

Fixed Function Compute Framework Proposal to SNIA

A Standard Interface for Near-Data Computing that Reduces Data Movement

AI, cloud, enterprise and IoT applications process unprecedented amounts of data that have put strain on every system resource, namely compute, memory, network and storage. To relieve this resource strain, NVMe Express™ introduced the [Computational Storage Feature](#) for NVMe™ specification 2.0 that provides a standardized, vendor-neutral and generic framework for connecting applications to NVMe computational storage devices, enabling them to process data locally. These SSDs offload work from the main host CPU that removes data movement bottlenecks and speeds up tasks like encryption, compression/decompression and data analytics. Adoption of this proposed specification is challenging due to interoperability concerns (varying hardware/software architectures) and finding the appropriate applications.

To move adoptability forward, KIOXIA proposed to standardize Fixed Function Compute (FFC) capabilities for NVMe devices through the SNIA® Computational Storage Technical Work Group (TWG). The group's objective is to create a common, interoperable framework that enables NVMe devices to perform predefined computational tasks close to the data. The proposed FFC approach focuses on standardized interfaces and application programming interfaces (APIs) that enable host systems to orchestrate computational offloads, such as checksum generation, parity calculations, compression/decompression, deduplication and data integrity operations.

Why Fixed Function Compute Offload?

At present, there are no common standard interfaces available between host and compute offload engines (hardware accelerators). The KIOXIA near-data computing¹ approach resolves this interface void through FFC interface standardization which has the following benefits:

- Ensures less data movement, which in turn reduces system power consumption and energy costs (depicted in Figure 3)
- Provides hardware accelerated, power-efficient compute engines with fixed functionality when compared to general-purpose compute capabilities via CPUs in servers
- Provides host-orchestrated operations
- Introduces a standard NVMe interface between the host and compute offload engines in SSDs
- Allows for scalable solutions as each storage system SSD offers the same compute capabilities
- Reduces system resource usage (i.e., compute, memory and network)
- Enables applications and system tasks to run faster by bypassing the typical bottlenecks associated with traditional data transfer
- Delivers an infrastructure designed to meet future offload requirements

If an SSD has some compute capabilities, such as parity computation, compression/decompression or vector distance calculation, using FFC framework, these compute capabilities can be exported to the host. The proposed FFC command and data flow are depicted in Figure 1.

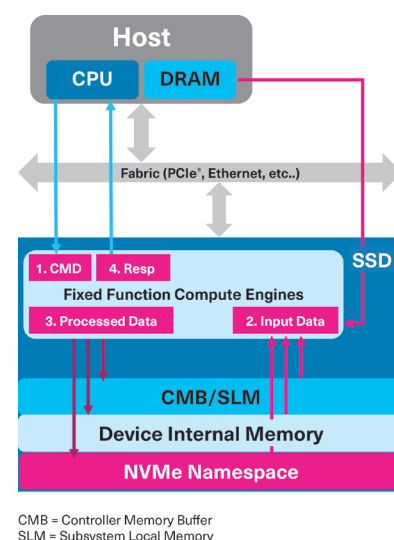


Figure 1: Proposed FFC command and dataflow
Created by KIOXIA America, Inc

Fixed Function Compute Architecture

The proposed FFC architecture introduces a generic API-based framework that supports multiple predefined compute operations through a common mechanism. The key design principles and core characteristics include:

- *Parameter-driven operation execution*
- *Secure memory addressing*
- *Unidirectional data flow*
- *Self-sufficient framework that does not impact any existing NVMe™ data structure defined in the specification*
- *End-to-end data protection support*

The FFC framework proposes two NVMe commands – one to initiate computation – the other to fetch computed data. A compute command includes many descriptors that specify different arguments required for a generic compute function. Examples include:

- *The 'function descriptor' specifies the compute operation that needs to be performed*
- *The 'function parameter descriptor' specifies function-specific arguments, like algorithms, etc.*
- *The 'source descriptor' specifies input data location and its length*
- *The 'destination descriptor' specifies where to place computed data*

This descriptor-based approach enables support for different computational workloads while maintaining a consistent interface. The different types of descriptors proposed for FFC command are listed in the image below:

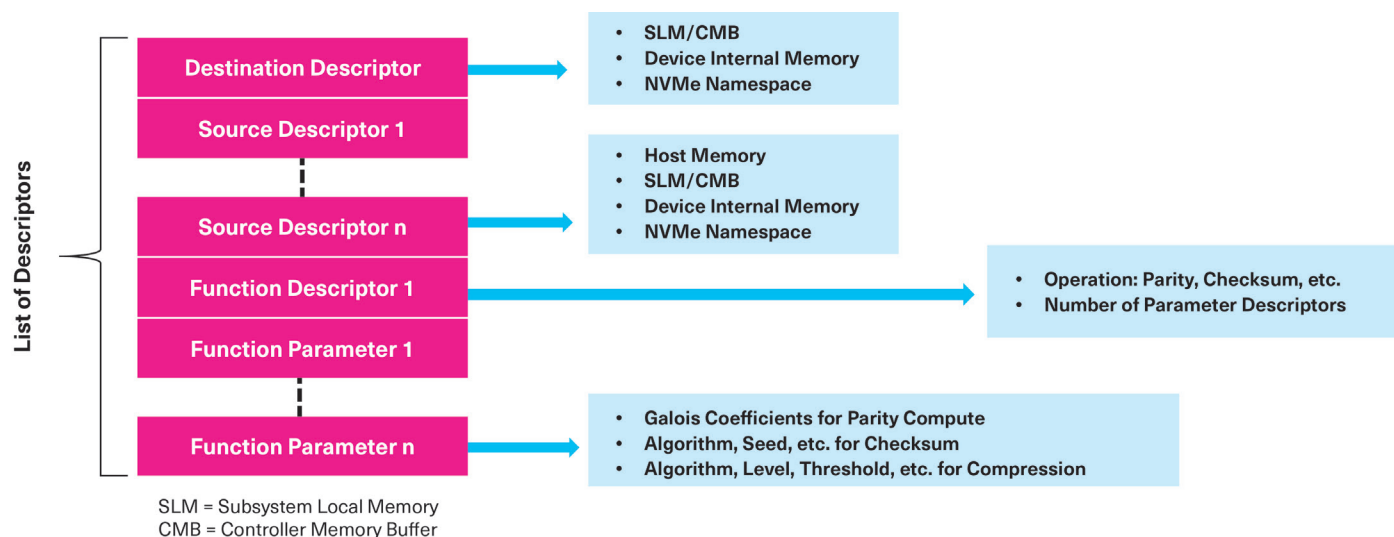


Figure 2: Proposed FFC command descriptors
 Created by KIOXIA America, Inc.

Candidate Compute Functions

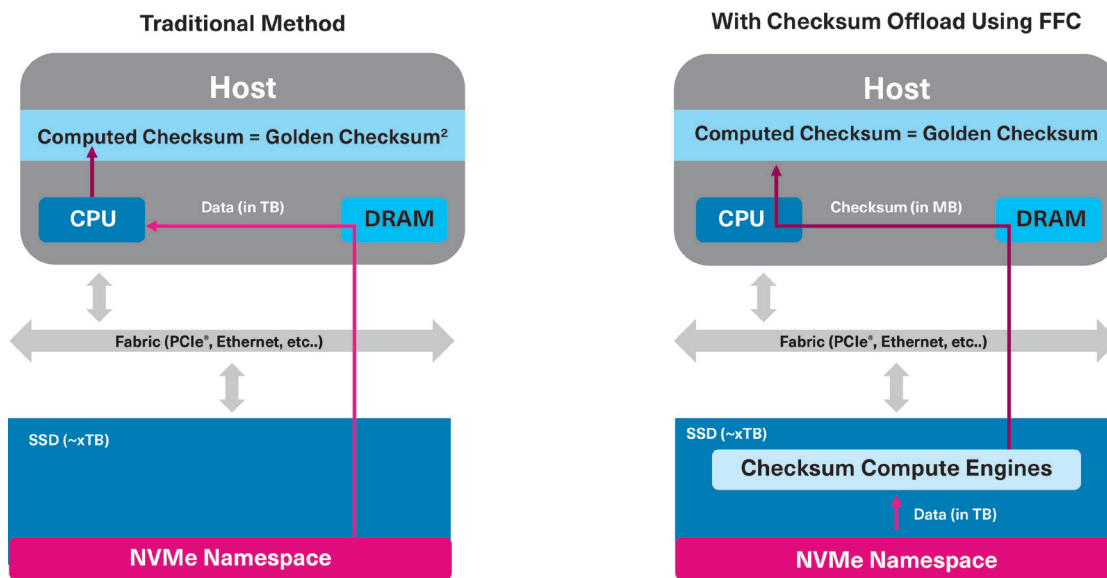
KIOXIA is evaluating several existing storage system-centric workloads as initial use cases for standardization in the SNIA® Computational Storage TWG. The compute functions under consideration include:

Compute Function	Usages
Checksum	Deduplication Object/File Signature Data Scrubbing Buffer Integrity
Parity Compute	Erasure Code Compare Data Scrubbing RAID Rebuild
Compression/Decompression	Storage/Network Optimization

These functions were proposed because existing enterprise storage systems are increasingly constrained with compute and memory requirements with increased underlying NVMe™ SSDs performance. It should be noted that the FFC framework is extendible to future compute offload requirements and is not limited to only these selected compute functions.

Example Application of Proposed Compute Functions: Checksum for Integrity Check

Many storage systems use checksums to check the integrity of stored data as a background operation. This background check is often postponed because the actual integrity check is typically expensive from a system resource utilization perspective. Using the FFC framework, checksum computations can be offloaded to SSDs (Figure 3) which compare data flow of an integrity check using the traditional method versus a checksum offload using the FFC framework:



TB = terabytes³ / MB = megabytes³

Figure 3: Integrity check data flow comparisons: Traditional method vs. FFC checksum offload
Created by KIOXIA America, Inc.

For a single SSD, one integrity check cycle using the traditional method - xTB (SSD size) of data passes through the network and CPU, and 2x xTB of data passes through DRAM. When checksum is offloaded using the FFC framework, only a few MB of data pass through the network, memory and CPU resulting in ~99% system resource (i.e., compute, memory and network) savings in addition to significant power savings.

KIOXIA has demonstrated FFC framework benefits with the parity compute function in:

- [RAID rebuilds](#)
- [Integrity checks through a proof-of-concept](#)

Summary

The KIOXIA proposed FFC framework enables a very efficient and generic method for SSDs to enable compute offload techniques with the following design features:

- *Better utilization of system resources primarily compute, memory, network and storage*
- *Enables scalable, secure⁴, adoptable and interoperable solutions*
- *Provides a generic offload solution designed to meet future offload requirements*

KIOXIA also intends to collaborate with the SNIA[®] Computational Storage TWG to enhance the FFC framework by supporting these future workloads:

- *Key-Value (KV) cache offload⁵ with compression, compaction and quantization*
- *Similarity searches performed on the device to reduce data movement on the network*
- *The KIOXIA proposed FFC interface is in an early design stage, and as such, the SNIA Computational Storage TWG is receptive to new ideas and suggestions. KIOXIA is open to collaboration for reviewing the FFC design or for receiving new compute offload ideas to incorporate in the FFC design.*

FOOTNOTES:

¹ Near-data computing (NDC) is an architectural design that places processing hardware directly next to data storage or memory. Instead of the traditional method of moving raw data across a system to a central CPU or GPU, NDC moves the computation to where the data resides.

² A computed checksum = golden checksum means that the data received, downloaded or read matches the original source file completely and has not been altered or corrupted.

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 1Gbit = 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 FFC does not allow executable code to be downloaded from the host to an SSD that can alter data stored on the drive and thereby enables a higher level of data security.

⁵ A Key-Value (KV) cache workload in a Large Language Model (LLM) inference is a read-dominant data pattern where intermediate attention keys and values are stored during the prefill phase and repeatedly fetched during sequential token generation.

TRADEMARKS:

NVMe and NVM Express are registered or unregistered trademarks of NVM Express, Inc. in the United States and other countries. PCIe is a registered trademark of PCI-SIG. SNIA is a registered trademark of the Storage Networking Industry Association. 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 tech 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.