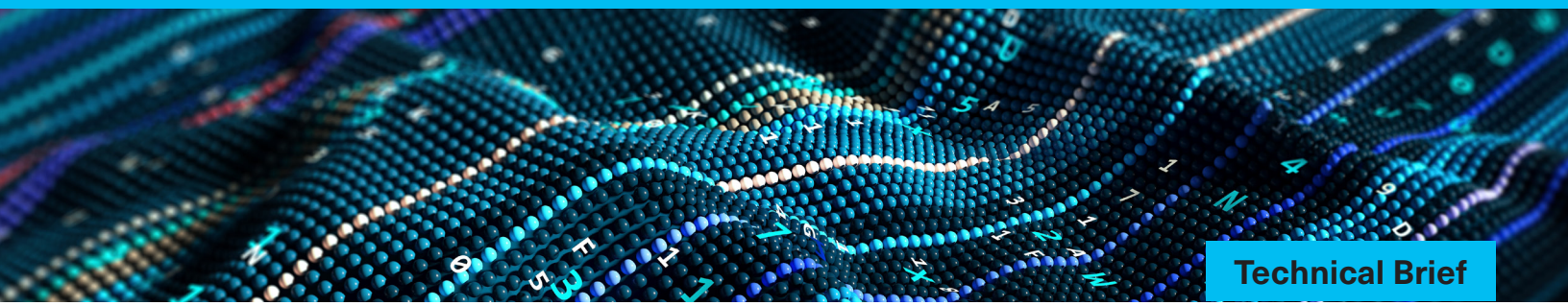
KIOXIA



Technical Brief

Introducing KIOXIA Data Scrubbing Offload Technology

Offloading Data Scrubbing to SSDs where Data Resides Eliminates Unnecessary Data Movement without Sacrificing Data Quality

Collecting, refining and leveraging data has become crucial for most industries and governments today to maintain a competitive edge. These data processes give rise to challenges, of which data integrity and data movement are included. Data integrity is crucial for ensuring the accuracy, consistency and reliability of data. Maintaining data integrity helps organizations make better informed decisions, prevents errors and builds trust in the data-driven processes. However, methods to ensure data integrity, such as data scrubbing, require a lot of unnecessary data movement, resulting in resource waste within data center architectures. As data continues to grow exponentially, these challenges are worsening.

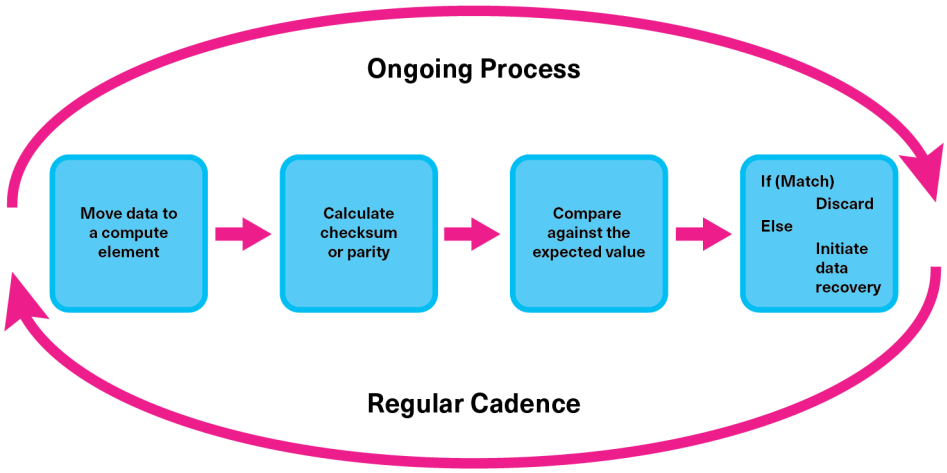
Data scrubbing is an error correction technique for ensuring data validity when applications need to access the data. It is a background task that periodically inspects main memory or storage for errors, then corrects detected errors using redundant data in the form of different checksums or copies of data. Data scrubbing reduces the likelihood that single correctable errors will accumulate, leading to reduced risks of uncorrectable errors.

