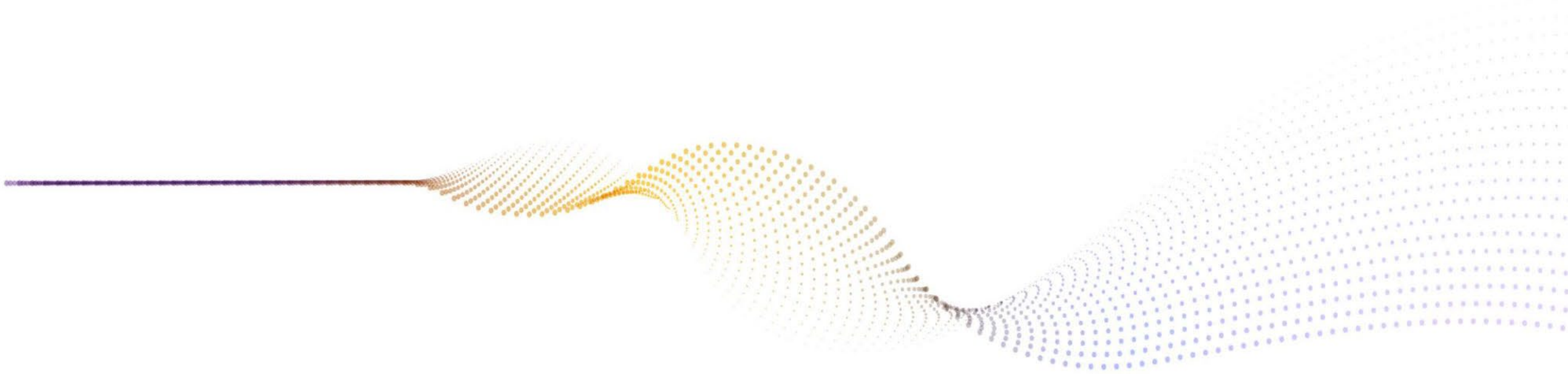
Capacity



I/O Density

- Two general options to increase I/O Density in HDDs
  - Bandwidth
  - Command efficiency
- Increasing bandwidth comes with cost/power tradeoffs
- Command efficiency can be approached with a better host/drive queuing protocol (i.e., CDL)

Today, we'll review the CDL QoS model and CDL results on production Linux



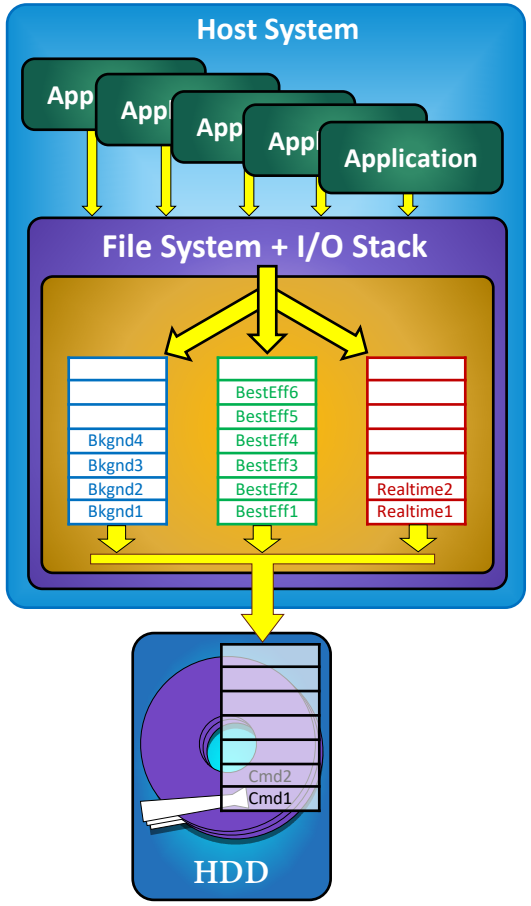
The CDL Model

Toward Command Efficiency: Command Level QoS

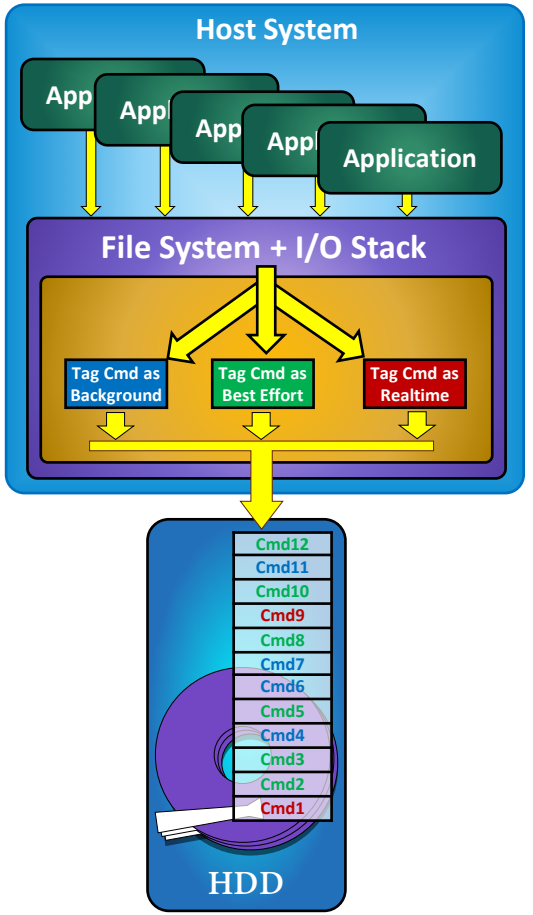
Legacy

- **QD = ~1:** Applications limit queue depth to achieve predictable latency; IOPS lost due to less effective seek ordering, etc.
- **Vague host guidance:** Most priority constructs use abstract priority model, giving only vague latency guidance to drive

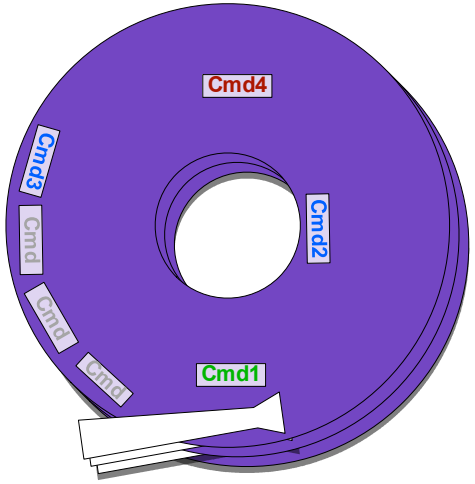
Example Priority Class Schemas				Implied Latency
Real-time	High	#1	…	Low
Best Effort	Medium	#5	…	Medium
Background	Low	#10	…	High



What we want



- **Goal:** Maximize IOPS while maintaining latency targets for specified commands
- **Solution:** Deadline scheduling; application/host can better communicate scheduling and latency intent to drive

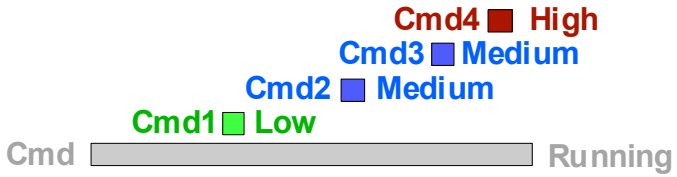


Central Idea of CDL: Deadline vs. Priority Scheduling

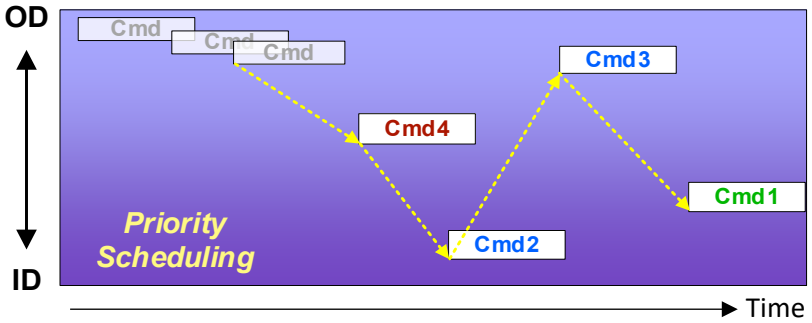
- With deadline scheduling, drive can interpret time limits as relative priorities, but can also use them as actual time
  - Execution urgency increases as deadline approaches

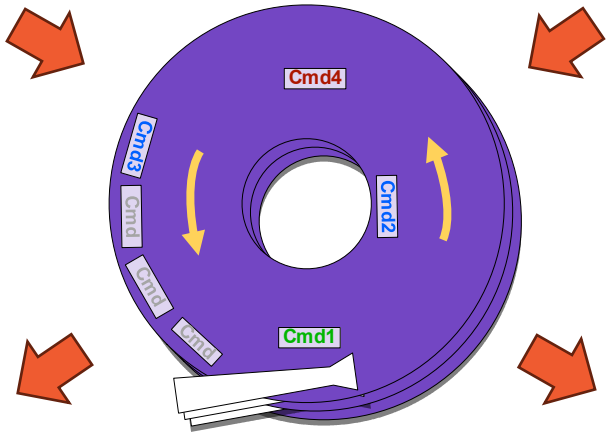
Priority Scheduling

Command Arrival



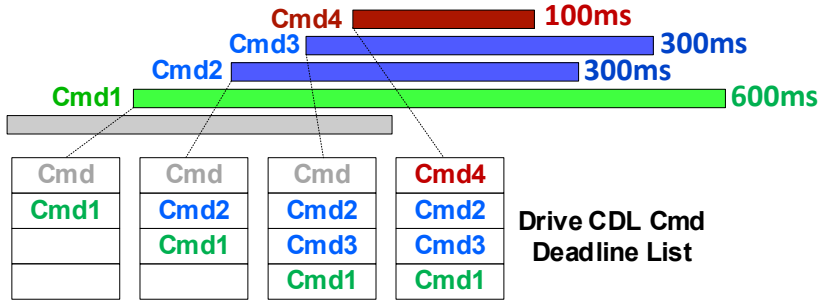
Command Execution



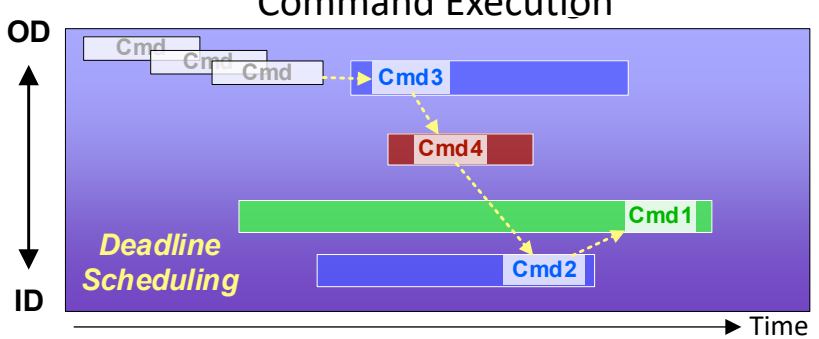


Deadline Scheduling

Command Arrival



Command Execution



CDL Model Overview

1

- Host provisions device with CDL QoS requirements (i.e., Duration Limits)
- Duration Limits are expressed as Time Limits about phases of CDL commands (Active, Inactive, Total) and as Time Limit Policies associated with those Time Limits; i.e., device directives if a Time Limit exceeded)

2

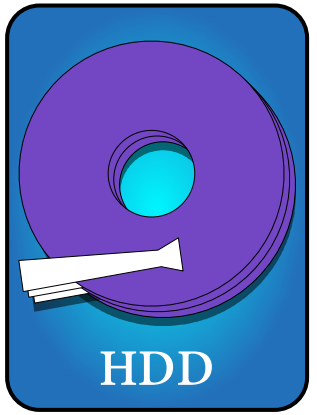
- Host issues duration limited commands that select Duration Limits by selecting a CDL Descriptor using a 3-bit CDL Index

CDL Log/Mode Page
CDL Descriptors
…

CDL Descriptor Table	
Read 1	Write 1
Read 2	Write 2
…	…
Read 7	Write 7

CDL Descriptor
Active Time Limit
Active Time Limit Policy
Inactive Time Limit
Inactive Time Limit Policy
Total Time Limit
Total Time Policy
…

