

Benefits of Using the Controller Memory Buffer in Enterprise NVMe SSDs

Leveraging Underutilized SSD DRAM Bandwidth to Reduce SSD Latency and Improve System-Level Performance

AI, machine learning and analytics dominate today's compute landscape and need the utmost in high data throughput and low latency from the SSDs that support them. KIOXIA has improved drive performance and enabled system-level performance for data-intensive applications that reduces overhead associated with data transfers by utilizing the Controller Memory Buffer (CMB) feature in the NVMe™ 2.0 specification.

Enterprise NVMe SSDs typically include gigabytes¹ (GB) of on-controller DRAM to help deliver high performance. This DRAM is used primarily by the SSD controller, particularly for large Flash Translation Layer (FTL) tables that enable low-latency address translation and efficient media management. Although this memory capacity is essential, its available DRAM bandwidth is not always fully utilized by internal SSD activity.

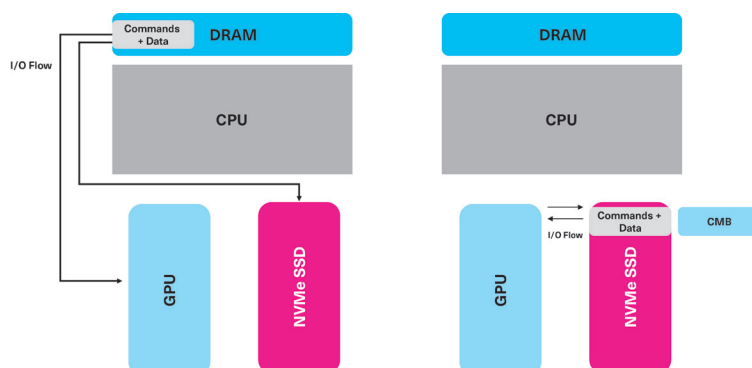
The NVMe CMB capability allows a portion of this controller-resident memory to be exposed to the host through a PCIe® base address register² (BAR), making it directly host-addressable. This exposed memory region, together with its otherwise underutilized bandwidth, can be leveraged by host software and applications to offload selected tasks, and is especially valuable in memory and bandwidth-constrained systems.

Instead of requiring the NVMe controller to fetch input/output (I/O) from host DRAM across the PCIe bus, the host can place Submission Queues (SQs), Completion Queues (CQs), command entries and associated data descriptors - including Physical Region Pages (PRPs) and Scatter Gather Lists (SGLs) - directly into the CMB. These capabilities move command-path metadata out of host DRAM and into device-local DRAM, allowing the controller to consume these structures without issuing PCIe memory read transactions for command fetch and descriptor traversal. The result is reduced PCIe read traffic and improved command-path efficiency.

More broadly, the same host-addressable CMB region can be used by system software or applications to stage additional metadata, transient buffers or Direct Memory Attached (DMA)-related control structures enabling partial offload of memory-intensive I/O processing from the host. In memory-constrained environments, CMB is not just for queue placement optimization, but is a practical mechanism for reducing host memory pressure and reclaiming host DRAM bandwidth for application use while exploiting underutilized SSD DRAM bandwidth.

The image to the right shows simplified data flow (left-side) and big accelerator memory³-like data flow (right-side). Any PCIe endpoint device could take the place of the GPUs shown. Selected KIOXIA enterprise NVMe SSDs support Proof-of-Concept (PoC)-level CMB implementations.

Getting the most out of available compute resources is key to any system architecture, and as such, data directly on SSD memory yields fewer read or write operations from host DRAM as part of the overall data transfer sequence. In other schemes, a GPU, Network Interface Card (NIC) or other PCIe device can push data directly onto an SSD (via Remote Direct Memory Access/RDMA), which bypasses host DRAM and compute resources entirely. These capabilities improve command transfer latency, bandwidth efficiency and host resource utilization. This tech brief provides an overview, benefits and applications of CMB as specified in the NVMe 2.0 specification.



Why CMBs?

The table below summarizes estimated on-device DRAM bandwidth utilization for a 15 terabyte¹ (TB) data center NVMe™ SSD across representative enterprise and emerging workload profiles. For a conventional 70:30 read/write (R/W) enterprise storage workload, total SSD DRAM bandwidth consumption is typically limited to approximately 10% to 15% of available on-controller DRAM bandwidth. These limitations are typically due to the read-dominant traffic placing comparatively low demand on SSD DRAM bandwidth as data-path movement for read operations is largely handled through controller-managed DMA engines rather than from sustained DRAM-intensive staging. In contrast, emerging AI and data-intensive pipelines are increasingly trending toward 50:50 read/write ratios that can materially increase internal metadata activity, write staging pressure and aggregate DRAM bandwidth demand within the device.

