KALRAY
THE POWER OF MORE

# TRANSPARENT QLC USING DPU TECHNOLOGY

Tim Lieber
Lead Solution Architect

Remy Gauguey
Lead Software Architect

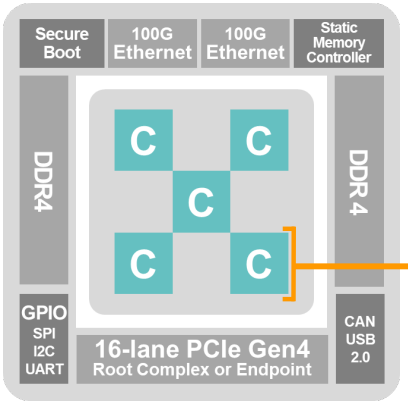
Flash Memory Summit 2023

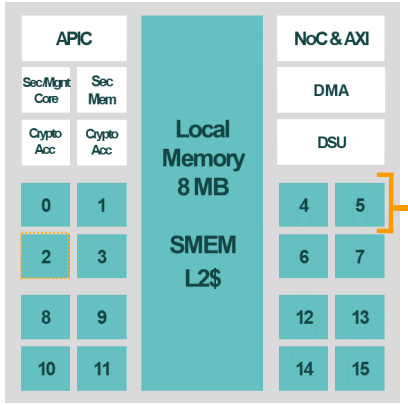
www.kalrayinc.com

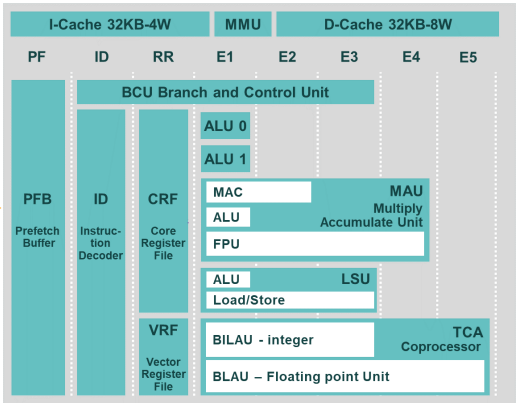
# MPPA® DPU ARCHITECTURE

## The I/O Processor for Next Gen Intelligent Systems

PATENTED




### MULTI CLUSTER ARCHITECTURE

**5 Clusters: 80 cores + 80 co-processors**

- Load Balancer / Packet Parser
- 2x100Gbps Ethernet
- PCIe Gen4
- DDR4 - 3200

**AXI Bus + NoC Bus**

- L2 refill in DDR and direct access to DDR from clusters
- DMA-based highly efficient data connection

### CLUSTER

**Architecture**

- 16 cores
- 1 safety/security dedicated core
- 1,2 GHz

**Memory**

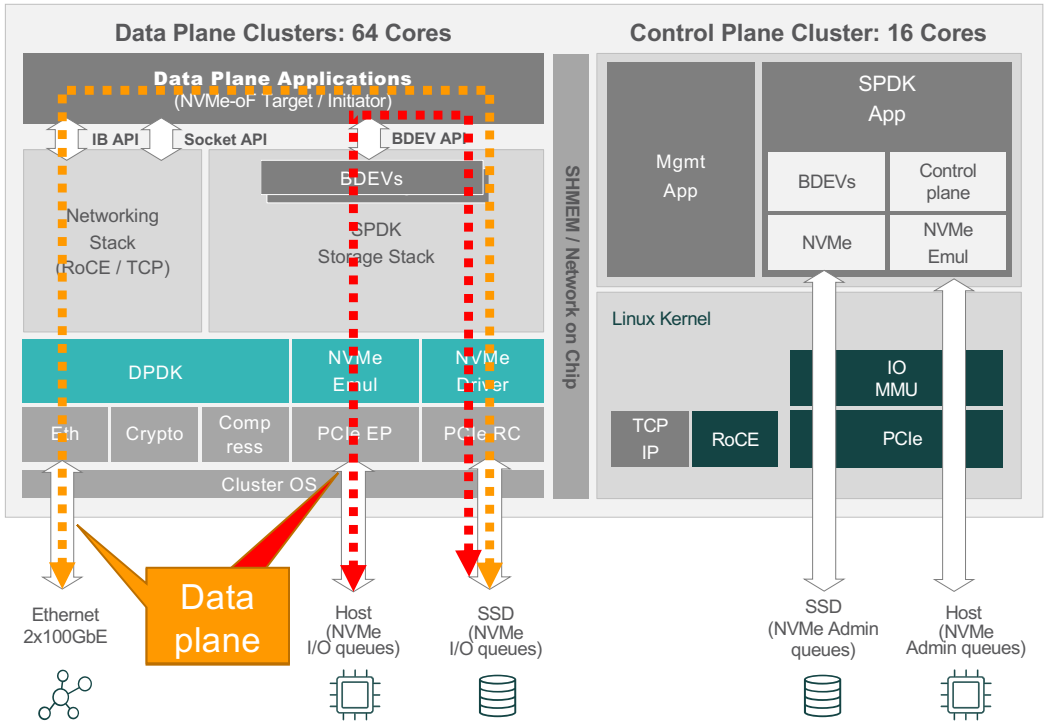
- L1 cache coherency (configurable)
- 8MB configurable memory (L2 cache)
- 256 bits / bandwidth up to 614GB/s