CDL-enabled IO Command		
CDL Index		



3

- Device makes seek optimization and command retirement decisions based on:
  - ① Pre-provisioned Time Limits and Time Limit Policies in the device
  - ② CDL Descriptors selected by IO commands carrying a non-zero CDL Index

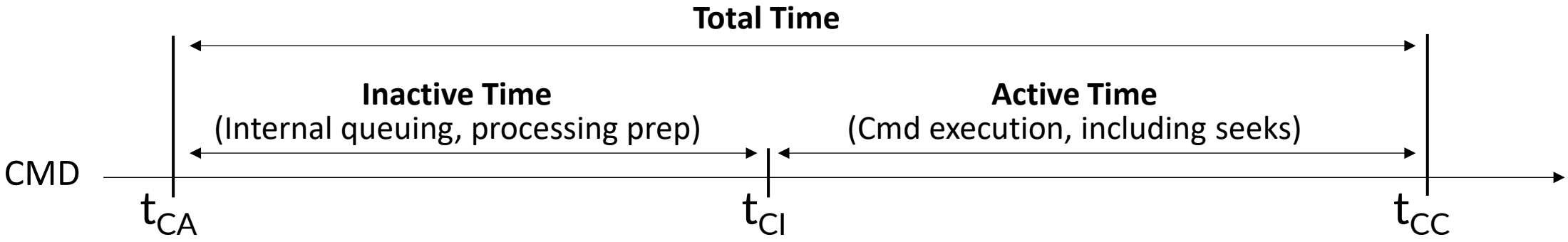
Time Limits for CDL-Enabled Commands

Time **points**

- **Command Acceptance (t_{CA}):** Time at which the device ingests the command
- **Command Initiation (t_{CI}):** Time at which device initiates actions to transfer or act upon data for the command
- **Command Completion (t_{CC}):** Time at which device completes the processing of the command

Time **intervals**

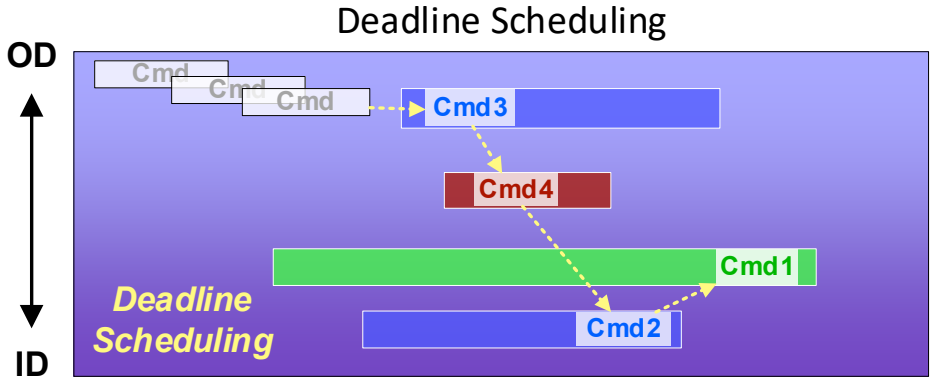
- **Inactive Time (t_{CI} - t_{CA}):** Device preparing command for execution, includes device queuing, scheduling calc.
- **Active Time(t_{CC} - t_{CI}):** Device executing command; i.e., acting on data for the command, including seek time
- **Total Time(t_{CC} - t_{CA}):** Inactive Time + Active Time



Inactive, Active, and Total Time all have associated Time Limits

Interpreting CDL Time Limits

- Time Limit can be configured to be interpreted as a Scheduling Guideline
  - Host direction on **relative importance of a command**
  - Focus on command scheduling as related to IOPS



- Time Limit can be configured to be interpreted as a Command Timeout
  - Host direction on **processing of a command in flight**
  - Focus on command retirement as related to tail latency

"Stop processing this command vs. continuing to try recovery because I have a duplicate of the data somewhere else." (i.e., Fast Fail)

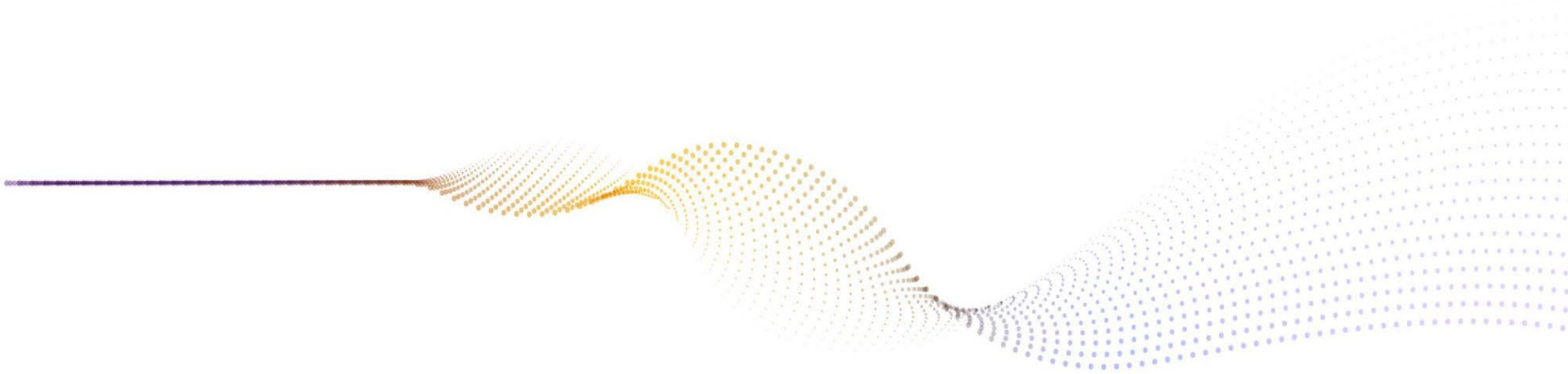
These two semantics are central to how hosts set, and devices process, Time Limits and Time Limit Policies to achieve CDL QoS goals

CDL Time Limit Policies: What if a Time Limit is Exceeded?

Time Limit Semantic	CDL 1.0 Codepoints (ACS5/SPC6)	CDL 2.0 Codepoints (ACS6/SPC7)	Device Policy
Scheduling Guideline	0h	4h	Complete the command at the earliest possible time after the specified Time Limit is exceeded (i.e., best effort)
	1h	3h	Use the next CDL Descriptor in the Descriptor Log and process the Duration Limits as defined in that descriptor
	2h	5h	Process the command as if no CDL descriptor has been specified (i.e., no Duration Limits for this command)
Command Timeout	Dh	Dh	Complete the command, returning normal status, with additional status “data currently unavailable”
	Eh	Eh	Report command aborted; may indicate range of data processed before abort **(SCSI only)**
	Fh	Fh	Report command aborted with a status indicating a command timed out before (inactive limit) or during (active limit) processing

Segue to some real-life results

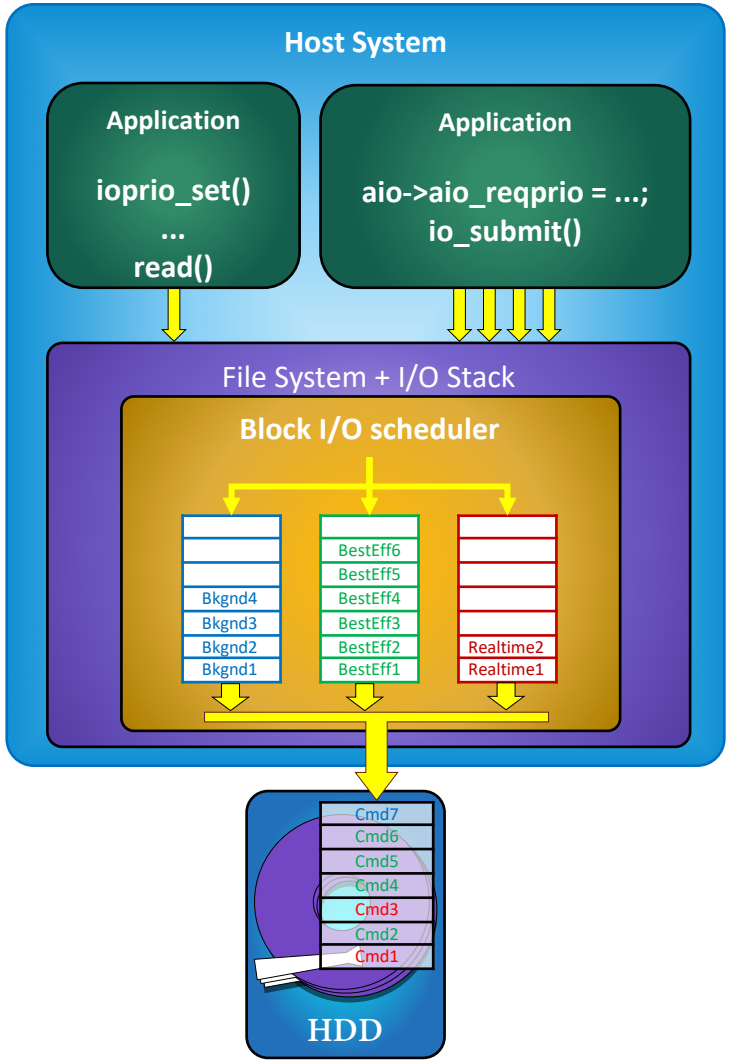
- We have shown the basic model of CDL
  - Time Limits (Active, Inactive, Total) and Time Limit (Exceeded) Policies
  - Host can express QoS for command scheduling and command retirement
- CDL v1.0 was published in ACS-5 and SPC-6
  - CDL v2.0 is coming in ACS-6 and SPC-7
  - Some new semantics but maintains backward compatibility
- We have not covered a related feature set: Sequestered Commands
  - More encouragement for hosts to offload more commands to drives
- Now we'll show some data on CDL based on up-streamed Linux code and real drives



Linux Support

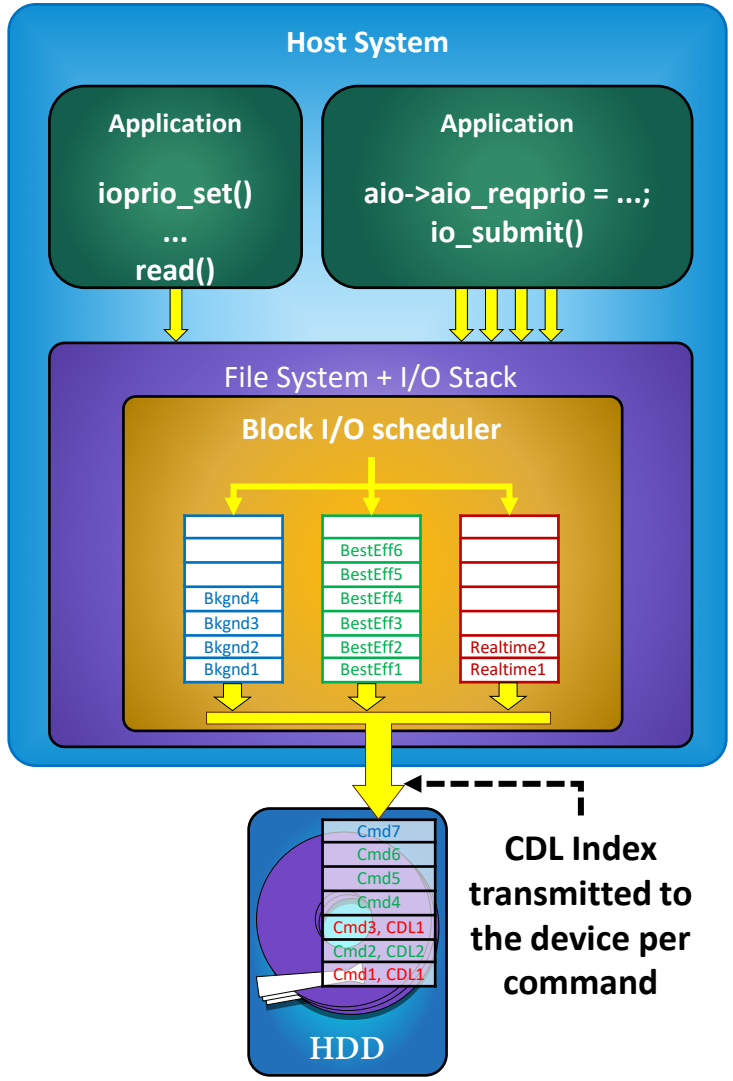
Linux Kernel and I/O Priorities

- Linux I/O priority interface defines 3 classes
  - Real-time (high priority)
    - Maps to high-priority commands for devices supporting NCQ Priority
  - Best-effort (default)
  - Idle (lowest priority for background traffic)
- I/O priorities can be specified per-process or per I/O
  - ioprio_set() system call
  - Per I/O is possible only for asynchronous I/O with the aio_reqprio field of asynchronous I/O descriptors
- The Kernel I/O scheduler supports I/O priority scheduling
  - Traditional elevator scheduling (increasing LBA order) within each priority class to reduce seek overhead
  - Includes starvation prevention for low-priority I/Os