Workload	Data Bandwidth	DRAM Bandwidth Consumption	DRAM Utilization	Estimated Spare DRAM Bandwidth
Sequential Read	13-14 GB/s*	Negligible (metadata)	<~1%	~24.6 GB/s
Sequential Write	~7 GB/s	14 GB/s	~60%	~10 GB/s
Random Read (@2,700 KIOPS*)	~10.8 GB/s	Negligible (metadata)	<~1%	~24.6 GB/s
Random Write (@600 KIOPS)	~2.4 GB/s	~4.8 GB/s	~20%	~21.2 GB/s
Mixed 70:30 (R/W) (@~1,320 KIOPS Read, ~395 KIOPS Write)	~5.28 GB/s	~3 GB/s	<~12%	~25.6 GB/s
Mixed 50:50 (R/W)* (@~1,350 KIOPS Read, ~300 KIOPS Write)	~6.6 GB/s	~2.4 GB/s	<~10%	~22.8 GB/s

NOTE: The metrics presented for data bandwidth, DRAM bandwidth used, spare DRAM bandwidth and typical utilization are high level estimates based on published specifications as of this publication date and are approximations.

*GB/s = Gigabytes per second

* IOPS = Input/output operations per second

*For emerging AI workloads

The table above is based on the following assumptions:

Estimated DRAM bandwidth inside of the 15 TB PCIe® Gen 5 enterprise SSD:

- PCIe Gen 5 x4 interface bandwidth: 16 GB/s
- DDR5 DRAM bandwidth: 25.6 GB/s (2 channels)

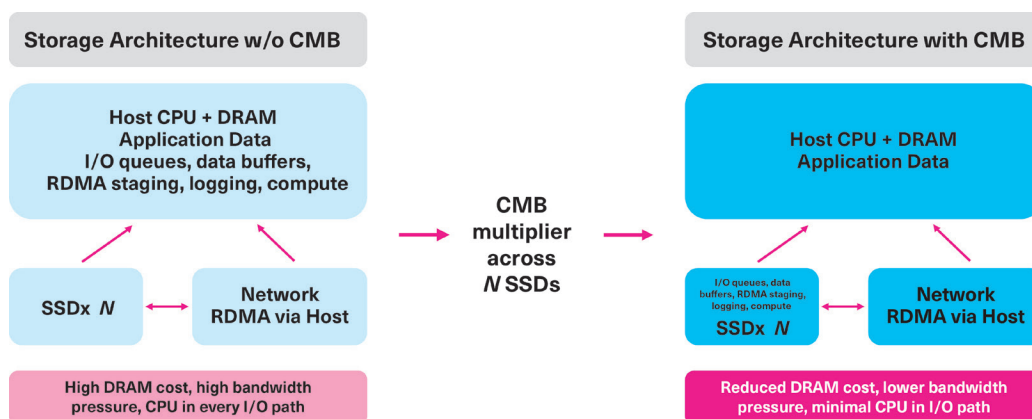
The DDR5 DRAM bandwidth is between 12.8 GB/s and 25.6 GB/s (assuming 1-2, 3,200 megabytes per second (MB/s) channels at both 32-bit and 64-bit).

In a conventional NVMe host/device model, SQs and CQs, command-related metadata, data buffers, PRPs and SGLs are allocated in host DRAM. As a result, the SSD controller must issue PCIe memory read transactions to fetch command entries and as required, traverse host-resident PRP/SGL structures before executing the I/O. These accesses consume PCIe bandwidth, introduce command-path latency and may incur additional penalties when transactions traverse one or more PCIe switch hops. At high queue depths or high IOPS rates, the cumulative overhead of repeatedly fetching command-path metadata from host memory can become significant enough to constrain overall system efficiency.

From an architectural standpoint, on-controller DRAM is provisioned primarily to support internal SSD functions such as FTL mapping-table storage, write coalescing and staging, firmware execution and other controller-local metadata operations. For high-capacity enterprise SSDs, FTL capacity alone can scale to roughly 1 GB of DRAM per 1 TB of NAND capacity, depending on mapping granularity and implementation. Although this DRAM capacity is mission-critical, its instantaneous bandwidth utilization is often well below peak, leaving a meaningful amount of otherwise idle memory bandwidth that is available for external use.

By exposing a portion of this memory through the NVMe CMB feature and where supported through Subsystem Local Memory (SLM) mapping, the system can repurpose SSD-local DRAM bandwidth for host-visible functions. This repurposing enables a selective relocation of the queue state, transport metadata, staging buffers, logging structures, RDMA descriptors and peer-to-peer (P2P) transfer state out of host DRAM and into device-local memory. For memory or bandwidth-constrained hosts, repurposing effectively converts unused SSD DRAM bandwidth into a system-level resource.

At rack and node scale, storage performance is frequently limited, not only by media bandwidth, but by host memory bandwidth and I/O software overheads. CMB-based memory exposure provides a path to aggregate device-local memory resources across 'N' SSDs, creating a distributed memory-bandwidth pool that can offload portions of the I/O control path and select data-movement functions from the CPU complex and host DRAM subsystem. The distributed memory-bandwidth pool can reduce host DRAM DIMM requirements, lower CPU cycle consumption in the storage stack and improve PCIe® transaction efficiency as depicted in the chart on page 2, which in turn can result in a lower-cost system architecture while preserving or improving end-to-end (E2E) throughput (see image below):



CMB Use Cases

CMB can offload both NVMe™ I/O command queues and small data buffers. By placing SQs and associated PRPs or SGLs directly in CMB, the controller eliminates host-initiated PCIe read transactions for queue access. For small payloads, data can also bypass host DRAM enabling lower-latency transfers while supporting P2P data movement. RDMA can further extend the value of CMB by using it for queue management and zero-copy data transfers. In these models, CMB can be mapped directly into the NIC address space, allowing inbound RDMA writes to place data directly into the storage or accelerator controller without staging them through host DRAM. This capability can reduce host memory bandwidth consumption and can lower E2E latency, as the SSD controller processes data directly from its local CMB.

The SNIA® Computational Storage Technical Working Group (TWG) is standardizing a CMB use case for fixed-function compute offload. In this model, CMB is used to stage data and commands and execute fixed operations such as checksum, deduplication, hashing, compression and RAID parity calculations. By performing these operations locally within the SSD, PCIe traffic and host DRAM usage can be reduced, freeing host resources for application workloads.

In addition, emerging fixed-function offload use cases are being investigated for memory-bandwidth-constrained AI workloads, including vector searches on quantized vectors and key-value (KV) cache compression.

CMB Is and Is Not

In essence, CMB can unlock trapped spare memory bandwidths, but **it is not**:

- A DRAM replacement
- A direct comparison to system DRAM and should not be claimed as low-latency memory
- Intended to saturate PCIe® bandwidth

However, **it is**:

- A source of spare memory bandwidth within an SSD controller
- Efficient for DMA-based data transfers
- Suitable as a bounce buffer or temporary memory
- Well suited for P2P data transfers
- Positioned for power-efficient compute offload applications

More information on KIOXIA enterprise SSDs is available [here](#).

FOOTNOTES:

¹ Universal Flash Storage (UFS) is a product category for a class of embedded memory products and devices built to the JEDEC[®] UFS standard specification. The v5.0 specification is the current release issued by JEDEC and announced in February 2026.

² Product density is identified based on the density of memory chip(s) within the Product, not the amount of memory capacity available for data storage by the end user. Consumer-usable capacity will be less due to overhead data areas, formatting, bad blocks, and other constraints, and may also vary based on the host device and application. For details, please refer to applicable product specifications. The definition of 1 KB = 2¹⁰ bytes = 1,024 bytes. The definition of 1 Gb = 2³⁰ bits = 1,073,741,824 bits. The definition of 1 GB = 2³⁰ bytes = 1,073,741,824 bytes. 1 Tb = 2⁴⁰ bits = 1,099,511,627,776 bits. 1 TB = 2⁴⁰ bytes = 1,099,511,627,776 bytes.

³ 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 1 Gbit = 2³⁰ bits = 1,073,741,824 bits, 1GB = 2³⁰ bytes = 1,073,741,824 bytes, 1TB = 2⁴⁰ bytes = 1,099,511,627,776 bytes and 1PB = 2⁴⁰ 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.

⁴ Read and write speed may vary depending on various factors such as host devices, software (drivers, OS, etc.) and read/write conditions.

⁵ All product images shown are representations of the design models and not accurate product depictions.

⁶ Based on preliminary specifications provided by KIOXIA Corporation, and subject to change, and this published technology topic: <https://www.kioxia.com/en-jp/rd/technology/topics/topics-83.html>.

⁷ Based on preliminary specifications provided by KIOXIA Corporation, and subject to change, and this published technology topic: <https://www.kioxia.com/en-jp/about/news/2026/20260703-1.html>.

TRADEMARKS:

JEDEC is a registered trademark of the Joint Electron Device Engineering Council (JEDEC) Solid State Technology Association. NVMe is a registered or unregistered trademark of NVM Express, Inc. in the United States and other countries. PCIe is a registered trademark of PCI-SIG. All other company names, product names and service names may be trademarks of third-party companies.

DISCLAIMERS:

© 2026 KIOXIA America, Inc. All rights reserved. Information in this technical brief, including product specifications, tested content, and assessments are current and believed to be accurate as of the publication date of the document, but is subject to change without prior notice. Technical and application information contained here is subject to the most recent applicable KIOXIA product specifications. Images within are for illustration purposes only.