



# AI Inference Storage Powered by CSAL on Bluefield-3 DPUs

Wayne Gao, Bo Li, Mariusz Barczak, Sarika Mehta,  
Scott Wentz, Kapil Karka

**SOLIDIGM**

## Abstract

The rapid development of Large Language Models (LLMs) has posed an urgent demand for efficient Key-Value cache (KV cache) management to optimize AI inference performance in multi-turn conversations and long-context processing. This paper explores the integration of the Cloud Storage Acceleration Layer<sup>1</sup> (CSAL) with the BlueField-3 Data Processing Unit<sup>2</sup> (DPU), addressing the storage challenges of KV Cache in high-concurrency AI workloads through a flexible storage architecture.

This architecture combines high-capacity SSDs and high-performance SSDs to deliver system level configuration flexibility. Pairing CSAL with one or more DPU, it dynamically allocates data between high-capacity SSD, high-performance SSD, or cache-layer storage resources based on the specific needs of different AI workload stages such as data preparation, training, inference, and retrieval-augmented generation (RAG), thereby significantly improving throughput and substantially reducing Time-to-First-Token (TTFT).

Innovative technologies like CacheBlend<sup>3</sup> and AttentionStore<sup>4</sup> reduce computational overhead by enhancing KV Cache reuse and prefetching capabilities, while maintaining response quality. CSAL further improves storage efficiency by reducing SSD storage wear and enhances the ability to manage multiple data streams, providing support for scalable, cost-effective AI inference. This architecture seamlessly integrates with frameworks such as vLLM<sup>5</sup> and SGLang,<sup>6</sup> supporting advanced applications like RAG, and delivering high performance and resource efficiency for enterprise-grade AI deployments.

---

<sup>1</sup> <https://www.solidigm.com/products/software/csalm.html>]

<sup>2</sup> <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf>

<sup>3</sup> <https://arxiv.org/abs/2405.16444>

<sup>4</sup> <https://arxiv.org/html/2403.19708v2>

<sup>5</sup> <https://docs.vllm.ai/en/latest/index.html>

<sup>6</sup> <https://docs.sglang.io/>

## Table of Contents

1	AI inference KV Cache storage industry trend.....	3
1.1	CacheBlend analysis uncovers the need for high-capacity NVMe SSD.....	3
1.2	AI inference needs hierarchical KV Cache storage with prefetch.....	5
2	Solidigm CSAL flex storage solution .....	9
2.1	High level overview for AI Inference software stack with CSAL and Bluefield-3 .....	9
2.2	Disk I/O requirements in different stages of LLM models.....	10
2.3	DPU-native flexible storage solution with CSAL .....	10
3	AI Inference Storage Reference Architecture .....	12
3.1	AI inference KV Cache ecosystem .....	12
3.2	AI tiered storage reference stack for AI inference with Bluefield-3.....	13
3.3	CSAL cache solution advantage for KV Cache stack.....	14
4	Cloud Storage Acceleration Layer Architecture .....	15
4.1	CSAL Distributed FTL .....	17
5	Performance Evaluation and Test Analysis.....	19
5.1	Performance of External KVCache Storage .....	19
6	CSAL KV Cache interface and AI industry partnerships .....	21

## 1 AI Inference KV Cache Storage Industry Trend

The need for long-context processing and multi-turn dialogs using LLMs has positioned the use of KV Cache as a pivotal element for efficient AI inference. Industry insights from a paper presented at [FAST 2025](#) indicates that external KV Cache storage significantly reduces computational overhead by reusing cached key-value (KV) pairs, eliminating redundant processing of static input tokens during decoding<sup>7</sup>. In high-concurrency workloads with large input prompts, inference frameworks that recompute KV Cache for unchanged token sequences suffer from severe inefficiencies, driving up latency (e.g., Time to First Decoded Token) and imposing massive memory demands.

In a 70B parameter model, each token contributes approximately 2.56 MB of key-value pairs. Defining a context window of one million input tokens can necessitate over 2.5TB of memory in FP16 precision. To address these escalating demands, industry leaders have pioneered the Hierarchical KV Cache Store design concept, harnessing local SSDs and network storage within a disaggregated prefill-decoding architecture.

Paired with asynchronous KV Cache management and high-bandwidth transfer engines (e.g., 50 GB/s via RDMA), this solution achieves up to a 525% throughput improvement,<sup>8</sup> as demonstrated by cutting-edge inference platforms.

A paramount challenge in KV Cache management arises from its relentless capacity growth, placing unprecedented strain on storage resources, especially during the decoding phase. Each newly generated token triggers the computation of a new KV Cache entry, which is appended to the historical KV Cache and fully utilized in subsequent decoding steps, exponentially increasing memory needs.

In persistent conversational settings with high user volumes, memory and storage consumption growth is constrained only by the inference framework's context window, as decoding continuously generates tokens in response to user inputs.

For example, 1,000 users each generating 10,000 tokens daily could inflate the KV Cache by 25.6 TB per day in FP16 precision, accumulating an overwhelming 9.3 PB annually, posing a critical capacity bottleneck. Even high-capacity solutions like Solidigm™ SSDs, offering over 100s of terabytes, fall short of meeting this explosive demand, necessitating advanced KV Cache lifecycle management strategies to ensure sustainability.

### 1.1 CacheBlend Analysis Uncovers the Need for High-capacity NVMe SSD

RAG enhances LLMs by integrating multiple retrieved text chunks into prompts, providing critical context for high-quality responses. However, the prefill phase—computing the KV Cache for long inputs—incurrs significant latency, especially in long-context scenarios. For example, prefilling a 4,000-token input on Llama-70B takes ~6 seconds on an A40 GPU, causing high TTFT delays and throughput bottlenecks. Reusing precomputed KV Cache data mitigates prefill overhead. Existing methods like prefix caching, limited to the first chunk, and full KV reuse, ignoring cross-attention between chunks, compromise response quality.

---

<sup>7</sup> <https://www.usenix.org/system/files/fast25-qin.pdf>

<sup>8</sup> [Mooncake: A KVCache-centric Disaggregated Architecture for LLM Serving](#)

## CacheBlend: A Novel Approach to KV Cache Fusion

CacheBlend overcomes these limitations by reusing KV Cache for all prompt chunks, not just the prefix, while maintaining generation quality. It selectively recomputes the KV Cache for ~15% of tokens to restore critical cross-attention, which full KV reuse overlooks as Figure 1-1 indicates. CacheBlend reduces TTFT by 2.2x to 3.3x and boosts throughput by 2.8x to 5x compared to full KV recompute, with minimal quality loss.

For instance, recomputing 15% of tokens for Yi-34B model<sup>9</sup> with a 4K-token context takes ~3ms per layer, while loading a layer's KV Cache from a gen5 NVMe SSD takes ~8ms. Pipelining KV Cache loading and selective re-computation eliminates additional latency.

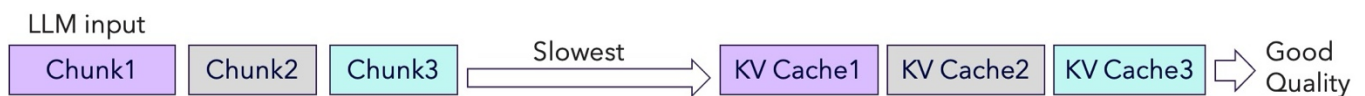


Figure 1-1.a Full KV re-compute, Prefill on entire input

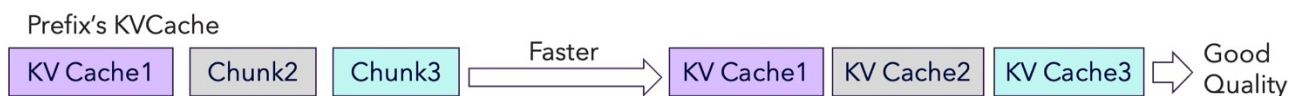


Figure 1-1.b Prefix caching, only reusing prefix's KV Cache

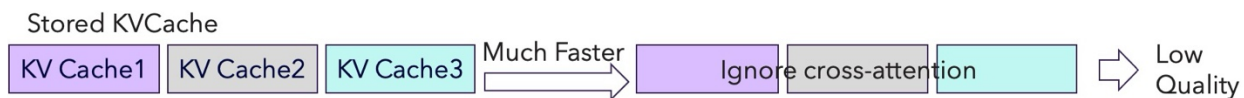


Figure 1-1.c Full KV reuse, Reusing all KV Cache data, ignoring cross-attention

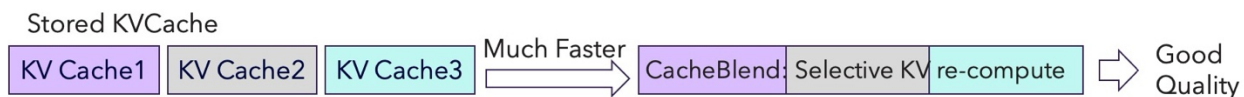


Figure 1-1.d CacheBlend, Reusing all KV Cache but recomputing small fraction

Note: Figures & data from CacheBlend paper, refer to footnote 3

### Key Takeaways:

- **Prompt chunk caching:** Unlike prefix caching, CacheBlend caches all RAG prompt chunks, increasing storage needs. High-capacity Solidigm™ NVMe SSDs are essential to store the millions of sequences represented, e.g., 61 million at 2MB each.
- **Decoding continuously generates new KV Caches:** During decoding, each token generated from user queries creates new KV Caches, which are stored as historical data for future reuse in multi-turn dialogues, necessitating scalable storage solutions like [Solidigm D5-P5336](#) 122TB high-capacity NVMe SSDs to accommodate the ever-growing KV Cache history. For each new decode token, KV Cache becomes a vital part of the historical record, enabling the development of more sophisticated AI applications for enterprise and scientific

<sup>9</sup> <https://vstorm.co/glossary/yi-34b/>

purposes. Compared to the human brain's memory capacity, the physical memory in GPUs remains very limited. By leveraging KV cache techniques to store additional knowledge and content on SSDs, the system can theoretically achieve near-unlimited expansion, leading to more accurate and effective reasoning and decision-making.

- **FullKV accuracy:** FullKV, retaining complete KV Cache history without compression, achieves superior accuracy in long-context tasks (e.g., LongBench Needle-In-A-Haystack<sup>10</sup>), ensuring optimal information retention for contexts up to 128k tokens, as shown in Figure 1-2 from the DynamicKV paper.<sup>11</sup>

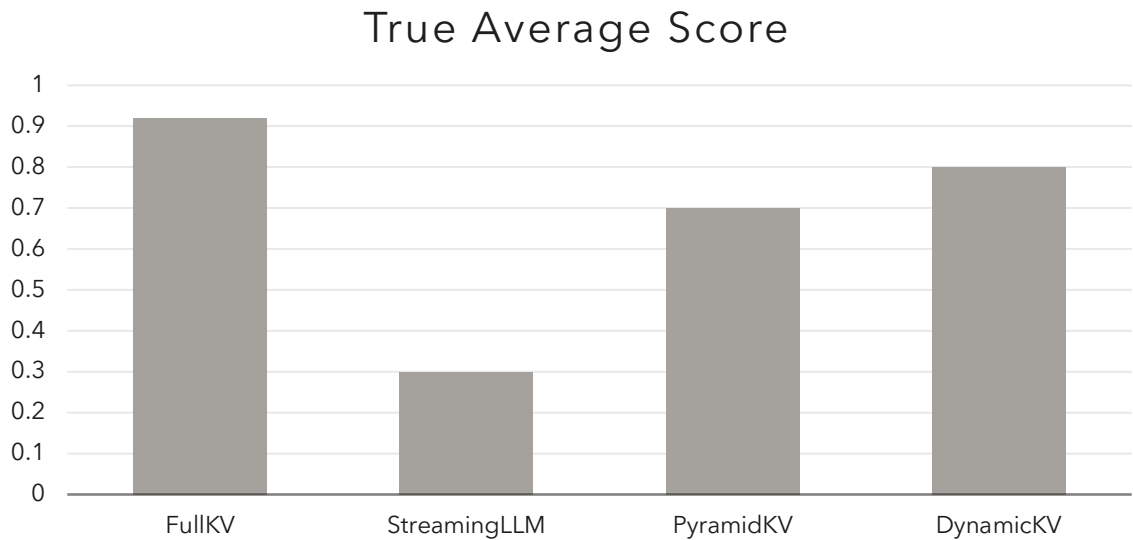


Figure 1-2. LongBench Needle-In-A-Haystack KV Cache comparison

## 1.2 AI Inference Needs Hierarchical KV Cache Storage with Prefetch

AttentionStore<sup>12</sup> enhances LLM inference for multi-turn conversations and long-context workloads by reusing KV Cache across turns. Its hierarchical KV caching system spans GPU HBM, host DRAM, and SSDs, interconnected via RDMA for high-speed data transfer. With layer-wise prefetching and asynchronous saving, AttentionStore reduces TTFT by up to 87%, boosts prefilling throughput by 7.8x for multi-turn dialogues, and cuts inference costs by 70%. For long-context tasks of 28K-token sequences, it slashes TTFT by 95% and increases throughput by 22x.

<sup>10</sup> <https://arxiv.org/abs/2308.14508>

<sup>11</sup> [DynamicKV: Task-Aware Adaptive KV Cache Compression for Long Context LLMs](#)

<sup>12</sup> [AttentionStore: Cost-effective Attention Reuse across Multi-turn Conversations in Large Language Model Serving](#)

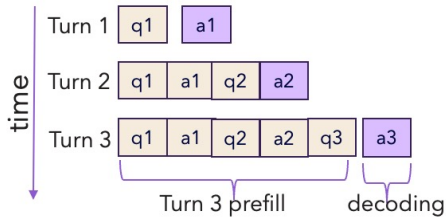


Figure 1-3.a Re-computation

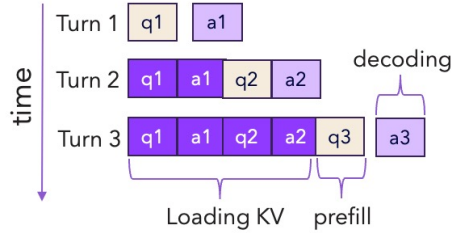


Figure 1-3.b AttentionStore

### Multi-layer KV Cache Design

AttentionStore addresses KV Cache storage demands for multi-turn conversations, represented by 73% of ShareGPT<sup>13</sup> and long-context workloads that can be viewed as having 30% exceed 4K tokens workloads using:

- Hierarchical Storage:** Utilizes GPU HBM (80 GB per A100), host DRAM (512-1024GiB), and SSDs (30-122 TB) to store KV Cache data overcoming HBM's limited capacity (e.g., LLaMA-65B fills 190 GB HBM around 12 seconds). SSDs provide vast storage, while DRAM acts as a faster intermediary, achieving cache hit rates of 90% to 92% across models like LLaMA-13B and Falcon-40B. Figure 1-4 shows high-level architecture of AttentionStore spanning over HBM, host memory and NVMe SSD.

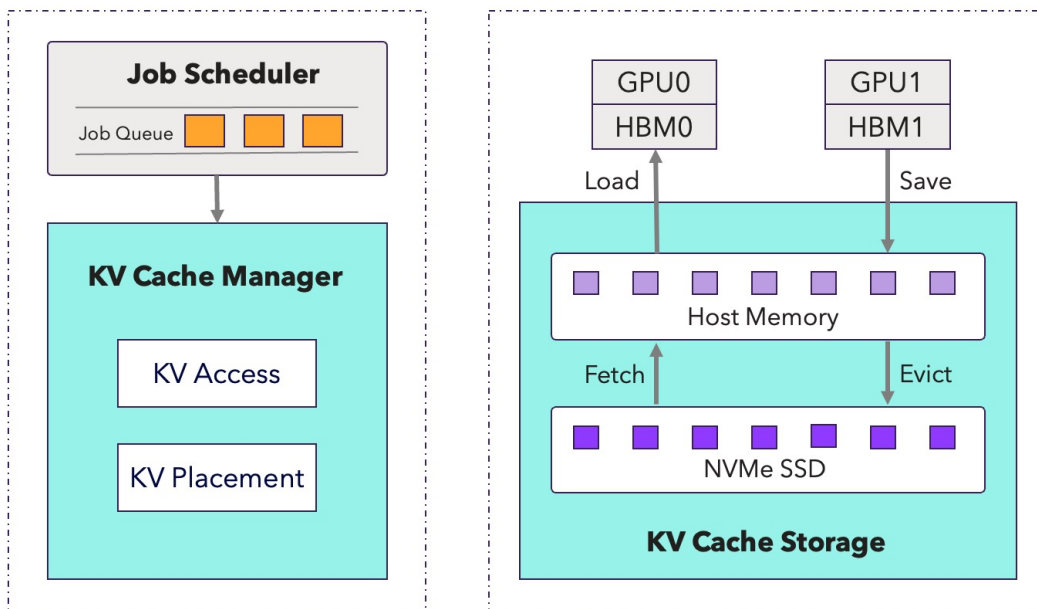


Figure 1-4. High-level AttentionStore Architecture

<sup>13</sup> : [ShareGPT Datasets - a bunnycore Collection](#)

- Scheduler-aware placement:** Prefetches KV Cache from SSDs to DRAM using job scheduler hints, ensuring low-latency access. Eviction prioritizes fewer reusable caches, improving hit rates by 27% to 31% over LRU/FIFO. The total data cost includes the cost of DRAM memory and SSDs. It accounts for 9% to 16.4% of total inference cost for models like LLaMA-13B and Falcon-40B and is far below GPU cost that account for rest of the infrastructure cost.

## Prefetch and Overlap Mechanisms

AttentionStore minimizes access overheads for historical tokens (up to 99% of prefilling time):

- Layer-wise Pre-loading:** Overlaps KV Cache loading time from DRAM to HBM with inference, reducing TTFT by 61% using a 15-layer read buffer. A customizable HBM buffer eliminates job gaps when loading time exceeds computation as Figure 1-5 illustrates.

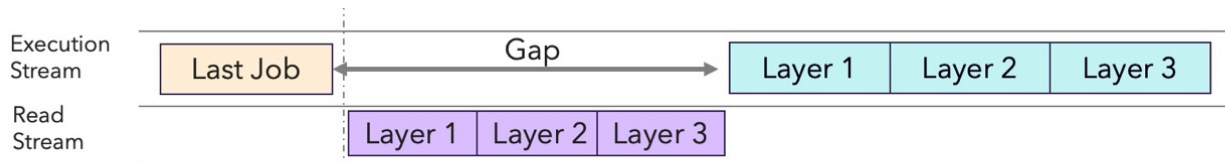


Figure 1-5.a Baseline: KV Cache loading without concurrent operations



Figure 1-5.b Layer-wise pre-loading without buffer

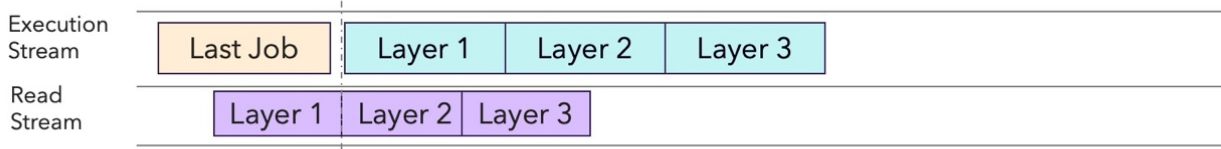


Figure 1-5.c Layer-wise pre-loading with buffer

- Asynchronous saving:** Saves KV Cache layer-by-layer from HBM to DRAM/SSDs during prefilling/decoding, cutting execution time by 13% to 15% for 1K-1.6K-token prompts.
- Scheduler-aware fetching:** Prefetches KV Cache to DRAM based on job queues, ensuring low-latency access for random conversation requests as indicated in Figure 1-6.

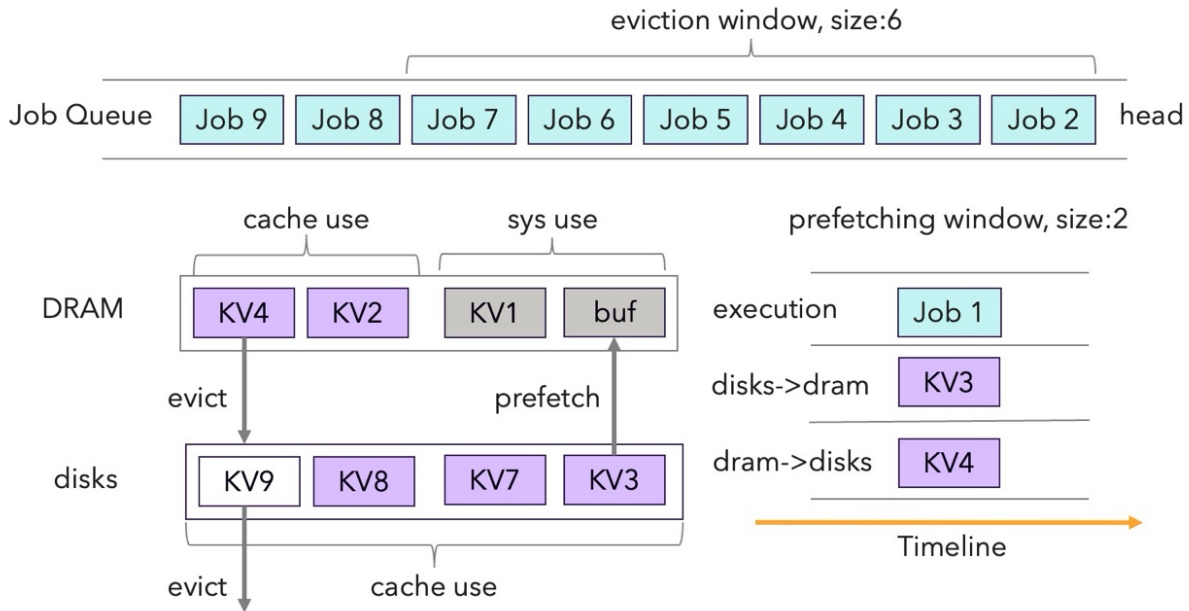


Figure 1-6 prefetch pipeline

### Key Takeaway

AttentionStore's scheduler-aware prefetching uses prompts and documents to prefetch KV Cache from SSDs to DRAM, achieving a 99.6% hit rate. By eliminating historical token recomputation, it saves GPU computing power while maintaining decode speed for multi-turn and long-context LLM workloads.

## 2 Solidigm CSAL Flex Storage Solution

### 2.1 High Level Overview for AI Inference Software Stack with CSAL and Bluefield-3



Figure 2-1 An End-to-End View of the LLM Inference Ecosystem with CSAL and Bluefield-3

Scalable LLM inference depends on efficient KV Cache management and vector database (VectorDB) integration to reduce redundant computations and enable advanced applications such as RAG, Knowledge-Augmented Generation (KAG), and Context-Augmented Generation (CAG). As shown in **Figure 2-1**, LLM inference ecosystem integrates the Hierarchical KV Cache store with VectorDBs, leveraging high-performance inference engines like those similar to vLLM and SGLang for model execution.

The Hierarchical KV Cache Store serves as an external storage system for caching key-value pairs, while VectorDBs manage high-dimensional embeddings for quick context retrieval, supported by efficiency data access mechanisms. These inference engines interact with the Hierarchical KV Cache Store to optimize KV Cache handling, enhancing overall inference efficiency. This approach supports scalable and responsive LLM deployment in practical scenarios in combination with application orchestration and monitoring systems.

## 2.2 Disk I/O Requirements in Different Stages of LLM Models

The storage subsystem is a critical component in the application of large-scale AI models. It plays an integral role across every stage of the model lifecycle. The storage subsystem must manage a wide range of I/O patterns:

- Data ingestion during the preprocessing phase
- Frequent checkpoint saving in the training process
- Rapid model loading for inference
- Random small-block data access required for KV Cache and RAG

These include both sequential read/write operations for large data blocks and random read/write for smaller blocks. This diversity places significant demands on storage system design, requiring a careful balance of performance, latency, and throughput to meet the needs of AI workloads effectively.

During the implementation process most system integrators procure or deploy tailored storage solutions to address the specific demands of different stages in large model development.

For example, the data cleansing phase requires efficient sequential reading of large data blocks, while the checkpoint saving phase necessitates rapid sequential writing of large data volumes, leading to the selection of storage solutions optimized for high-performance large-block read/write operations.

In contrast, the RAG and KV Cache phases demand the opposite, emphasizing fast read/write capabilities for small data blocks and requiring storage solutions with superior random I/O performance.

As a result, AI data centers often feature multiple storage systems designed for diverse purposes, which not only drives up costs but also increases maintenance complexity. Could there be a more flexible solution to tackle these challenges?

## 2.3 DPU-native Flexible Storage Solution with CSAL

CSAL was designed to optimize features associated with adopting high-capacity SSDs such as wear prevention and lower write performance. By utilizing a high-performance SSD as a cache disk, CSAL temporarily stores incoming data in the cache, opportunistically consolidating small data blocks into larger ones before their deferred writing to the high-capacity disk.

This approach effectively mitigates issues related to lower small IO write performance and write amplification. As illustrated in the figure, CSAL also incorporates features at the cache layer, including data protection via RAID support, IO traffic shaping, compression, and deduplication, further enhancing the efficiency and reliability of the storage system.

With the widespread adoption of AI applications and the need to better serve diverse application requirements and frequent changes in data I/O patterns, we have made certain modifications to

CSAL. These include the integration of BlueField-3 and NVMe-oF technology stacks, enabling the delivery of storage services with greater flexibility.

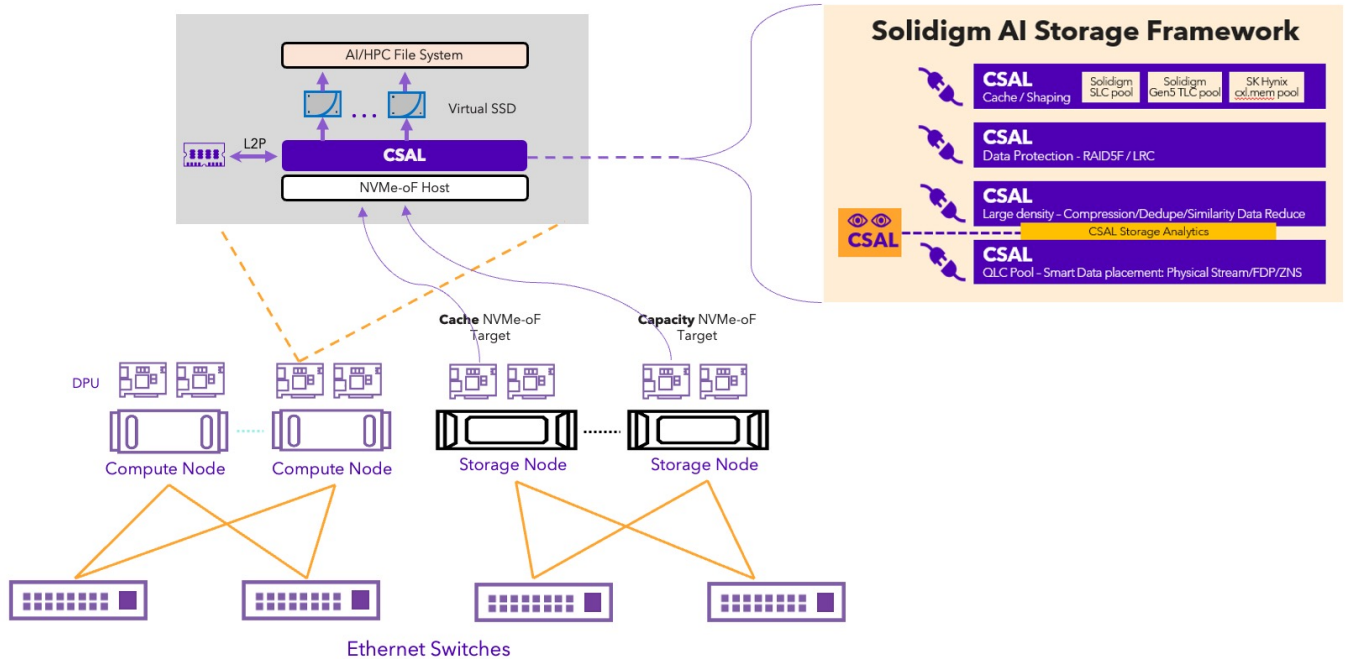


Figure 2-2. Solidigm Flexible Storage Solution for AI

As illustrated in Figure 2-2 above, the storage nodes are equipped with a significant number of high-performance SSDs and high-capacity SSDs. Both the compute nodes and storage nodes are fitted with BlueField-3 cards. All SSDs leverage these BlueField-3 cards to provide various NVMe target services over the network.

AI workloads operate on compute nodes, with storage requirements varying across different stages. Customers can leverage infrastructure management software to dynamically configure storage, ensuring they meet the diverse needs of each phase.

During the inference phase, the integration of RAG services necessitates storage space for a vector database, where the I/O pattern shifts to random read/write operations on small data blocks. Through the virtualized infrastructure management software, we can control CSAL to create a virtual disk on the BlueField-3 of the compute node, comprising a high-performance SSD cache target paired with a high-capacity SSD target.

This virtual disk is then allocated for use by the vector database. Given the small-block random I/O nature of the vector database, the high-performance cache target efficiently manages I/O alignment best suited for large writes by consolidating data into larger blocks (16/64K) before writing to the backend high-capacity SSD target.

In another scenario, during the data cleansing phase of the training process on a compute node, the management software can direct CSAL to create a virtual disk on the node's BlueField-3, composed

of two NVMe-oF target endpoints. Since this stage primarily involves sequential writing of large data blocks, a cache disk is unnecessary. Direct writes to the high-capacity SSD are enabled optimizing write performance.

Through the two examples above, we can see that, with the integration of BlueField-3 and NVMe-oF protocols, CSAL has evolved into a cross-network storage solution, offering greater flexibility and efficiency. Customers can dynamically configure NVMe storage setups based on application needs and once completed, quickly release storage resources for other applications, significantly improving resource utilization.

### 3 AI Inference Storage Reference Architecture

This section details the features of AI inference frameworks and distributed KV Cache stores and proposes a cost effective and efficient AI inference storage reference architecture to meet the diverse demands of AI applications with optimized performance and utmost flexibility in storage infrastructure.

#### 3.1 AI Inference KV Cache Ecosystem

The Table 3-1 lists most popular AI inference frameworks and high-performance frameworks, their key features and challenges.

Framework	Key Features	Challenges
vLLM	<ul style="list-style-type: none"> <li>- PagedAttention: Block-based KV Cache with hash-based prefix caching</li> <li>- Continuous batching for high concurrency</li> <li>- Modular block management (block pool, free queue, hash table)</li> </ul>	<ul style="list-style-type: none"> <li>- Less optimized for prefix-heavy workloads</li> <li>- Potential hash collisions in prefix caching</li> </ul>
SGLang	<ul style="list-style-type: none"> <li>- RadixAttention: Radix tree for prefix caching</li> <li>- Cache-aware scheduling for prefix reuse</li> <li>- Disaggregated inference (GPU/CPU support)</li> </ul>	<ul style="list-style-type: none"> <li>- Radix tree complexity (O(L) operations)</li> <li>- Higher scheduling overhead</li> </ul>
Ollama	<ul style="list-style-type: none"> <li>- Lightweight, local-first inference</li> <li>- Simplified KV Cache management (in-memory, no advanced caching)</li> </ul>	<ul style="list-style-type: none"> <li>- Lacks advanced KV Cache optimizations</li> <li>- Not designed for distributed or high-concurrency scenarios</li> <li>- Limited scalability for enterprise use</li> </ul>

Table 3-1: inference frameworks

KV Cache Store	Key Features	Integration with Frameworks
External KV store	<ul style="list-style-type: none"> <li>o Distributed KV Cache store</li> <li>o Cross-node cache reuse</li> <li>o Extended cache pools</li> <li>o Prefill-decode disaggregation</li> </ul>	<ul style="list-style-type: none"> <li>✓ vLLM: Full via LMCache</li> <li>✓ SGLang: In progress (via KVConnectorBase)</li> </ul>
LMCache	<ul style="list-style-type: none"> <li>o Disk-based KV cache</li> <li>o storageCacheGen for encoding,</li> <li>o CacheBlend for composition</li> <li>o 100x more storage capacity</li> </ul>	<ul style="list-style-type: none"> <li>✓ vLLM: Seamless integration</li> <li>✓ SGLang: Exploratory (via KVConnectorBase)</li> </ul>

Table 3-2: Distributed KV Cache store

Table 3-2 shows the most popular KV Cache stores for the inference framework along with their pros and cons.

### 3.2 AI Tiered Storage Reference Stack for AI Inference with Bluefield-3

The Solidigm reference design shown in Figure 4-1 was developed with external parties related to the Bluefield-3 implementation. Let's walk through the steps indicated in Figure 4-1 showing the end-to-end inference design.

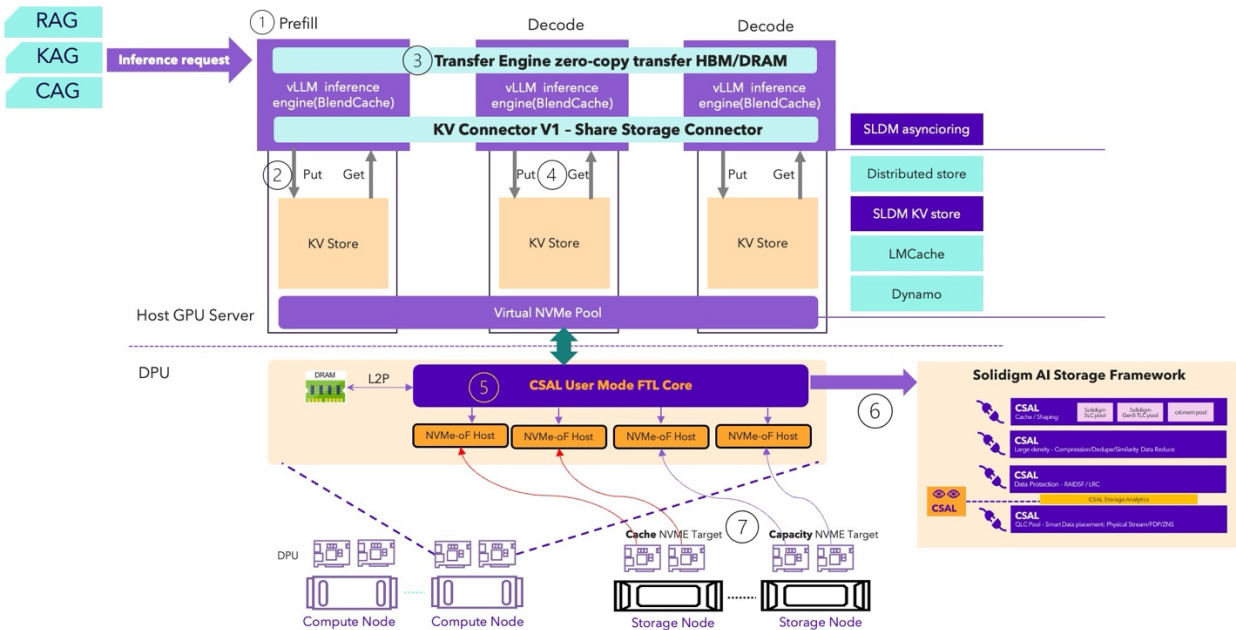


Figure 3-1 Solidigm AI inference SW stack reference design with CSAL

1. **Prefill:** Inference requests are initiated and sent to the inference engine for initial processing (prefill stage) which computes the KV Cache for the input prompt.
2. **Put/get operations:** The inference engine interacts with an external KV Cache store via put/get operations to store and retrieve KV Cache tensors, optimizing data access efficiency.
3. **Zero-copy transfer:** A high-performance transfer engine performs zero-copy transfers to move KV Cache data between different nodes in CPU DRAM.
4. **External distributed KVStore management:** The Hierarchical distributed KV Cache Store, a high-performance external storage system, manages KV Cache storage and retrieval. It integrates with advanced caching and optimization mechanisms from vLLM to support scalable inference.
5. **CSAL user-mode FTL core:** CSAL user-mode Flash Translation Layer (FTL) core facilitates communication between compute nodes (BlueField-3) and storage nodes (NVMe targets), ensuring efficient data cache and placement.<sup>14</sup>

<sup>14</sup> <https://www.solidigm.com/products/technology/dpu-native-distributed-flexible-storage.html>].

6. **Cache and capacity NVMe target:** NVMe targets provide scalable storage capacity, supporting the Solidigm AI Storage Framework for efficient data access and management during inference.

The SLDMKVCacheStore<sup>15</sup> offers a streamlined and scalable approach and provides sample code to build an external KV Cache store for vLLM. By leveraging the Hierarchical KV Cache Store concept and vLLM's KVConnectorV1 and KVTransferConfig interfaces, developers can create high-performance, distributed storage systems that enhance LLM inference efficiency.

This end-to-end design empowers the deployment of responsive, cost-effective AI applications, from conversational agents to advanced RAG workflows, in diverse real-world scenarios. In addition to the KV Cache implementation in KVStore, the industry has also adopted shared file system approaches, where each KV Cache layer is stored as a file. Solidigm developed the SLDM asynciouring module for this scenario, innovatively combining the advantages of asynchronous IO with coroutines to facilitate asynchronous IO programming for developers. Internal bench testing shows promising results of over 8x single-core IO efficiency in Python vLLM. We encourage partners and customers to reach out to Solidigm to gain access to this module.

### 3.3 CSAL Cache Solution Advantage for KV Cache Attack

#### Optimizing KV Cache Management with RAID0 Gen5 NVMe SSDs

In LLM inference, the KVC ache stores attention states for efficient token generation, but its size (e.g., 2MB/sequence) creates capacity challenges for long-context and multi-turn workloads.

Offloading the KV Cache to storage frees GPU VRAM (40GB to 80GB on A100 GPUs) but demands high bandwidth to match DRAM's 60 GiB/s and CPU-to-NVMe Gen5's 10 GB/s. Combining multiple Gen5 NVMe SSDs addresses this challenge by scaling both bandwidth and capacity.

A single [Solidigm D7-PS1010](#) Gen5 NVMe SSD delivers ~9.4 GB/s sequential writes. Six drives achieve ~82.9GiB/s, outperforming the 60 GiB/s DRAM cap and enabling KV Cache eviction and retrieval without bottlenecking inference. Solidigm D5-P5336 122TB high-capacity SSD store KV Cache for ~61 million sequences (122TB ÷ 2MB), ideal for enterprise-scale multi-turn or long-context tasks. High-capacity SSD 7GiB/s read performance and 16KB indirection unit optimize read-intensive decoding, with write-heavy eviction limited to initial offloads.

#### Benefits and Considerations

- **Bandwidth matching:** 6xGen5 SSDs (~82.9GiB/s) support high-throughput KV Cache eviction/retrieval preventing inference bottlenecks.
- **Density scale-out:** 122TB high-capacity scales to millions of sequences with strong read performance for decoding.
- **GPU VRAM savings:** Offloading KV Cache maximizes VRAM for computation, supporting more requests or longer contexts.
- **CSAL software-defined SSD solution:** CSAL enables efficient KV Cache management by initially writing KVCache data to high-performance SSDs. During periods of workload idleness CSAL

---

<sup>15</sup> [SLDMKVCacheStore one demo showcase how to design one KVCache distribute store](#)

compacts this data, migrating it to the denser high-capacity SSD layer for long-term storage which optimizes both performance and capacity.

## 4 Cloud Storage Acceleration Layer Architecture

CSAL FTL has been successfully deployed across thousands of servers, demonstrating its scalability and practical applicability. It is hardware-agnostic, making it a flexible solution for a variety of data center architectures. It is built on top of SPDK and can be easily deployed and fully offloaded into BlueField-3. This platform enables storage software customization and enhancements in a way that is transparent to the user providing an innovative and flexible storage controller design.

CSAL delivers numerous features that allow customers to build an advanced storage stack (see Figure 4-1) to increase performance with a TCO solution that fits the application needs. Write shaping sequentializes user workloads to high-capacity SSD while leveraging both high sequential bandwidth and dense capacity. Write shaping provides scalable performance when running CSAL FTL on multiple CPU cores, which is a must feature when CSAL works with RAID volumes spanning over several drives.

To guarantee robust data redundancy, CSAL can be deployed on top of RAID5F volumes. RAID5F is RAID5 which accepts only full stripe writes. It gets rid of the RAID write hole exposure and avoids read-modify-write when the RAID parity needs to be recalculated and updated. Handling the generation of full-stripes and RAID functionality are managed by CSAL in a host-transparent fashion.

CSAL's read cache reuses write-shaping FTL logic to provide a client-side ultra-fast cache. It maintains cache device Write Amplification Factor (WAF) at the level of 1.0 and therefore delivers high performance and high quality of service (QoS) over traditional caching solution.<sup>16</sup>

In addition to the above features, users can benefit from robust and mature SPDK infrastructure itself. Users can tap into SPDK features and services to compose full stack storage solutions like leveraging NVMe over Fabric (RDMA/TCP) and vhost targets, using various storage device drivers and connectivity protocols (NVMe PCIe drive, initiators of NVMe-oF (RDMA/TCP), or io\_uring). The configured storage stack can be consumed by applications which use SPDK directly, or over block interface, or over fabrics.

---

<sup>16</sup> <https://www.snia.org/sites/default/files/2025-10/SNIA-SDC25-Barczak-Mehta-CSAL-with-Core-Scaling-RAID5F.pdf>

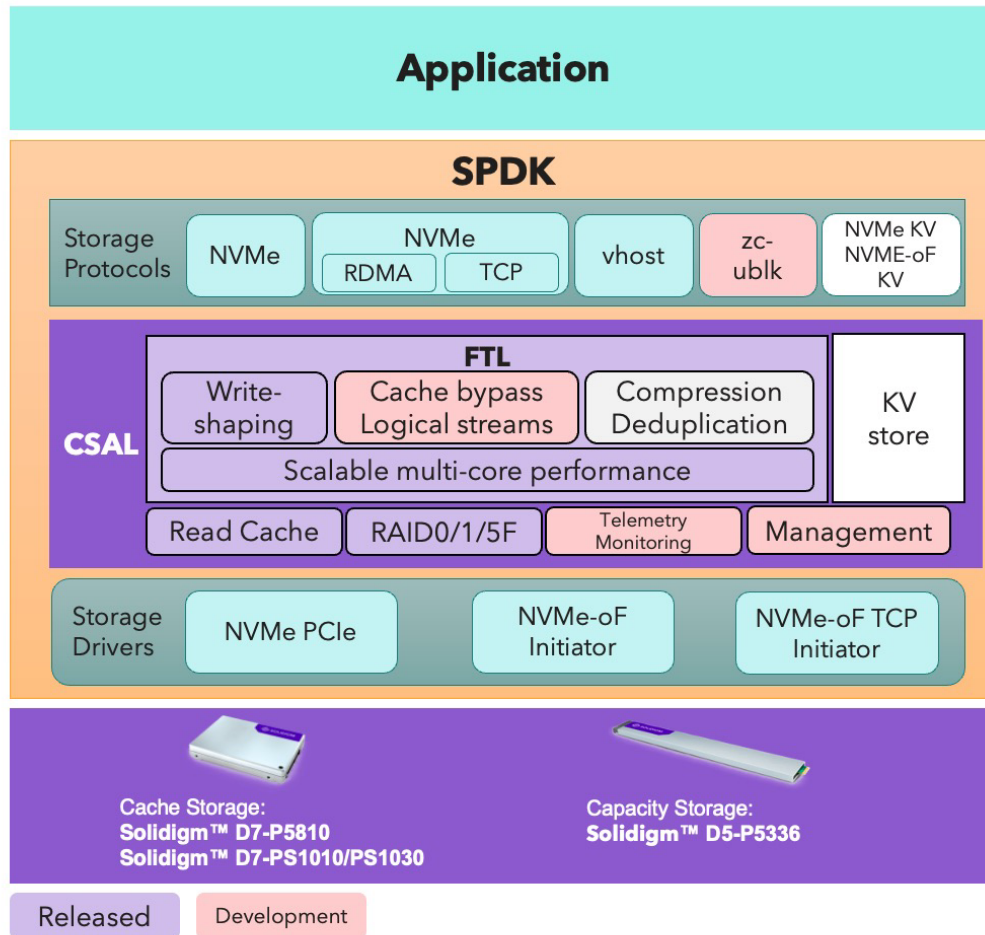


Figure 4-1 Cloud Storage Acceleration Layer

Deploying high-capacity SSD storage for AI and multi-tenant systems provides potential architectural design challenges. As drives grow to tens or hundreds of terabytes, it becomes harder to maintain consistent performance, endurance, and availability. high-capacity SSDs provide excellent density and attractive cost per terabyte, but in a scenario that needs these drives to serve small IO writes, the absence of traffic shaping in the host stack can impact the available endurance and impact the expected performance due to internal data management. Therefore, an effective caching solution would be ideal.

CSAL feature set elevates the management of high-density storage and provides even more advance features to address these challenges.

- Cache Bypass is a selective mechanism that allows certain workloads—such as sequential ingestion or bulk data movement—to skip the cache layer. This prevents cache pollution, stabilizes and scales bandwidth, and improves overall system efficiency under mixed I/O patterns.

- Logical Streams enable fine-grained classification and placement of data based on workload characteristics efficiently supporting multi-tenant environments.
- Compression and deduplication reduce backend write volume, extend high-capacity SSD endurance, and increase effective capacity—delivering more usable storage without raising cost.
- By combining years of FTL innovation with advanced caching techniques, CSAL delivers a high-performance KV store and cache solution that unlocks the full potential of a flexible storage architecture based on high-performance SSD and high-capacity SSD. It provides high capacity and line-rate performance capable of saturating the fastest NICs—while using minimal CPU and memory resources.
- Zero-copy ublk target exposes an SPDK block device back to the kernel, allowing legacy applications to use a POSIX filesystem without any modifications. The data and I/O path is based on a zero-copy mechanism, which guarantees no additional memory or CPU utilization overhead.
- CSAL will enhance this solution with the introduction of manageability, telemetry, and monitoring, enabling full operational visibility.

*Note: CSAL can be deployed in RAID or non-RAID configurations.*

## 4.1 CSAL Distributed FTL

Using CSAL's logical streams feature, CSAL distributed FTL mode is established. CSAL implements logical streams (Figure 4-2) by identifying incoming workload as sequential or random and placing the data into separate logical bands, preventing mixed-pattern data from landing in the same physical region of the SSD. Sequential streams bypass the cache, being written directly into their own dedicated bands (e.g., Stream 1 Band, Stream 2 Band) to maintain clean, large-block contiguity. Random streams, by contrast, are directed into the cache first; when the cache becomes full, only these random writes are moved into a compaction band. This prevents sequential and random data from becoming mixed, which is the key mechanism that reduces WAF and improves performance.

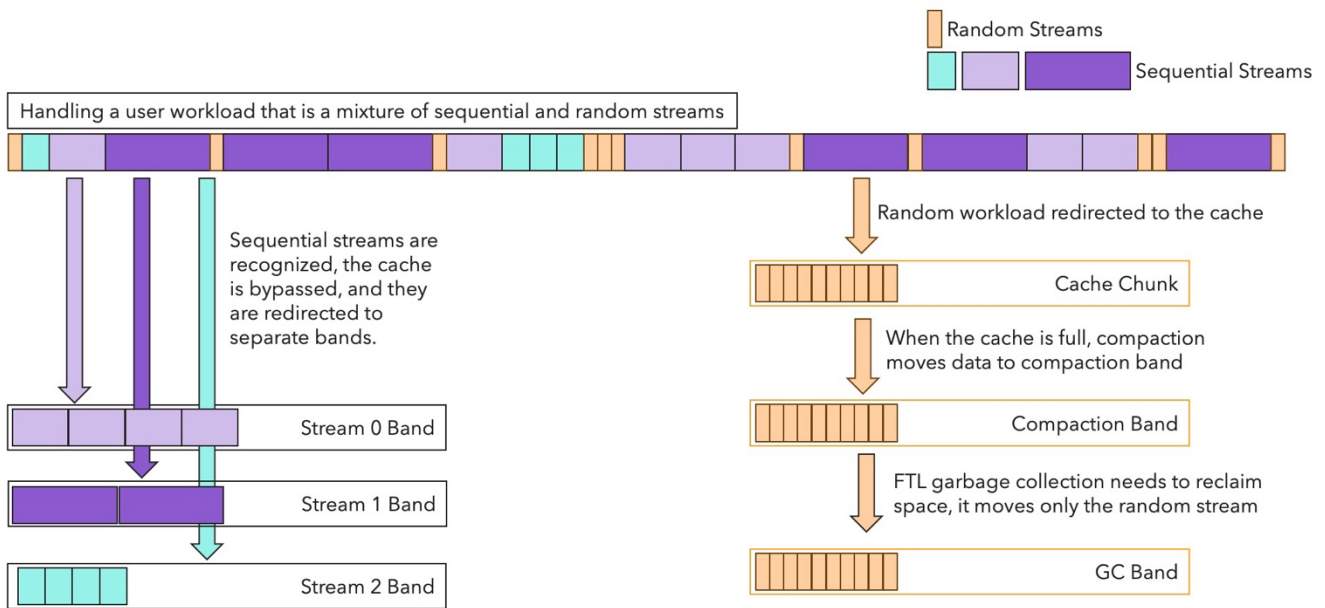


Figure 4-2 CSAL FTL logical streams data management

This implementation allows the host-based FTL to forward each logically separated stream into a corresponding drive-level placement ID, enabling the device to maintain this separation internally.

Finally, CSAL FTL addresses the increasing performance demands of AI deployments by introducing scalable solutions that leverage a set of various data-placement protocol supporting NVMe devices. Figure 4-3 illustrates how CSAL forwards logical streams to a pool of NVMe drives. These drives are grouped into a large-capacity logical volume (on the order of petabytes). Depending on workload requirements, CSAL FTL can allocate and utilize individual physical data placements across SSDs to ensure optimal tenant isolation, quality of service (QoS), data grouping, and performance scaling.

Summarizing, based on presented examples, CSAL as an innovative software defined storage controller design empowers storage systems with advanced features that optimize performance, reduce overhead, simplify data management, and extend storage endurance. CSAL distributed FTL, built on top of CSAL's user-mode FTL, can be deployed on a single DPU to present it as one massive SSD. Aggregation of all SSDs from all DPUS with CSAL distributed FTL, to create a scale-up, software-defined distributed namespace is under research.

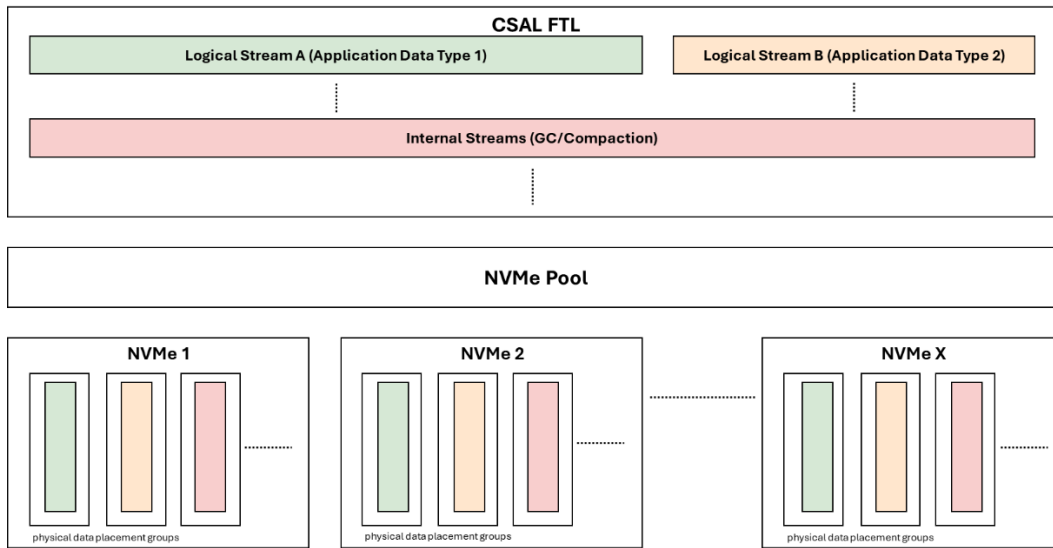


Figure 4-3 CSAL Distributed FTL

Note: CSAL Distributed FTL can be deployed using standard NVMe SSDs or drives that support data placement technologies in all cases, users benefit from CSAL's logical data-separation capabilities, which isolate sequential and random workloads to reduce write amplification and improve performance consistency.

## 5 Performance Evaluation and Test Analysis

### 5.1 Performance of External KVCache Storage

The key-value cache (KV Cache) is a critical technique for optimizing LLM inference by caching key and value vectors of previous tokens in decoder layers, thereby eliminating redundant computations.

Popular external KV Cache storage systems, built on the Hierarchical KV Cache Store design concept, leverage high-bandwidth interfaces and scalable storage tiers, such as local SSDs and networked storage, to achieve exceptional performance. These systems support high-concurrency workloads with read throughput reaching up to 40 GiB/s, facilitated by high-speed network interfaces (e.g., 400 Gbps per node).

CSAL distributed FTL performance evaluation for maximum performance capability with Solidigm Gen5 D7-PS1010 in RAID0 configuration:

Storage Server	
<b>BIOS Version</b>	1.09.00
<b>OS</b>	Fedora 31 (Server Edition)

<b>Kernel</b>	5.4.119-19-0010
<b>CPU Model</b>	AMD EPYC 9654 96-Core Processor
<b>NUMA Node(s)</b>	2
<b>DRAM Installed</b>	376GB
<b>Huge Pages Size</b>	2048 kB
<b>Drive Summary</b>	6x Solidigm Gen5 high-performance SSD D7-PS1010 7.68TB (SOLIDIGM SB5PH27X076T)
<b>CSAL</b>	25.08 - Solidigm Test version
<b>FIO</b>	3.39

Table 1: FIO and HW configuration

Test Case			6xD7-PS1010	Easily saturates 400Gbs RDMA
No	Workload	Parameters	GiB/s	
1	Large Read IO BW	4 read jobs: 128K/seq/r/qd128	82.9	7M IOPS
2	Large Write IO BW	4 write jobs: 128K/seq/w/qd128	50.7	
3	Small Read IO BW	4 read jobs: 4K/seq/r/qd128	39.4	
4	Small Write IO BW	4 write jobs: 4K/seq/w/qd128	27.8	

Figure 5-1: CSAL distributed FTL performance assessment - RAID0 w/Solidigm D7-PS1010

For a single LLM model, the KV Cache write workload can be simplified into two distinct streams:

- Large sequential writes for the KV matrix
- Small metadata writes for KV metadata

Using Solidigm's CSAL with its distributed FTL capabilities, we can shape these writes to achieve a Write Amplification Factor (WAF) of ~1, maximizing SSD endurance and performance.

In next-generation cloud ("neo-cloud") environments where multiple LLM models run concurrently, the resulting mixed write patterns can increase WAF. Fortunately, we can fine-tune distributed FTL to maintain WAF ~1 even under these demanding multi-model workloads.

For additional use-cases, detailed workload studies, and to explore collaboration opportunities please reach out at the email included at the end of the paper.

## 6 CSAL KV Cache Interface and AI Industry Partnerships

As this paper showcases, Solidigm works closely with the BlueField-3 team<sup>17</sup> and industry AI ISV partners to build the best performance and cost-effective AI inference infrastructure for storage.

The upcoming KV interface from CSAL introduces a user-space FTL (host-side Flash Translation Layer) that tightly integrates SSD capabilities and resources. This may enable significantly better support for the LLM inference software ecosystem – including LMCache, Dynamo, and the latest Inference Context Memory Storage Platform (**ICMSP**) framework built on top of Dynamo. Server and JBOF vendors can more easily and efficiently integrate this solution into leading inference platforms.

For any inquiries, please email: [dl\\_csal@solidigm.com](mailto:dl_csal@solidigm.com)

---

<sup>17</sup> <https://www.nvidia.com/gtc/session-catalog/sessions/gtc26-p81046/>

## Notices & Disclaimers

All product plans, roadmaps, specifications, and product descriptions are subject to change without notice. Nothing herein is intended to create any express or implied warranty, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, or any warranty arising from course of performance, course of dealing, or usage in trade. The products described in this document may contain design defects or errors known as "errata," which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your Solidigm representative or your distributor to obtain the latest specifications before placing your product order. For copies of this document, documents that are referenced within, or other Solidigm literature, please contact your Solidigm representative. All products, computer systems, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

© Solidigm. "Solidigm" is a trademark of SK hynix NAND Product Solutions Corp (d/b/a Solidigm). "Intel" is a registered trademark of Intel Corporation. Other names and brands may be claimed as the property of others.

Other trademarks may be the property of their respective owners.

Solidigm may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Solidigm reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase.

Performance results are based on testing as of dates shown in the configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Solidigm or Intel optimizations, for Solidigm or Intel compilers or other products, may not provide optimized performance to the same degree for non-Solidigm or Intel products. Solidigm or Intel technologies may require enabled hardware, software, or service activation. Your costs and results may vary. Solidigm does not control or audit third-party data. You should consult other sources to evaluate accuracy. Some results have been estimated or simulated using internal Solidigm analysis or architecture simulation or modeling, and provided to you for information purposes only. Any differences in your system hardware, software or configuration may affect your actual performance.