CDL and Linux I/O Priorities

- A command duration limit is specified by adding a **hint** to a Linux I/O priority
  - priority = IOPRIO_PRIO_VALUE_HINT(class, 0, **hint**)
  - The hint value corresponds to a CDL descriptor number
    - Value of 1 to 7, with the value "0" reserved for "no limit"
- The kernel and block I/O scheduler do not act on a priority hint value
  - I.e. short CDL limit I/Os issued with a low priority may suffer long latencies due to the host queuing time
  - Users are responsible for using an appropriate I/O priority class for a particular CDL limit
    - E.g. higher I/O priority class for short CDL limit (urgent) I/Os
      - Reduces the host-level queuing time and improves the precision of CDL limits as seen by the user

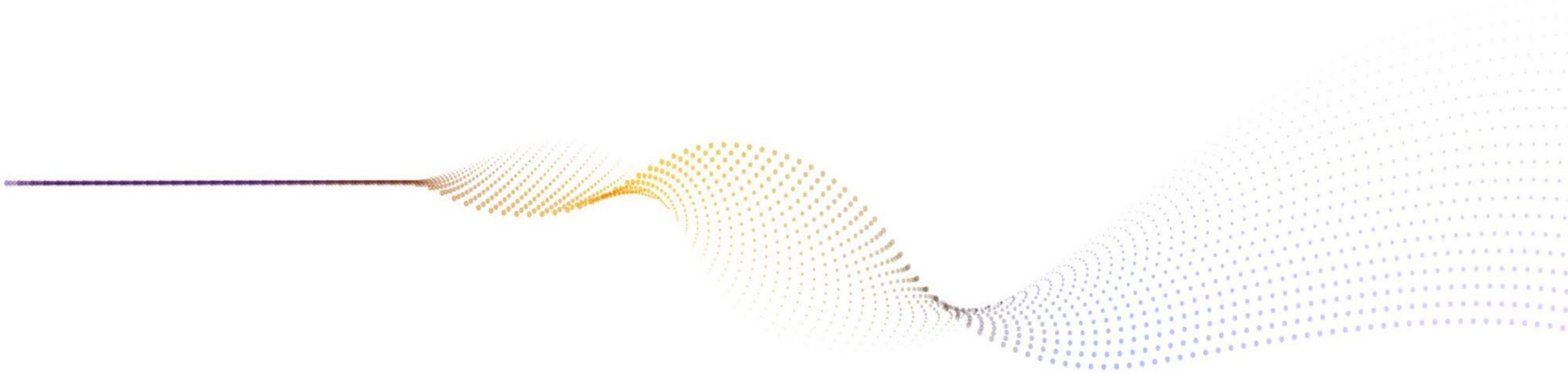


Linux Tooling for CDL

- ***cdl-tools*** project
  - Provides the *cdladm* command line utility to manage the CDL descriptors of an HDD
    - Allows consulting and changing CDL descriptor pages and manage CDL statistics
  - Hosted on github
    - https://github.com/westerndigitalcorporation/cdl-tools
- ***fio*** v3.36 and above include options for controlling I/O priorities
  - Example: 128 KiB random read workload with 20% of read commands using CDL index 1 with the best effort I/O priority class

```
$ cdladm show /dev/sdk
Device: /dev/sdk
   Vendor: ATA
   Product: WDC  WUH722626AL
   Revision: WJ20
   50782535680 512-byte sectors (26.000 TB)
   Device interface: ATA
    ACS version: ACS-5
    SAT Vendor: LSI
    SAT Product: LSI SATL
    SAT revision: 0008
   Command duration limits: supported, disabled
   Command duration guidelines: supported
   High priority enhancement: not supported, disabled
   Statistics: supported
   Duration minimum limit: 20000000 ns
   Duration maximum limit: 4294967295000 ns
System:
   Node name: xxx
   Kernel: Linux 6.15.8-200.fc42.x86_64 #1 SMP PREEMPT_DYNAMIC Thu Jul 24 13:26:52 UTC 2025
   Architecture: x86_64
   Command duration limits: supported, disabled
   Device sdk command timeout: 60 s
Page T2A: read descriptors
 perf_vs_duration_guideline : 20%
 Descriptor 1:
   max inactive time       : No limit
   max inactive policy     : best-effort
   max active time         : 50 ms
   max active policy       : best-effort
   duration guideline      : 100ms
   ...

$ fio --name="test" --filename=/dev/sdk --random_generator=tausworthe64 \
                    --rw=randread --ioengine=libaio --iodepth=32 --bs=128k \
                    --direct=1 --runtime=60 --continue_on_error=none \
                    --cmdprio_class=2 --cmdprio_hint=1 --cmdprio_percentage=20
```

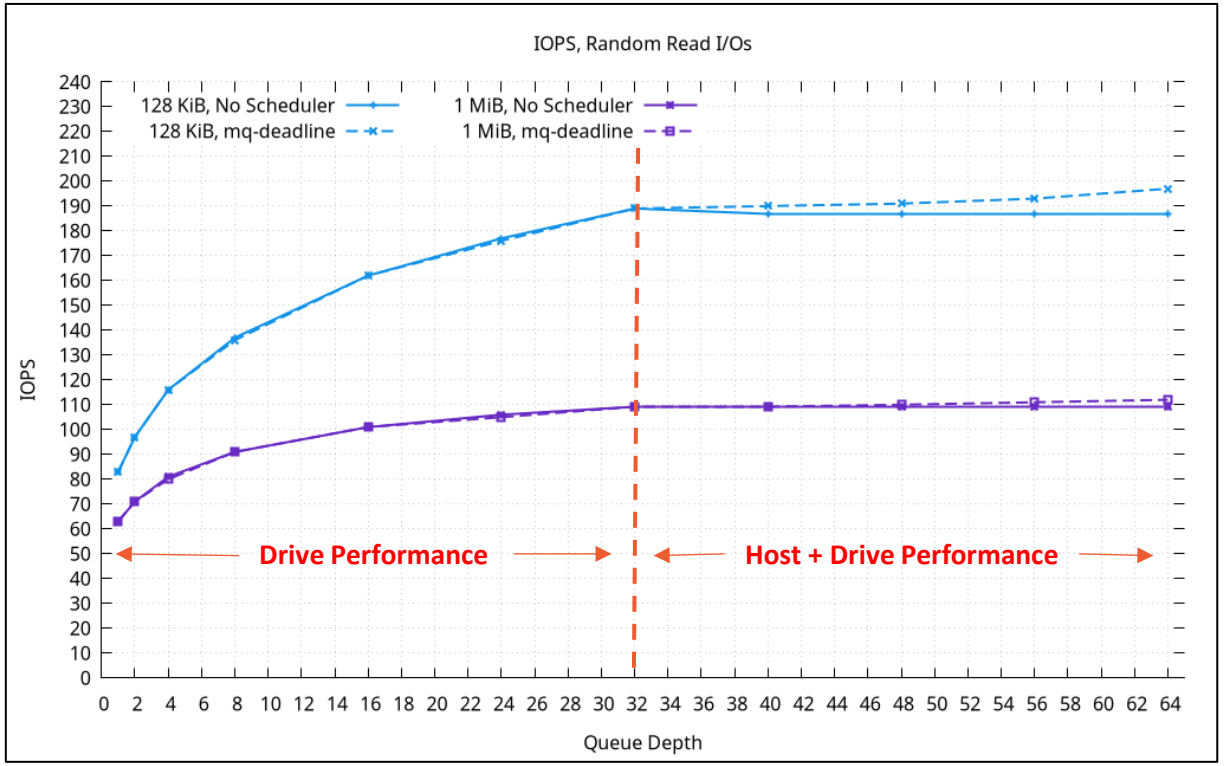
CDL Use Examples

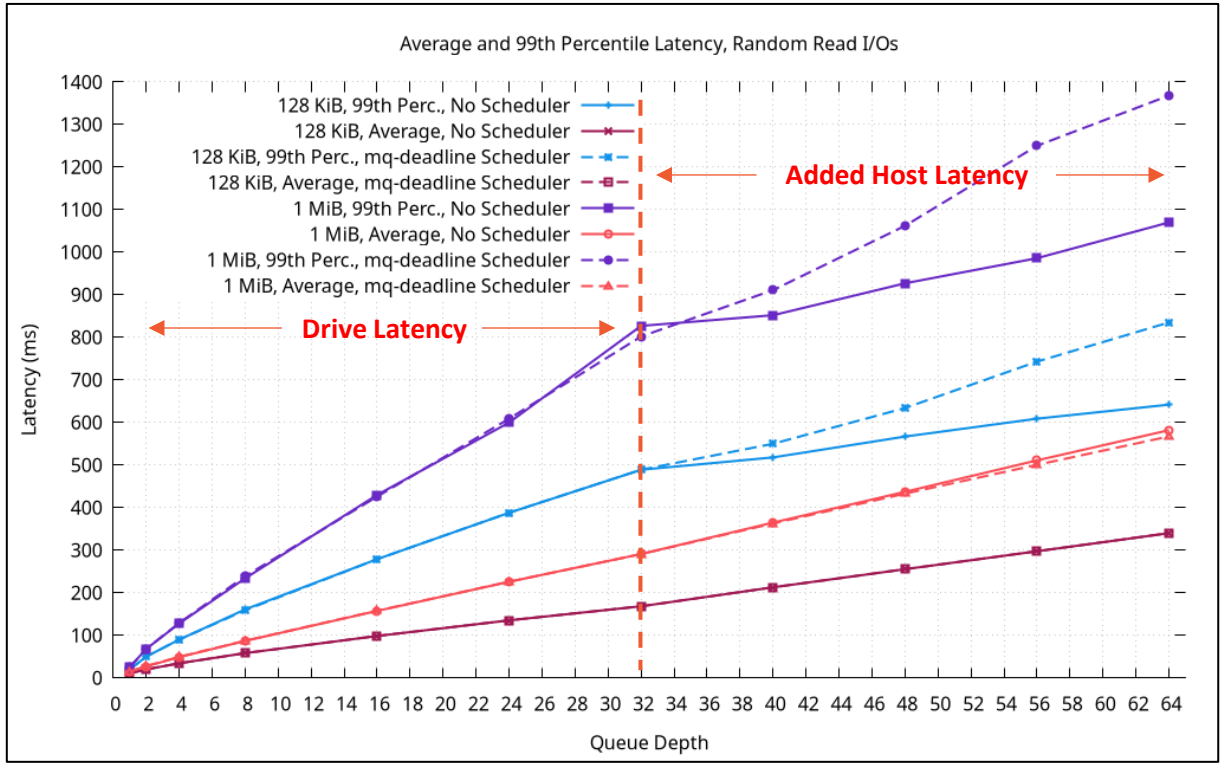
Environment and Workloads

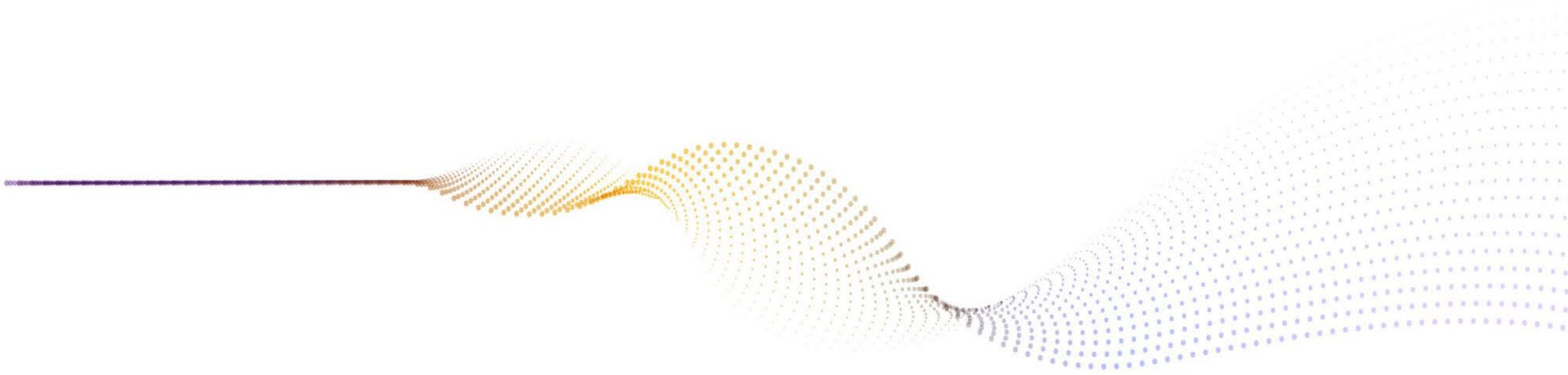
- Host
  - 8-cores (16 threads) x86_64 CPU, 128 GiB DRAM
  - 26 TB SATA CMR HDD connected to an AHCI SATA port
  - Fedora 42 OS
    - Kernel version 6.15.9
- Workloads
  - 128 KiB and 1 MiB random read workloads with varying queue depth
  - CDL with best-effort (no abort policy) single limit and two limits
  - CDL fast-fail examples (CDL policies 0xD and 0xF)
- The performance metrics used are IOPS and the 99th percentile tail latency of I/Os

