SNIA DEVELOPER CONFERENCE

SDC 24

BY Developers FOR Developers

September 16-18, 2024
Santa Clara, CA

Data Redundancy and Scrubbing with RAID Offload

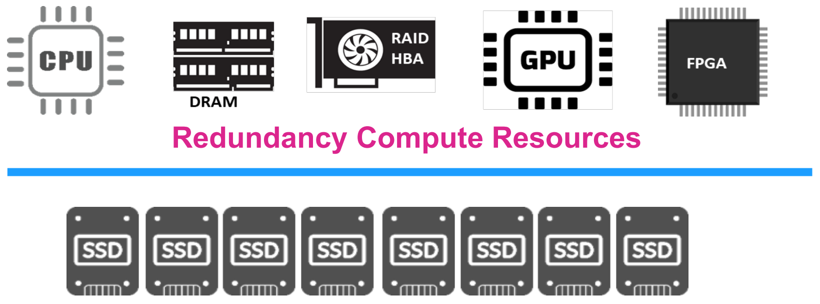
Mahinder Saluja, Director SSD Technology, KIOXIA

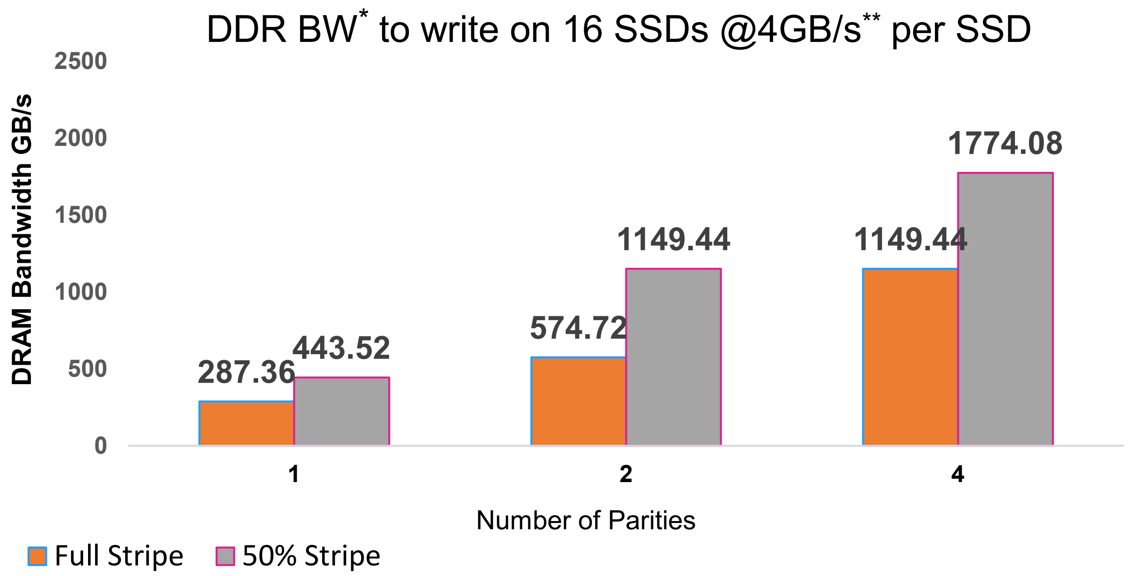
Agenda

- Data Redundancy Challenges and RAID Offload proposal
- RAID Offload Adoption Environment
- Applications of RAID Offload
- Summary
- Call to Action

Data Redundancy Challenges

- Redundancy applications are critical for aggregating storage and protecting the data
- NVMe™ SSD performance improvement is continuously shifting the bottlenecks to redundancy applications
- RAID complexity is increasing with additional redundancy
- Challenges being addressed by:
  - Assigning dedicated CPU cores and DRAM bandwidth
    - Memory bandwidth is limited
  - Hardware accelerators
    - Requires additional power - costly and not scalable
  - Mirroring the data
    - Doubling the storage cost





- BW = Bandwidth ** GB/s = gigabytes per second.

NVMe is a registered or unregistered mark of NVM Express, Inc. in the United States and other countries. All images, graphs, and/or graphics within this presentation are the property of KIOXIA America, Inc. (KIOXIA) and are reproduced with the permission of KIOXIA.