To efficiently address data integrity and data movement challenges, Kioxia Corporation developed data scrubbing offload technology, which offloads data scrubbing to SSD storage where the data resides, eliminating wasted resources through unnecessary data movement without sacrificing the quality of the data.

This technical brief discusses the need for data scrubbing offload, the existing method used and its challenges and presents KIOXIA Data Scrubbing Offload technology.

What is Data Scrubbing?

To detect errors, data scrubbing leverages checksum, parity re-compute or a similar function. Early detection and correction of errors reduces the risk of uncorrectable errors, so, correcting errors, when small in numbers, prevents data corruption. The image below depicts a typical data scrubbing operation using RAID (redundant array of independent disks).



A checksum is a small-sized block of data, derived from the data of interest. As shown in the table below, the checksum function outputs significantly different values, even when small changes are made to the input.

Input Text	Checksum Output
Hello World	b10a8db164e0754105b7a99be72e3fe5
Hello, World	82bb413746aee42f89dea2b59614f9ef
Hello, World!	65a8e27d8879283831b664bd8b7f0ad4

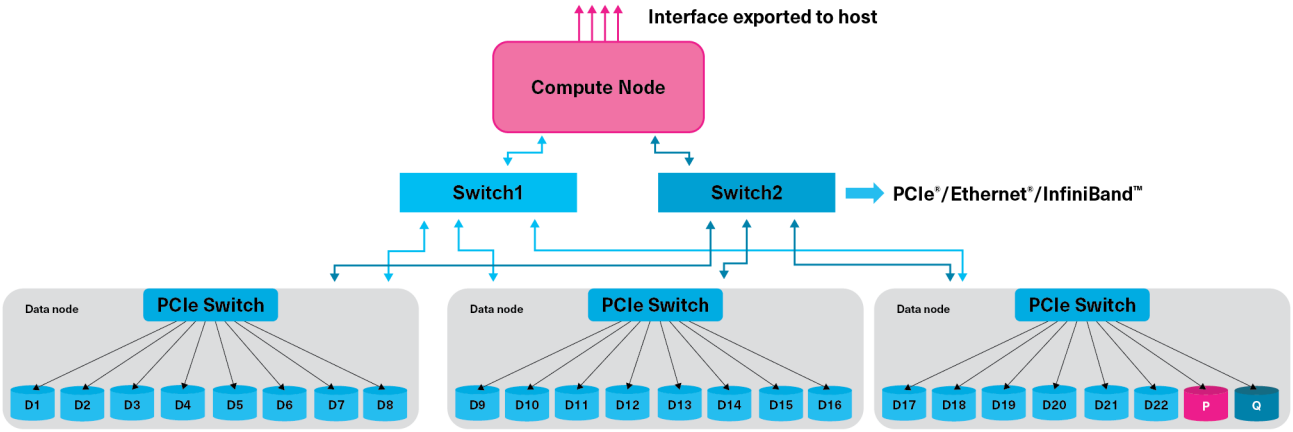
If the computed checksum for the current data input matches the stored value of a previously computed checksum, there will be high probability that the data was not accidentally altered or corrupted. It should be noted that checksums are often used to verify data integrity but are not relied upon to verify data authenticity.

The implementation of data scrubbing varies by the algorithm employed to scrub data, the frequency of the scrub operation, and the level at which data is scrubbed (block-level or file-level).

The biggest challenge with existing data scrubbing operations is unnecessary data movement. Data being scrubbed must move through the transport (i.e., PCIe®, SAS or Ethernet®) and the memory subsystem. If errors were found during the scrub, the host will initiate the data recovery process. The desired and best-case outcome of the scrubbing process is zero errors, which means that 100% of the energy spent in reading and processing this data is overhead, and the main motivation for data scrubbing offload technology.