HDD Baseline Performance

- IOPS and tail latency increase with the workload queue-depth
- Average tail latency increases with the workload queue depth
  - Increase more pronounced beyond max QD; even more pronounced with IO scheduler
    - Host queuing time increases as more time spent in IO scheduler queues





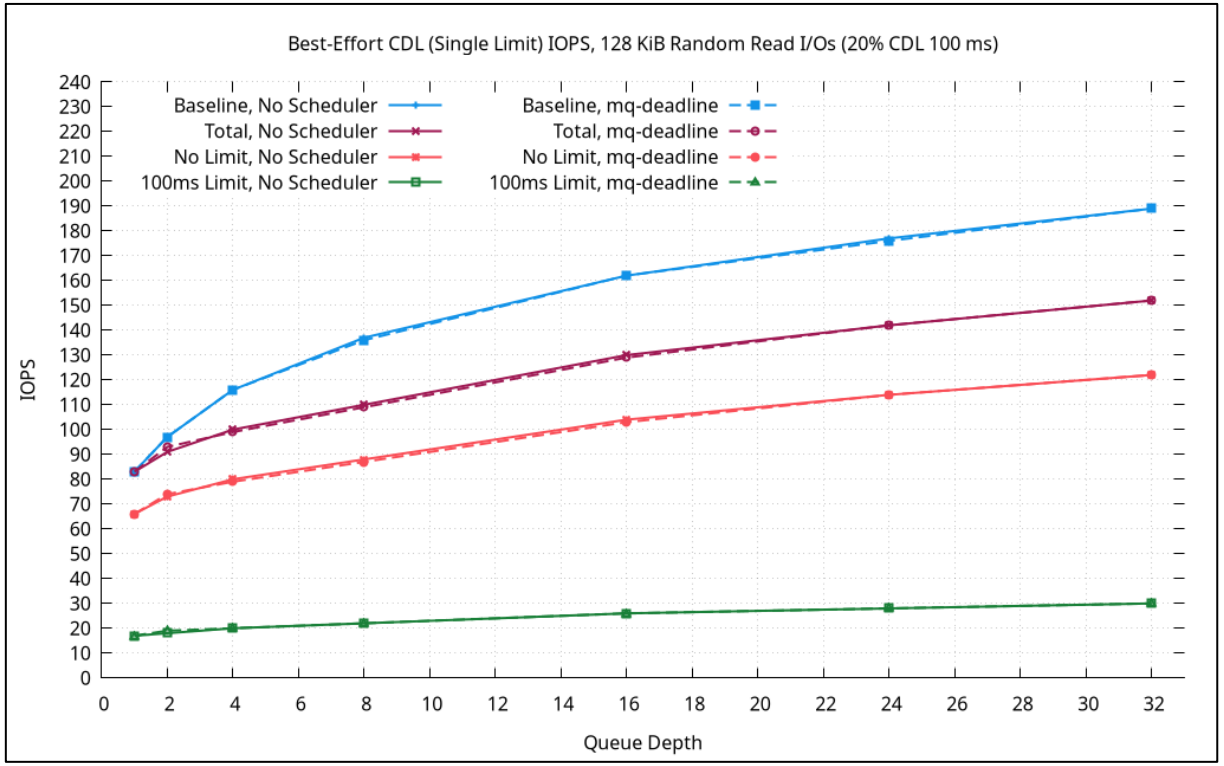


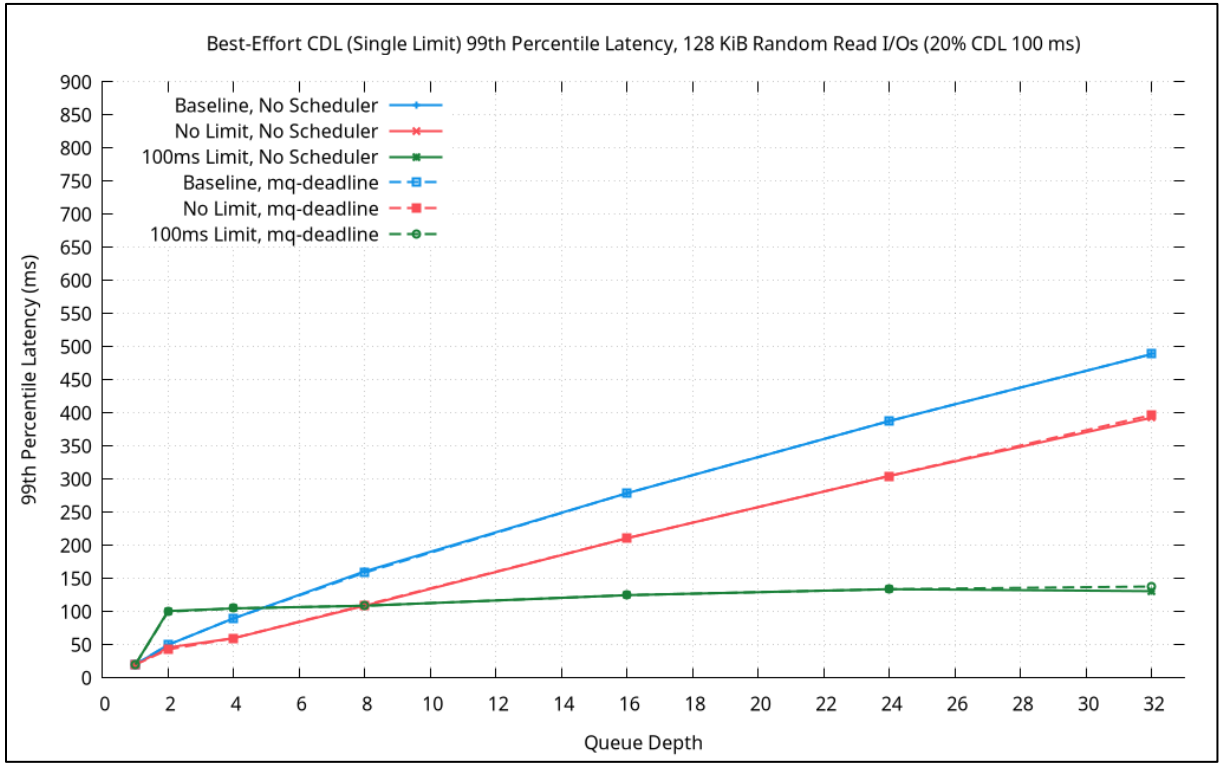
CDL Single Limit Example

20 % of I/Os with 100ms Duration Guideline Limit

CDL I/Os as Best-Effort Priority, Queue Depth <= 32

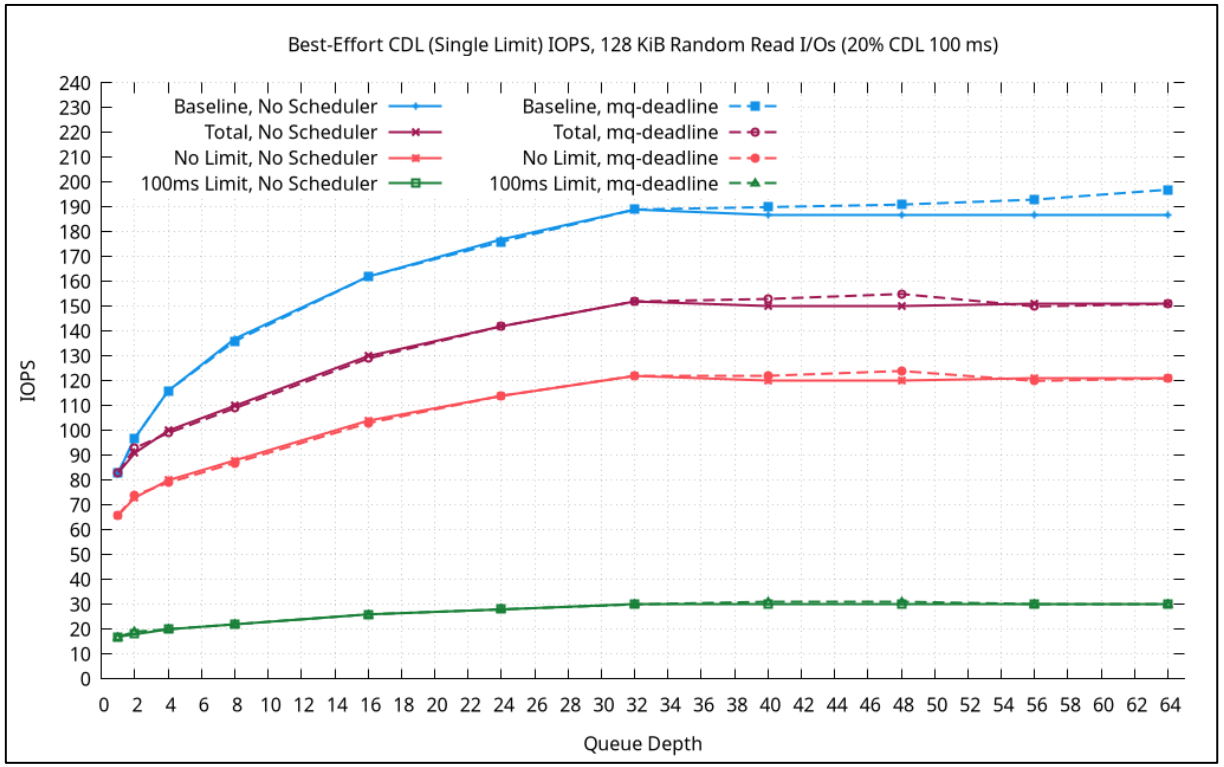
- Overall Result: IOPS reduced but latency targets maintained
  - Tail latency of I/Os with no CDL limit also improves
  - Kernel scheduler (mq-deadline) has little to no effect