### 3RD GENERATION KALRAY CORE

- VLIW 64-bit core
- 6-issue VLIW architecture
- MMU + I&D cache (32KB+32KB)
- 16-bit/32-bit/64-bit IEEE 754-2008 FPU
- Vision/CNN Co-processor (TCA)

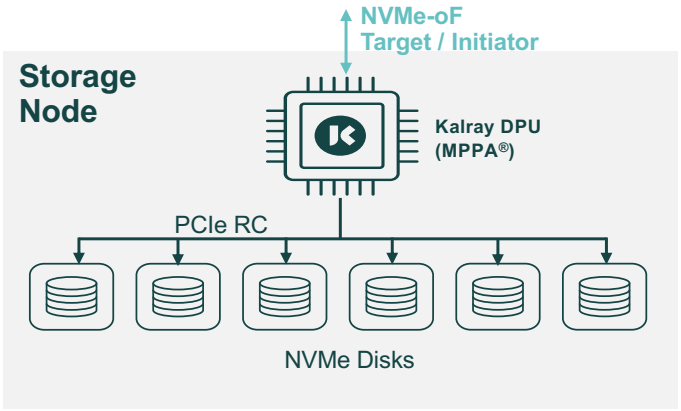
# ACCESSCORE STORAGE: SW ARCHITECTURE

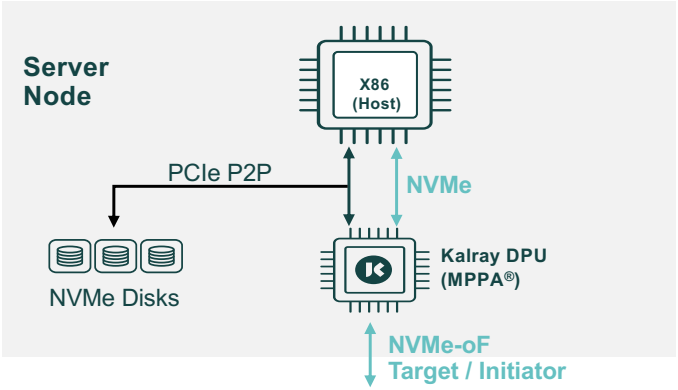
SPDK Has Never Been So Fast …




- Modular storage framework leveraging SPDK
- Driver framework leveraging DPDK
- Rich set of SPDK vBDEVs available, that can be composed to implement complex storage services
- Support for NVMe emulation, NVMe-oF Target and Initiator
- Easy porting from SPDK X86

# DPU USE CASE CONFIGURATIONS




## Use-Case #1: Storage Node in a storage cluster

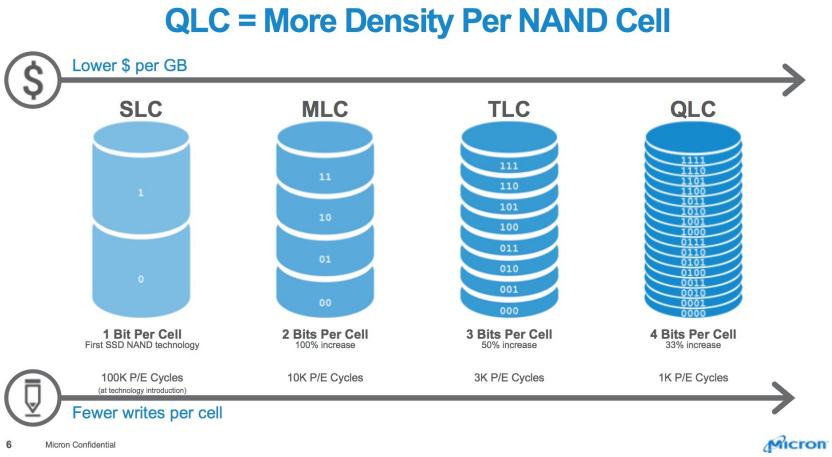
- Reduced cost from server based appliance
- High performance Backend Storage
- Distributed Erasure Coding
- Data reduction
- Encryption
- QLC Write-shaping

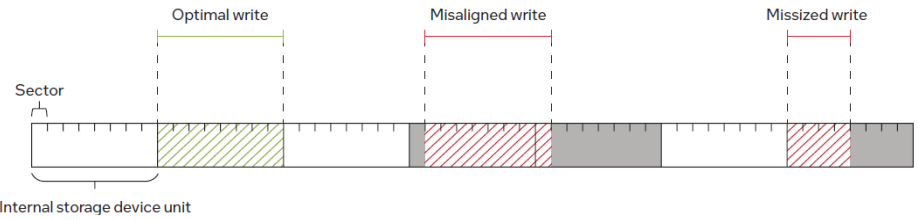
## Use-Case #2: Storage Service Offload from Host to DPU

- Declustered RAID / Erasure Coding
- Local/Remote device management
- Data reduction
- Encryption
- QLC Write-shaping

# THE REALITY OF QLC NAND DEVICES

## Characteristics and limitations




**+**

- QLC (4 bits per cell) and PLC (5 bits per cell) NAND reduce $/GB SSD closer to HDD and enable large capacity SSD (30..60+ TB)

**-**

- NAND SSDs' write **performance** per TB decreases with storage density going up
- Available Program/Erase cycles decline with higher density NAND. This reduces the **endurance** lifetime (Drives Writes per Day)
- Large capacity SSDs need **larger internal storage unit** (i.e. indirection unit, or IU) to reduce the DRAM BOM cost (ex on 64TB SSD, **64K IU** for a 4GB DRAM)
- Executing 4K random writes results in 16x write amplification (**WAF**) due to read-modify-write penalty. Writes have to be IU aligned

# QLC, NOT MEANT FOR ALL WORKLOADS

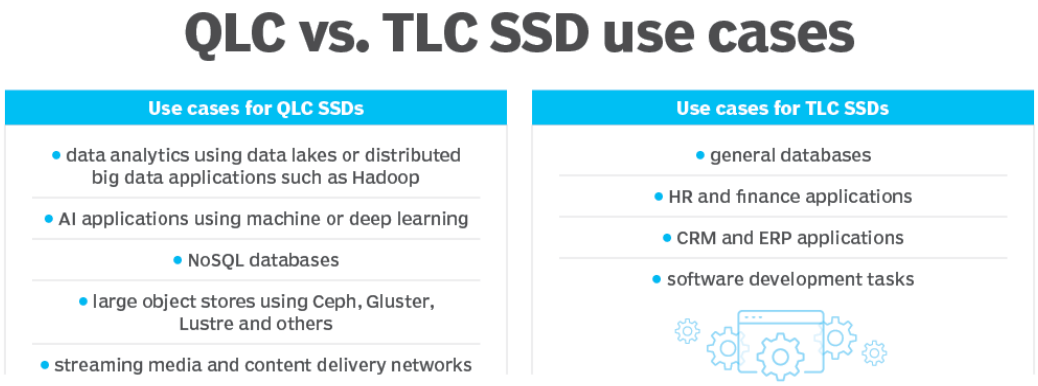
Best fit and techniques to overcome limitations

\+

- QLC SSDs best fit for Read Intensive workload

\-

- Not good fit for Writes Intensive/Mixed workloads
- Require large IU alignment to avoid WAF which affects endurance
- Not compatible with majority of FS with 4KB block size

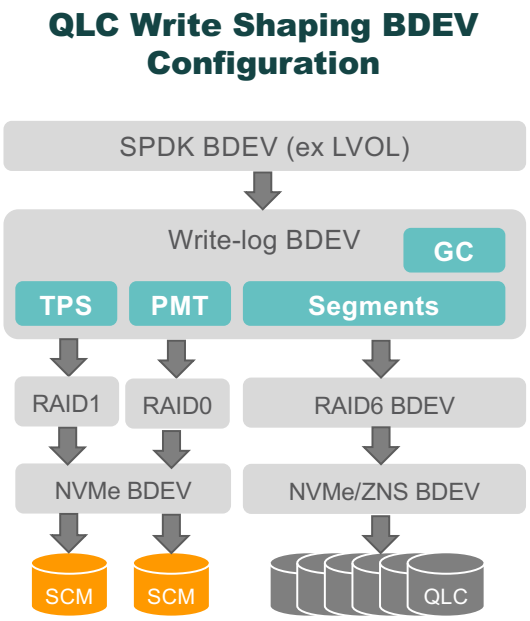



*Source : techtarget.com*

Write Intensive workloads require technique known as « **Write Shaping** »

# SEQUENTIALIZED DATA FOR QLC

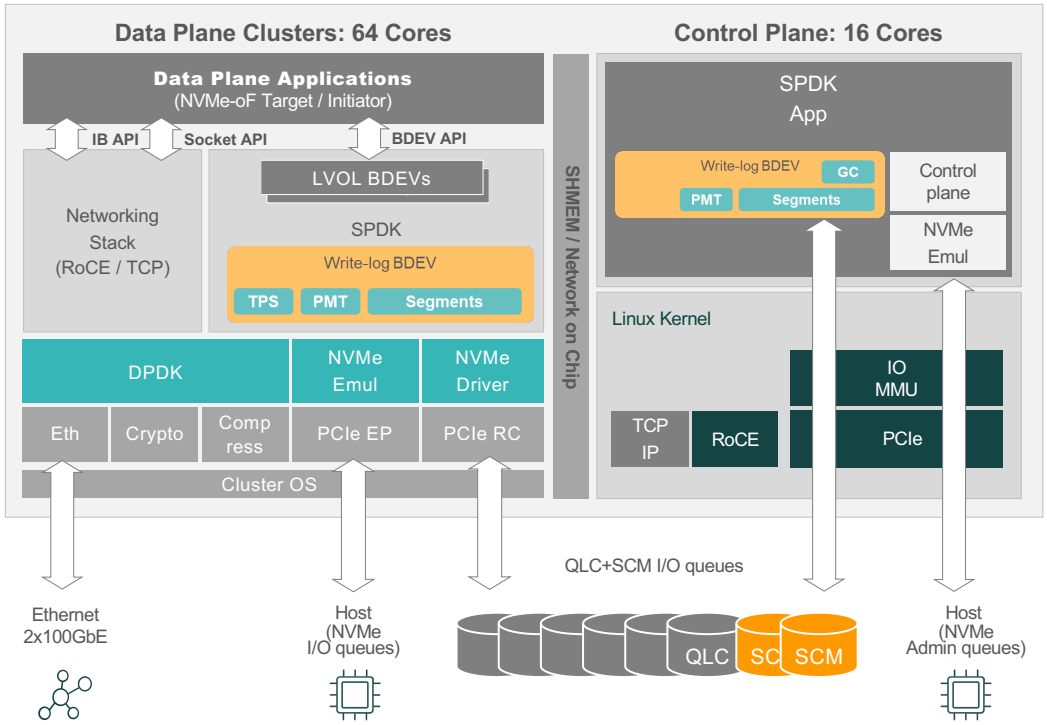
SPDK « write-log » block device




- New host FTL and data layout
- "**Sequentialize**" random writes data in '**segments**' (1MB) temporary saved in fast tier (SCM NVMe or NVRAM) aka **TPS** (Temporary Persistent Storage)
- **Commit Full Segments** as optimal IU aligned and sequential writes to QLC SSDs on '**bands**' granularity (several contiguous segments)
- Maintain a Page Mapping Table (**PMT**) for LBA to PBA translation (4KB granularity). Huge table on SCM with cache in DDR memory
- Background **Garbage Collector (GC)** to clean/move segments with obsolete data
- Best fit for :
  - Full stripe RAID6 writes
  - RAID6 write hole prevention
  - ZNS 'append' semantic
  - QLC erase block

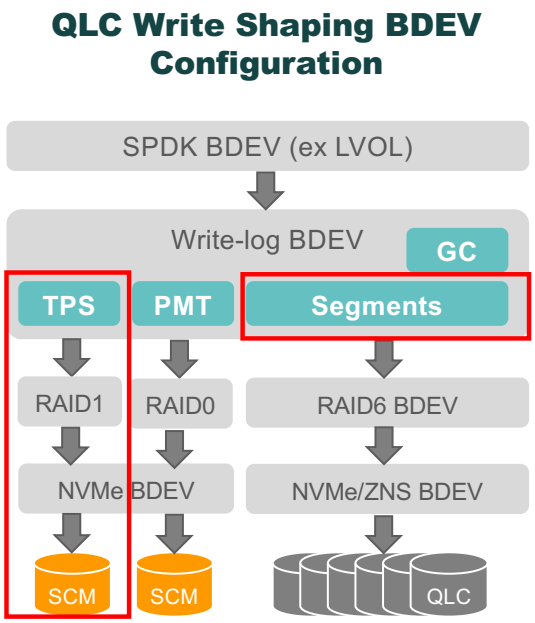
# SPATIAL SEPARATION FOR PERFORMANCE

A log-structured write shaping with spatial separation




- Data plane cores (aka ‘Front End’)
  - **Data aggregation**
  - **T**emporary **P**ersistent **S**torage
  - **P**age **M**apping **T**able
  - **P**ower **F**ail **S**afety
- Control plane cores (aka ‘Back End’)
  - **G**arbage **C**ollector with minimal interference with data plane
  - **P**age **M**apping **T**able
- Interface to host
  - PCIe NVMe emulation volume
  - NVMe-oF volume

# DATA AGGREGATION PROCESS




**On every data plane core and for every write :**

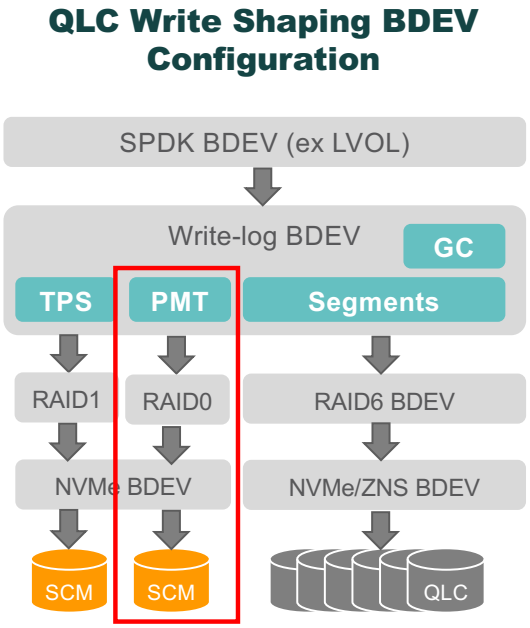
- Aggregate random write I/Os in RAM segments (ex 1MB segment)
- Maintain metadata on 1st 4KB in segments (GC, PMT recovery)
- Save I/Os in redundant Temporay Persistent Storage (**TPS**) cache : reduced latency
- TPS bypass if segment full
- Update PMT (LBA => RAM pointer)

**Upon full segment :**

- Compute RAID6 parities
- Commit to QLC drives : append only on current ‘band’ (1 band = 1 or more ‘segments’), with PFS
- Update PMT (LBA => PBA on RAID6 BDEV)

# PAGE MAPPING TABLE

SCM table write-back cache

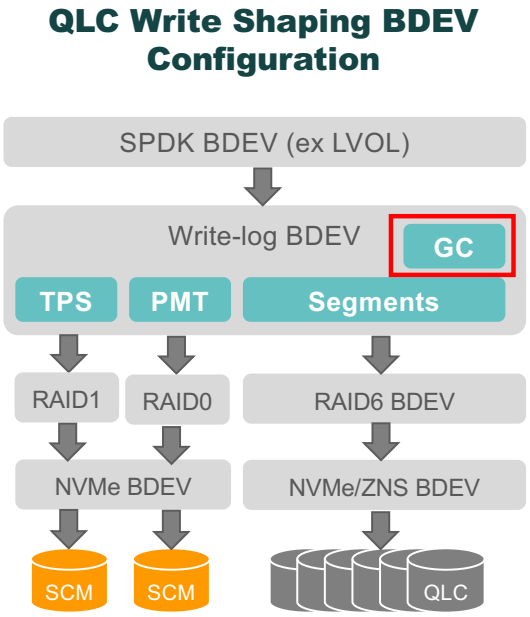



**Page Mapping Table (PMT)**

- Logical to Physical translation @4KB granularity
- 64bits entry per 4KB => With 8+2 QLC of 30 TB : 240TB usable capacity / 480GB PMT size
- Stored on SCM (RAID0) plus 8 ways associative write back cache in memory
- Reconstructed from segments metadata in case of dirty shut-down
- Snapshoting (version number) to speed-up reconstruction
- Updated by data plane and garbage collector

# GARBAGE COLLECTOR

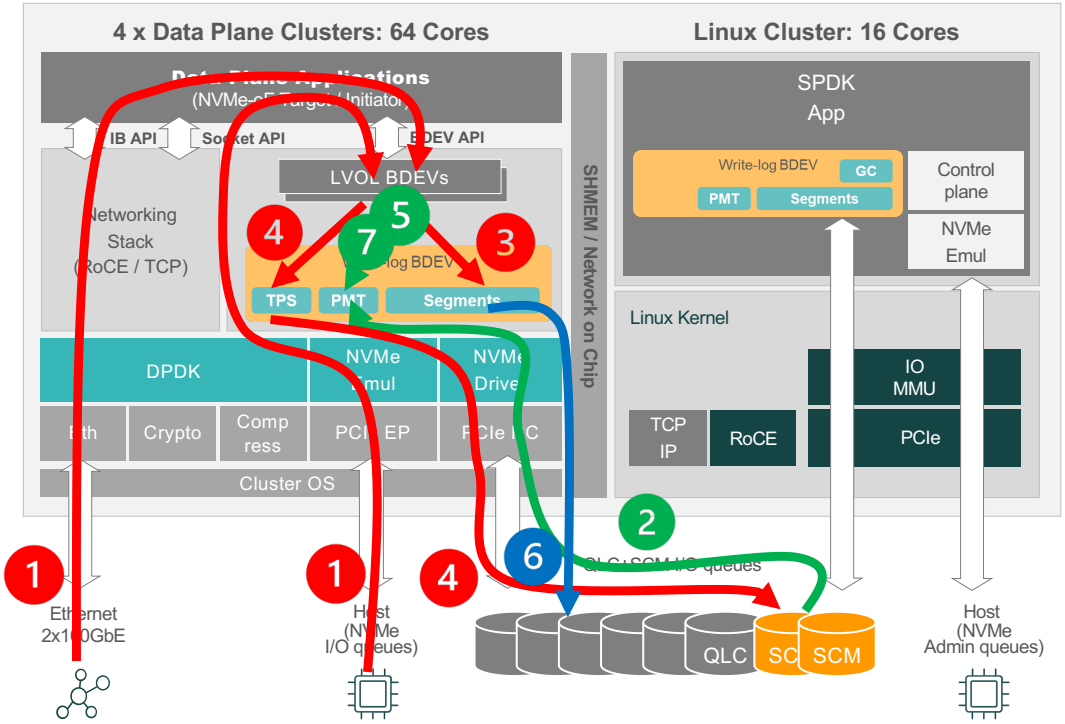
## Spatial Separation Between GC and User I/O




### Garbage Collector (GC)

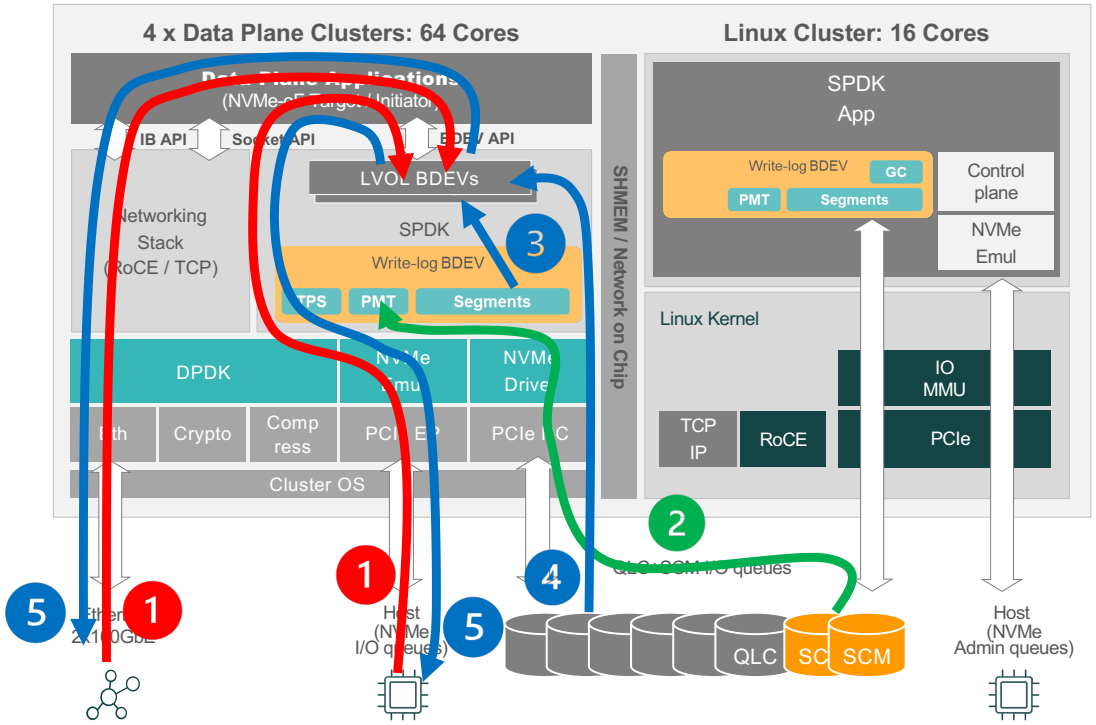
- Band based garbage collector (band >= QLC erase zone ~100MB)
- Executed asynchronously on ‘Control Plane’ cluster to minimize interference with user I/Os
- Identify « obsolete » blocks through version number
- Move valid blocks if any to new segment / band
- Update PMT accordingly

# WRITE DATA FLOW




1. Write command received from PCIe or NVMe-oF
2. PMT cache refill if cache miss (4K read from SCM) – can cause eviction and write back
3. Append write IO to segment in memory
4. Write to TPS if IO doesn't allow to complete segment else bypass TPS
5. Update PMT cache entry to memory segment
6. If segment full : compute RAID6 parities and commit full segment to QLCs
7. Update PMT cache entry to QLC location

# READ DATA FLOW




1. Read command received from PCIe or NVMe-oF
2. PMT cache refill if cache miss (4K read from SCM) – can cause eviction and write back

