STORAGE DEVELOPER CONFERENCE

SDC21

BY Developers FOR Developers

Virtual Conference
September 28-29, 2021

A SNIA Event

Unleashing the Performance of Multi-Actuator Drives

Arie van der Hoeven, Principal Project Manager

Tim Walker, Principal Engineer

Seagate Technology

Agenda

- The Multi-Actuator Era
- Developing for SAS and SATA multi-actuator drives
- Deploying SATA multi-actuator drives
- Performance guidance

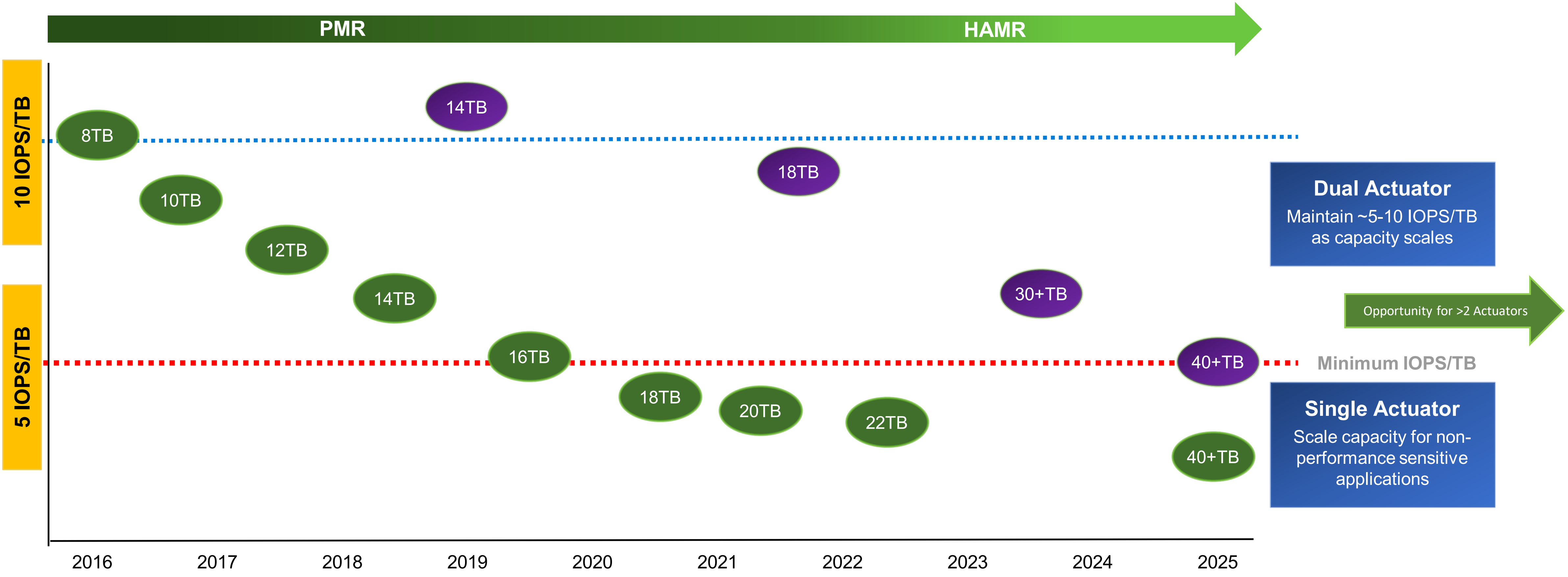
You'll leave knowing how to

- Deploy systems to take advantage of multi-actuator performance
- Prepare applications and workloads to take advantage of multi-actuator environments

The Multi-Actuator Era

"All that data cannot sit behind a single actuator"

Solving the Performance Challenge for Cloud Architectures



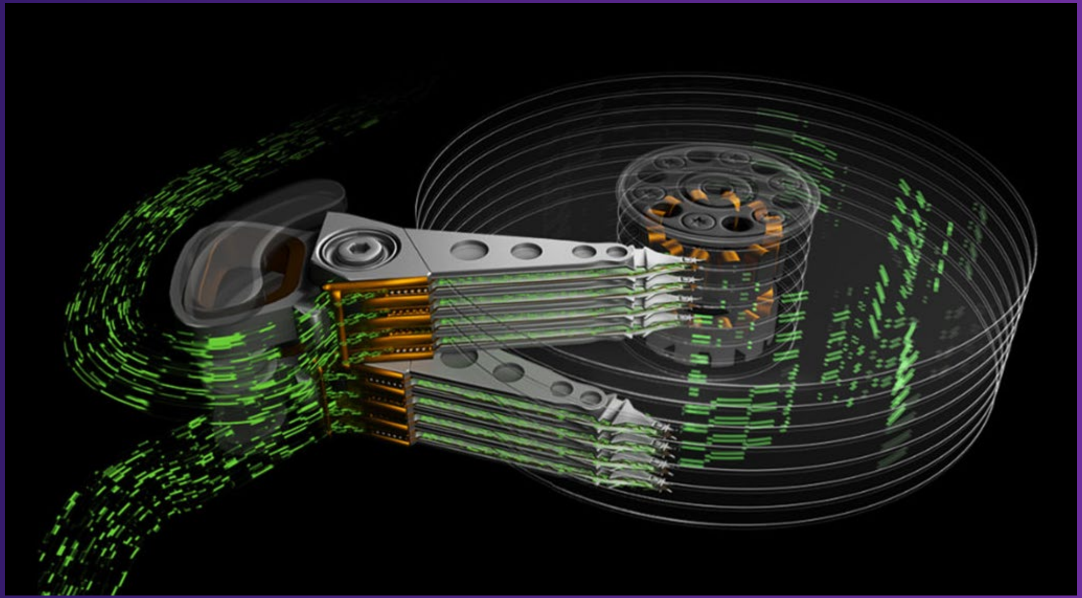
The Challenge: Maintaining Performance Density

To enable continued optimal use of the highest-capacity hard drives, latency-bounded I/O performance must be increased.

This requires an increase in HDD raw servo-mechanical capability (IOPS).