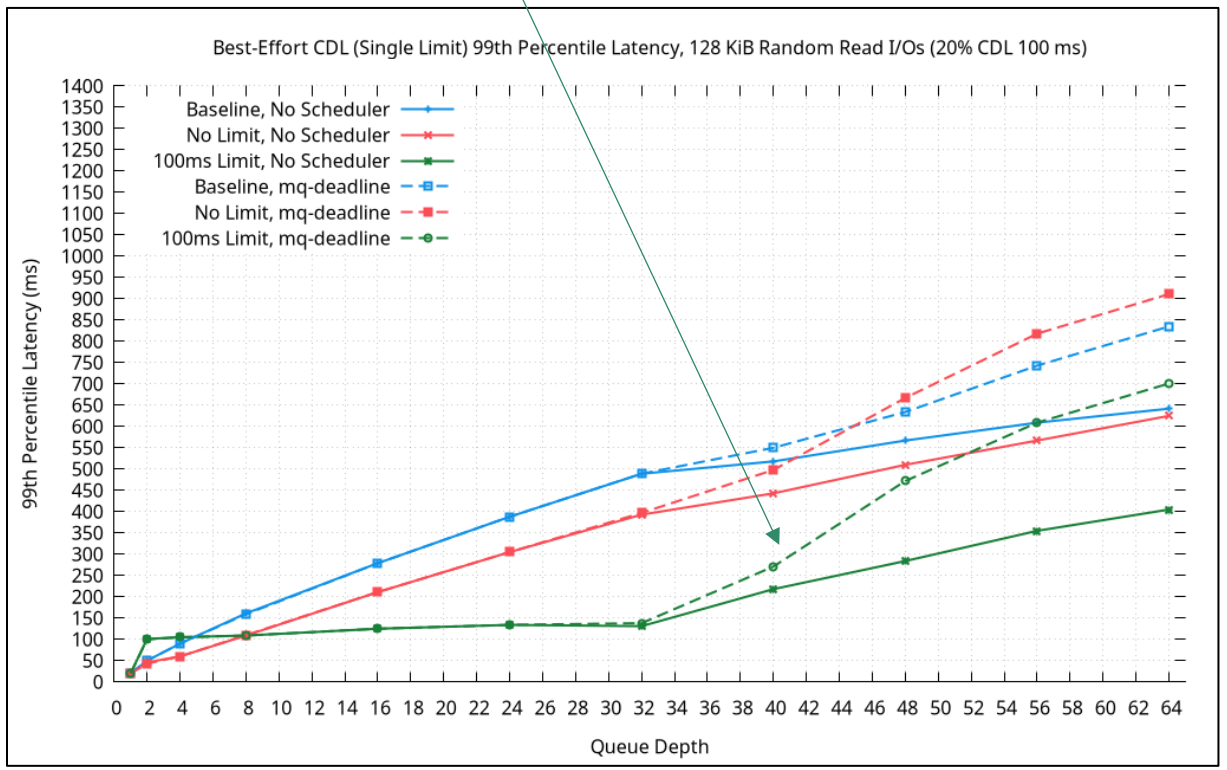




CDL I/Os as Best-Effort Priority, Queue Depth > 32

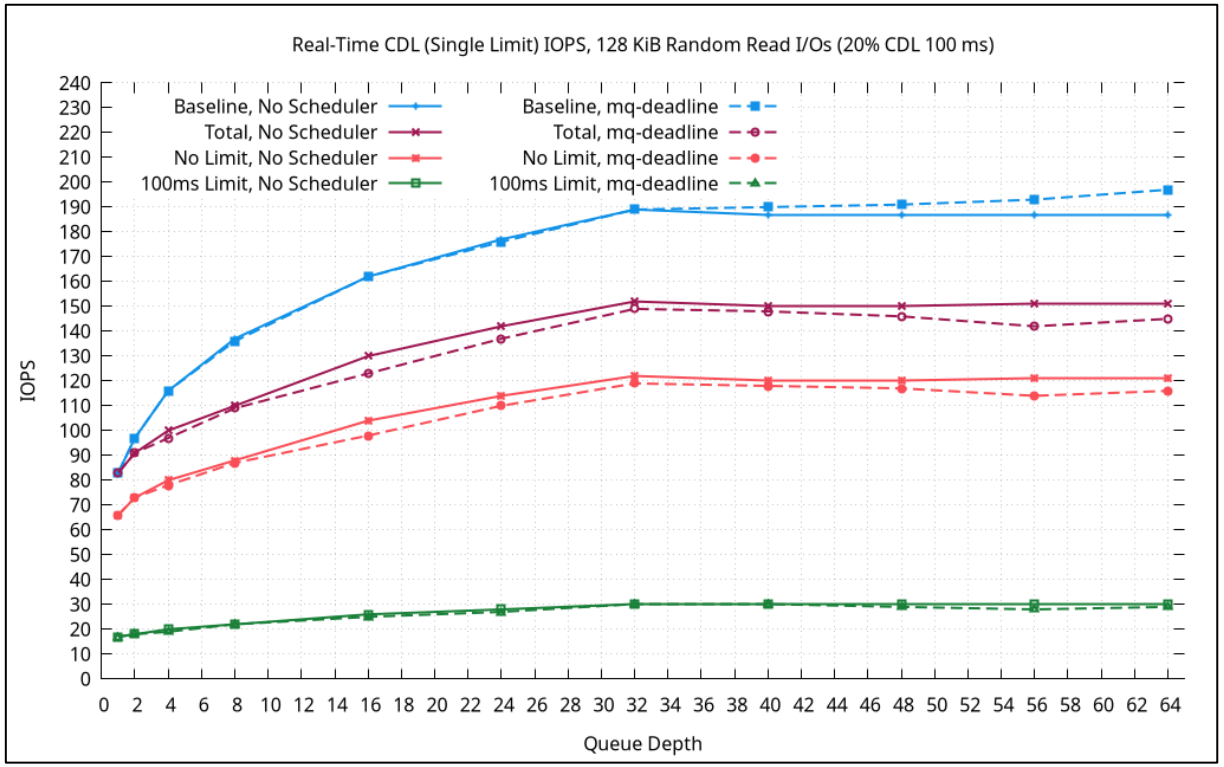
- In general, tail latency for CDL I/Os increases significantly when QC > 32
- Block I/O scheduler increases host queuing time for the CDL limited I/Os
  - Scheduler has no knowledge of CDL and thus increases tail latency even more

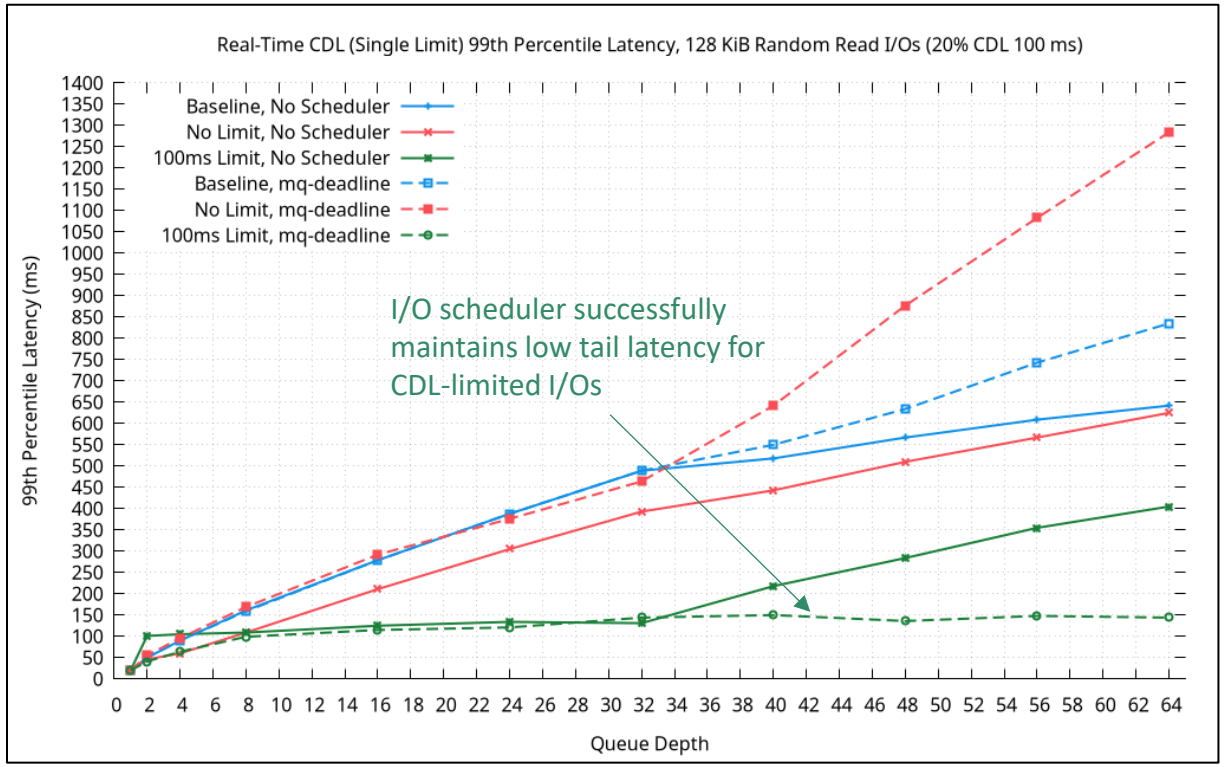




CDL I/Os as Real-Time Priority, Queue Depth > 32

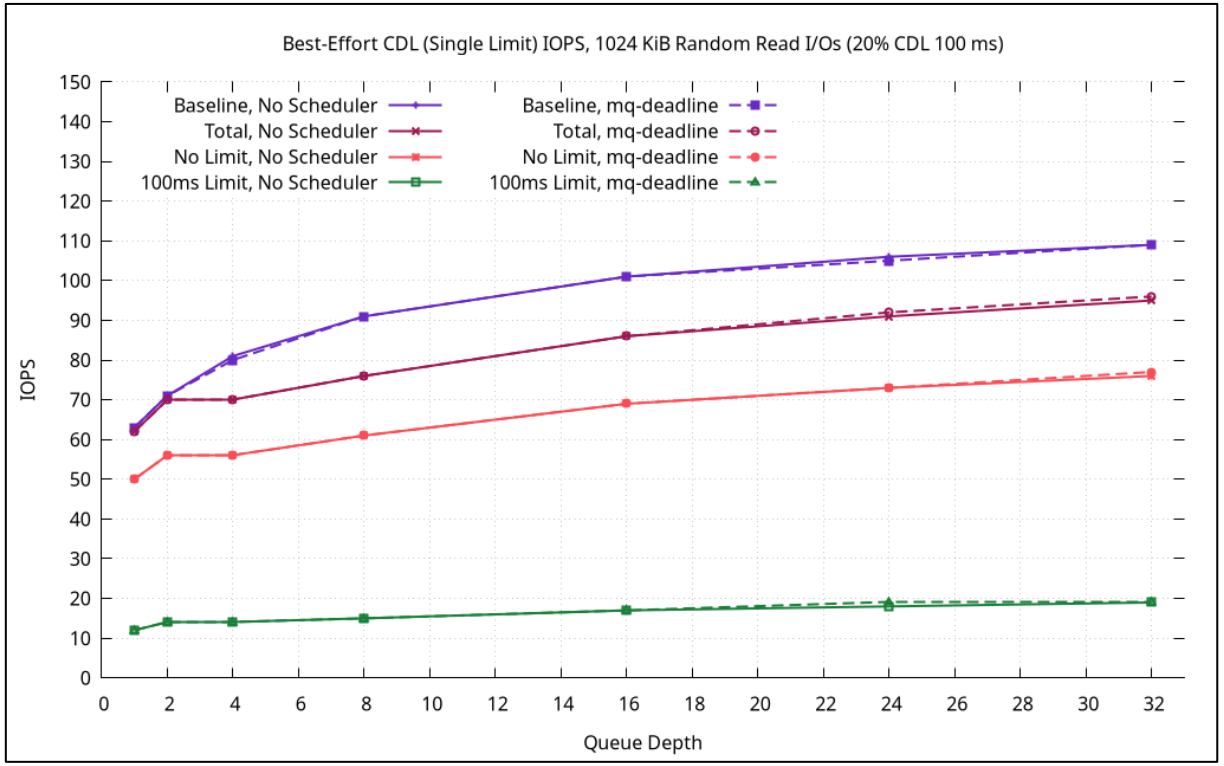
- Host I/O scheduler prioritizes time limited commands, resulting in low tail latency, even beyond the HDD maximum queue depth (32)
  - Very slightly lower IOPS

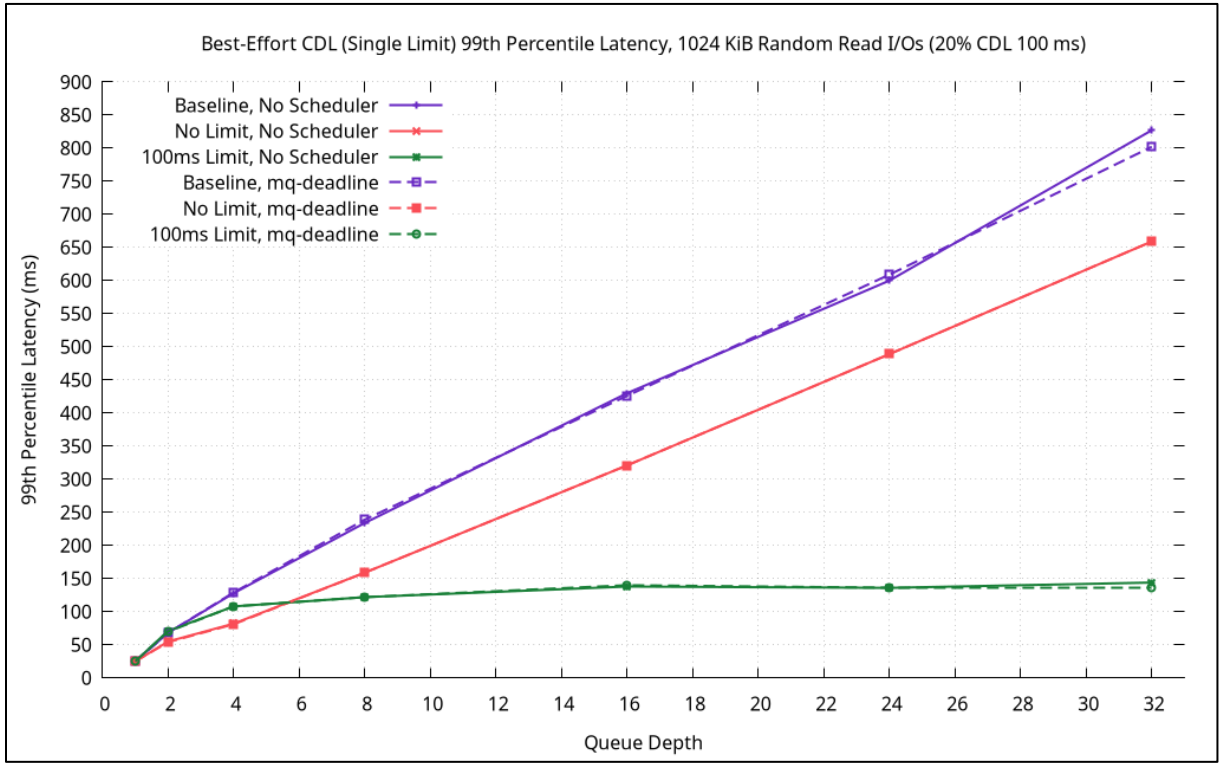


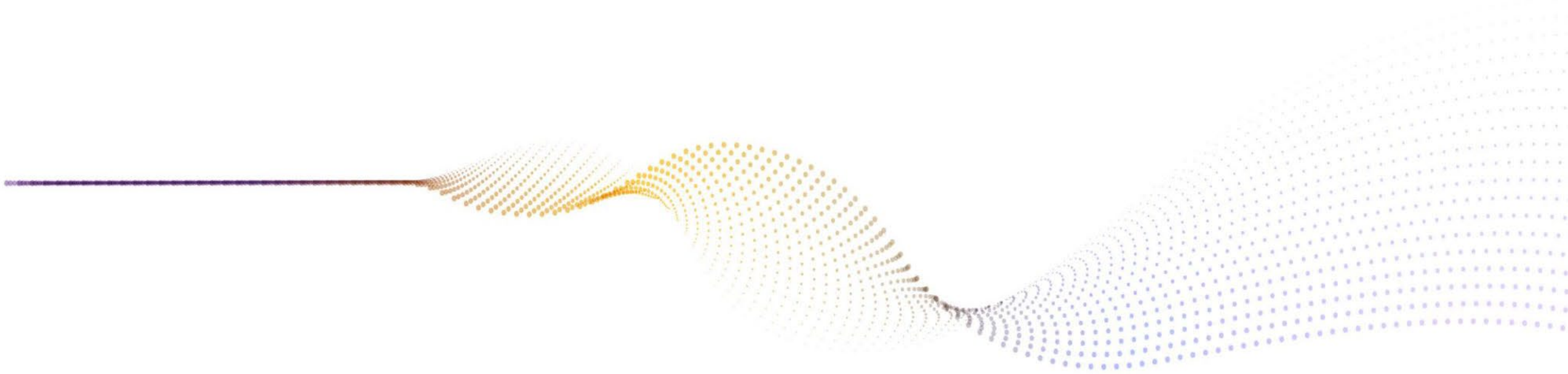


Large 1 MiB CDL I/Os as Best-Effort Priority, QD <=32

- Tail latency for large CDL I/Os remains well controlled
  - HDD internal command scheduling aware of longer command data transfer times







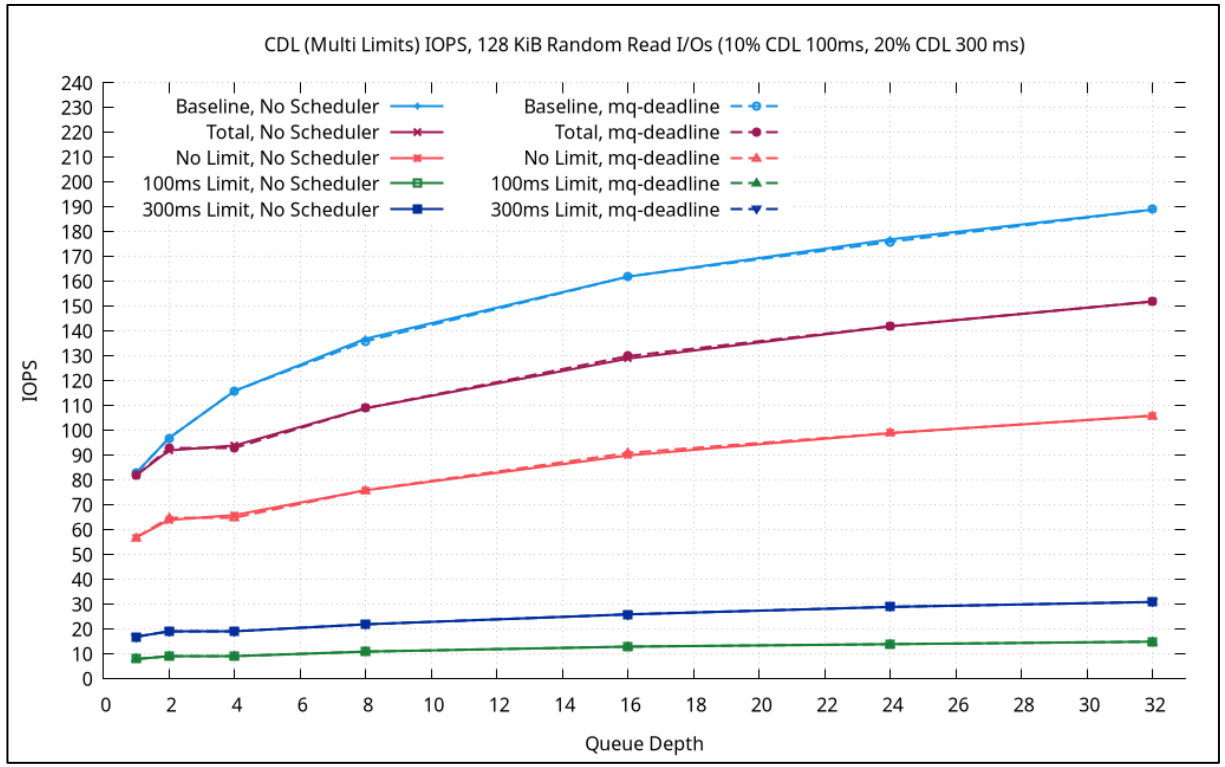
CDL Dual Limit Example

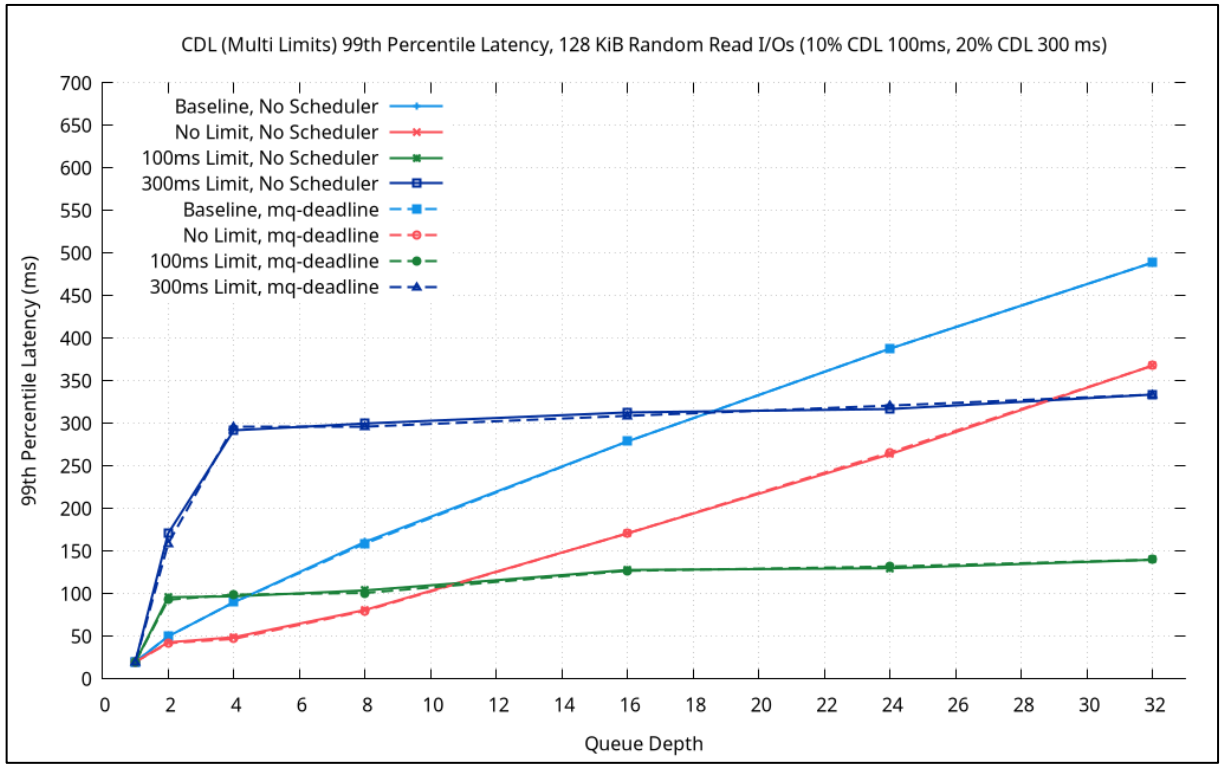
10 % of I/Os with 100ms Duration Guideline Limit

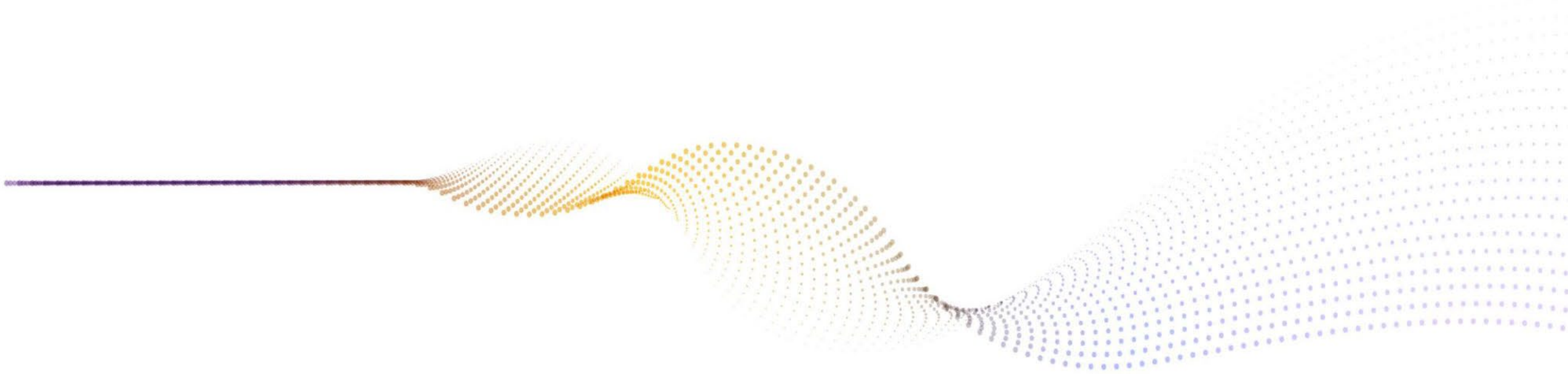
20 % of I/Os with 300ms Duration Guideline Limit

Multi-Class CDL I/Os as Best-Effort I/Os

- The two CDL-limited I/O classes show the expected tail latency
  - Overall IOPS is the same as for the single CDL limit case
  - The tail latency of I/Os with no CDL limit remains low







CDL Fast Fail Example

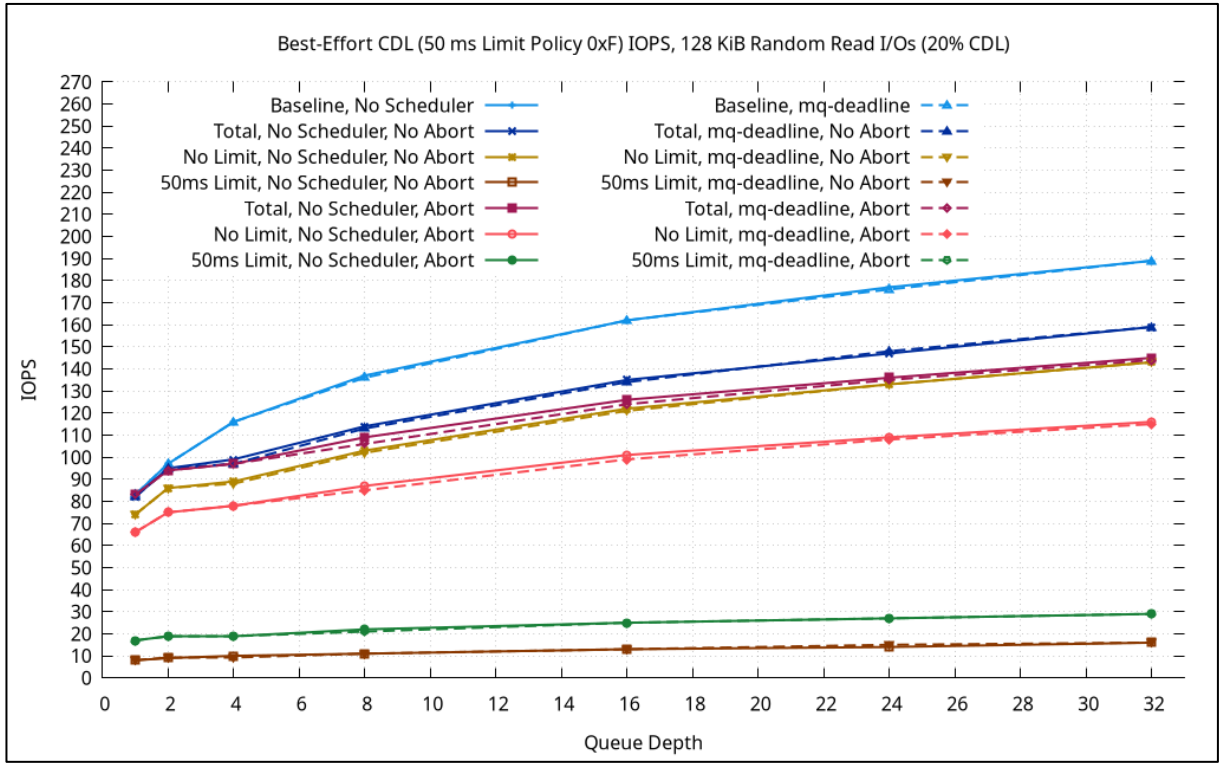
20 % of I/Os with 50ms Duration Guideline Limit and 50ms inactive time limit

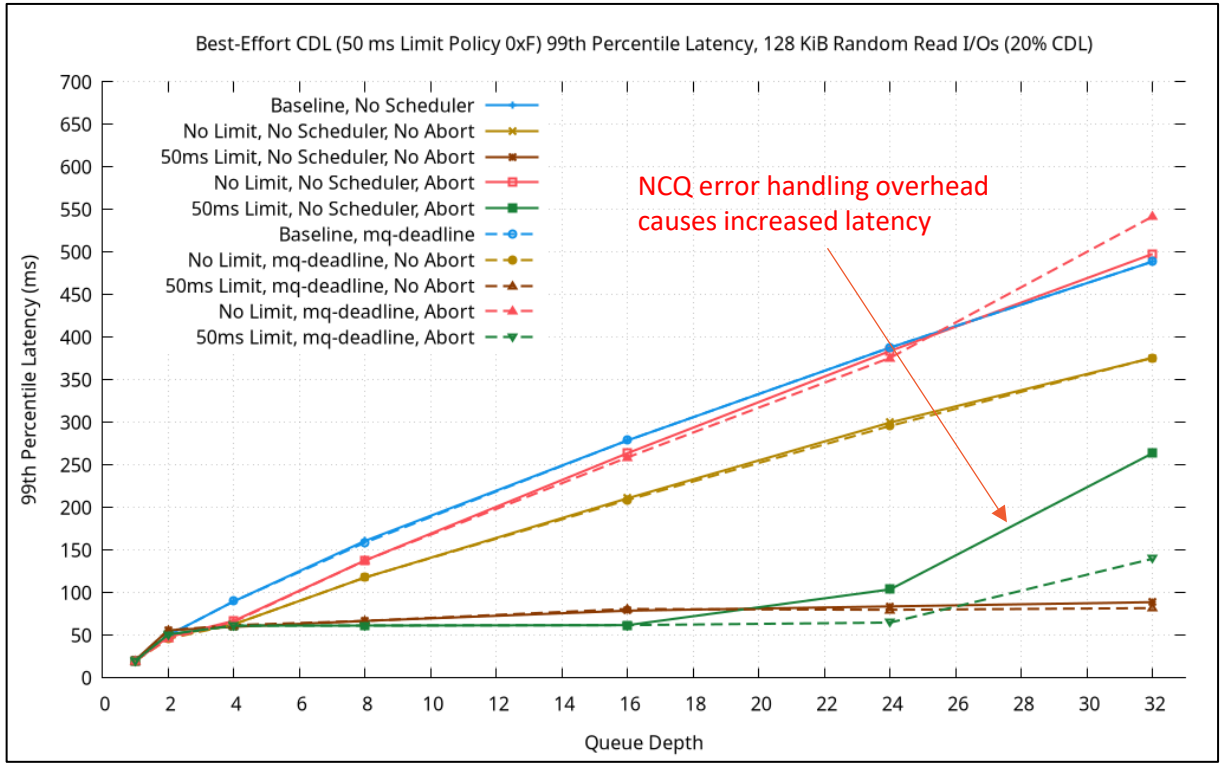
When to Use Fast Fail ?

- When a user I/O is split into multiple I/Os issued to different devices
  - Erasure coded storage system (e.g. distributed object store or RAID)
    - A device I/O failing can be recovered using parity/erasure coding
- CDL policies allow controlling the latency of the entire user I/O by forcing an abort of device I/Os exceeding the user I/O deadline
  - Avoids long user I/O latencies if one drive is slow (e.g. a bad sector is accessed)
- Two CDL command timeout policies apply
  - **0xD**: If command misses its limit, abort it with success status and send sense data indicating "data not available"
    - With SATA drives, this requires reading a log page to obtain the command sense data
    - In Linux, this is currently done using the NCQ error recovery mechanism, that is, using a non-NCQ read log command which causes a drive queue stall
      - Improved handling of policy 0xD is being developped
  - **0xF (or 0xE with SAS HDDs)**: If the command misses its limit, abort it with an error status
    - With SATA drives, this causes an NCQ queue abort (all queued commands are aborted)

CDL Fast-Fail I/Os and Policy 0xF

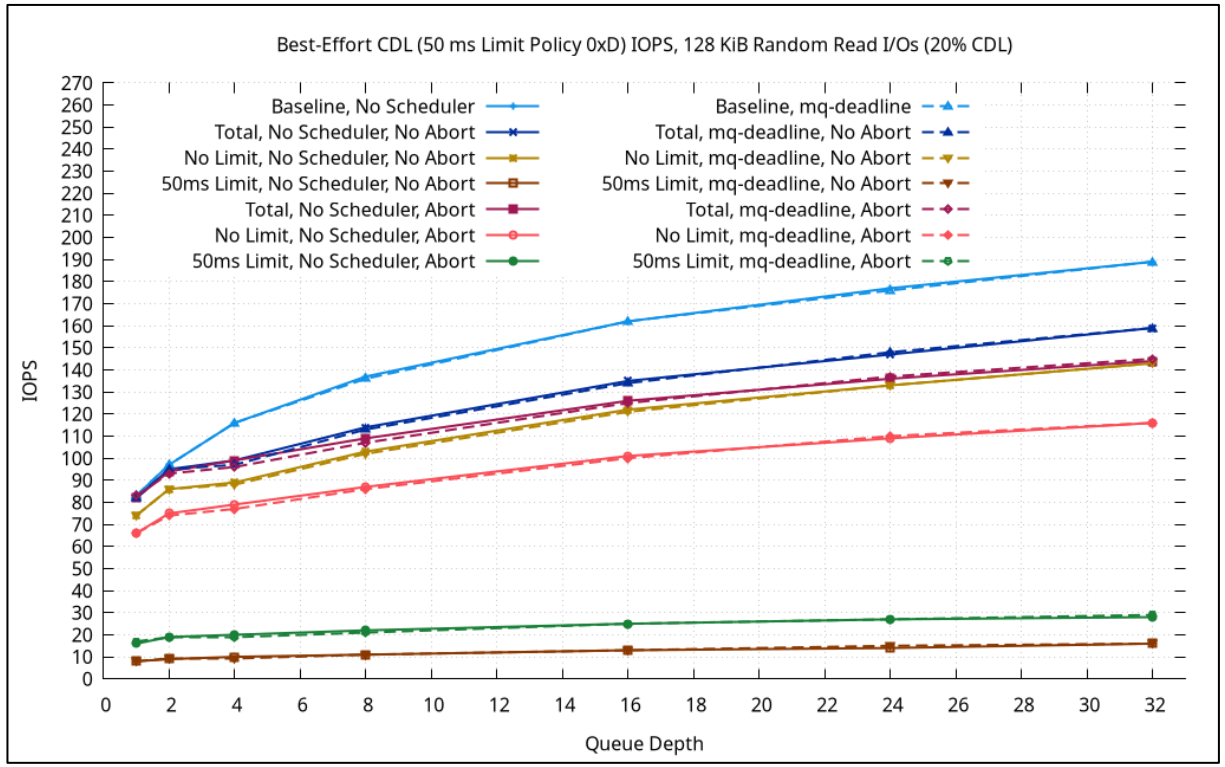
- Lower total IOPS, but higher IOPS for CDL-limited commands
  - The overhead of NCQ-error handling degrades total IOPS and increases the tail latency of CDL-limited commands at high queue depth