Conventional Data Scrubbing Method

In many aggregated or disaggregated storage systems, the conventional data scrubbing method uses RAID stripes that span across many data nodes, as depicted in the image below:



In this configuration, the RAID stripe is spread across twenty-four drives (twenty-two data drives and two parity drives) that span across three data nodes. If two SSDs fail in this configuration, it is capable of recovering the data stored on the two failed drives. If the drive size is assumed to be four terabytes[1] (TB), then the RAID group size would become ninety-six TB. Also, in one RAID stripe, the data segments are defined as D1 to D22, and the parity segments are P and Q.

The conventional compute node method of data scrubbing requires the following steps:

1. *Choose a RAID stripe and lock it so no incoming writes can change the stripe data*
2. *Read all data segments and parity segments of one RAID strip*
3. *Compute these two equations*

   a. P + D1 + D2 + D3 + D4 ..... + D22 = 0
   b. Q + g1.D1² + g2.D2 + g3.D3 + ... + g22.D22 = 0
4. *Once the stripe is found to be correct, it is unlocked, and all data/parity segments are discarded from the compute node*
5. *Repeat steps 1-4 for each RAID stripe verification*

The resources used for this conventional data scrubbing method were determined to be:

System Resources to Scrub 96 TB	Conventional
Data passes through CPU for parity computation	96 TB
Data passes through DRAM controller	192 TB
Data passes through network	96 TB
Data passes through PCIe interface	96 TB

Data Scrubbing Method Using RAID Offload Technology

The new and advanced approach to data scrubbing is based on KIOXIA RAID Offload technology, and 'Best of Show' award winner at the Future of Memory and Storage 2024 symposium in August, in the 'Most Innovative Technology' SSD technology category. In simplest terms, this technology is an architecture that offloads compute and DRAM bandwidth demands to SSDs delivering the following benefits:

- ***Increased usage optimization of compute resources***
  - *Frees up host CPU, cache and memory resources to focus on primary applications*
- ***Improvement in system-level performance***
  - *Initializes/rebuilds a RAID volume at the maximum sequential write performance of an SSD*
- ***Efficient scaling***
  - *RAID parity compute throughput scales proportionally as the number of SSDs increases*
  - *Helps address the memory wall issue*
- ***Easy integration with the existing infrastructure***
  - *Utilizes the existing mature RAID stack and user interface*
  - *Agnostic to RAID geometry*
- ***Increased TCO optimization***
  - *Increases power efficiency in server and storage systems*

Similar concepts invented for KIOXIA RAID Offload technology helped to spur the development of KIOXIA Data Scrubbing Offload technology, which falls into two classifications:

1. *Data scrubbing using RAID Offload per stripe*
2. *Data scrubbing using RAID Offload across multiple stripes*

Data scrubbing using RAID Offload per stripe

This method of data scrubbing offload is best suited where the compute node does not want to lock more than one stripe to allow for incoming writes. To detect imperfections of the stripe itself, this method isolates one stripe at a time with an additional objective of reducing excess data traffic on the network fabric. There are three steps involved in this per stripe approach:

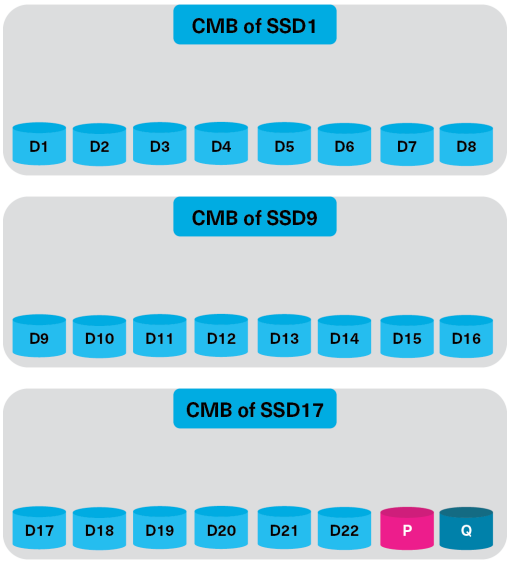
1. *In parallel, read data from all SSDs in a given data node to the Controller Memory Buffer (CMB) of one SSD and calculate an interim parity for that data node*
2. *Transfer all interim parities to the CMB of one SSD*
3. *Calculate the final parity – should be zero representing a good stripe*