Existing Strategies for improving IO Performance

- System-level caching to improve write performance efficiency
- Queuing (combined with latency management) to improve actuator seek efficiency
- IO Prioritization / Stale Command Timers / Command Duration Limits (CDL) to manage QOS requirements
- Longer transfer lengths
- IO Schedulers / File system optimizations



The Solution: Multi-Actuator Technology

- Multiple LUN SAS, SATA and Single LUN SAS

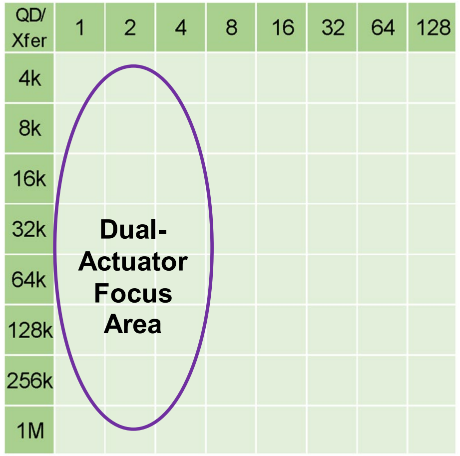
Accelerate your data	Dual actuators drives doubles drive performance using two independent actuators to transfer data concurrently
Maximize drive capacity	Realize utilization and performance gains without compromising latency
Built for applications that need performance	Ideal solution for content delivery networks (CDN), video streaming, software-defined storage, Ceph, Hadoop, virtualization, and more

Performance Goals

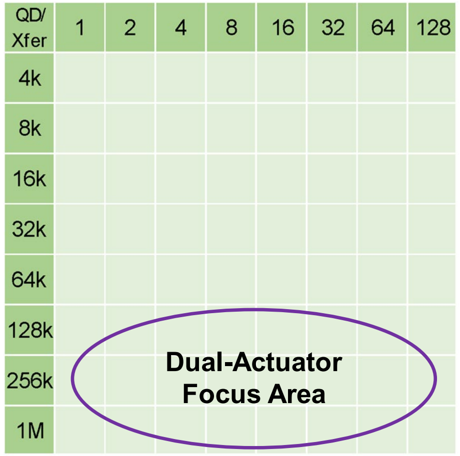
- Enable higher capacity drives in data centers by enabling IOPs/TB improvements

- Implement independent actuators to provide parallelized random access
  - Focus on low queue depths
- Implement independent read and write paths to allow simultaneous parallel transfers
  - Focus on long transfer lengths
- Manage power to minimize impact to existing infrastructure

Random I/O

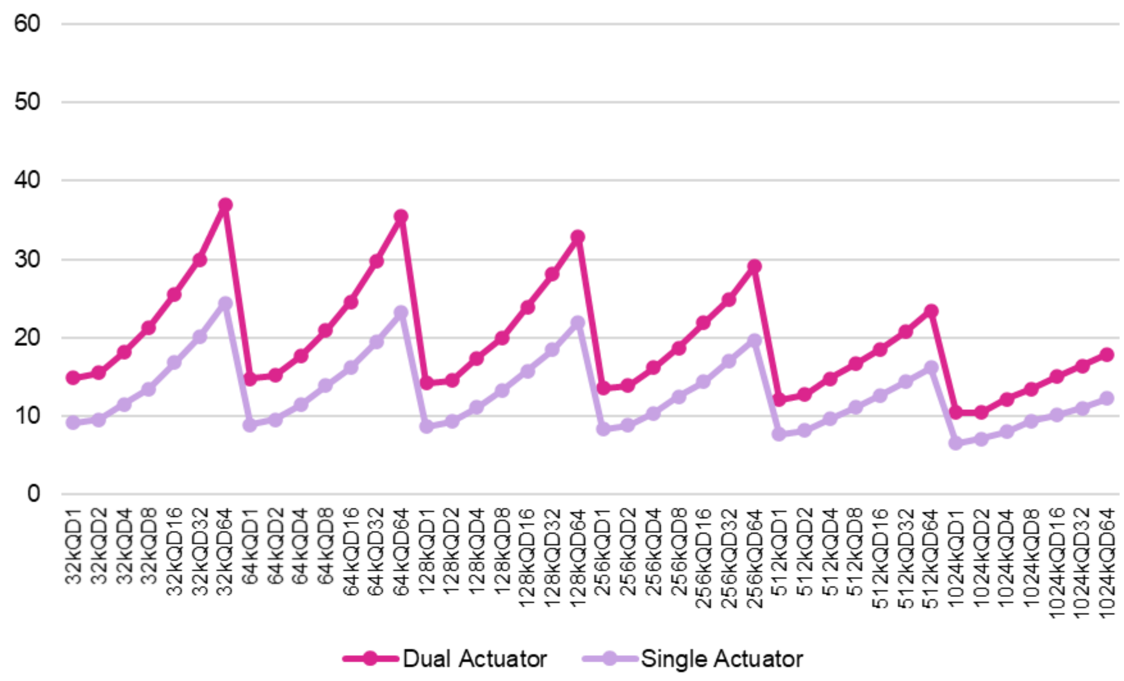


Sequential I/O

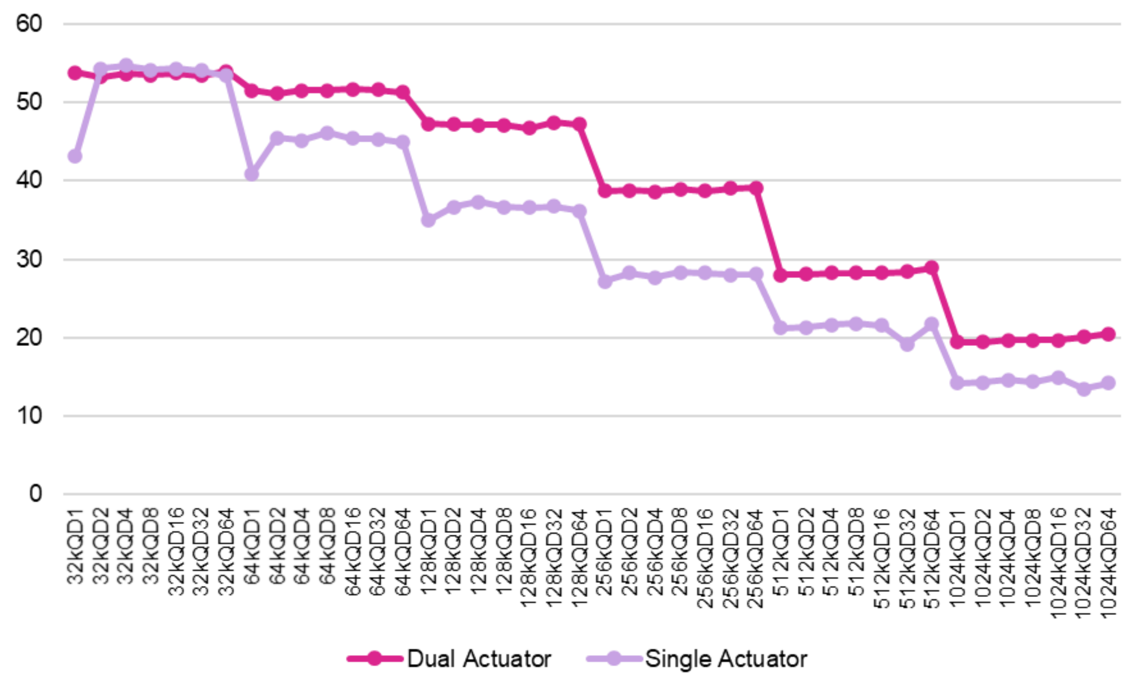


Power Efficiency Favors Dual Actuator

Random Read Performance Efficiency (IOPs/W)



