SEAGATE SK hynix

WHITE PAPER

Enabling inference at massive scale with hybrid storage for KV cache offloading.

Combining hard drives with SSDs extends Key-Value (KV) cache stores beyond the limits of memory.

Authors

Kyung Soo Lee, Principal Engineer, SK hynix
Jin Na Yang, Staff Engineer, SK hynix
Thomas Prohofsky, Principal Engineer, Seagate

Contents

Executive summary 02
Trends and opportunities 03
Challenges 04
Solution overview 05
Workload requirements and design goals 06
Solution architecture 07
Demonstration setup 08
LMBenchmark runs 10
LMBenchmark results 11
Future work 15
Conclusion 15

Executive summary

A tiered system for KV cache blocks can turn storage into a strategic advantage

The AI trial phase is over. As enterprises and cloud builders move pilots into production, inference workloads are scaling faster than the infrastructure designed to support them. Context windows are growing, driven by agentic workflows that involve multi-turn exchanges. With each inference request, a Key-Value (KV) cache accumulates the model's attention values from the tokens computed by the GPU. Efficiently storing KV cache values means the GPU can avoid recalculating prompts, reducing cycles and power consumption.

KV caches are typically stored in high-bandwidth memory (HBM) or dynamic random-access memory (DRAM), both of which are straining as cache sizes grow. Hyperscalers have already seen positive results implementing hybrid storage tiers of SSDs and hard drives for AI applications.[1] This multi-tiered system balances capacity, cost, and latency. By following their lead, enterprises and AI infrastructure builders can turn storage into a strategic advantage that enables them to expand the number of users leveraging inference and agentic services and extend the life of their sessions.

SK hynix, a leader in flash and SSD storage, and Seagate, a leader in hard drive storage, partnered together to explore the use of SSD and hard drive storage tiers for KV cache offloading over remote direct memory access (RDMA)-accelerated networking. In the proposed joint solution:

- NVM Express® over Fabrics (NVMe-oF) composability creates a flexible, high-performance, and cost-effective storage pool with GPU direct memory placement performance.
- SSDs serve latency-sensitive operations for disaggregated LLM inference, while hard drives provide mass capacity.
- The Storage Performance Development Kit (SPDK) and Linux Logical Volume Manager (LVM) enable dynamic allocation of NVMe-oF targets from resource pools composed of both types of drives.
- NVIDIA Dynamo KV Block Manager uses composed logical volumes to efficiently utilize GPU resources for multi-turn inference.
- Prefill workers build a distributed KV store, avoiding prompt reprocessing by decode workers, yielding faster Time to First Token (TTFT) and lower power consumption.

The paper concludes with performance projections, cost analyses, and drive type mix for infrastructure architects, platform teams, and CTO leadership as they build storage architectures.

About Seagate

Seagate (NASDAQ: STX) is a pioneer in mass-capacity data storage, accelerating the ability to harness the full value of data. Our portfolio of advanced storage solutions helps hyperscale cloud providers, enterprises, and consumers protect, create, and manage the data that powers their transformation and growth. For more than 45 years, Seagate has driven breakthrough innovations that bring sustainable, high-performance storage to the world at scale.

About SK hynix Inc.

SK hynix Inc., headquartered in Korea, is the world's top-tier semiconductor supplier offering Dynamic Random Access Memory chips ("DRAM") and flash memory chips ("NAND flash") for a wide range of distinguished customers globally. The Company's shares are traded on the Korea Exchange, and the Global Depository shares are listed on the Luxembourg Stock Exchange. Further information about SK hynix is available at www.skhynix.com, news.skhynix.com.

Trends and opportunities

The moment has arrived for AI and agentic pilots to scale

AI is no longer a novelty but a catalyst for businesses to extract value from their data. Teams that have had years to experiment are now scaling their most promising use cases: bespoke LLMs that give role-specific guidance, support agents that can resolve issues on their own, and predictive agents that forecast problems before they cause downtime. Simple exchanges with chatbots have evolved into multi-turn conversations and agentic workflows that span multiple departments and take place over weeks or months.

Scaling is increasing the demand on inference systems

But the use cases that worked smoothly in the pilot stage are hitting roadblocks at scale. Demand on AI infrastructure has shifted from training-centric (model building) to inference-centric (model use). While training often runs on an isolated GPU cluster, inference scales much larger, serving many concurrent users for each model deployed.

Agentic workflows are increasing context size and complexity

At the same time, context volume and prompt sizes have grown exponentially with the proliferation of AI agents that work together and engage in multi-turn exchanges. Compared with conventional chatbots, agentic AI generates up to 15× more tokens.[2]

KV cache is becoming more central to managing context

As inference scales up, in both the number of users and context length, the KV cache can become a bottleneck. The KV cache is the memory store for *key* and *value* vectors from previously computed tokens. It allows a model to support long, multi-turn exchanges in a single session without recalculating previous prompts (Figure 1). By using a KV cache to skip computations, the inference engine can deliver a faster TTFT while reducing compute burden on the GPU. Ultimately, this delivers a highly responsive user experience and lower GPU power consumption.

The KV cache grows with each additional user and prompt in the context window. But because the KV cache usually lives in the GPU's HBM or the CPU's DRAM, most deployments are running out of capacity to retain these contexts over the life of the user's session or dialog.

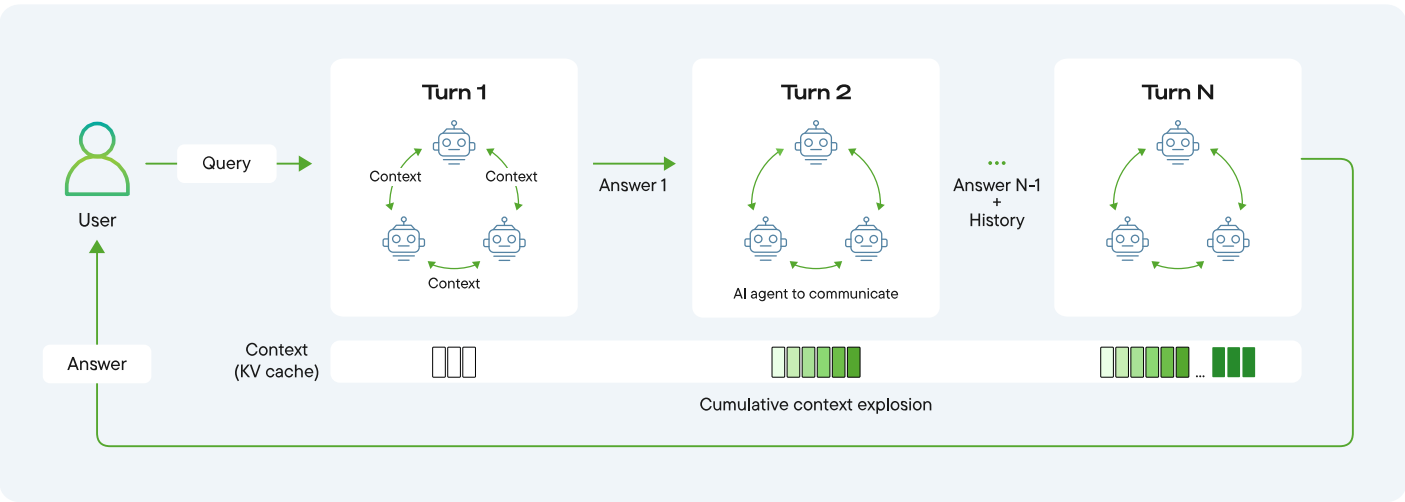

Figure 1. The KV cache grows at each step in a multi-turn workflow that involves multiple AI agents.