Step 1:
Since each SSD has a CMB and parity computation, the host can now read the per node data segments into the CMB of one of the SSDs in that node. For example, using the conventional data scrubbing image presented earlier, the host performs three steps in parallel for the three data nodes as follows:

a. *Read D1 to D8 in the CMB of SSD1 for data node 1:*

b. *Read D9 to D16 in the CMB of SSD9 for data node 2:*

c. *Read D17 to D22, and parity segments P and Q, in the CMB of SSD17 for data node 3:*



As the three steps above occur in parallel for the three data nodes, the host, also in parallel, calculates the interim parity in each node as follows:

Data Node	SSD CMB	Calculations
Data Node 1	SSD1	PQNode1 = D1 + g1.D1^2 + . . . D8 + g8.D8
Data Node 2	SSD9	PQNode2 = D9 + g9.D9^2 + . . . D16 + g16.D16
Data Node 3	SSD17	PQNode3 = D17 + g17.D17^2 + . . . D22 + g22.D22

Step 2:
Transfer interim parities, PQNode1 and PQNode2, to the CMB of SSD17, of data node 3.

Step 3:
Calculate the final parity using all three interim parities, and compare it with zero.

The calculation of data node 3, for the CMB of SSD17, should be PQNode1 + PQNode2 + PQNode3 = 0. When Step 3 achieves a zero result, the compute node can conclude that all data and parity in a given stripe are correct, representing a good stripe.

When compared with the conventional method of data scrubbing, the results below show system resource usage savings when data scrubbing offload is used per scrub cycle in a compute node:

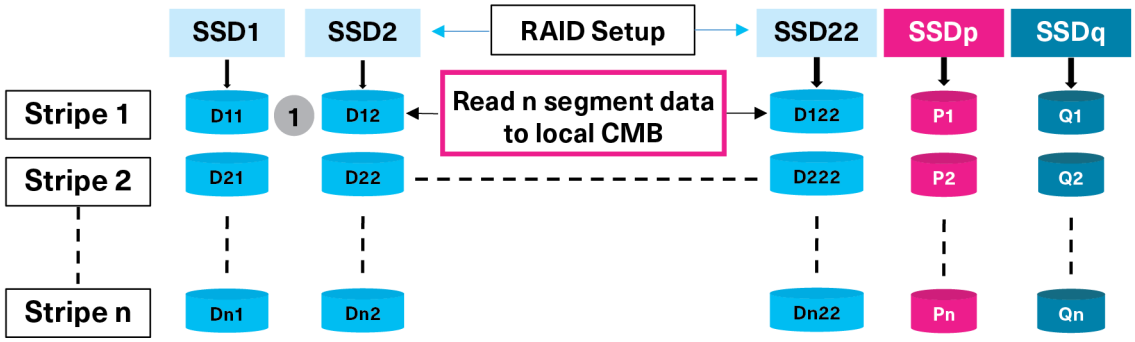
System Resources to Scrub 96 TB	Conventional Method	KIOXIA Data Scrubbing Offload	% of Savings
Data passes through CPU for parity computation	96 TB	0 TB	100%
Data passes through DRAM controller	192 TB	0 TB	100%
Data passes through network	96 TB	8 TB	92%
Data passes through PCIe® interface	96 TB	92 TB	5%

This data scrubbing using RAID Offload per stripe method demonstrates high efficiency except with an inability to significantly reduce PCIe traffic. The second data scrubbing method using RAID Offload across multiple stripes is better suited to address this data traffic deficiency.