Random Write Performance Efficiency (IOPs/W)



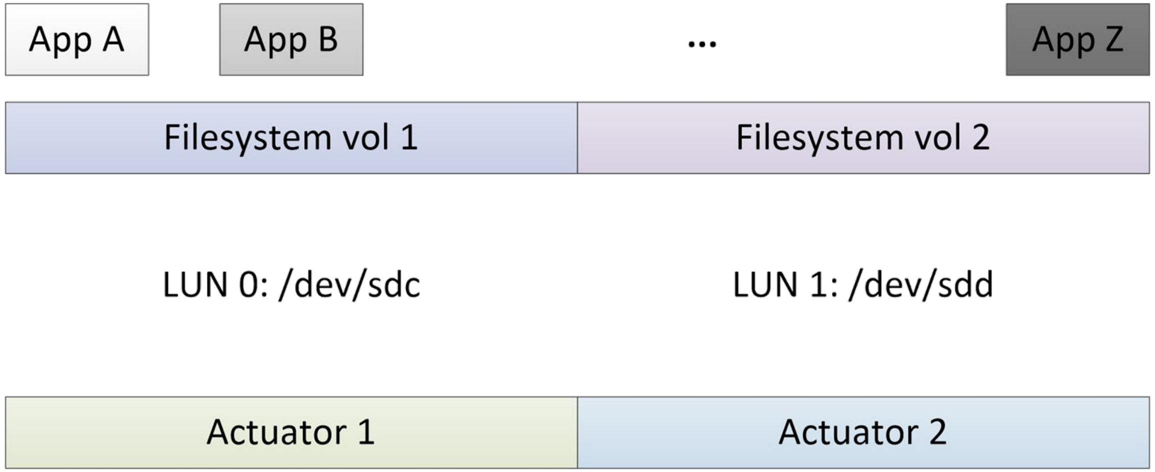
54% higher average IOPs/W in random reads

26% higher average IOPs/W in random writes

SAS Interface

14TB SAS Exos 2x14

- Dual LUN
- 1 Filesystem per actuator



Linux device listing

```
[0:0:0:0]    disk    SEAGATE  ST14000NM0001     K003  /dev/sda    /dev/sg0
[0:0:0:1]    disk    SEAGATE  ST14000NM0001     K003  /dev/sdb    /dev/sg1
[0:0:1:0]    disk    SEAGATE  ST14000NM0001     K003  /dev/sdc    /dev/sg2
[0:0:1:1]    disk    SEAGATE  ST14000NM0001     K003  /dev/sdd    /dev/sg3
[1:0:0:0]    disk    ATA      CT240BX500SSD1    R013  /dev/sde    /dev/sg4
[2:0:0:0]    disk    ATA      SAMSUNG MZ7WD480  NS00  /dev/sdf    /dev/sg5
```