Source: KIOXIA in-house testing and calculation

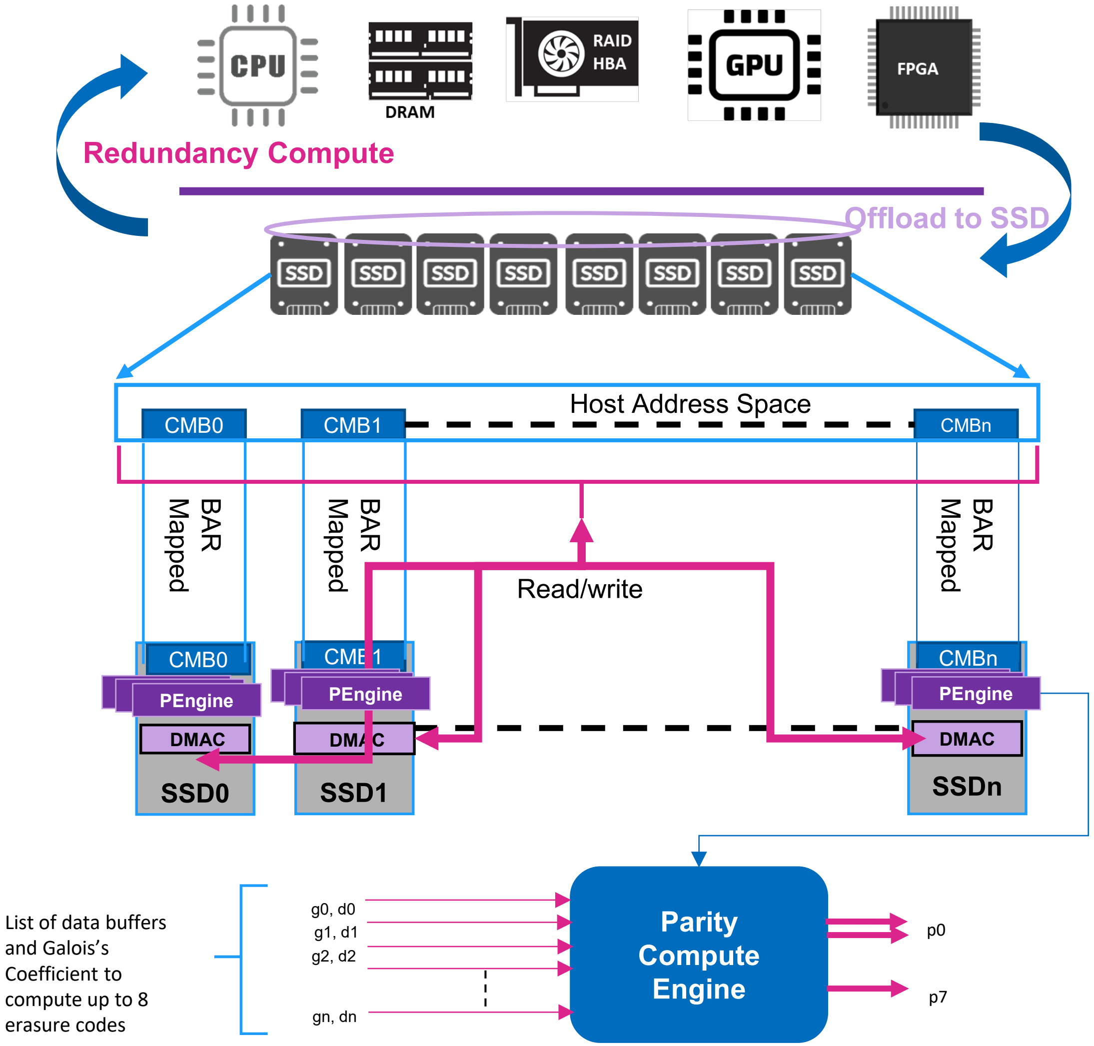
RAID Offload Proposal

Goals:

- Addresses the challenges of increased complexity
- Cost effective, scalable and standards based
- Existing redundancy applications should require minimum change

KIOXIA Proposes:

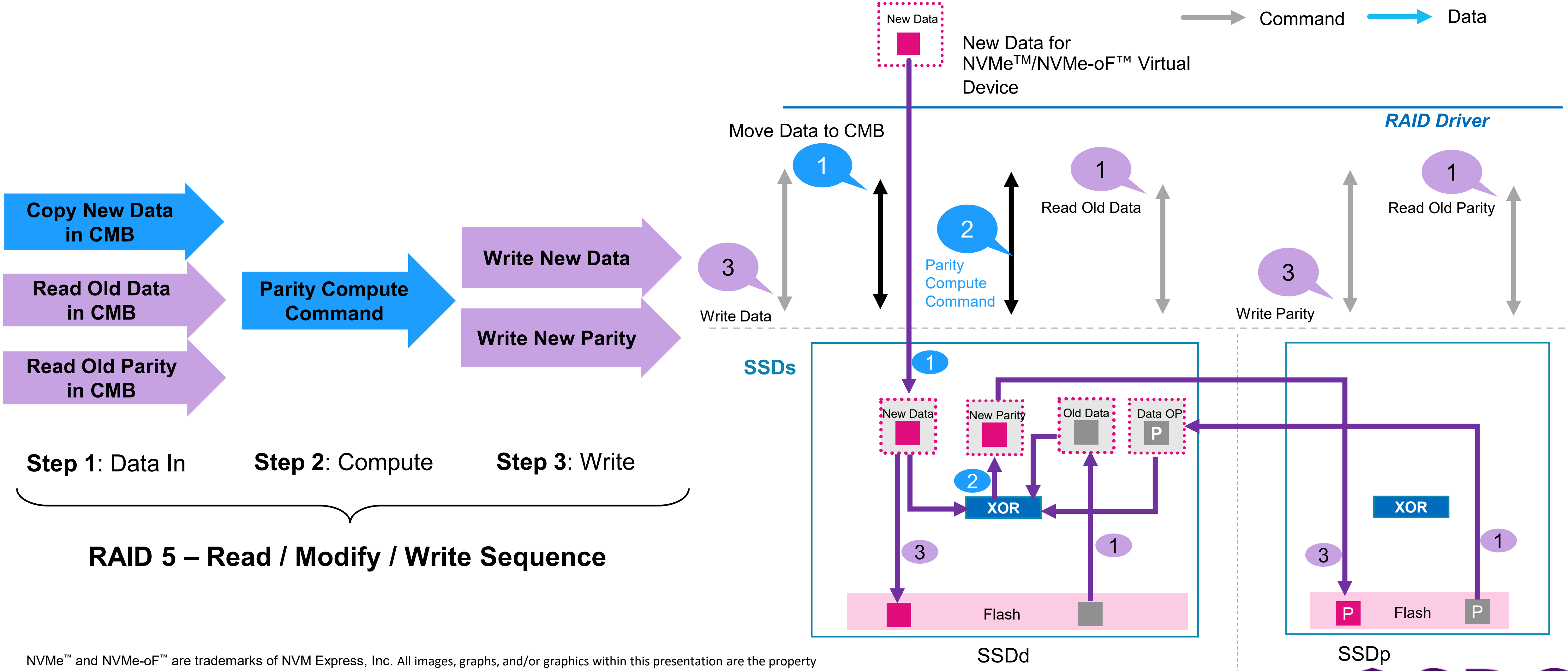
- Host orchestrated parity/EC(Erasure Coding) compute and memory bandwidth offload to SSD
  - Leverage NVMe[TM] subsystem attached memory such as CMB for memory bandwidth offload
  - Parallel parity compute function on SSD
  - DMA (direct memory access) engine for buffer to buffer copy for mapped addresses
  - Commands to control the functions
  - RAID applications use commands to offload and continue to handle faults



RAID Offload Proof of Concept (PoC)

- Implemented **PoC** with 2 commands that identifies memory buffers using familiar NVMe™ PGP/SGL descriptions
  - **Buffer Copy Command**
    - Copies buffers from Host buffer to NVMe™subsystem using the NVMe™ subsystem's DMA engine
  - **Parity Compute Command**
    - The command allows a number of parity blocks to be computed
    - Activates parity engine to compute parity by input parameters
    - The command identifies data buffers for each parity computation and corresponding weights for each data buffer in that computation
    - Each parity output is sent to an identified destination buffer

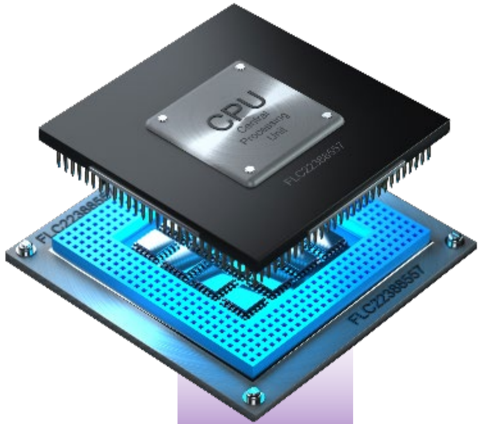
Command and Data Flow : RAID Partial Stripe Write



NVMe™ and NVMe-oF™ are trademarks of NVM Express, Inc. All images, graphs, and/or graphics within this presentation are the property of KIOXIA America, Inc. (KIOXIA) and are reproduced with the permission of KIOXIA.

Proof of Concept (PoC) : mdraid5 and KIOXIA CM7 Series SSD

Subject to Change Without Notice



1. Reduce CPU workload for RAID computation
2. Reduce DRAM bandwidth Utilization
3. Improve host CPU utilization; contribute to energy efficiency for PCIe® Gen 5 server

Leverage High Performance of PCIe® 5.0 SoCs

RAID 5

Parity Calculation Offload

RAID Offload : PoC Results (with KIOXIA CM7 & mdraid5)

System	KIOXIA CM7 Gen4 x4 – mdRAID 5#	RAID Offload	% Benefit
Number of SSDs	5	5	
Full Stripe Write 512 kibibytes (KiB)			
CPU Utilization	42%	37%	12% Saving
DRAM Bandwidth in mebibytes (MiB/s)	3450	340	91% Saving

workload: Flexible I/O tester (FIO) 512K Random Write @ 950 megabytes per second (MB/s)

System DELL® PowerEdge™ R650xs Xeon™ Gold 6338N 2.2GHz(2 Socket, 32 Cores) PCIe Gen4 , SSDs : 5xCM7 Gen4 (1.92TB)

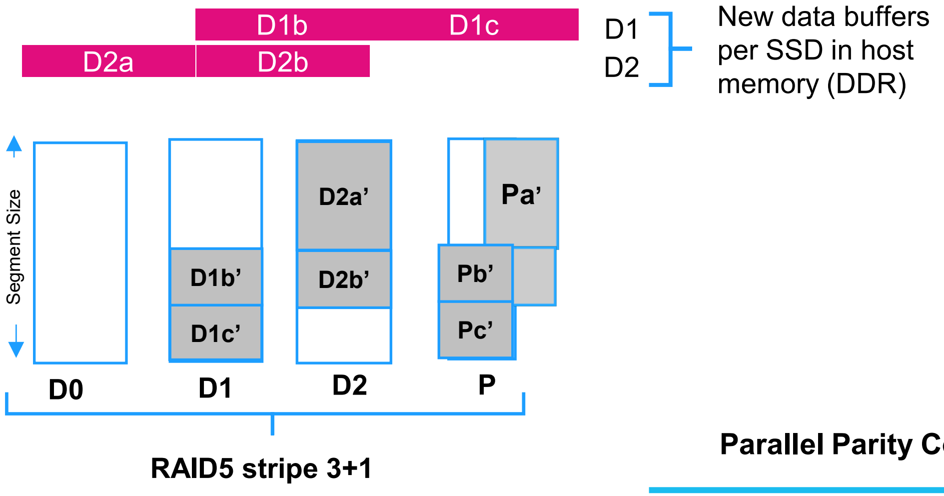
The KIOXIA product images shown are a representation of the design model and not an accurate product depiction.

Parallel Parity Engine Use Cases

Parallel XOR* : Multi-drive Partial Stripe Write Request

Command
Data
New Write Data
P New Parity
Old Data
P Old Parity

Conventional RAID



Traditional way of handling this case is as follows:

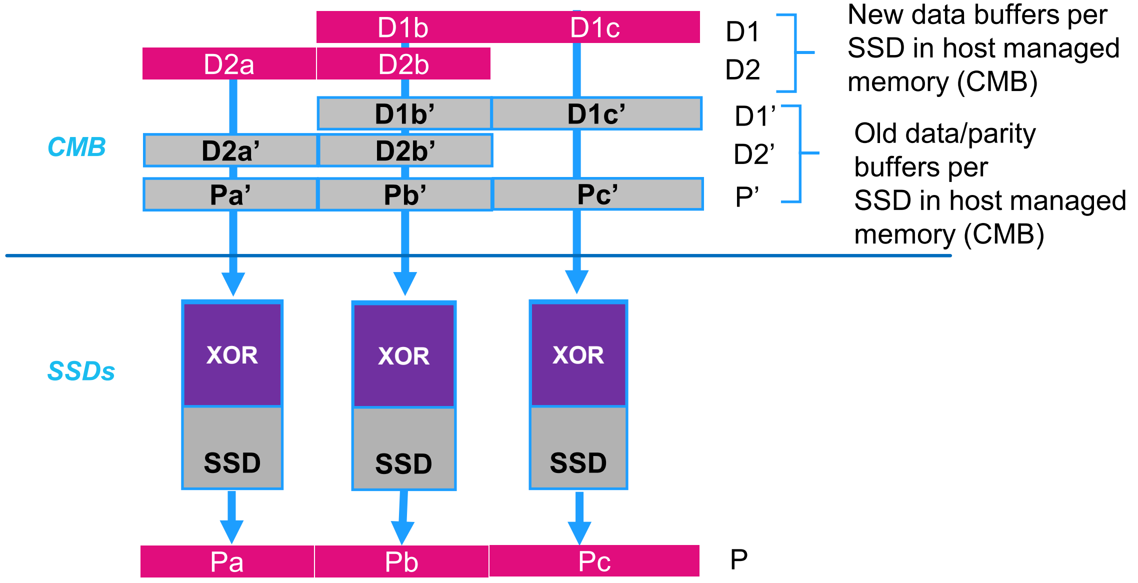
1. Read old data **D1b'c'** from D1 and parity **Pb'c'** from P
2. Calculate new parity **Pbc = D1bC⊕D1b'c⊕'Pb'c'**
3. Write new data to D1 and new parity to P
4. Read old data **D2a'b'** from D2 and parity **Pa'b** from P
5. Calculate new parity **Pab = D2ab⊕D2a'b'⊕Pa'b**
6. Write new data to D2 and new parity to P

Parallel Parity Computation

Eliminates serialization and improves performance

Parallel XOR to avoid steps 4, 5 and 6

RAID Offload



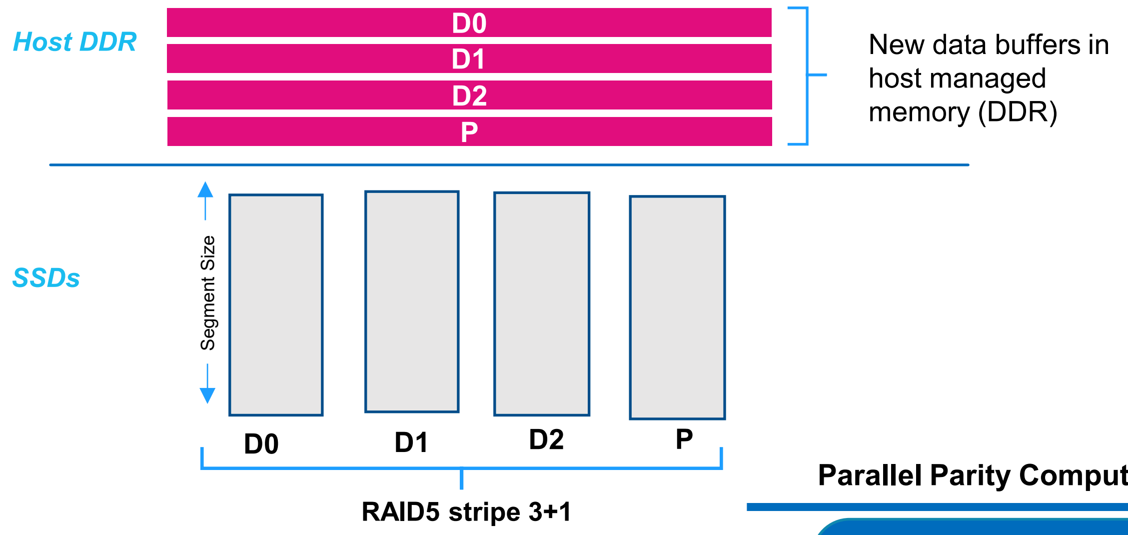
1. Host perform two steps in parallel
   a. Host transfers new data in CMB
   b. Host issue NVM read commands to read old data/parity in CMB (parallel 3 read)
2. Host issue 1 command to compute **3 parities**
   Pa= D2a ⊕D2a'⊕Pa | Pb= D1b⊕D2b⊕D1b'⊕D2b'⊕Pb' | Pc=D1c ⊕ D1c' ⊕ Pc'
3. Host issue NVM write command to update new data/parity (3 parallel write)

* XOR = Exclusive OR

SDC 24

Parallel XOR : Multi-drive Stripe Write Request

Conventional RAID Method



1. Host calculate new parity serially

D0⊕D1 = Pi

D2⊕Pi = Pi

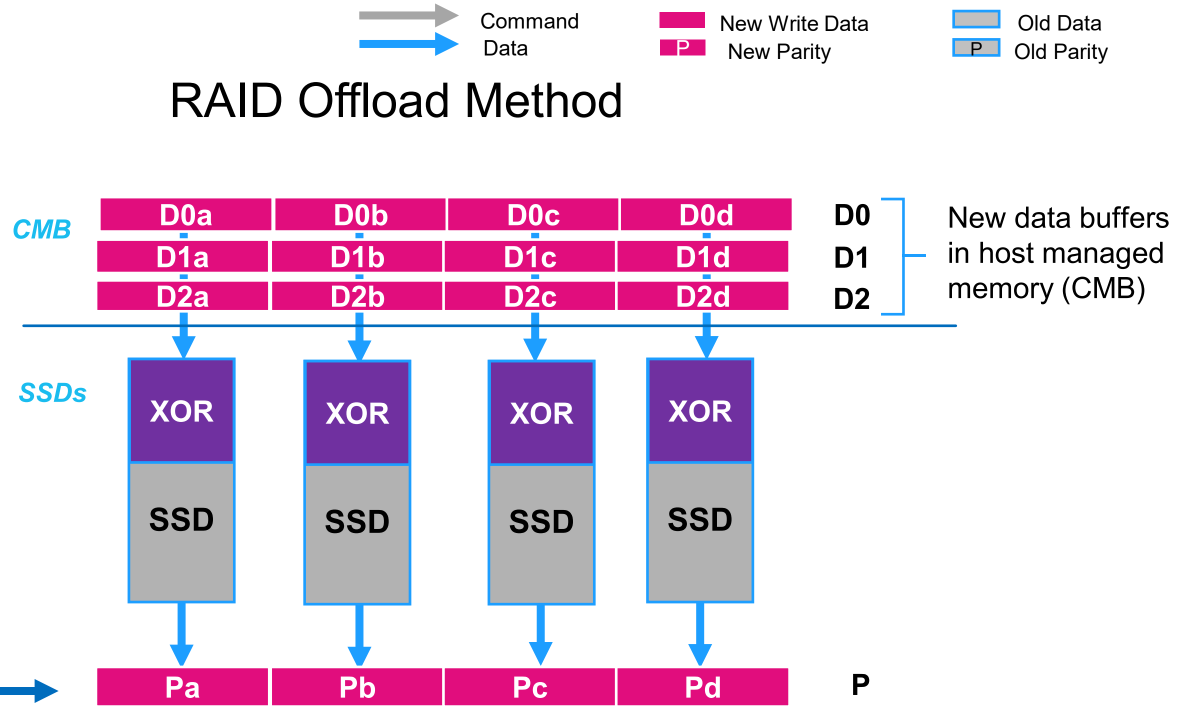
D3⊕Pi = P

2. And write new data and new parity

Parallel Parity Computation

Flexibility to optimize for Latency OR high bandwidth

RAID Offload Method



1. Host can calculate new parity column-wise (low latency) or row-wise for high throughput

D0a⊕D1a⊕D2a⊕D3a= Pa

D0b⊕D1b⊕D2b⊕D3b= Pb

D0c⊕D1c⊕D2c⊕D3c= Pc

D0d⊕D1d⊕D2d⊕D3d= Pd

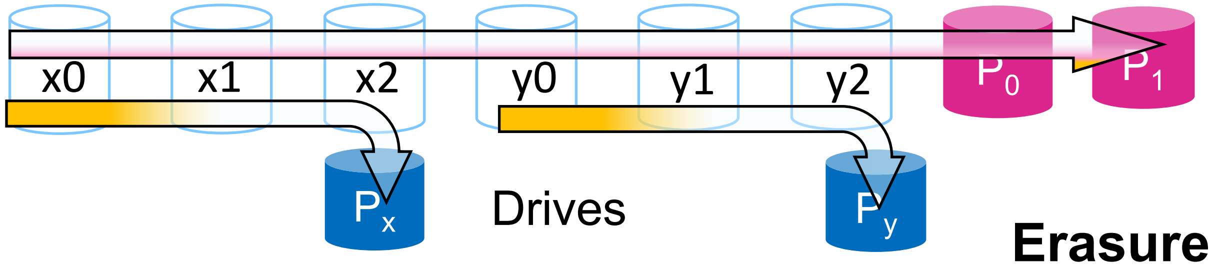
OR

Throughput

D0⊕D1⊕D2⊕D3= P

2. Host then write new data and new parity

Parallel XOR: Example Build 4 EC in One Command



Erasure Code command for **4 parity** compute

Parity P_x (XOR)	
Src buf	x0, x1, x2
Galois coefficient	1,1,1,1
Output buffer address	Px
Operation type	XOR

Parity P_y (XOR)	
Src buf	y0,y1,y2
Galois generator	1,1,1,1
Output buffer address	Py
Operation type	XOR

Parity P_0 (Weighted XOR)	
Src buf	x0, x1, x2,y0,y1,y2
Galois coeffficient	$\alpha_0, \alpha_1, \alpha_2, \beta_0, \beta_1, \beta_2$
Output buffer address	P_0
Operation type	XOR

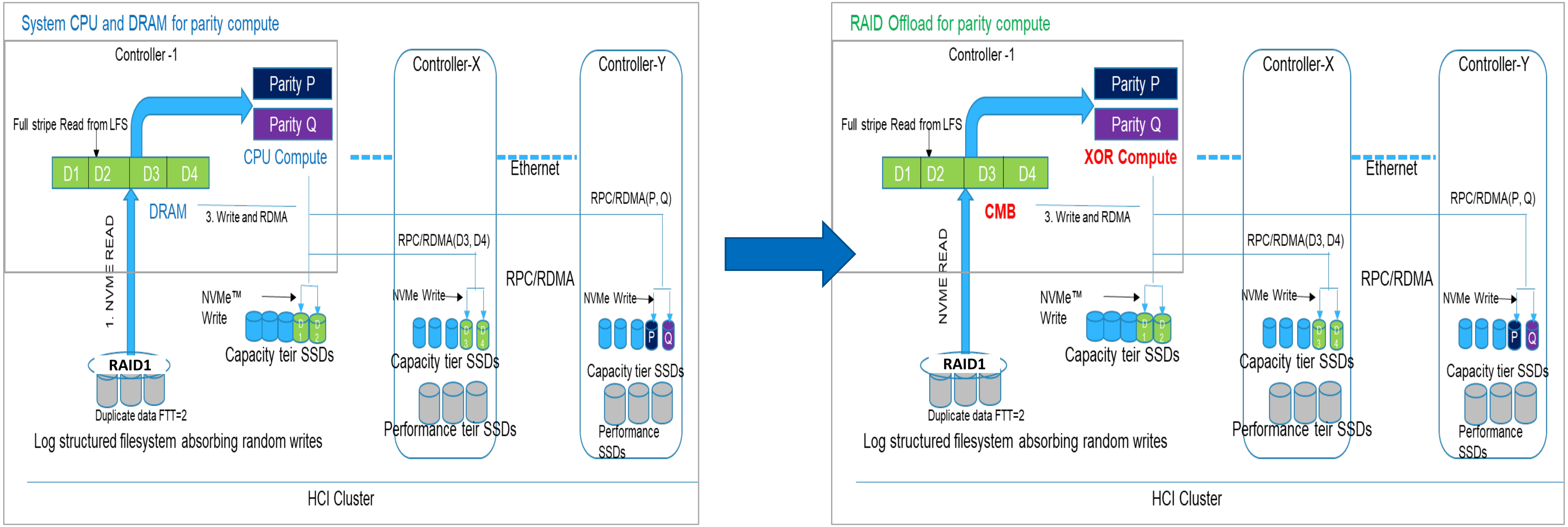
Parity P_1 ((Weighted XOR)	
Src buf	x0, x1, x2,y0,y1,y2
Galois coefficient	$\alpha_0^2, \alpha_1^2, \alpha_2^2, \beta_0^2, \beta_1^2, \beta_2^2$
Output buffer address	P_1
Operation type	XOR

Parallel XOR: Rebuild With Minimum Overhead

- Each SSD can participate in rebuilding the failed drive
- Each Parity command can build multiple stripes
- System does not pay compute and DDR bandwidth cost in rebuild
- Saturate write bandwidth of the destination drive

RAID Offload Adoption Environments

RAID Offload for Software Defined Storage (SDS)



- Leverage SSDs to offload parity compute

NVMe is a registered or unregistered mark of NVM Express, Inc. in the United States and other countries. All images, graphs, and/or graphics within this presentation are the property of KIOXIA America, Inc. (KIOXIA) and are reproduced with the permission of KIOXIA.

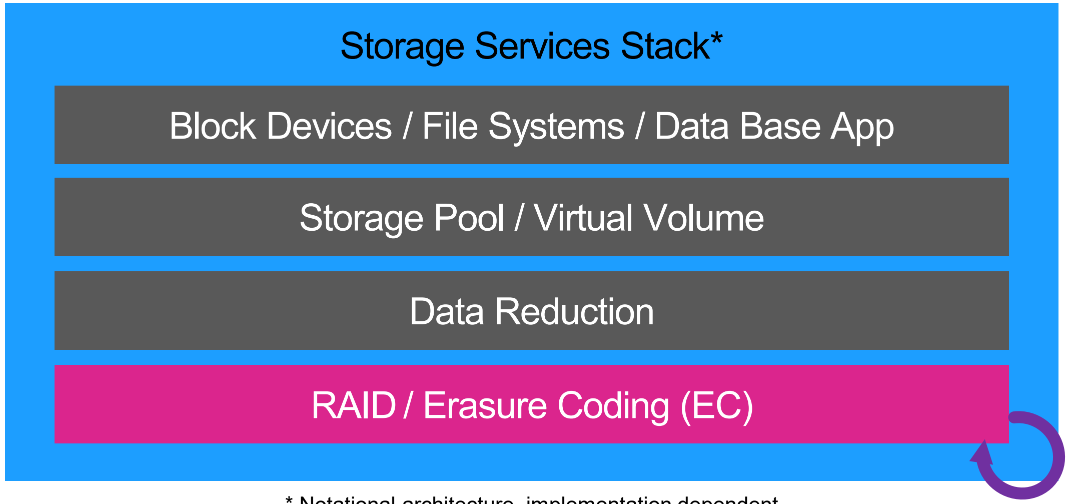
Graphics and product icons created by KIOXIA

NVMe is a registered or unregistered mark of NVM Express, Inc. in the United States and other countries. PCI is a trademark of PCI-SIG

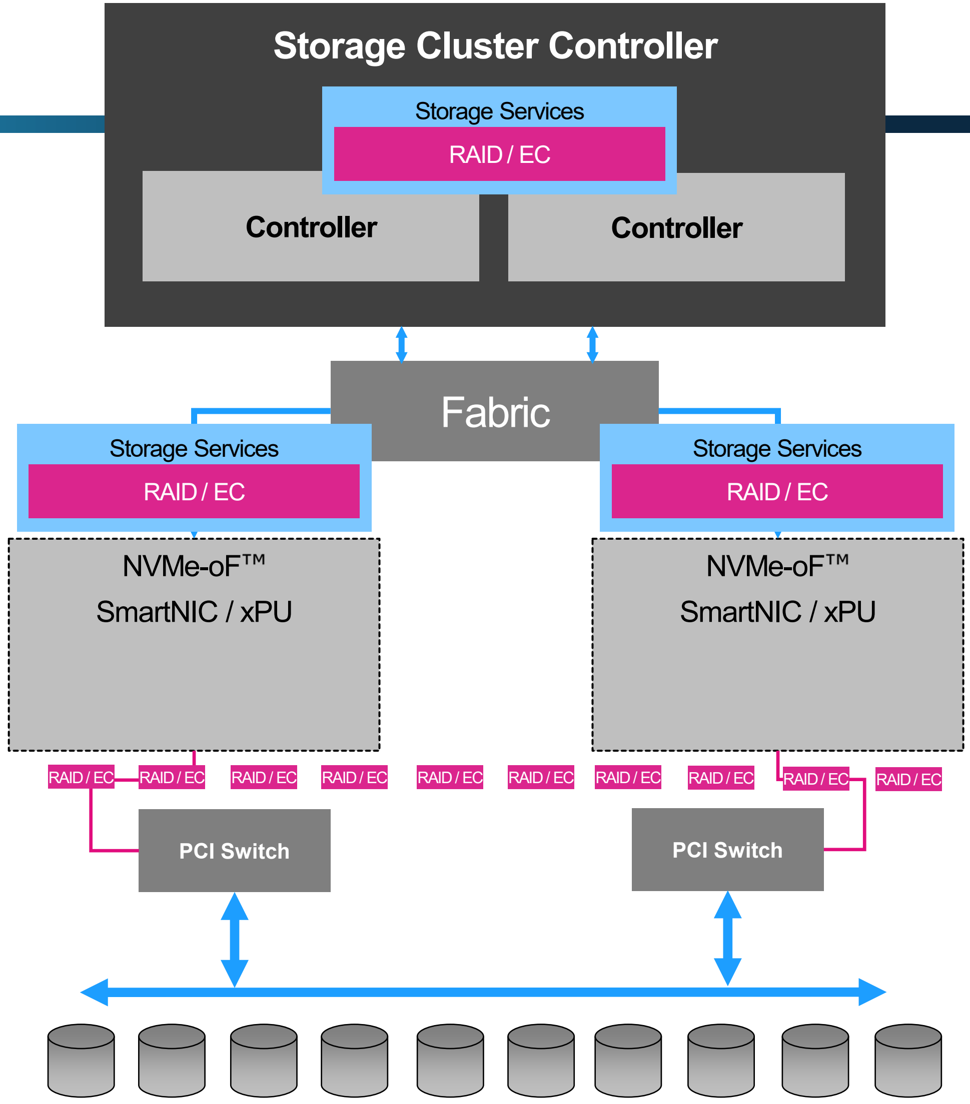
RAID Offload with xPU/DPU

- xPUs* are making inroads to offload storage services stack
- xPUs will be challenged for performance as Network bandwidth improvements in future
- xPUs and SSD can collaborate to offload storage services for cost effective scaleable storage solutions

*Based on the PCIe® evolution as published by PCI-SIG.

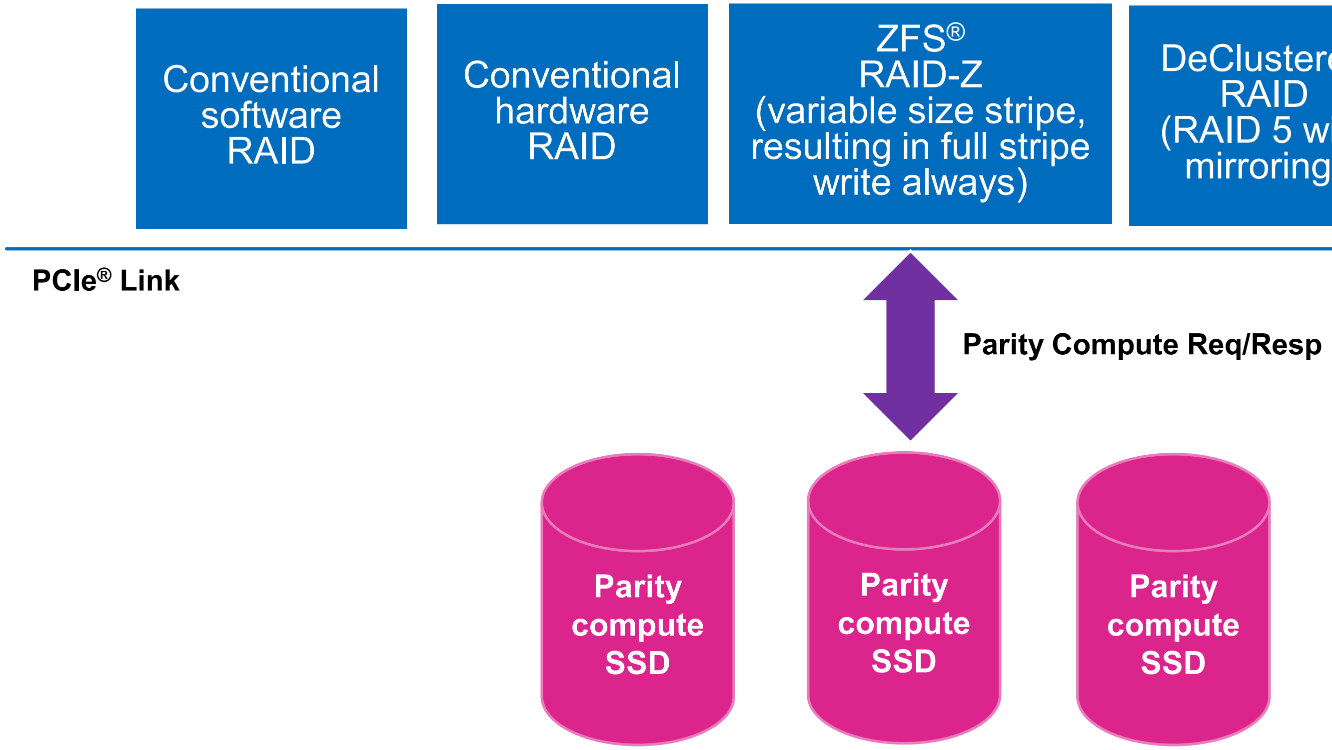


* Notational architecture, implementation dependent



* xPU represents a portfolio of architectures (i.e., CPU, GPU, FPGA and other accelerators), depending on the application. NVMe™, and NVMe-oF™ are trademarks of NVM Express, Inc. PCIe is a registered trademark of PCI SIG. All images, graphs, and/or graphics within this presentation are the property of KIOXIA America, Inc. (KIOXIA) and are reproduced with the permission of KIOXIA.

RAID/EC Type Agnostic



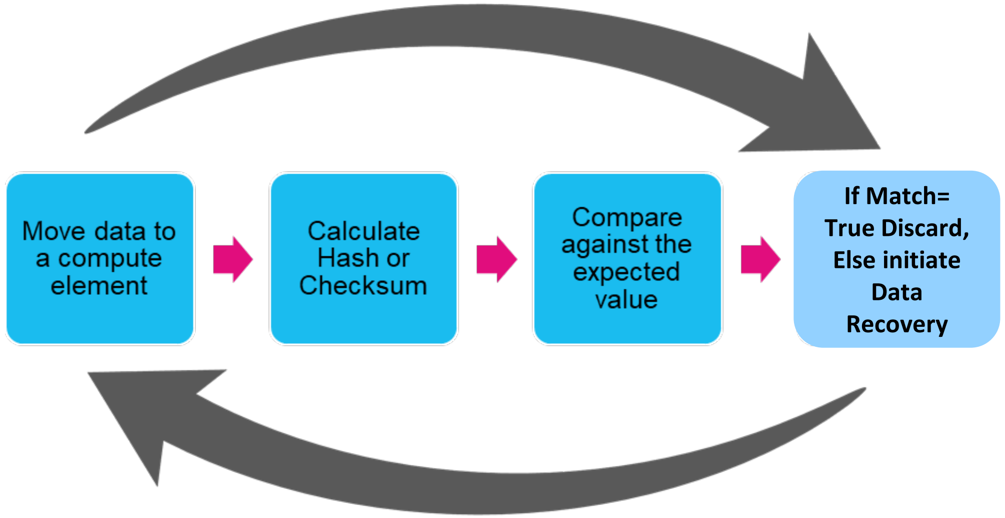
- RAID orchestration logic remains with application
- Any software or hardware-based solution can use parity compute offload
- Each SSD can compute parity at enough performance to scale
- Application's choice to use CMB or host DDR
- Flexible interface for parity computation
- Single command can contain multiple independent parity requests

* LFS = Log-structured file system. ZFS is a registered trademark of Oracle. PCIe is a registered trademark of PCI SIG. VMware, VMware ESXi, VMware vMotion, VMware vSAN, VMware vSAN Express Storage Architecture and VMware vCenter are registered trademarks or trademarks of VMware Inc. in the United States and/or various jurisdictions. All other company names, product names and service names may be trademarks or registered trademarks of their respective companies. All images, graphs, and/or graphics within this presentation are the property of KIOXIA America, Inc. (KIOXIA) and are reproduced with the permission of KIOXIA.

RAID Offload Application

Data Scrubbing in Conventional Setup

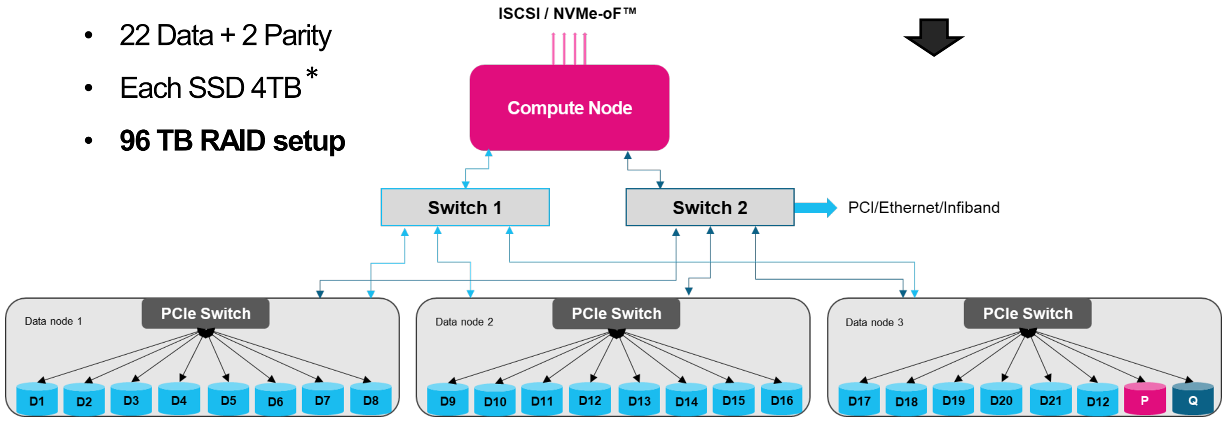
Data Scrubbing: early detection and correction of errors



All data movement during scrubbing operation is an overhead penalty paid to ensure data integrity

Assumptions:

- 22 Data + 2 Parity
- Each SSD 4TB*
- **96 TB RAID setup**



Compute node performing disk scrubbing for one stripe using RAID

P + D1 + D2 + D3 + D4 …… + D22 = 0

Q + g1.D1 + g2.D2 + g3.D3 + ….. + g22.D22 = 0

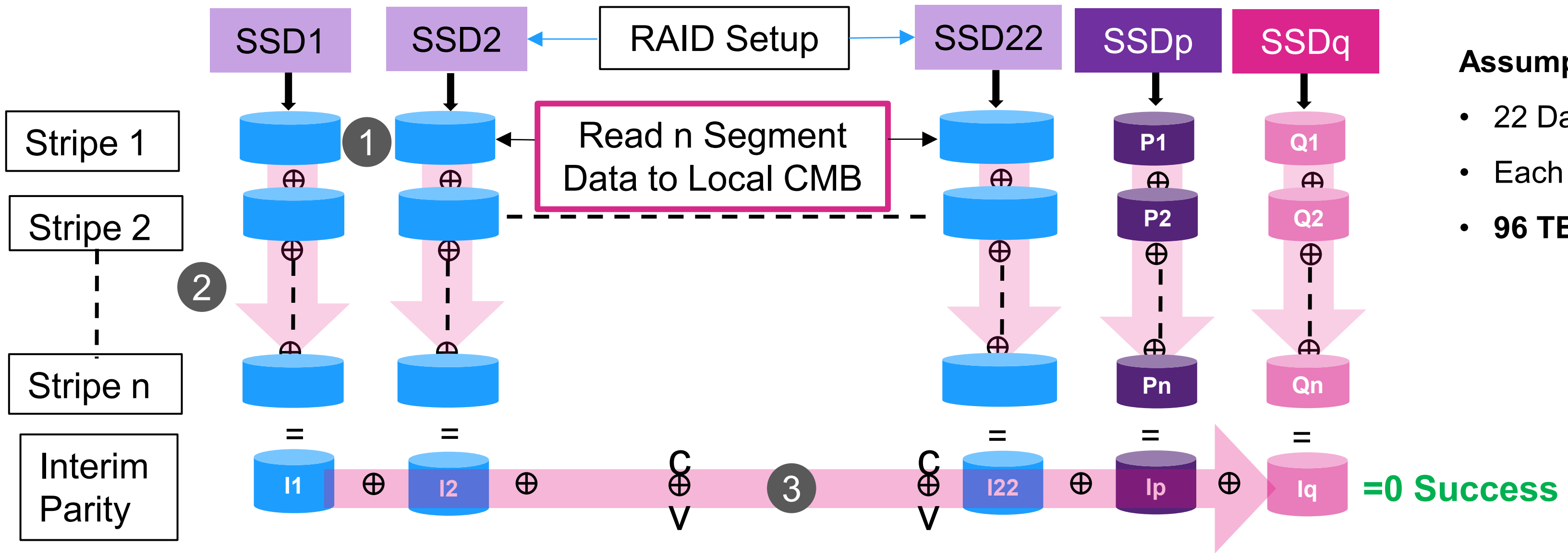
In above setup, 96 TB data moves through the PCIe® bus, network, CPU and 192 TB through memory subsystem during each scrubbing cycle

Assumptions created by KIOXIA in-house engineering team

* TB = terabytes. PCI is a trademark of PCI-SIG. KIOXIA Corporation defines a megabyte (MB) as 1,000,000 bytes, a gigabyte (GB) as 1,000,000,000 bytes and a terabyte (TB) as 1,000,000,000,000 bytes. A computer operating system, however, reports storage capacity using powers of 2 for the definition of 1GB = 230 = 1,073,741,824 bytes and therefore shows less storage capacity. Available storage capacity (including examples of various media files) will vary based on file size, formatting, settings, software and operating system, and/or pre-installed software applications, or media content. Actual formatted capacity may vary. All images, graphs, and/or graphics within this presentation are the property of KIOXIA America, Inc. (KIOXIA) and are reproduced with the permission of KIOXIA.

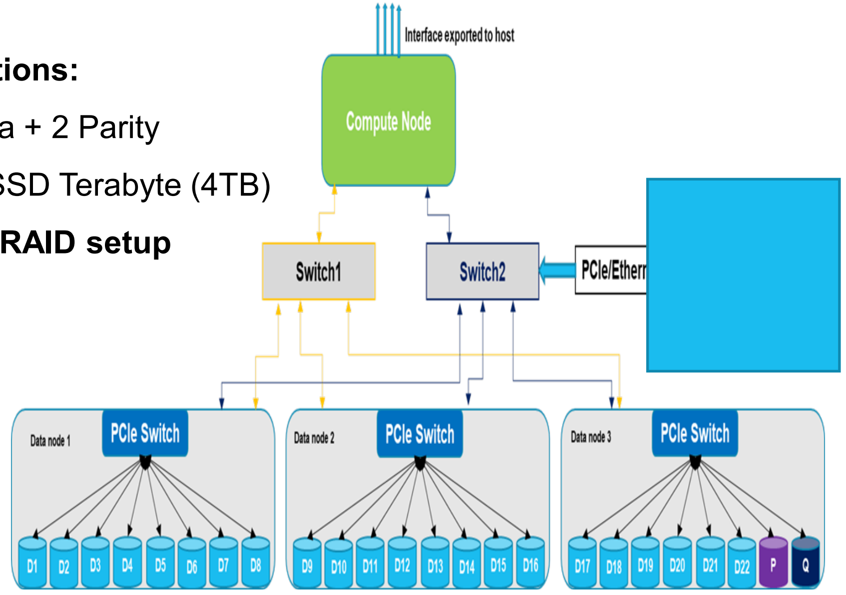
SDC 24

Data Scrubbing with RAID/EC Offload PoC



Assumptions:

- 22 Data + 2 Parity
- Each SSD Terabyte (4TB)
- **96 TB RAID setup**



2. I1= D11 + g1.D11 + D21 + g2.D21 + ……+ Dn1 + gn.Dn1
3. I1 + I2 + I3 + ……. + Ip + Iq == 0 ➔ **Success**

- Using 3 step process, ~99% data movement can be reduced
- No data passes through CPU and DRAM on compute node
- For n stripes, only one stripe moves over network and PCIe®
- Data scrubbing proof of concept data shown in table is for 9 SSDs

Resource Utilization	Offload Disabled	Offload Enabled
Scrubbing Time	129s	91s
DRAM Bandwidth	10.24 GB/s	1.43 GB/s
Total CPU Utilization	99.5%	~70%
L3 Cache Misses	14.7M	4M
Total PCIe Write (MB/s)	3694 MB/s	159 MB/s

Assumptions created by KIOXIA in-house engineering team

Assumptions created by KIOXIA in-house engineering team

Summary

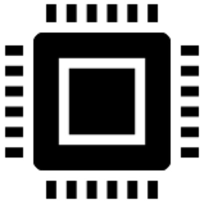
- RAID Offload is cost effective, scalable and sustainable technology
- Scales linearly with number of drives
- No additional CPU and DRAM required for redundancy computation
- Existing RAID/EC applications can offload with minimum change
- Rebuild at the max performance of target drive
- Data Scrubbing saves significant system resources

SDC 24

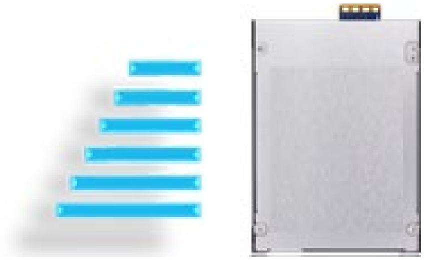
Future Possibilities: A Call to Action

- Evaluate RAID Offload with the KIOXIA PoC platform
- Participate in NVMe™ standardization discussions
- Collaborate with KIOXIA to explore offload functions beyond RAID

Standards Based



Host Controlled



Hardware Accelerators

(Memory, Compute, DMAC)

RAID Offload Brief

We are ready to collaborate!

Please take a moment to rate this session.

Your feedback is important to us.