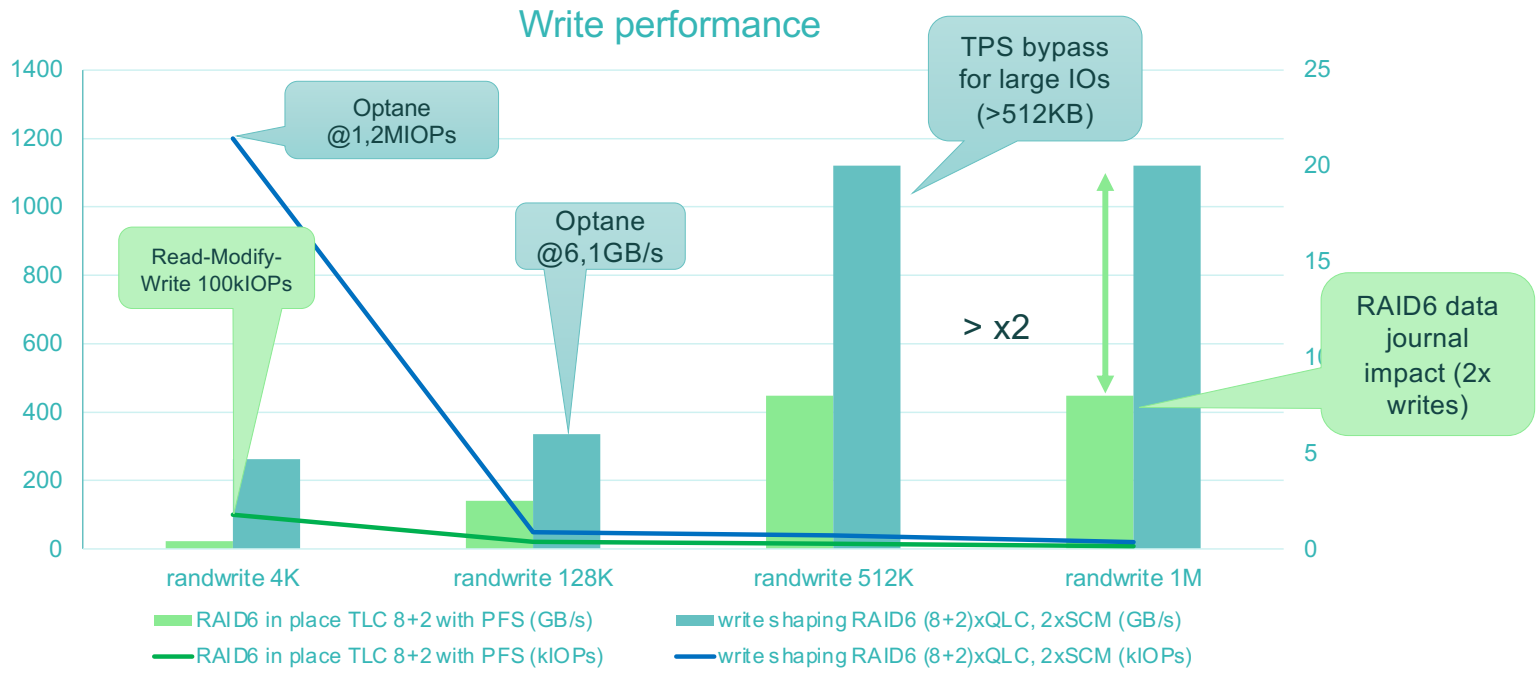
2 possibilities :

3. data in current accumulating segment : read from memory
4. data is in a commited segment : read from QLC RAID array

5. Send data to host (PCIe / RDMA / TCP)

# RAID PERFORMANCE WITH WRITE SHAPING

RAID6 in place vs write shaping comparison




Design is Future-planned to add SCM-CXL byte addressable drives.

# CONCLUSIONS, PREDICTIONS, OBSERVATIONS

1. QLC/PCL TECHNOLOGY ADOPTION IS VITAL FOR COST REDUCTION IN THE DATA CENTER.
2. MOST DATA CENTER SOFTWARE WILL NEED TO CHANGE TO ADAPT TO QLC/PLC
3. DPUS CAN OFFLOAD TECHNOLOGY ANOMALIES TO SMOOTH PATH FOR ADOPTION
4. THE RESULT IS LOWER COST AND HIGHER PERFORMANCE STORAGE


THANK YOU
KALRAY
THE POWER OF MORE
www.kalrayinc.com

# DISCLAIMER

Kalray makes no guarantee about the accuracy of the information contained in this document. It is intended for information purposes only and shall not be incorporated into any contract. It is not a commitment to deliver any material, code or functionality, and should not be relied upon in making purchasing decisions. The development, release and timing of any features or functionality described for Kalray products remains at the sole discretion of Kalray.



KALRAY

THE POWER OF MORE

www.kalrayinc.com