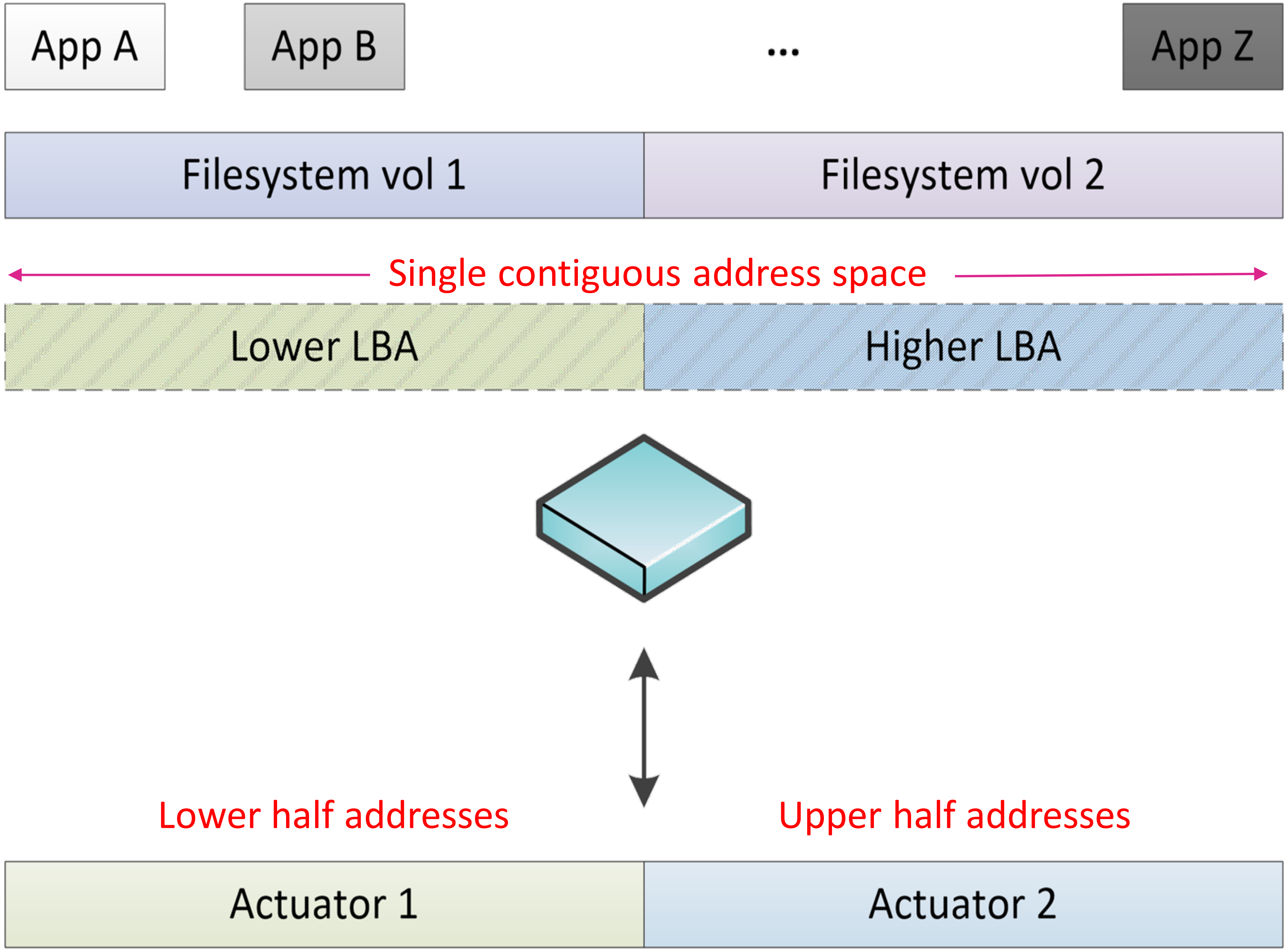

Dual Actuator Implementation Alternatives

- **Treat each actuator like an individual drive**
  - ***Pro***:
    - Best performance potential
    - User controls the movement of data
    - Works well with Dual LUN SAS
  - ***Con***:
    - In order to get the best performance, each LUN/actuator must be kept busy
    - Requires partitioning for Split LBA Space SATA

- **Span both LUNs/actuators into one large volume**
  - ***Pro***:
    - User doesn’t have to manage LUNs/actuators
    - Easy solution for migrating full databases from single actuator to dual actuator
  - ***Con***:
    - Must be aware of the actuator boundaries to be sure that commands are being issued to the whole drive
    - Application layer must ensure IO balance across actuators / LBA space

- **Software RAID 0 - Striping LUNs/actuators together**
  - ***Pro***:
    - User doesn’t have to manage LUNs/actuators (single volume vs two volumes)
    - Commands are automatically divided between actuators
  - ***Con***:
    - Stripe size affects performance – too small of a stripe size and single commands will cross actuator boundaries, causing both actuators to work on the same command. This situation will negatively affect performance and latency.
    - Operating system overhead can affect performance potential

SATA Interface



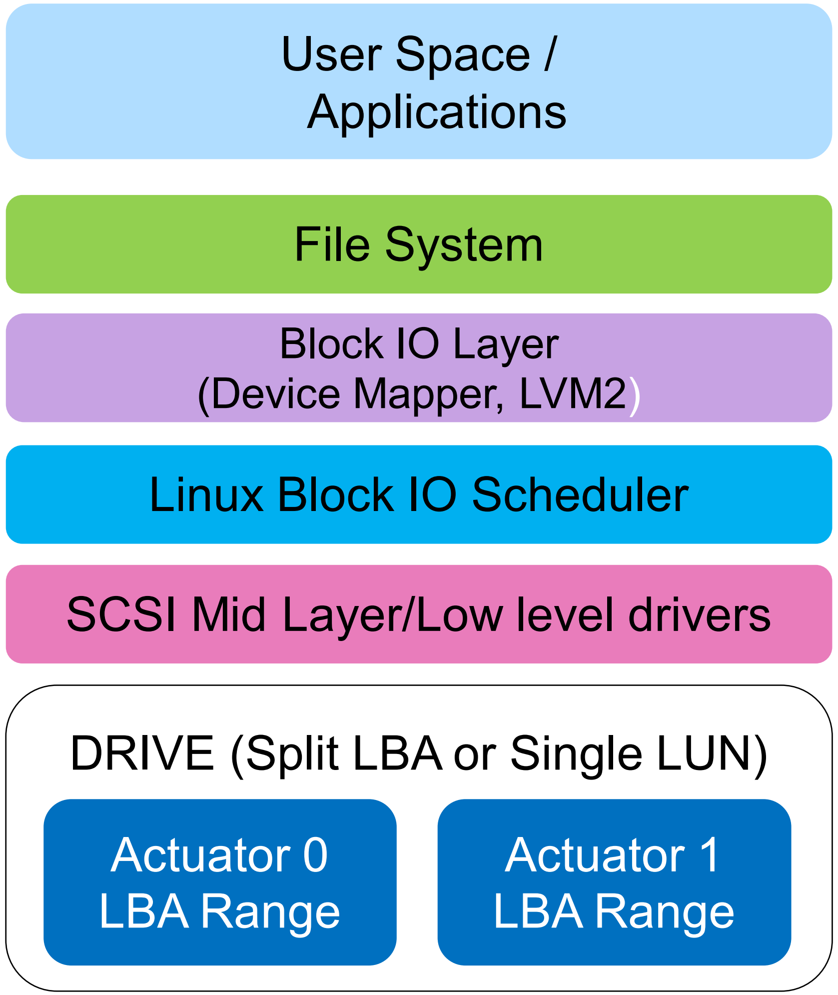
Deploying SATA and Single LUN SAS Drives

The Next Step in Multi-Actuator Drive Adoption

Options for SATA and Single LUN SAS

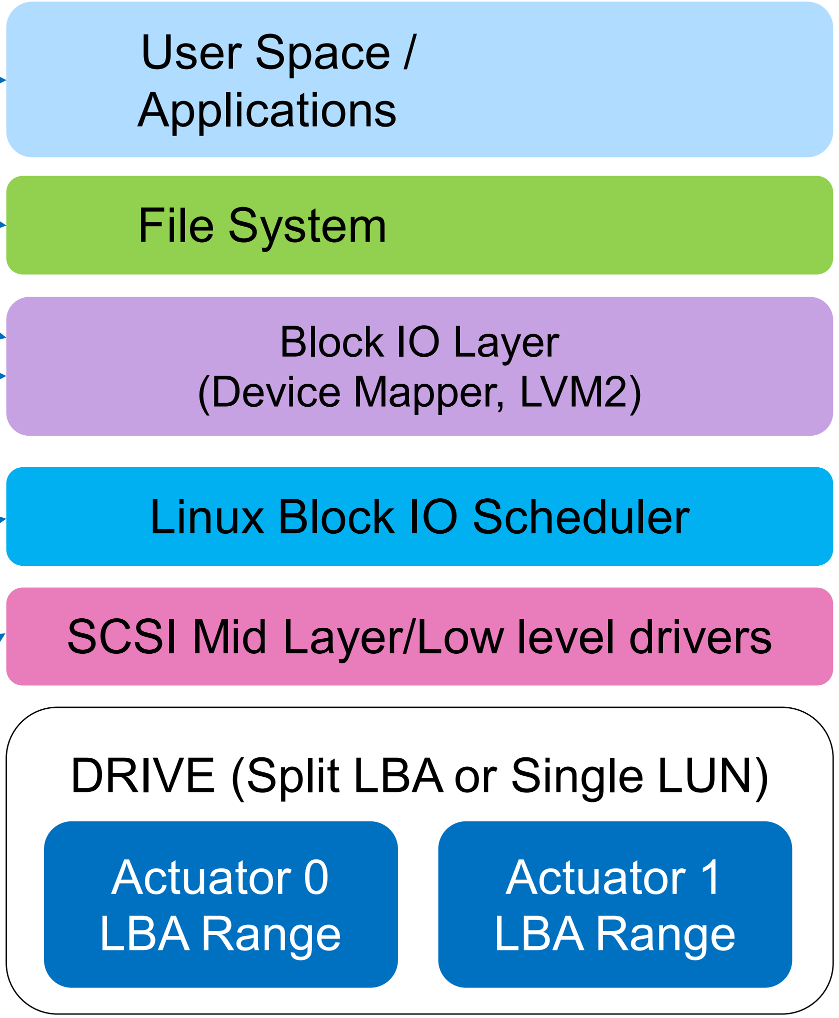
- Multi-Actuator will be a must have solution for improving Data-Center TCO
- HAMR and other hard drive technologies will greatly increase aerial densities a capacity
- Dual LUN SAS Exos 2x14 MACH.2™ Drives are here today, SATA and Single LUN SAS are coming
- SATA drives haves a split LBA space with ranges described in ACS-5 (SATA) Log
- Linux support by WDC to advertise this information to kernel & user space for both SAS and SATA
- Software stack tuning and workload management will achieve optimal performance

Storage Stack



Additional Options: Apps/Partitioning

- Applications can be modified to become multi-actuator aware)
- File System: (kernel dependent, difficult)
- Device mapper target splits underlying block device at the actuator split
- Linux Block Device Partitioning
  - Use GPT to create two independent devices
  - Persistent / Kernel Dependent
- IO Scheduler Optimization
  - Manages commands (using a variety of algorithms) to provide command streams for the ”best” overall IO performance.
- SCSI Subsystem
  - Normally shouldn't redistribute workload
  - Kernel and Legacy complications limit flexibility



What Does an IO Scheduler do, anyway?

- Linux traditionally controls only the maximum number of commands (requests) that are outstanding (in-flight) to the HDD. The ***dispatch queue*** is set to the maximum queue depth appropriate for the underlying hardware: i.e. 32 or 128
- Good IO programming *groups and sorts and prioritizes commands* to maximize the locality and sequential aspects of HDD sector addresses.
- As processes generate IO, either directly or via a file system, we tend to see long runs of sequential or localized IO arriving in multiple streams from the active processes. Without an IO scheduler, these raw IO streams arrive unmanaged to the disk. The scheduler can manage requests (using a variety of algorithms) to provide IO streams to optimize overall IO performance.
- Deadline schedulers optimize for the best read performance while ensuring writes are not excessively delayed. Fair schedulers, such as BFQ (Budgeted Fair Queuing) add the originating process to the algorithm to ensure all processes get a fair portion of the IO bandwidth.

BFQ for Split Actuators

upper
lower

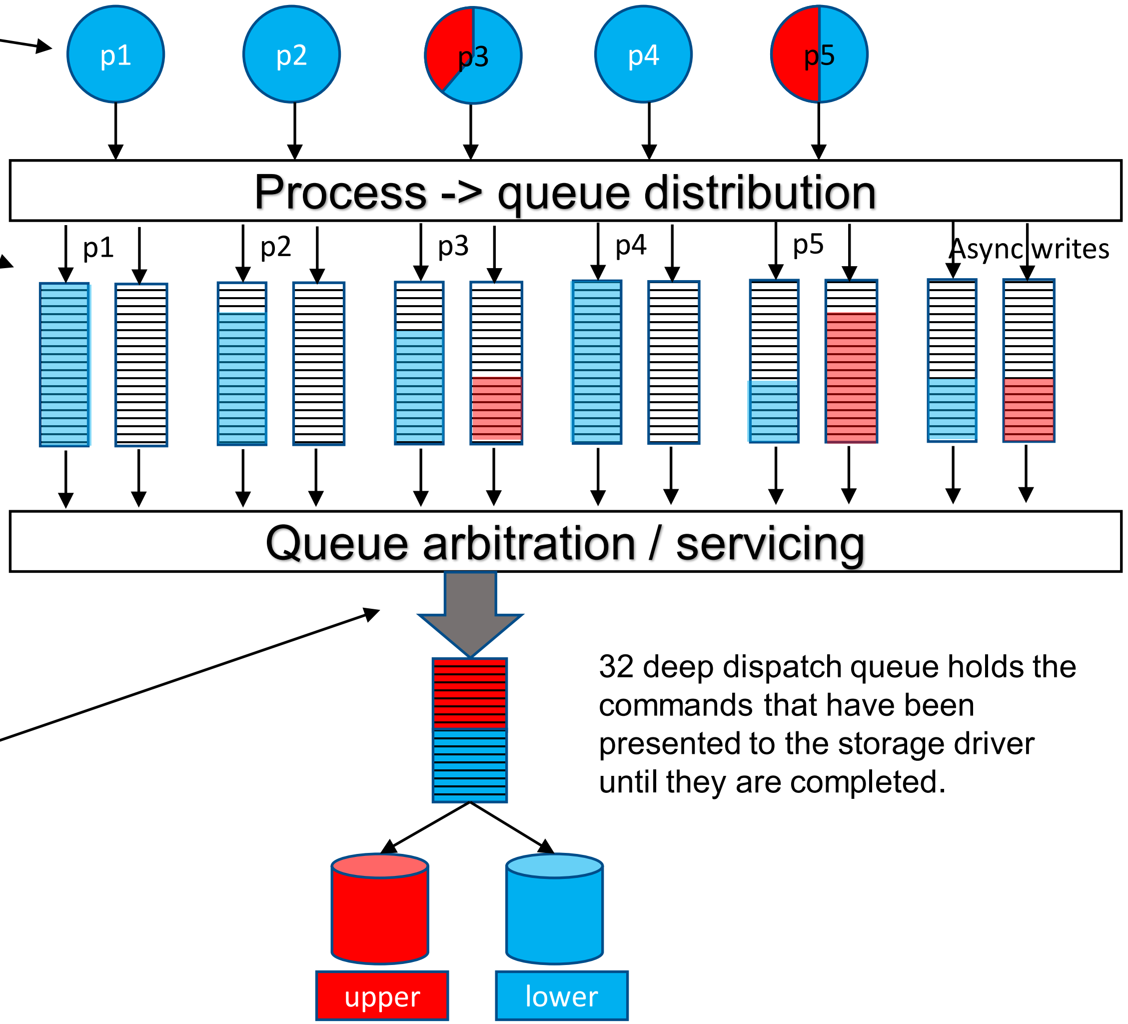
Processes generate IO to upper, lower, or both actuators

BFQ puts each request into an upper or lower queue per process

BFQ maintains TWO queues per process; one each for upper and lower actuators.