Challenges

Scaling infrastructure with context growth

Fundamentally, HBM and DRAM aren't equipped for massive capacity. As inference workloads grow in volume and duration, storing the KV cache in HBM and DRAM breaks down. What worked for short interactions does not hold for persistent, multi-turn, and agent-driven workloads.

Three constraints drive this limitation: capacity, cost, and compute inefficiency.

1. Memory capacity does not scale with context growth

HBM and DRAM deliver low latency but offer limited capacity. Microsoft demonstrated a throughput of 1.1 million tokens per second on a 72-GPU rack.[3] Based on this level of performance, the amount of KV cache data generated may reach approximately 92TB per GPU per day. Under the same assumptions, the NVIDIA H100 GPU features 80GB of HBM that can store roughly 1.2 minutes of KV cache, while a 1TB CPU DRAM can store about 16 minutes. The KV cache grows continuously across users and sessions and may be stored for days or weeks, quickly exceeding what memory can hold. Systems are forced to evict data, losing the ability to reuse prior computation.

2. Memory economics break at scale

HBM and DRAM are among the highest-cost resources in the data center. As of 2025, HBM media (HBM3E 12Hi) cost about $14/GB, and DRAM media (DDR5) about $8.50/GB.[4] As KV cache sizes and retention windows increase, storing context in memory becomes economically unsustainable at scale.

3. Recomputing context wastes GPU capacity and power

When KV cache is evicted from memory, it must be recomputed by reprocessing all prior prompts of the session. This increases TTFT, reduces effective GPU throughput, and drives unnecessary power consumption—especially in multi-turn, bursty workloads.

Together, these constraints create a mismatch between modern inference workloads and memory-centric design. Scaling inference requires a new approach to how context is stored and reused.

Solution overview

SSDs and hard drives bring massive capacity to KV cache storage

To provide the enormous capacity needed for multi-turn KV caches, system architects have begun offloading KV cache blocks to SSDs and hard drives. Hard drives provide durable, high-capacity storage, while SSDs serve as a cache layer for low-latency transfers. KV cache blocks can be loaded back onto the GPU when user activity resumes, avoiding the need to recompute prior prompts.

Where HBM and DDR5 offer capacity in the gigabyte range, SSDs provide terabytes of storage, and hard drives can store hundreds of terabytes. With so much capacity, SSDs and hard drives can efficiently store KV caches for days compared to minutes (Figure 2). Once full, the KV cache must evict the least-used block, thereby discarding the opportunity to reduce recalculation for a returning user.

Storage devices hold KV caches at a fraction of the cost

SSDs and hard drives offer a substantial cost advantage over DRAM. As KV caches, workloads, and memory costs scale rapidly, architects can dramatically relieve the pressure on capital expenses by using these storage devices.

As of 2025, DRAM is 44× the cost per byte deployed of NVMe SSD storage.[5] By tiering storage devices with memory and using a mix of provisioning across all tiers, architects can help control deployment costs while reducing GPU recalculation costs.

The best cost efficiency equation comes from strategically placing hard drives in tiered storage architectures. Over the last 10 years, SSD cost per TB has been about 10× higher than hard drives on average. As of 2026,[6] SSDs are 16× the cost per TB (Figure 3).

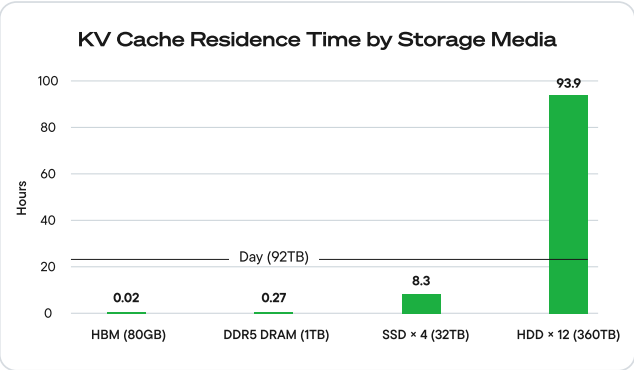

Figure 2. Example of retention duration by storage tier.

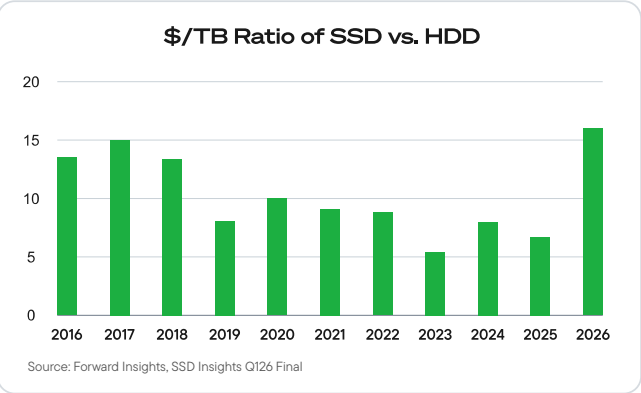

Figure 3. The SSD cost per TB is 16× higher than hard drives.

Workload requirements and design goals

Designing KV cache systems for performance and scale

KV cache offloading and prefix caching. Prefill is compute-bound,[7] making KV reuse critical to reducing TTFT. In multi-turn scenarios, KV cache can be offloaded and stored, then prefix caching loads it when needed (Figure 4). This reduces repeated computation and improves response time and GPU utilization while lowering power consumption.

Large-capacity KV cache store. To increase the prefix caching hit ratio, a KV cache pool must retain caches from multiple GPUs for periods ranging from days to weeks. Larger retention windows increase the chance that prior context can be reused instead of recomputed.

Low-latency and high-bandwidth KV cache transfer. Reusing KV caches can save GPU computation, but it may introduce I/O transfer bottlenecks. The KV cache store must not only provide the maximum bandwidth of a GPU form factor, PCI Express® (PCIe®) 5.0 ×16, but also minimize latency.

Managing KV cache across hybrid storage tiers

Storage-aware tiering. A tiered architecture combines hard drives with SSDs. Hard drives provide durable, high-capacity storage, while SSDs provide active, high-performance storage. Storing KV caches on hard drives reduces re-computation, while using SSDs as a cache layer delivers low-latency transfer. In multi-turn scenarios, a reusable KV cache is promoted from hard drives to SSDs while being delivered to the GPU. Subsequent KV cache requests mapped to the SSD are handled entirely within the SSD, without accessing the hard drive.

Reduced CPU bottleneck. GPU direct storage (GDS) enables the GPU to read KV cache data from storage into GPU memory (HBM) without CPU intervention. Compared with conventional POSIX read/write interfaces, GDS achieves up to 2.8× higher bandwidth while reducing CPU overhead by 3-5×.[8] Furthermore, it can lower overall server power consumption by approximately 7W.[9]

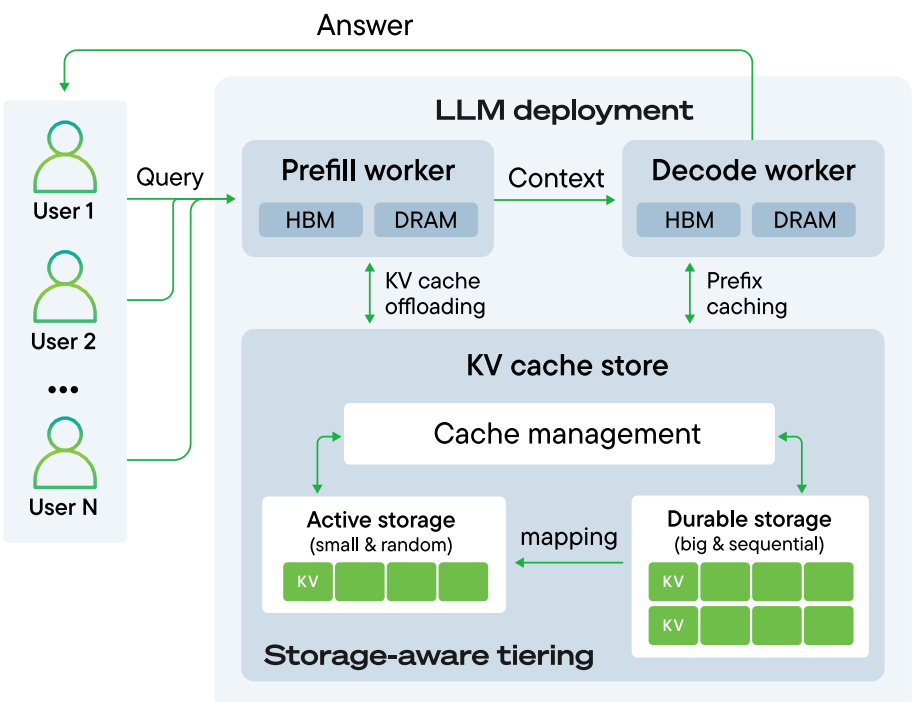

Figure 4. KV cache offloading and prefix caching via a tiered KV cache store.

Solution architecture

Components and disaggregation

The proposed solution employs SK hynix PS1010 3.84TB SSDs and Seagate Exos® 30TB hard drives, powered by Mozaic™ (HAMR) (Table 1). It uses NVIDIA Dynamo, an open-source distributed inference framework for serving generative AI models.

In LLM inference, disaggregating prefill workers and decode workers can lower the total cost of ownership.[10] KV cache backing storage is deployed as a shared NVMe-oF target, allowing multiple worker nodes to use RDMA hardware automation to optimize performance. When NVMe-oF targets are SSD/hard drive hybrid storage, the burst of activity is serviced from the SSDs during quick turns. Mass-capacity hard drives can save days or weeks of context, avoiding GPU recalculation time and cost.

Selecting NVMe-oF targets as durable storage

In NVIDIA Dynamo, the KV Block Manager leverages the NVIDIA Inference Xfer Library (NIXL) layer as a high-performance I/O abstraction. This allows multiple prefill and decode workers to compose NVMeoF targets as the backing store for the KV block cache. NIXL sees the targets as EXT4-mounted file systems on each node and is configured for peer communication to other nodes for distributed KV access.

This enables KV blocks to:

- Persist for days or weeks independent of any single worker
- Share across nodes
- Reclaim without binding to local memory

Prefill workers can share KV blocks once committed to the durable NVMe-oF target. Decode workers map and fetch those blocks on demand through NIXL, avoiding redundant computation and memory duplication.

The KV Block Manager coordinates block lifecycle metadata (allocation, pinning, eviction, and reuse) independently of physical placement. This allows base-tier storage to scale across nodes while preserving consistency and throughput (Figure 5).

This design decouples compute from memory capacity, supports elastic multi-worker inference pipelines, and enables efficient disaggregated KV caching using standard NVMe-oF infrastructure, which is fully GDS-enabled for RDMA acceleration.

	GPU Server	SSD Server	HDD Server
	Hardware specification		
CPU	Intel® Xeon® 6767P	Intel® Xeon® Gold 6548Y+	Intel® Xeon® CPU E5-2630 v2
GPU	NVIDIA H100 HBM2e 80GB PCIe		
Memory	2.0 TiB DDR5 RAM	503 GiB DDR5 RAM	32 GiB DDR3 RAM
Storage		SK hynix PS1010 3.84TB × 4 (15.36TB)	Seagate Exos 30TB × 12 (360TB)
	Software specification		
OS	Ubuntu 22.04.5 LTS	Ubuntu 22.04.5 LTS	Rocky Linux 9.6
Kernel	6.8.0.-94-generic	6.8.0.-94-generic	5.14.0-570.58. el9_6x86_64
Framework	Cuda 12.8 NVIDIA Driver 570.86.10		
Benchmark	LMBenchmark		

Table 1. GPU, SSD, and hard drive server specifications.

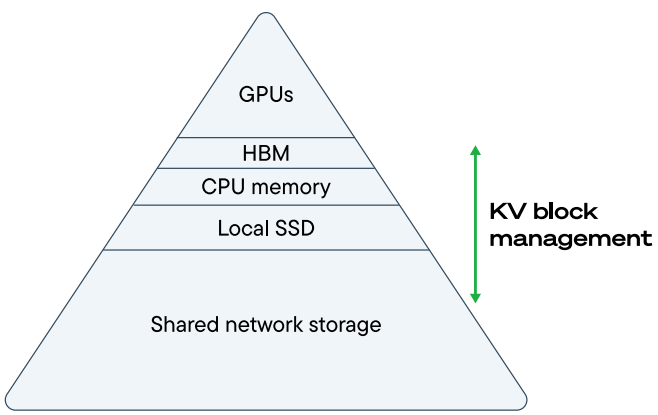

Figure 5. KV blocks are managed across HBM, CPU memory, local SSD, and shared network storage.

Hybrid SSD and hard drive targets

Using SSDs alone isn't enough. The demand for large-scale deployments to support AI requires the cost efficiency of a mixed storage solution. In this experiment, infrastructure teams add cost-efficient capacity to save days or weeks of KV blocks by creating hybrid hard drive targets with SSD caching. Managing active data in a cache within the NVMe-oF target provides fast responses to worker nodes as Dynamo orchestrates the user's repeated prompt turns to utilize available GPUs.

Seagate Exos 30TB drives, built on Mozaic HAMR technology, use conventional magnetic recording to deliver consistent, high-performance writes for KV block data, with the capacity headroom to scale KV cache stores. NVMe-oF provides a fast, hardware-accelerated data path between the requesting GPU and physical drive. End-to-end p99 latency is typically less than 100 microseconds.

The SK hynix 4TB PS1010 is a TLC-based SSD built on V-NAND technology, offering flexible power states and ultra-fast PCIe Gen 5 performance. It provides maximum bandwidth and parallel processing to deliver KV cache data cached from hard drives. By connecting four SSDs in parallel, it supports the maximum bandwidth of a PCIe 5.0 ×16 GPU interface. Hybrid storage combines the performance advantages of the SK hynix PS1010 with the capacity benefits of Seagate Exos, delivering consistent performance and reliability for AI inference as KV cache stores increase.

GPU → CPU → PCIe → NIC → RDMA → NIC → drive

Using an HTTP-based object protocol on the network adds thousands of microseconds from TLS handshake and HTTP request processing, even with a very fast object store service.

GPU → CPU → TCP → TLS → HTTP → load balancer → object store application → drive

Storage Performance Development Kit (SPDK)

The open-source SPDK provides tools and libraries for creating high-performance storage applications. It runs as a user-space application on the storage server or data processing unit (DPU), providing more features and performance than the Linux kernel alternative.

The Linux LVM provisions the backing storage pool and maps physical storage regions to NVMe-oF targets. This enables backing storage to be aggregated in parallel across multiple drives, maximizing the throughput for KV block access.

When more throughput is required, Open Cache Acceleration Software (Open CAS) is layered on top of the LVM volume as configured by the SPDK target application. In write-back mode, data is first sent to the SSD cache, and as the SSD needs more space it transfers data to the hard drive tier, which complements the SSD with larger and longer-term retention. This aligns with the Dynamo use case where prefill workers write KV cache blocks and subsequent decode workers read them through the SSD cache layer.

Using SPDK, architects can improve the robustness of LVM Shared Volume Groups by adding commands such as the Seagate In-Drive Mutex. This provides an open-source method for pooling NVMe-oF targets across multiple CPU and GPU servers, using a Kubernetes CSI driver to dynamically provision AI data pipelines.

NVMe-oF composability

Disaggregated storage composability allows raw block devices to be treated as flexible infrastructure resources within the data center or availability zone. In this architecture, the SPDK application running on a storage server, DPU, or cloud node instance can consume hybrid NVMe-oF targets.

Demonstration setup

This configuration is intended to demonstrate deployment optionality rather than define an optimized LMBenchmark infrastructure. In production, drive count and server resources can scale based on throughput, capacity, and workload requirements.

LLM serving system with hybrid storage

The sample LLM deployment includes two servers for composable storage (Figure 6). The SSD server holds four SK hynix PS1010 4TB SSDs (SSD × 4). These SSDs back the SPDK application, presenting hybrid targets of LVM-aggregated backing devices.

The hard drive server presents 12 Seagate Exos 30TB drives, exported as NVMe-oF targets using SPDK to the SSD server. Hard drives are configured with RAID storage aggregation to increase throughput.

The GPU server contains an NVIDIA H100 with 80GB HBM2e memory and two terabytes of DDR5 CPU memory. It acts as the NVMe-oF initiator, accessing both SSD-only and hybrid SSD/hard drive targets during testing.

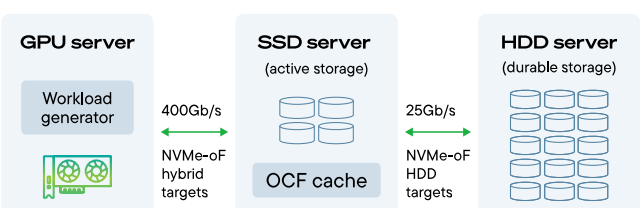

Figure 6. Combined servers composed of four SSDs and 12 hard drives to emulate hybrid storage.

Dynamo LLM deployment

Dynamo KV Block Manager deploys a disaggregated LLM with a publicly available DeepSeek-R1-Distill-Llama-8B model. Prefill nodes and worker nodes share KV cache data in multiple tiers, including GPU memory, CPU memory, local SSDs, or remote NVMe-oF targets (Figure 7). KV cache blocks are registered with a NATS broker that provides location information to other nodes using publisher/subscriber topics. The NIXL layer abstracts the access methods of the various layers, enabling GDS RDMA access for the NVMe-oF data plane.

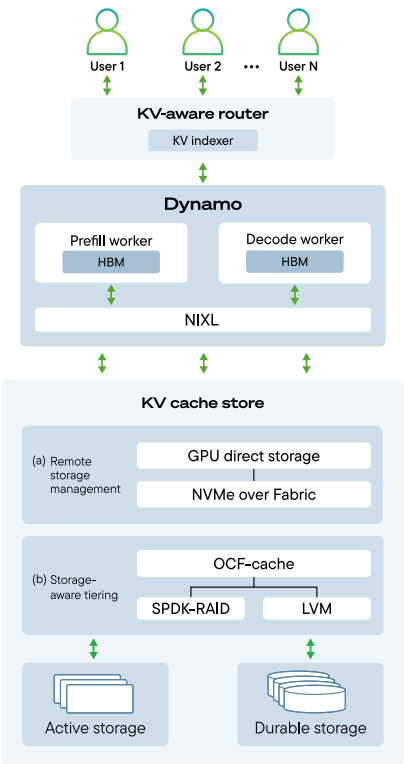

Figure 7. Prefill nodes and worker nodes share KV cache data in multiple tiers.

KV cache offloading scenario

By expanding the capacity of the KV cache store, a heterogeneous architecture of hard drives and SSDs enhances KV cache reuse and improves cost efficiency. In a multi-turn scenario, the user's prefix-cached KV data is loaded from the hard drive on the first connection (Figure 8). The subsequent KV cache is mapped to the SSD.

1. The user's initially generated KV cache is stored on the hard drive.
2. When the KV cache is reused, the saved data is copied from the hard drive to the SSD.
3. The KV cache on the SSD is loaded into the GPU. Subsequently, the reused KV cache is loaded from the SSD.
4. In the next round, in addition to KV cache hits, a small amount of newly generated KV cache data (previous responses + new user questions) are stored on the SSD.

KV cache and benchmark applications

All measurements were taken using KVBM. The LMBenchmark project, part of LMCache, was used to generate a synthetic user-prompt workload with multiple turns. Testing focuses on a long input, short output scenario to reflect common inference patterns with extended context and relatively small responses.

Assumptions and emulation of prompt bursts

While agentic AI workflows continue to evolve, the bursty nature of user activity is common across many deployments. This means users start, pause, and resume exchanges with the LLM. When pauses are long enough, the hybrid target migrates inactive KV cache data to the hard drive tier. Users returning to these contexts will see a one-time event of longer latency. The SSD tier services subsequent prompts during the burst of activity.

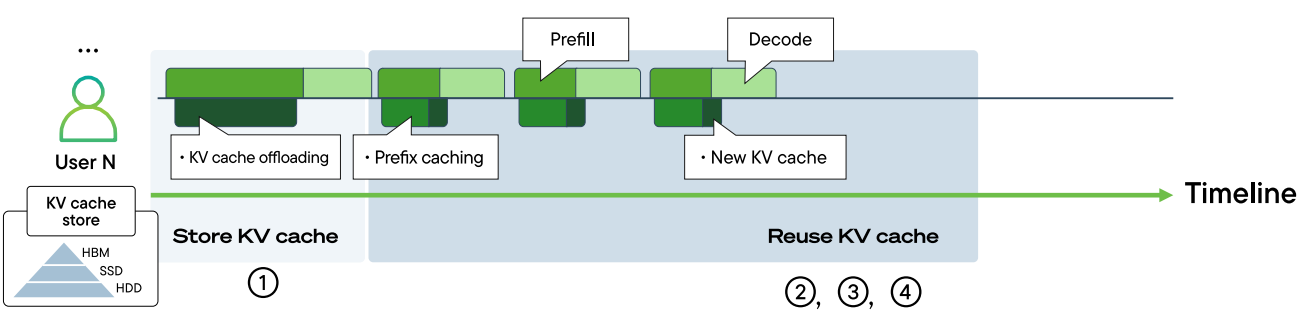

Figure 8. KV caches are offloaded, stored, and reused in a multi-turn scenario with storage-aware tiering.

LMBenchmark runs

Benchmark case

Baseline (no backing storage). This configuration measures the cost of regenerating the KV cache rather than reusing it. Prefix caching and KV offloading are disabled, requiring the KV cache to be recomputed from scratch during each prefill stage.

SSD block storage target. This configuration evaluates KV cache reuse with storage on SSD. Prefix caching and KV offloading are enabled, allowing KV data to persist across turns. The KV cache resides in both HBM and SSD.

Hybrid SSD–hard drive storage. This case evaluates a hybrid KV cache store designed for higher capacity and retrieval based on a foundation of hard drives and low-latency SSDs. This combines the benefits of SSD latency for repetitive actions and hard drive capacity for larger context saving. Because more context can be saved, there is a higher potential to avoid GPU recalculation. The KV cache spans HBM, SSD, and hard drives.

Benchmark scenario with SSD and hard drive block storage target

Preconditioning the hard drive tier with SSD bypass. To simulate a multi-turn scenario, the KV cache from turn one is written to the hard drive. The Open CAS Framework (OCF) cache policy is set to pass-through mode, causing the cache engine to bypass the SSD for all I/O operations. During turn one, LMBenchmark stores the KV cache for the initial system prompt and user query directly on the hard drive (as shown in the offloading phase in Figure 9).

KV cache reuse with write-back mode. After preconditioning, the KV cache from turn one is transferred from the hard drive to the SSD and loaded into HBM. The OCF policy is then switched to write-back mode, in which data is written first to the SSD (cache storage), and then as the SSD needs more space is transferred to the hard drive (backing storage) for longer-term retention.

From the second turn onward, the KV cache is mapped to the SSD: cached blocks are read from it, while newly generated blocks are written directly to it (as shown in the onboarding phase in Figure 9).

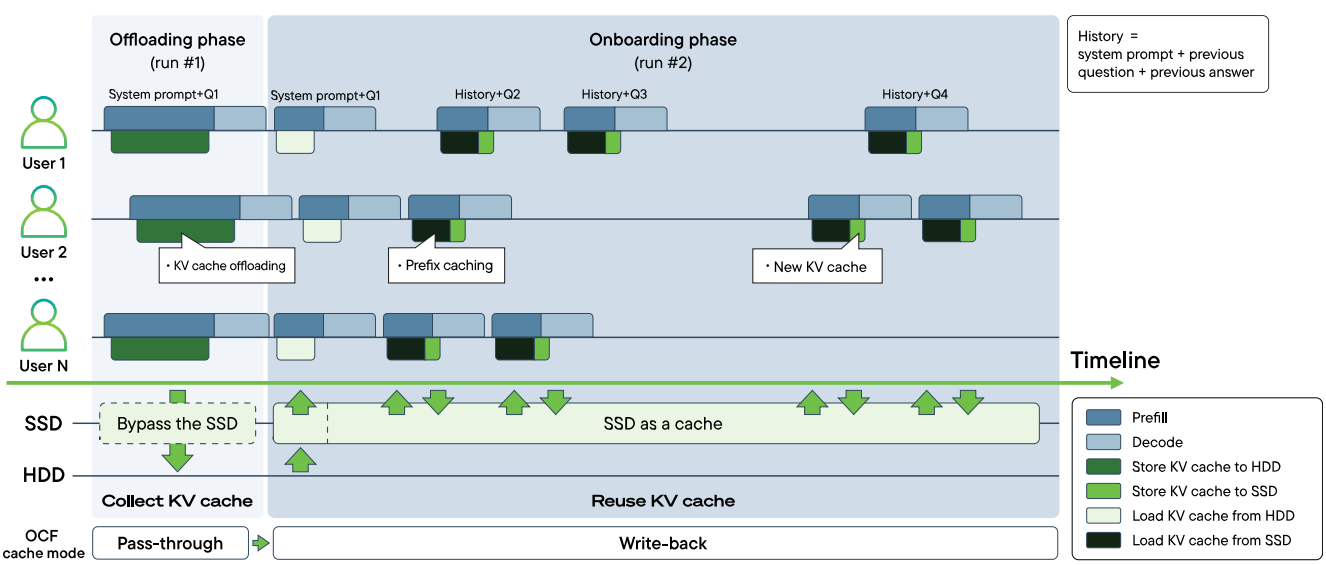

Figure 9. Multi-turn evaluation scenario for hybrid storage using LMBenchmark.

LMBenchmark results

Workload characteristic

Offloading phase. This phase represents the write stage of the KV cache. The observed I/O pattern consists of 72% random writes and 28% sequential writes. Each request is issued at 128KB, matching the maximum data transfer size (MDTS) supported by the hybrid storage (Figure 10).

Onboarding phase. This phase represents the read stage from the KV cache. The I/O pattern is dominated by 83% random reads and 17% sequential reads. As turns progress, additional KV cache write operations are generated at Output Sequence Length (OSL)-sized granularity; however, they account for less than 1% of total I/O and are therefore negligible (Figure 11).

Based on the I/O pattern, the KV cache store must be optimized for random access with data sizes of 128KB or larger.

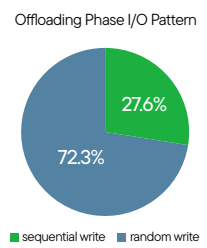

Figure 10. System I/O pattern during the offloading phase.

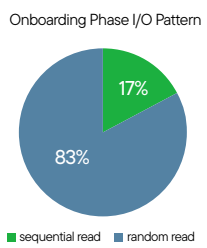

Figure 11. System I/O pattern during the onboarding phase.

LMBenchmark configurations: ISL = 21K, OSL = 100, number of users = 15, and queries per second (QPS) = 0.1.

TTFT vs. turns and users

In a multi-turn scenario, the user's prefix-cached KV data is loaded from the hard drive on the initial connection and subsequently achieves a 100% hit rate on the SSD.

Increasing turns. Figure 12 shows TTFT for each turn across up to 15 turns of inference without KV caching where the GPU is recalculating prior prompts. In contrast, a KV cache using 4 SSDs (SSD × 4) maintains stable performance even as the number of turns increases.

- **Request accumulation phase:** Initially, because KV cache generation is compute-bound and fully utilizes the GPU, the system could not immediately process continuously incoming user queries. As a result, the request backlog gradually increased, leading to an increase in TTFT.
- **Saturation phase:** As the accumulated requests filled the processing queue, the system reached a saturation state where TTFT remained consistently high.
- **Backlog draining phase:** Since LMBenchmark sends the next request only after receiving the response for the previous request, longer response times naturally reduced the effective request arrival rate. Consequently, the backlog was gradually drained, reducing queueing delay and lowering TTFT.

Increasing users. Figure 13 shows TTFT as the number of users increases. Because KV cache generation is compute-bound and limited by GPU capacity, per-user latency increases accordingly. When the number of users grows from 10 to 50, TTFT for KV cache regeneration increases by approximately 20×.

In contrast, KV cache reuse maintains stable performance even as user count scales.

Although an SSD-based KV cache store outperforms regeneration, relying solely on SSDs to retain KV cache data for days or weeks becomes impractical as model size and context length grow. For example, Qwen2.5-7B requires 56KB of KV cache per token, whereas Qwen2.5-72B requires 320KB per token. Combining SSDs with hard drives offers a more scalable solution by balancing cost and capacity.

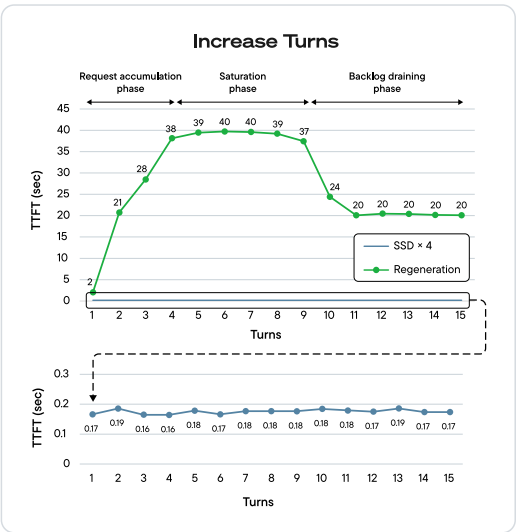

Figure 12. TTFT measurement as the number of turns increases. LMBenchmark configurations: ISL = 21K, OSL = 100, number of users = 15, and QPS = 0.5.

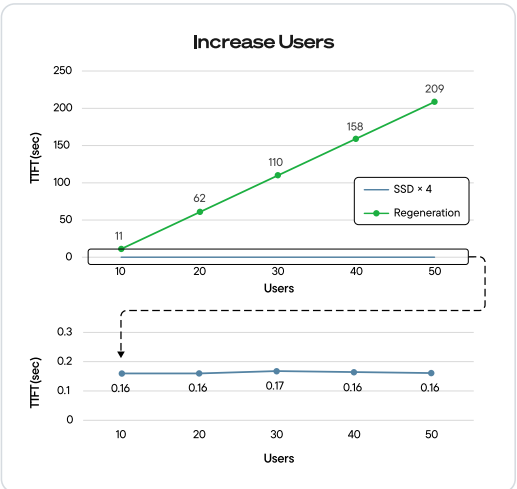

Figure 13. TTFT measurement as the number of users increases. LMBenchmark configurations: ISL = 21K, OSL = 100, QPS = 1, and turns = 10.

Performance of hybrid storage

TTFT comparison. Using hybrid storage, TTFT improves by 95% compared to regeneration (Figure 14).

End-to-end latency comparison. Hybrid storage reduces end-to-end latency by 80% (Figure 15). Because delays in the prefill stage directly propagate to the decode stage, improvements from KV cache reuse play a critical role in lowering overall latency.

Notably, this experiment does not use prefill–decode disaggregation, suggesting that the impact of prefill performance on decode latency may be even greater in disaggregated settings.

Retention time of hybrid storage

Capacity and performance. Hybrid storage achieves capacity and retention comparable to hard drives (Figure 16), enabling it to store approximately 11× more KV cache than SSD × 4.

As the number of turns increases, it maintains performance comparable to SSD × 4 while reducing cost by about 75% (Figure 17).

Increasing SSD capacity within the hybrid configuration would further improve performance consistency, allowing the system to sustain SSD-level performance over longer workloads.

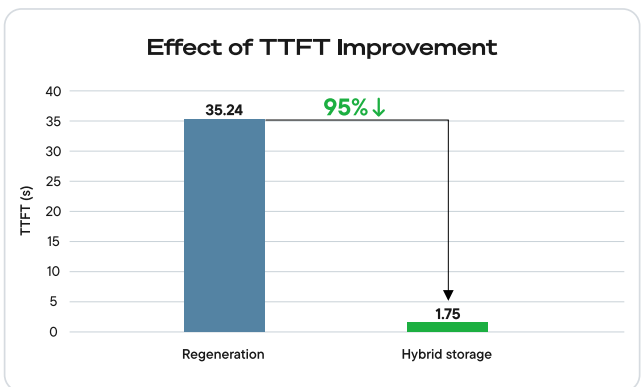

Figure 14. Hybrid storage provides a 95% improvement in TTFT compared to regeneration.

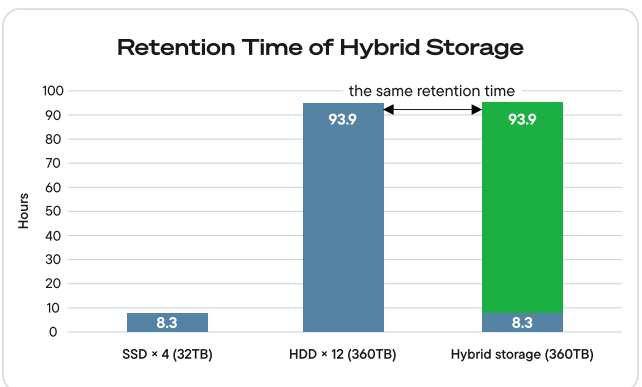

Figure 16. Hybrid storage can retain KV cache blocks significantly longer than SSD-only storage.

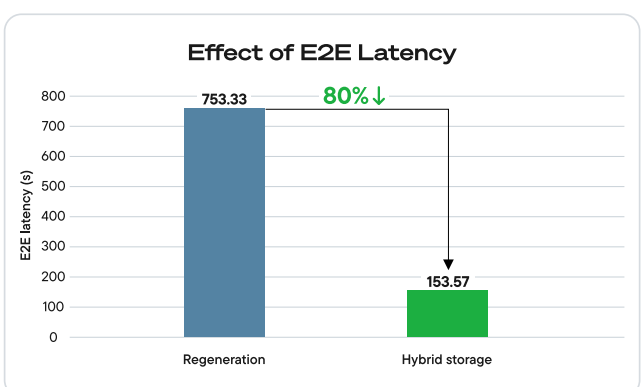

Figure 15. Hybrid storage provides an 80% improvement in latency compared to regeneration.

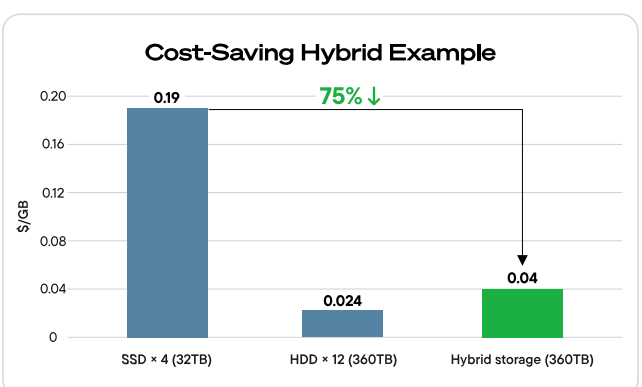

Figure 17. A hybrid storage example reducing costs by about 75% compared to SSD-only storage.

LMBenchmark configurations: ISL = 21K, OSL = 100, number of users = 15, QPS = 1, and turns = 10.

All-SSD vs. hybrid storage

Hybrid storage performance. Figure 18 shows the average TTFT across multi-turn scenarios, ranging from one to 50 turns. As the number of turns increases, hybrid storage performance converges toward the all-SSD configuration. The TTFT gap decreases from 7.6 seconds at one turn to 0.35 seconds at 50 turns.

This convergence occurs because the KV cache is read once from the hard drive and subsequently served from the SSD.

I/O bandwidth monitoring. Figure 19 shows the measured read bandwidth under a query rate of four queries per second. Consistent with the TTFT results, workloads with more turns benefit more from SSD caching, allowing hybrid storage bandwidth to approach that of SSD × 4. The bandwidth gap decreases from 88% at 100 seconds to 15% at 1,000 seconds.

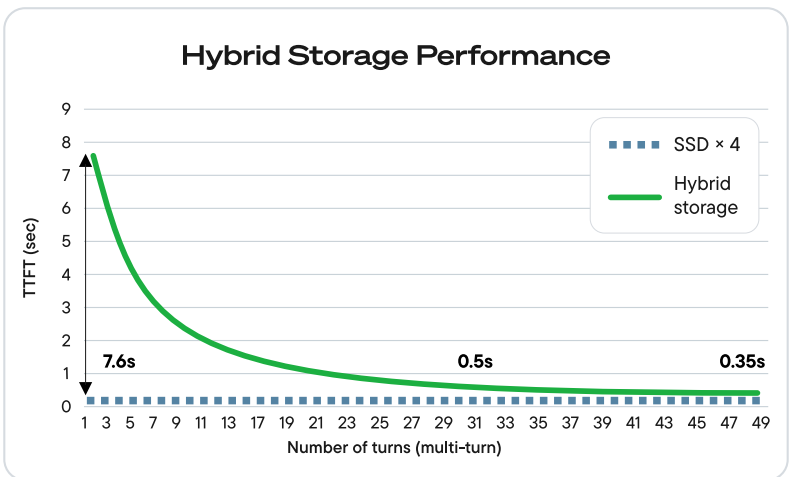

Figure 18. TTFT performance under various multi-turns. LMBenchmark configurations: ISL = 21K, OSL = 100, number of users = 15, and QPS = 1.

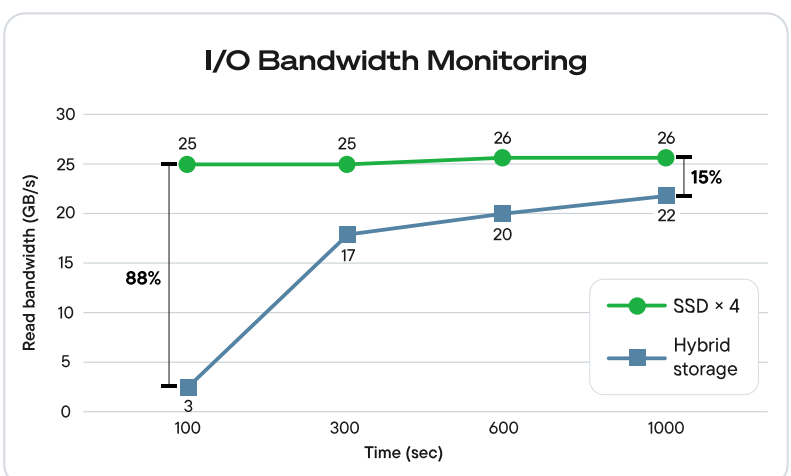

Figure 19. Read bandwidth measurement. LMBenchmark configurations: ISL = 21K, OSL = 100, number of users = 15, and QPS = 4.

Scaling hard drive throughput

This experiment was designed to accurately measure performance changes as hard drives scale. To achieve this, SSDs were excluded from the KV cache store, utilizing only hard drives. Execution time was adjusted to measure the TTFT for a single turn.

Scaling hard drive throughput. During the onboarding phase, the throughput performance of the first turn depends on hard drive bandwidth as the KV data is cached from the hard drive to the SSD. As the number of turns increases, more KV cache data is mapped to the SSD, and overall solution performance gradually approaches SSD bandwidth performance. This experiment combines 12 hard drives into a single logical volume. This allows data to be distributed across multiple hard drives, thereby increasing the effective hard drive bandwidth. Increasing the number of hard drives in the RAID configuration can further improve hard drive bandwidth, which can improve the performance of the first turn.

Figure 20 shows TTFT when increasing the number of hard drives. The results shown for up to 12 hard drives are based on actual measurements, while the results beyond 12 hard drives are estimated. When the number of hard drives increased from four to eight and 12, performance increased to 2× and 3×, respectively. When the number of hard drives increased from 12 to 24, TTFT decreased by approximately 50%. Beyond 36 hard drives, further scaling of hard drive bandwidth is not extrapolated because this system reaches the maximum network bandwidth limit of 25GB/s. Therefore, TTFT reduction saturates at approximately 60%.

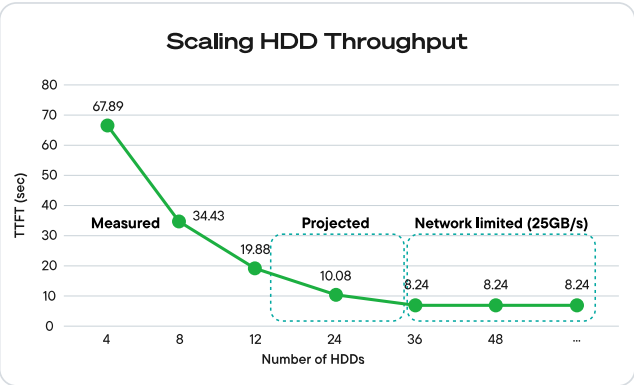

Figure 20. TTFT decreases with more hard drives.

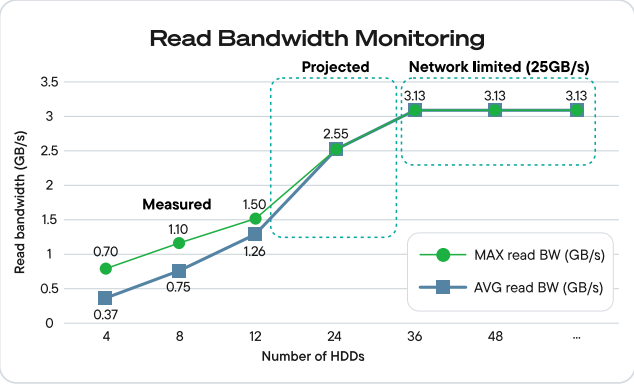

Figure 21. Read bandwidth increases with more hard drives.

LMBenchmark configurations: ISL = 21K, OSL = 100, number of users = 15, QPS = 1, and runtime = 50s, pass-through mode

Read bandwidth monitoring. As hard drive bandwidth increases, more user queries can be processed within the same amount of time (Figure 21). While the gap between the maximum and average bandwidth is about 2× when using four hard drives, it is only about 20% when using 12 hard drives. The GPU throughput demand is satisfied by 24 hard drives with the model under test.

GPU utilization and power reduction

GPU utilization in the prefill phase. By reusing KV caches, hybrid storage reduces GPU utilization during prefill by 76%. The freed GPU capacity can be redirected to generating new KV caches and serving additional queries (Figure 22).

Power utilization in the prefill phase. As a result, the measured power consumption was 52% lower than using GPU regeneration, reducing power consumption per session.

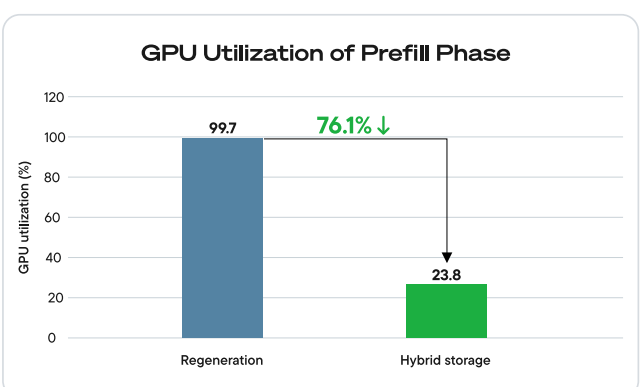

Figure 22. Hybrid storage for KV cache offloading significantly reduces GPU utilization compared to regeneration.

LMBenchmark configurations: ISL = 21K, OSL = 2, number of users = 15, and QPS = 1.

Future work

Dynamo Prefetch Hinting

The NATS broker provides a method to distribute the KV block mapping information to all workers in the cluster. By subscribing to topics that map to a backing storage, a prefetch hint could occur simultaneously with orchestration and assignment of the prefill worker node, reducing the latency of reads coming from the hard drive devices.

Comparison of local and remote S3 tier vs. NVMe-oF

Primary and high-performance S3 object storage tiers could be utilized, offering simple scalability and deployment. Object stores both near and far introduce latency from network communication complexities, where NVMe-oF using RDMA is very efficient over the wire with hardware automation supported in smart network adapters. A detailed comparison of the various deployment permutations was outside the scope of this paper, but worthy of research to understand the benefits and side effects.

Performance effects of using cuComp

The cuComp library offers very high throughput of compression and decompression operations. Reducing the data payload in and out of the GPU offers potential reductions in KV cache size and throughput requirements. Studying the impact on storage, token throughput, and TTFT may offer more methods to manage deployment costs.

Conclusion

Hybrid storage for KV caching reduces GPU load, power consumption, and TTFT

As inference workloads scale—driven by increasing user demand and longer context windows—enterprises and AI infrastructure builders require high-capacity KV cache architectures similar to those already adopted by hyperscalers. Offloading KV blocks to a composable NVMe-oF target eliminates the need to recompute prompts during multi-turn interactions, reducing GPU load and power consumption while improving TTFT and overall responsiveness.

Implementing targets as a hybrid SSD–hard drive tier further expands capacity, enabling KV blocks to be retained for days, weeks, or months. This approach balances cost, capacity, and performance by combining modern hard drives with SSDs to enable scalable data growth and low-latency caching.

Together, these benefits enable a lower total cost of ownership through reduced media costs and improved power efficiency.

Reference

1 AWS re:Invent 2025 - AWS storage beyond data boundaries: Building the data foundation (INV215), AWS, Dec. 3, 2025. Page 2.
2 Introducing Nemotron 3 Super: An Open Hybrid Mamba-Transformer MoE for Agentic Reasoning, NVIDIA, March 11, 2026. Page 3.
3 Breaking the Million-Token Barrier: The Technical Achievement of Azure ND GB300 v6, Microsoft, Nov. 3, 2025. Page 4.
4 SSD & HDD Storage Market Dynamics and Pricing Brief – February 2026, Omdia, March 4, 2026. Page 4.
5 Ibid. Page 5.
6 "SSD Insights Q126 Final," Forward Insights. Page 5.
7 Prefill and Decode for Concurrent Requests - Optimizing LLM Performance, Hugging Face, April 16, 2025. Page 6.
8 Accelerating Storage with Magnum IO and GPU-Direct Storage, NVIDIA GTC, pp14, 2020. Page 6.
9 Accelerating AI With High Performance Storage, Solidigm, September 2, 2025. Page 6.
10 Example Workload: Large MoE LLM Inference, NVIDIA, March 12, 2026. Page 7.