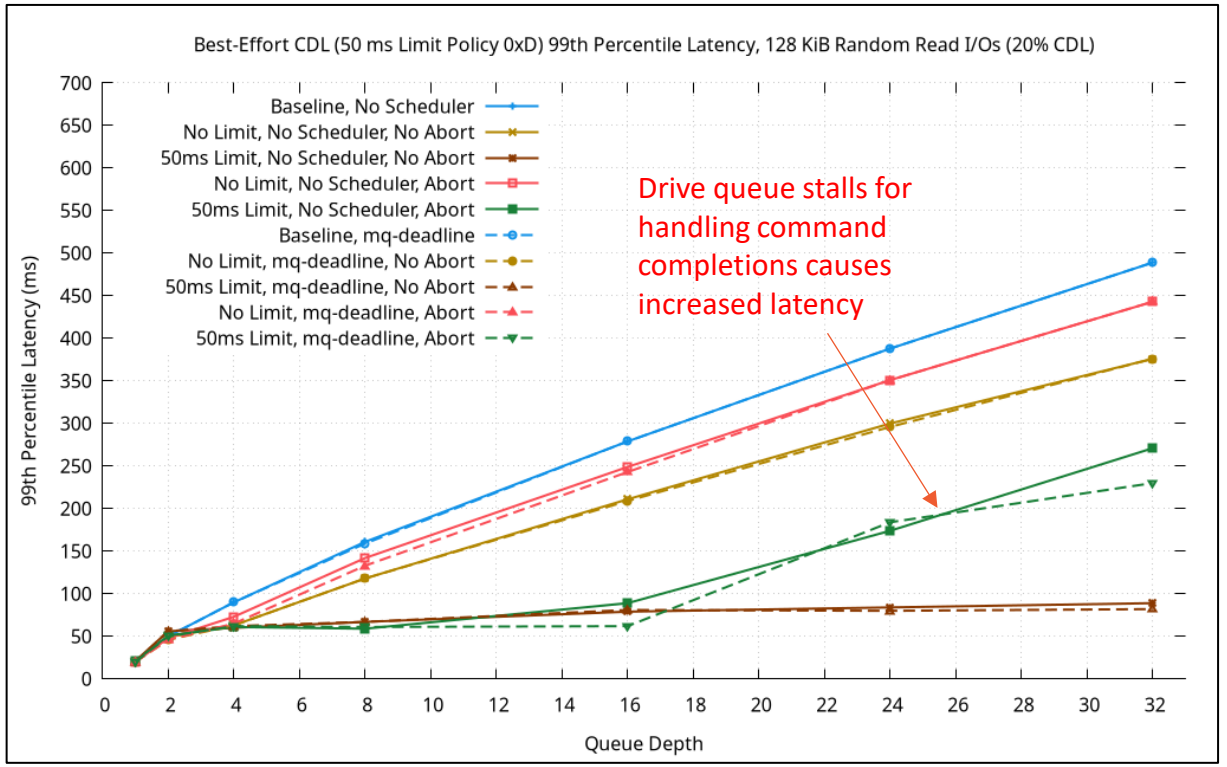




CDL Fast-Fail I/Os and Policy 0xD

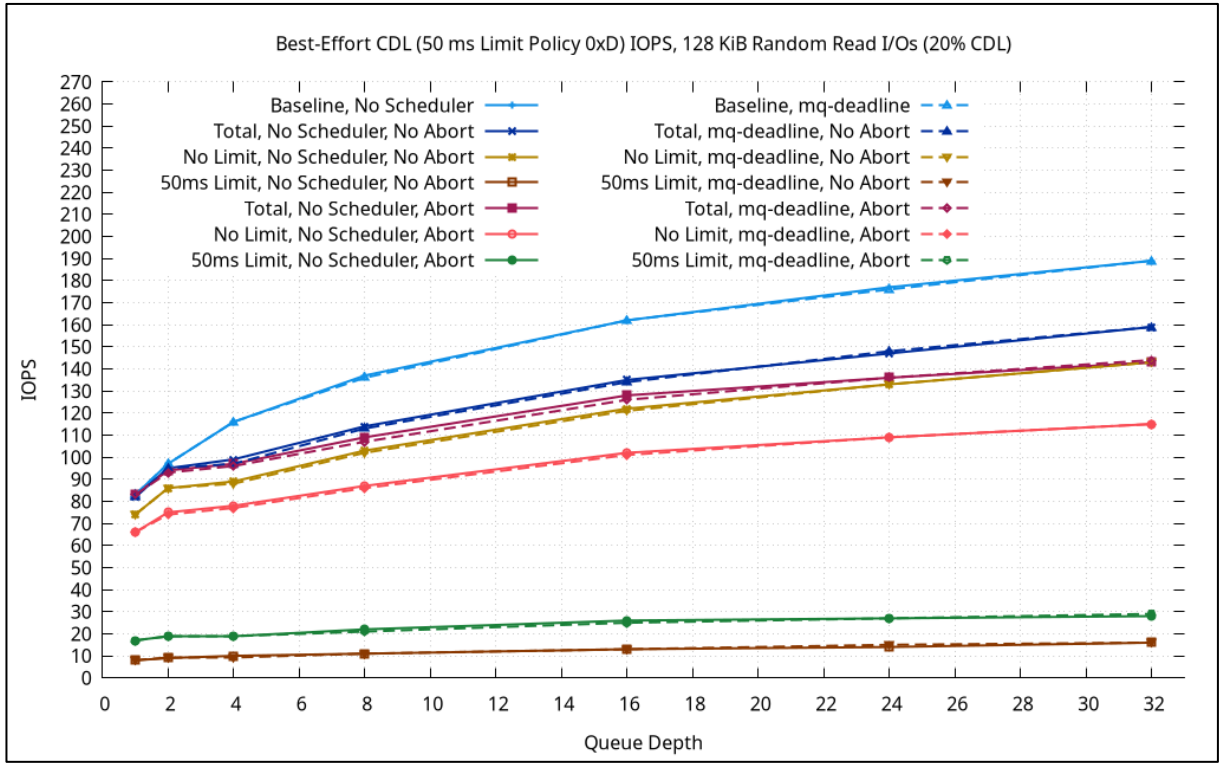
- Like policy 0xF, similar lower total IOPS and higher CDL-limited IOPS
  - Linux kernel libata (libsas) SAT handling of policy 0xD aborts is not efficient

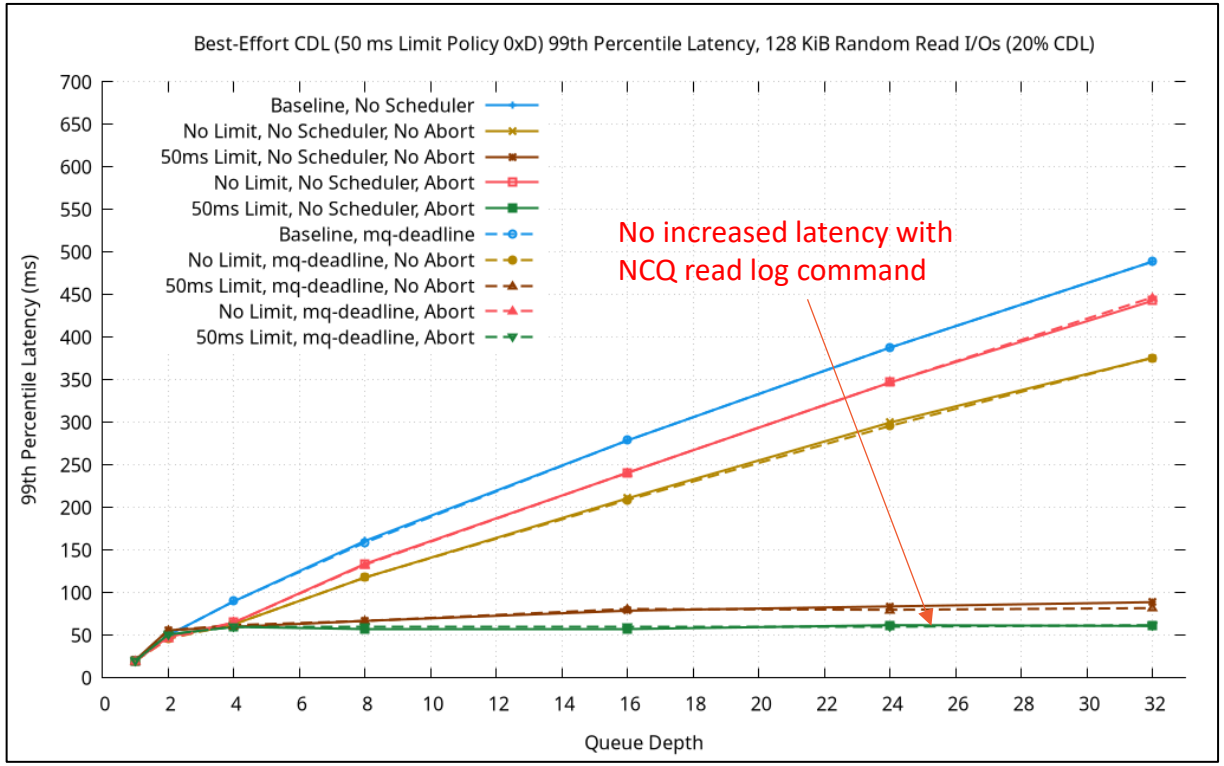




CDL Fast-Fail I/Os and Policy 0xD: Improved Handling

- Modified Linux kernel libata/libsas handling of policy 0xD aborts using an NCQ read log command to obtain command sense data
  - No degradation of the tail latency at high queue depth



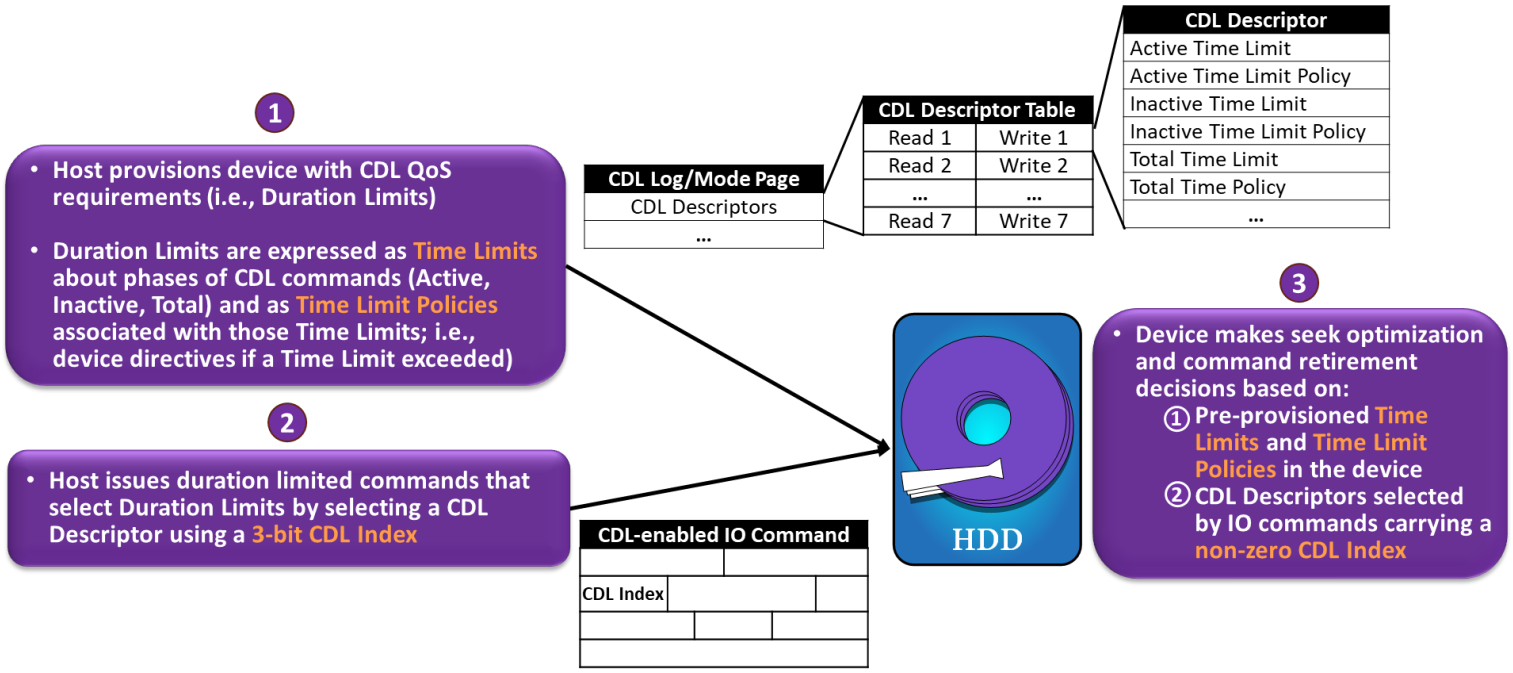


Summary: Host-Device Cooperation for Command Level QoS

- CDL QoS model is standardized
  - ACS (ATA) and SPC (SCSI)
- CDL supported in Linux
  - ***fio*** v3.36 and above include options for using CDL
  - ***cdladm*** utility to manage CDL descriptors
- CDL has shown results
  - Improves IOPS while maintaining tail latency
- CDL still evolving
  - Needs ecosystem involvement to perfect the model
  - Will fit into NVMe should NVMe HDDs emerge

Summary: Host-Device Cooperation for Command Level QoS

- CDL QoS model is standardized
  - ACS (ATA) and SPC (SCSI)
- CDL supported in Linux
  - ***fio*** v3.36 and above include options for using CDL
  - ***cdladm*** utility to manage CDL descriptors
- CDL has shown results
  - Improves IOPS while maintaining tail latency
- CDL still evolving
  - Needs ecosystem involvement to perfect the model
  - Will fit into NVMe should NVMe HDDs emerge



Thank you for attending!

damien.lemoal@wdc.com, dave.landsman@wdc.com

Please remember to rate this session. You can access the presentations at
http://sniadeveloper.org/conference