BFQ assigns a time slice to each QUEUE “fairly” and routes the requests to the dispatch queue. This is potentially fair and balanced unless the distribution of upper and lower is inherently imbalanced.

INJECTION is a feed-forward method that pulls commands from a different (non-fair) queue when needed. In our case, we INJECT commands from an actuator’s queue if that actuator is unloaded, regardless of fairness. We steal some of the fair IO to a heavily loaded actuator to ensure the other actuator maintains a minimum load threshold.



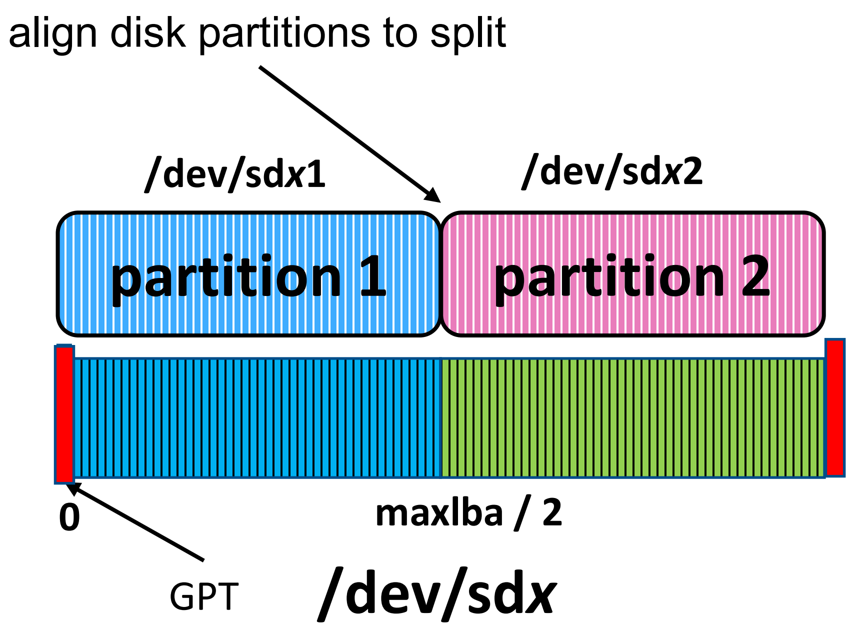
32 deep dispatch queue holds the commands that have been presented to the storage driver until they are completed.

STORAGE DEVELOPER CONFERENCE SDC 21

Linux Block Device Partitioning

Use Case: separate a split-address-space device into two independent devices

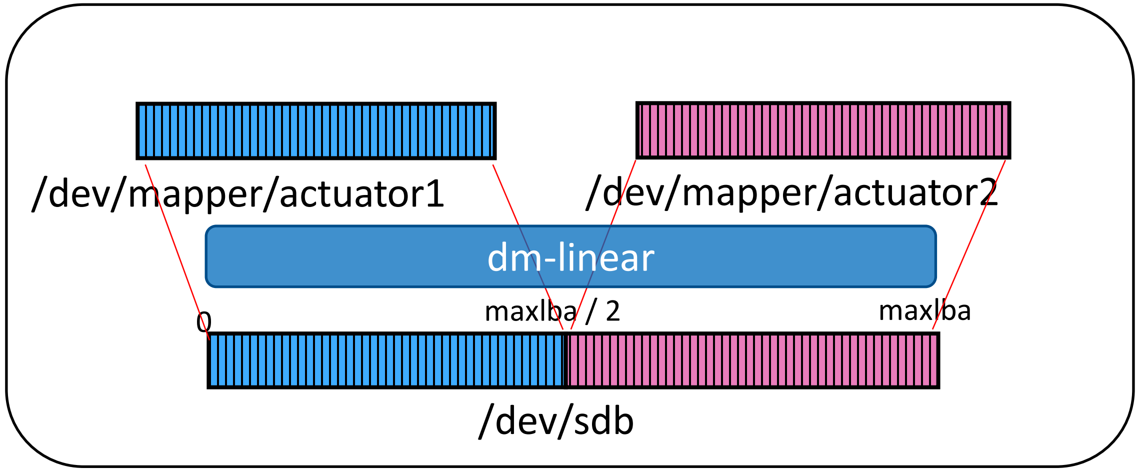
- Partitioning the device at the actuator boundary creates two subordinate block devices managed by the kernel
- Persistent: Kernel recreates the devices by default when it reads the partition table at device initialization
  - The backup partition table will be on the secondary actuator




```
#!/bin/sh
echo create two partitions on the supplied disk, dividing in half
dev=$1
device_size=`blockdev --getsize /dev/${dev}`
act1_sect0=$((device_size/2))
echo device size 512b blks: $device_size
echo act 1 first sector: $act1_sect0
echo use the following command to partition /dev/$dev
echo sudo parted /dev/$dev -s mklabel gpt mkpart act0 0% $(( $act1_sect0-1 ))s mkpart act1 ${act1_sect0}s 100%
```

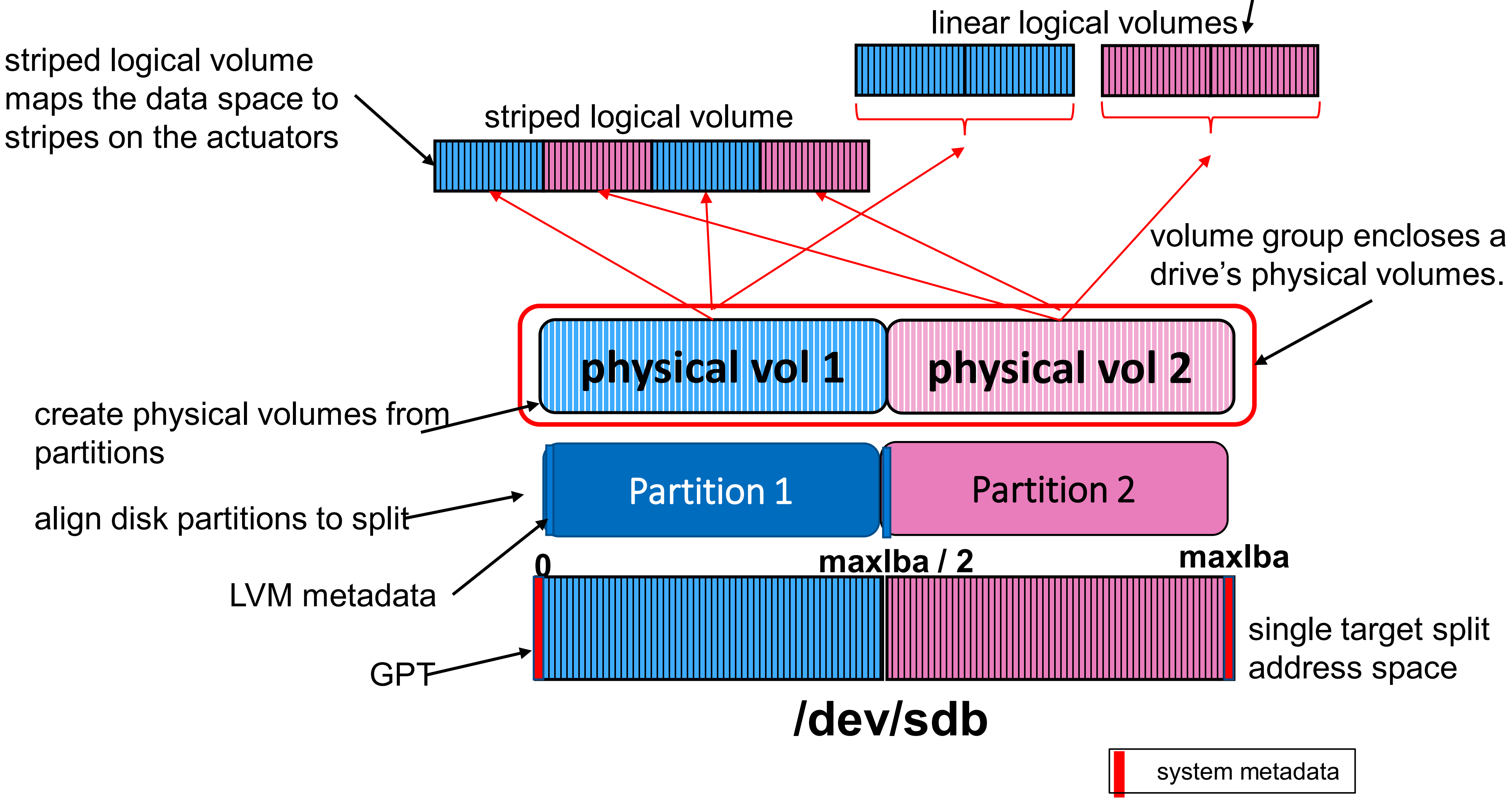

Device Mapper – Separate the Actuators

- The linear device mapper target can split the underlying block device at the actuator split point
- The bare device mapper does not store any metadata anywhere
- Not persistent: The setup must be re-done by udev or a startup script – it's not automatic, but provides the underpinning for LVM or other volume management approaches.
- Queue depth is still managed over the entire device: no controls to prevent one mapper endpoint from dominating the queue and causing actuator starvation.



LVM2 – Partitioned Disk

LVM2 is persistent across reboots.



Performance Guidance

Unleashing Multi-Actuator Drive Potential

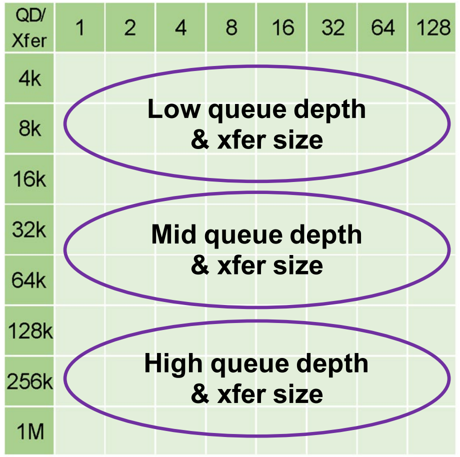
Sequential Reads

Low & mid queue depths & transfer sizes

- Sequential low & mid queue depths are not a primary design goal
- No hardware streaming – commands are overhead-dominated
- Expect slightly lower performance on LUN1 vs. LUN0 due to additional overhead
- Device performance will fall between LUN0-only and LUN1-only performance on any individual drive

High queue depths & xfer sizes

- 2x performance of single-actuator drive
- Will hit native disk data rate on each LUN
  - LUN0 & LUN1 usually show slightly divergent data rates due to BPI/TPI optimization
- Ideally: Device performance should be LUN0+LUN1



STORAGE DEVELOPER CONFERENCE SDC 21

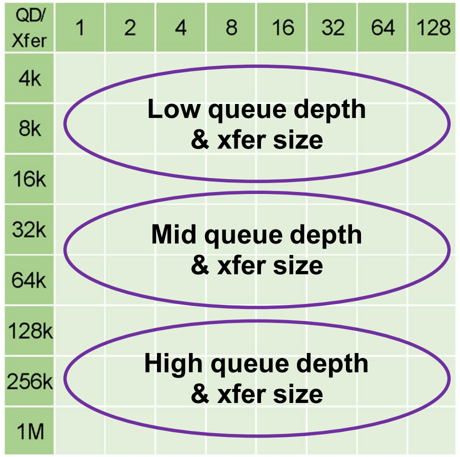
Sequential Writes

Low queue depths & short transfer sizes

- Performance dominated by slipped revs and system overheads.

High queue depths & xfer sizes

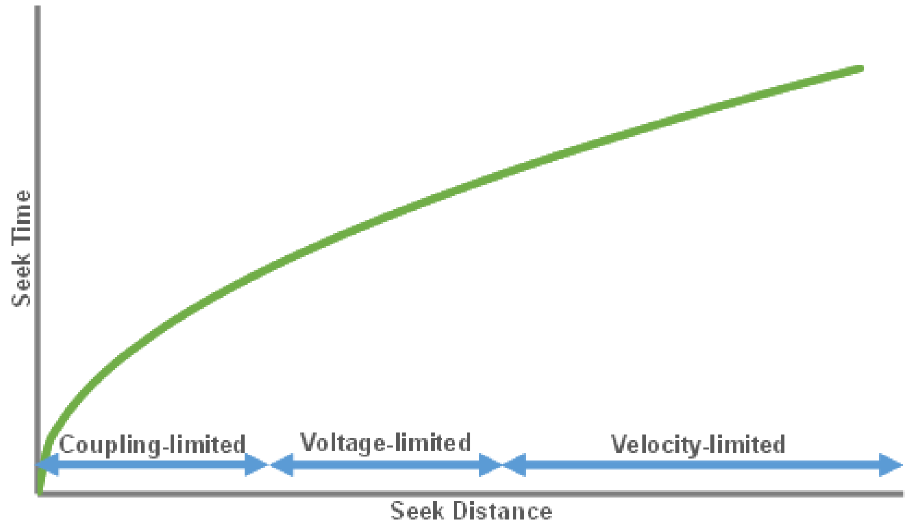
- 2x performance of single-actuator drive
- Will hit native disk data rate
  - LUN0 and LUN1 will have slightly different sequential data rates due to BPI/TPI optimization
- Ideally, device performance should be LUN0+LUN1



Random Performance

Harrier is optimized for shallow queue depth workloads

- Seek performance on mid and long seeks (shallow queue) is equivalent to other nearline drives
  - Bounded by available voltage and maximum design velocity (Kt/JR)
  - Performance scales very well with dual actuators
- Seek performance on short seeks (deep queue) dominated by settle times (like other nearline drives)
  - Settle times include mechanical coupling effects
    - **Short seeks tuned somewhat less aggressively to manage coupling**
  - Performance gains reduced at deep queues



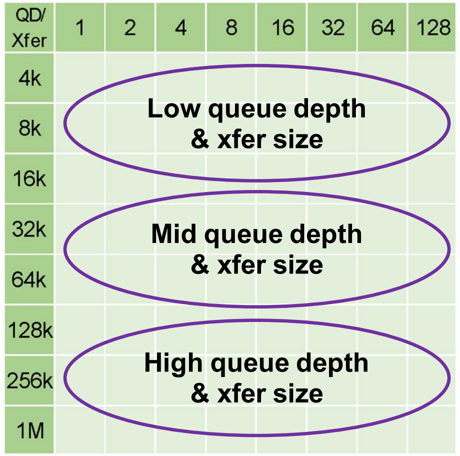
Random IO

Low queue depths

- Harrier optimized for low-queue random workloads
- ~95% performance (per LUN) of a single-actuator drive
- Device-level performance should be 1.9x single-actuator drive

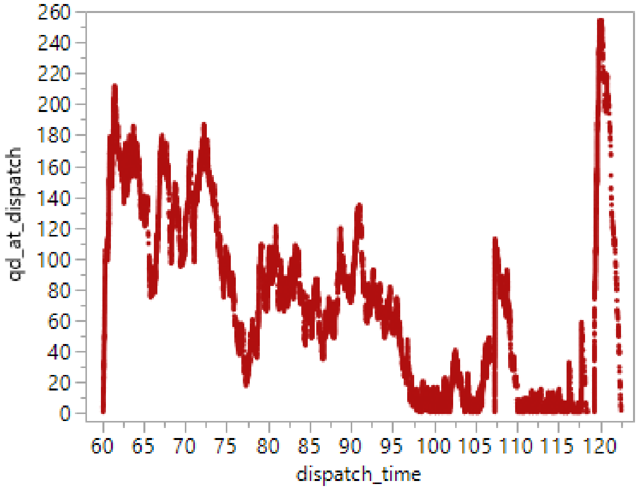
High queue depths

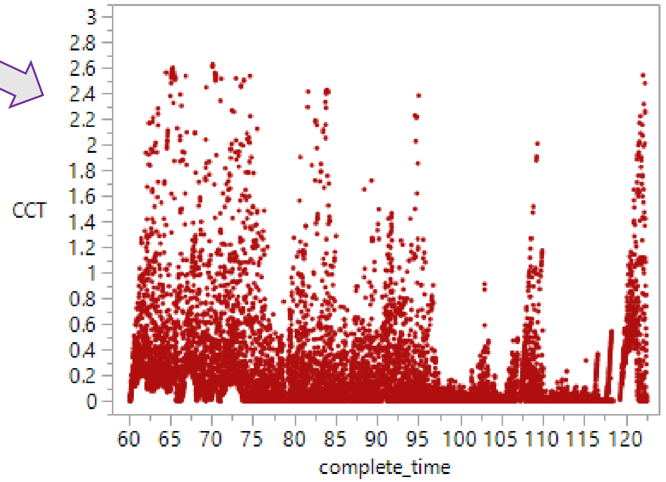
- Performance influenced by actuator coupling
- Longer transfer lengths will have performance closer to 1.8x
  - Longer transfers are more sensitive to data rate than short transfers



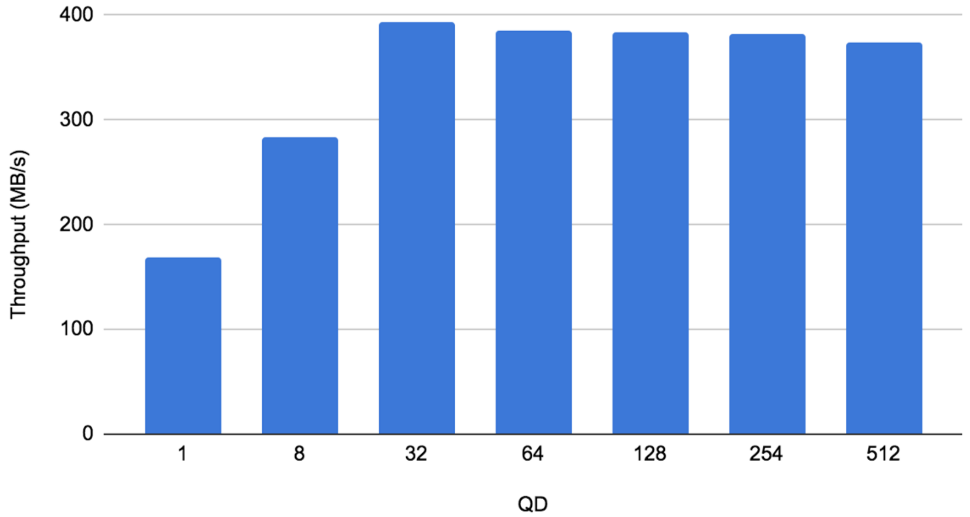
Lessons Learned: Queue Depth

Deeper queues at device level naturally tends to increase command completion times as requests spend more time in the queue.





Optimal System Performance is achieved through careful settings on system parameters, including queue depth at device level



Lessons learned

1. Maintain work balance between actuators
2. Set queue depth
3. Reduce metadata overhead
4. Reduce IO dependency between actuators (atomicity)
5. BTRFS RAID 0 can work well
6. Use 12Gb/s SAS

Please take a moment to rate this session.

Your feedback is important to us.