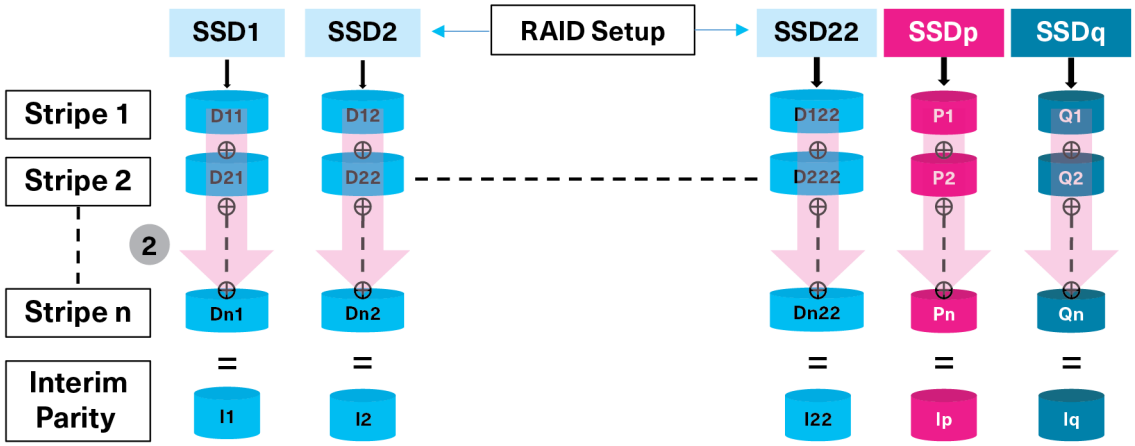
Data scrubbing using RAID Offload across multiple stripes

Since the bit error rate in NVMe™ SSDs are generally much less when compared with SATA SSDs or HDDs, data scrubbing offload can also be performed across multiple stripes. Large capacity NVMe SSDs should find this capability useful to reduce data scrubbing time, system resource utilization and power consumption. This method of data scrubbing offload also involves three steps, as follows:

Step 1:
This step involves reading and segmenting data from all SSDs to the local CMB as depicted in the image below:



Step 2:
Calculate the interim parity on each SSD vertically as depicted in the image below:



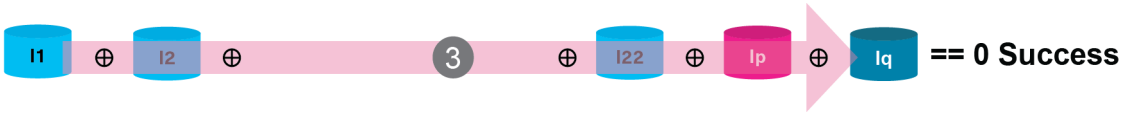
The calculations used to determine the interim parity included:

I1 = D11 + g1.D11^2 + D12 + g2.D12 + ...+ D1n + gn.D1n. Similar calculations were performed for I2 to Iq.

In Dnm, 'D' is data segment of RAID stripe n stored on SSDm and 'g' is the Galois coefficient. 'SSDp' and 'SSDq' are noted as fixed parity drives. RAID Offload supports rotational parity.

Step 3:

Calculate the horizontal parity. After the interim parity has been calculated, the host will transfer twenty-three interim parities to any chosen SSD CMB. On that SSD, the host performs the following action:



The horizontal XOR (exclusive OR) should provide zero output if all data and parity segments across the stripes are error free. When compared with the conventional method of data scrubbing, the results below show system resource usage savings when data scrubbing offload is used per scrub cycle in a 22+2 RAID configuration where each SSD capacity was four terabytes.

System Resources to Scrub 96 TB	Conventional Method	KIOXIA Data Scrubbing Offload	% of Savings
Data passes through CPU for parity computation	96 TB	0 TB	100%
Data passes through DRAM controller	192 TB	0 TB	100%
Data passes through network	96 TB	0.4 TB	99%
Data passes through PCIe® interface	96 TB	0.4 TB	99%

This data scrubbing offload method across multiple stripes demonstrates the highest efficiency for large capacity NVMe™ SSDs.

Data Scrubbing PoC Results Using mdRAID[3] 5

In this section, limited proof of concept (PoC) results are shown in this PCIe 4.0 mdRAID 5 configuration:

- *One Dell™ PowerEdge™ R650xs server with one Intel® Xeon® Gold 6338N processor (two-socket, thirty-two cores).*
- *Three, five or nine KIOXIA CM7-R Series Enterprise SSDs, each with 1.92 TB capacity.*
- *mdRAID 5 initialized the RAID volume at the time it was created. A 50 gigabyte[1] (GB) NVMe™ namespace was used to reduce initialization time.*

It should be noted that each KIOXIA CM7-R Series SSD could only process one request at any given time in this PoC platform. This limitation would not be present in product implementations as the resulting performance would be better.

The table below presents the results from the mdRAID 5 configuration:

	3 SSDs			5 SSDs			9 SSDs		
TEST	**No Offload**	**With Offload**	**Results w/ Offload**	**No Offload**	**With Offload**	**Results w/ Offload**	**No Offload**	**With Offload**	**Results w/ Offload**
Data Scrub Size	**150 GB**			**250 GB**			**450 GB**		
Parity Check Time	53 sec.	103 sec.	No Gain	69 sec.	95 sec.	No Gain	119 sec.	101 sec.	15% faster
Maximum Total DRAM Bandwidth	7,650 MB/s	200 MB/s	97% DRAM bandwidth reduction	9,580 MB/s	400 MB/s	95% DRAM bandwidth reduction	9,885 MB/s	900 MB/s	90% DRAM bandwidth reduction
Average CPU Utilization – mdRAID 5	99.4%	32.8%	67% CPU usage reduction	100.1%	30%	70% CPU usage reduction	100.5%	51%	49% CPU usage reduction

Observations:

- *In the PoC platform, each KIOXIA CM7-R Series SSD has one parity engine. As the number of SSDs increased in the mdRAID 5 configuration, parity compute capabilities also scaled in the same proportion. For three SSDs, the parity check time difference between data scrubbing offload and the 'no offload' solution was 50 seconds slower. For five SSDs, data scrubbing offload was 26 seconds slower. However, for nine SSDs with a higher level of parallel computing, data scrubbing offload was 15% faster than the host-based solution.*
- *DRAM bandwidth savings ranged from 90% with nine KIOXIA CM7-R Series SSDs up to 97% savings with three SSDs.*
- *CPU utilization reductions ranged from 49% with nine KIOXIA CM7-R Series SSDs up to 70% with five SSDs.*

Summary

KIOXIA Data Scrubbing Offload technology enables very efficient methods to detect errors early and correct them to reduce the risk of uncorrectable errors. It enables all SSDs in a RAID group to perform data scrubbing of different stripes, or multiple stripes in parallel. PoC results showed performance scaling as the numbers of SSDs in the configuration increased and delivered 15% faster parity check times with nine KIOXIA CM7-R Series SSDs when compared with the host-based solution. With data scrubbing offload, the PoC experienced DRAM bandwidth savings from 90% up to 97%, and CPU utilization reductions from 49% up to 70%. Existing RAID solutions can use data scrubbing offload either per stripe, or across multiple stripes, to save time, better utilize system resources and deliver scale up performance.

NOTES:

[1] Definition of capacity: Kioxia Corporation defines a megabyte (MB) as 1,000,000 bytes, a gigabyte (GB) as 1,000,000,000 bytes, a terabyte (TB) as 1,000,000,000,000 bytes and a petabyte (PB) as 1,000,000,000,000,000 bytes. A computer operating system, however, reports storage capacity using powers of 2 for the definition of 1Gbit = 2^{30} bits = 1,073,741,824 bits, 1GB = 2^{30} bytes = 1,073,741,824 bytes, 1TB = 240 bytes = 1,099,511,627,776 bytes and 1PB = 2^{40} bytes = 1,125,899,906,842,624 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.

[2] The 'g' in the equation is a Galois coefficient that is computed by the host on a polynomial equation. The 'g.D' in the equation means each byte of data segment 'D' is multiplied by 'g.' In erasure coding, many RAID solutions use proprietary polynomials to generate Galois coefficients.

[3] mdRAID 5 is Linux® software RAID that enables RAID functionality without a hardware RAID controller.

TRADEMARKS:

DISCLAIMERS: