The KV Cache Challenge: Optimizing the Memory and Storage Hierarchy for AI Inference

Author: Karthik Ganesan

The shifting AI landscape

The AI industry continues its relentlessly hectic pace of evolution. Hardware is getting faster, use cases are evolving, and everywhere you look, there is unprecedented growth.

Two trends have resulted in a dramatic shift that places memory and storage at the heart of AI systems:

Increasing data requirements of inference

Growth of inference's share of AI hardware

Together, these trends shine a spotlight on the KV cache, a technique developed for modern AI models to improve inference performance. The increasing data requirements result in KV caches becoming large, while the increasing share of inference amplifies the benefit to TCO from optimizing how we manage KV caches.

Increasing data requirements for inference

The development of new models, faster hardware, better training methodologies, and optimized inference workflows all lead to a foundational truth — more data and better data result in smarter AI.

Research has consistently demonstrated this principle. The Chinchilla scaling laws (Hoffmann et al., 2022) showed that model performance improves predictably with both model size and training data volume, establishing that data quantity is as important as parameter count. Subsequent work has extended this insight to inference: Longer context windows allow models to reason over more information per query, and richer input data produces higher-quality outputs.

Agentic AI systems amplify this further. In multi-turn agentic workflows, the memory of the entire chain of inputs and outputs — every tool call, every intermediate reasoning step, every prior response — is used to generate the next action. This means the effective context grows with each turn, and the KV cache must retain the full conversation history to maintain coherent, multi-step reasoning.

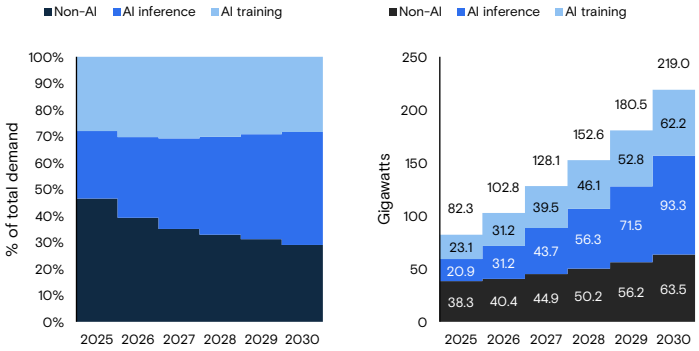
Growth of inference's share of total data center power demand

Training large language models was the first wave — capital-intensive, hardware-hungry, and concentrated among a handful of hyperscalers. But now, inference is the business. Already, more than half of all AI infrastructure is dedicated to inference, and this share is expected to grow rapidly as agentic AI use cases continue to rise. Source: McKinsey, February 2026

In 2026, the split between inference and training is roughly equal, but the 22% CAGR for training lags significantly behind the 35% CAGR for inference. As inference becomes the dominant workload, optimizing the efficiency and cost-effectiveness of inference systems becomes paramount.

By 2030, inference workloads could account for more than 40% of all data center demand and 60% of AI data center demand.[1]

Figure 1: Global data center power demand by workload type, 2025–2030: Share of total (left)



1. "How data centers and the energy sector can sate AI's hunger for power" (mckinsey.com).

What is the KV cache, and why does it matter?

When a large language model generates text, it doesn't process each token in isolation. The model uses the "Attention Mechanism" to represent how tokens relate to each other and how tokens relate to language overall — effectively a model's short-term and long-term memory.

Internally, a model does this by computing Key and Value vectors for every token in the context window. Initial LLMs would recompute the K and V keys and values for prior tokens when generating the next token, but longer context windows required caching these vectors in memory for successive token generation to prevent the compute required for each new token from growing exponentially.

Multi-turn interactions with LLMs (like in a back-and-forth conversation) reuse the same KV caches, even when the interactions are minutes, hours, or days apart. Recomputing the KV caches results in a long wait before the LLM can begin to reply. Still, by storing KV caches on disk, they can be read into the accelerator's memory to begin generating new tokens immediately.

The KV cache scales roughly proportionally to:

```
layers × heads × head_dim × sequence_length × bytes_per_element
```

Each component of this formula maps to a different dimension of the problem:

- Layers × heads × head_dim determined by model architecture (model size). Larger models have more layers and attention heads, directly increasing KV cache per token.
- Sequence length determined by the length of input and output. Longer context windows and longer generated responses linearly increase KV cache size.
- Bytes per element determined by numerical precision and quantization. Techniques like TurboQuant reduce this factor by compressing the KV cache per token.

Long context and many concurrent requests compound quickly. Here's the problem: KV cache sizes are growing rapidly.

Table 1: KV cache growth by model and context length[2]

Model	Parameters	KV heads	Head dim	KV cache per 100K context
Llama-3.3-70B	70B	8	128	31.3GB
DeepSeek-V3	671B (MoE)	128	128	~7GB
Qwen-2.5-72B	72B	8	128	~32GB
GPT-4o class (est.)	~200B+	~16	128	~94GB
Gemini 2.5 Pro (est.)	~500B+ (MoE)	~32	128	~120GB

A single 100K-token request on a 70B-class model requires over 30GB of KV cache. Scale that to 32 concurrent users on a single GPU, and even a mid-size model demands nearly 1TB of KV cache storage. GPU HBM, typically 192GB on current platforms, cannot hold it all.

Table 2: KV cache per token for representative models, 2020–2025[3]

Year	Representative model	Attention	KV per token (BF16 unless noted)	vs. 2020 baseline
2020	GPT-3 (175B)	Multi-Head Attention (MHA)	4.72MB	1x (baseline)
2023	Llama-2-70B	Grouped-Query Attention (GQA, 8 KV heads)	0.33MB	~14x smaller
2024	DeepSeek-V3 (671B MoE)	Multi-head Latent Attention (MLA)	0.070MB	~67x smaller
2025	70B GQA + TurboQuant	GQA + 3.5-bit / 2.5-bit KV quantization	0.072 / 0.051MB	~66x / ~92x smaller

2. Values derived from published model architectures and Micron DCWE analysis. MoE models use an active subset of parameters but a full KV cache across all layers.
3. Calculated from published model configurations as 2 (K and V) × layers × KV heads × head dim × bytes per element: GPT-3 (96 layers, d_model 12,288, MHA); Llama-2-70B (80 layers, 8 KV heads, head dim 128); DeepSeek-V3 (61 layers, 512 + 64 cached MLA latent). TurboQuant row applies 3.5-bit (quality-neutral) and 2.5-bit KV quantization to the Llama-2-70B configuration.

While KV cache per token has improved dramatically, total KV cache demand continues to grow because models are getting larger, context windows are getting longer, and concurrency requirements are increasing. The efficiency gains are being consumed by growing demand.

The three forces driving KV cache pressure

The KV cache challenge is not a static problem. Three simultaneous forces compound it:

- Context window growth
- Reasoning models
- Agentic AI

Context windows are growing exponentially

Between 2018 and 2025, context length grew by approximately 19,500x in 7 years. This far outpaces Moore's Law, which predicts roughly 2x every 2 years, or approximately 11x over 7 years. The average annual growth factor for context windows has been approximately 4.5x per year since 2022.

Table 3: Context length growth: 2018–2025[4]

Year	Industry max context (tokens)	YoY multiplier	Cumulative growth
2018	512	—	1x
2019	1,024	2x	2x
2020	2,048	2x	4x
2021	2,048	1x	4x
2022	4,096	2x	8x
2023	100,000	~24x	~195x
2024	2,000,000	20x	~3,906x
2025	10,000,000	5x	~19,531x

The industry trend toward longer context windows is accelerating: GPT-4o supports 128K tokens, Claude 3.5/4 supports 1M tokens, and Gemini 2.5 Pro supports 1M+ tokens with plans for 10M. Each new model generation increases context length, and each increase multiplies the KV cache storage requirements.

Multi-document planning, large software repositories, and multi-modal applications are pushing real-world context lengths to 1M+ tokens. Serving 1M context for a 400B+ model with sub-1-second latency requires more than 2TB of combined memory across the hierarchy — and concurrency compounds it further.

Reasoning models amplify the problem

Modern reasoning models (chain-of-thought, tree-of-thought) generate intermediate tokens that expand the effective KV cache by up to 3x compared to standard inference. These intermediate reasoning tokens must be cached just like user-facing tokens, dramatically increasing the memory footprint of each request.

Data from Epoch AI's analysis of inference compute trends shows that reasoning-intensive models like OpenAI's o1/o3 and DeepSeek-R1 routinely generate 3–10x more tokens internally than they return to the user. OpenRouter usage data confirms this pattern: Reasoning model requests consume significantly more compute and memory per query than standard chat completions, with median output-to-input token ratios 2–5x higher for reasoning models.[5]

4. Values based on published model specifications: 2023: Claude 2 (100K), 2024: Gemini 1.5 Pro (2M), 2025: Gemini 2.5 Pro (10M context planned).
5. See Epoch AI, "Trends in AI Inference" (2025) for more information.

As AI moves from single-turn chatbots to multi-step agents that plan, execute, and iterate, the KV cache becomes something that must persist — not for milliseconds, but for hours to days. An AI agent managing a complex workflow needs to recall its prior reasoning steps, which means the KV cache must be stored, retrieved, and reused across sessions.

The need to store KV caches for systemic learning and reuse transforms the KV cache from a transient GPU memory structure into a persistent storage problem.

Combined Effect: Memory capacity lags compute growth, creating the primary bottleneck in inference economics. Context length (~30x annual growth), reasoning tokens (up to 3–10x expansion), and agentic persistence (hours-to-days cache reuse) compound to create unprecedented demands on memory and storage.

Can algorithmic advances solve the problem?

A natural question arises: Can software and algorithmic innovation compress the KV cache enough to eliminate the hardware bottleneck? The answer is nuanced — algorithms bend the curve, but they don't break it.

It is important to recognize that algorithmic improvements to KV cache efficiency are not a one-time event — they have been happening continuously and are expected to continue. Each generation of model architecture brings new techniques that reduce KV cache overhead, from multi-head attention to grouped-query attention to multi-head latent attention.

Table 4: KV cache reduction methods: Progressive improvements[7]

Method[6]	Year	Core result	Key trade-offs
GQA (Grouped Attention)	2023	4–16x KV reduction vs. MHA (8x typical)	Requires pre-training; now standard
MLA (Dimensionality Reduction)	2024	~57x KV reduction vs. MHA (~3.6x vs. GQA-8)	Must be architected from scratch
Cross-Layer Attention	2025	2x KV reduction	Reuses KV blocks across layers
Sparse Attention / KV Compaction	2025	Linearizes attention	Decreases accuracy; hallucination risk
TurboQuant	2025	~4.6–6.4x KV reduction vs. 16-bit (3.5–2.5 bits)	Preserves dot-product accuracy; early-stage

A critical caveat: These reduction methods do not all stack multiplicatively. GQA and MLA are alternative architectural choices made at model design time; a model uses one or the other, not both, so their reductions cannot be multiplied together. Quantization, such as TurboQuant can be applied on top of either, but practical bit-widths limit its contribution to roughly 4.6–6.4x versus 16-bit. Measured on representative models, per-token KV cache has fallen roughly 65–90x from the 2020 MHA baseline — from 4.72MB for GPT-3 to about 0.05–0.07MB for MLA or quantized GQA models (see Table 2) — not the ~740x that naively multiplying the best-case figures in Table 4 would suggest.

TurboQuant, published by Google in April 2025, represents the current state of the art in KV cache quantization. Rather than preserving element-wise accuracy like standard quantization, it optimizes for preserving the dot product (Q × K) that the attention mechanism actually computes. Standard quantization only preserves element-wise accuracy that reduces volume — INT8 is usually fine, INT4 starts degrading, and INT2/3 often breaks attention. TurboQuant says: don't optimize for reconstructing K exactly; optimize for preserving Q × K. It stores K as a coarse quantization plus a tiny correction tuned for dot product accuracy in very low precision (1 bit). The result: 6x KV cache reduction when compared to FP16 with no measurable loss in model quality.

But here's the critical insight: Any reduction in KV capacity gets immediately reinvested. Smaller KV caches per token mean operators can serve longer contexts or more concurrent users — consuming the freed capacity. Coupled with the approximately 30x annual growth in leading model context windows, algorithmic advances delay the scaling pressure but do not eliminate it.

6. To learn more about each of the methods shown, see the following:
   GQA: "Introduction to KV Cache Optimization Using Grouped Query Attention" (pyimagesearch.com).
   MLA: "A Gentle Introduction to Multi-Head Latent Attention" (machinelearningmastery.com).
   CLA: "Reducing Transformer Key-Value Cache Size with Cross-Layer Attention" (arxiv.org).
   Sparse attention: "Compressing KV cache memory by half with sparse attention" (cerebras.ai).
   TurboQuant: "TurboQuant: Redefining AI efficiency with extreme compression" (research.google.com).
7. Source: Micron CMBU & CDBU preliminary analysis

Epoch AI's analysis of software efficiency improvements in AI shows an approximately 3x annual improvement in algorithmic efficiency, meaning that each year, the same performance can be achieved with roughly one-third the compute (and by extension, one-third the memory).[8] TurboQuant's 6x reduction fits neatly within this extrapolation as approximately two years of expected algorithmic progress. This means TurboQuant is not a paradigm shift — it is an expected improvement arriving at the expected time.

While KV reduction can delay offloading to warm and cold storage tiers for even larger contexts, it also democratizes the use of larger contexts; currently, only about 25% of deployments use long context. As algorithmic advances make long context more accessible, overall KV cache demand grows.

Key takeaway: Algorithmic improvements are continuous and expected; they bend the memory curve but do not break memory scaling. Each improvement gets reinvested in longer contexts and more users. No signal on demand reduction to memory or storage TAM.

Optimizing the memory and storage hierarchy for inference

The memory hierarchy challenge

GPU HBM capacity is limited, typically 192GB on current platforms, scaling to 288GB of HBM4 on next-generation platforms. Meanwhile, KV cache demands grow unbounded as context length, concurrency, and model size increase. Memory capacity lags compute growth, creating the primary bottleneck in inference economics.

Data movement is the cost. Context length and model size increases put enormous pressure on the memory and storage subsystem. Compute is growing faster than memory capacity, driving inference demand that the memory hierarchy cannot satisfy. The solution requires optimizing the entire memory and storage hierarchy for inference workloads, not just a single tier, but a coordinated approach across all levels.

KV cache memory and storage tiering

The KV cache problem cannot be solved by any single memory or storage technology. Instead, it requires a tiered approach where KV cache data flows through multiple layers of the memory and storage hierarchy, with each tier optimized for a specific access pattern, latency requirement, and data residency duration. As KV cache data ages and is accessed less frequently, it is evicted down through the tiers.

Table 5: KV cache memory and storage tiering in inference[9]

Storage tier	Capacity per GPU	Latency	Bandwidth	Data residency	Role in KV cache	Micron products
Near memory (GPU HBM)	10 – 80GB	~100ns	1 – 5 TB/s	Milliseconds to seconds (active token generation)	Active generation (seconds)	HBM4, HBM3E, GDDR7
Main memory (CPU DDR5)	200 – 400 GB to TB	~100ns	100 – 300 GB/s	Seconds to minutes (long context queries)	Long context overflow (minutes)	MRDIMM, high-capacity DDR5, SOCAMM2
Expansion memory (disaggregated)	100s of GB to TB	~1–10µs	10 – 50 GB/s	Minutes to 10s of minutes (context switching, scheduling)	Context switching (10s of minutes)	Disaggregated DDR5
Context memory storage (local NVMe™ SSD)	10s to 100s of TB	~20µs (P50)	20 – 200 GB/s	Minutes to hours (multi-turn workflows)	Multi-turn persistence (hours)	Micron 9650 SSD Micron 7600 SSD
Networked data lakes (network storage)	PB	~100–500µs	< 10 GB/s	Hours to days (RAG, persistent context)	RAG / long-term (days)	Micron 6600 ION SSD

8. See "ALGORITHMIC PROGRESS IN LANGUAGE MODELS" (arxiv.org) to learn more.
9. Source: Micron Technology KV cache tiering analysis, 2026.

Each tier serves a distinct purpose in the inference pipeline:

- Near memory (HBM): Holds the active KV cache for tokens currently being generated. This is the hottest tier — every attention computation reads from here. Capacity is limited (10–80GB per GPU) but bandwidth is extreme (1–5 TB/s). At typical generation rates, HBM holds seconds of active context.
- Main memory (DDR5): Provides overflow capacity for KV caches that exceed HBM. With 200–400GB per CPU socket, main memory can hold minutes of KV cache for long-context queries. Bandwidth (100–300 GB/s) is sufficient for prefill and context switching operations.
- Expansion memory (disaggregated DDR5): Extends memory capacity beyond what a single server can hold. Useful for query scheduling and context switching across tens of minutes of accumulated KV cache data.
- Context memory storage (local NVMe SSD): The workhorse tier for multi-turn inference. Stores KV caches for hours of conversation history, enabling fast retrieval for returning users and multi-step agentic workflows. High-performance NVMe SSDs provide the bandwidth (20–200 GB/s) and capacity (tens to hundreds of TB) needed.
- Networked data lakes: Long-term storage for RAG (retrieval-augmented generation) contexts and persistent KV caches that span days to weeks. Capacity scales to petabytes but bandwidth is limited.

The key insight: Optimizing inference is not about choosing one tier, it's about orchestrating data movement across all tiers to minimize latency while maximizing capacity utilization. Each tier of the memory and storage hierarchy plays a critical role.

KV cache offload to NVMe SSDs: Validated and production-ready

Among the tiers, NVMe SSD-based KV cache offload represents the most immediate and commercially actionable optimization. When the KV cache exceeds GPU HBM capacity, KV tensors are transparently offloaded to NVMe storage for later retrieval. This is necessary because DRAM residency alone doesn't scale — the cost of adding enough DRAM to hold all active KV caches is prohibitive.

KV cache offload directly improves inference throughput, GPU utilization, and margin — making it immediately monetizable. At approximately 13 KB per token, the storage requirements are well-matched to high-capacity NVMe SSDs. The software ecosystem is already responding: vLLM's LMCache, SGLang's RadixAttention, and TensorRT-LLM all support NVMe KV cache offload.

Validated performance: Zero throughput penalty

This isn't theoretical. Micron's Data Center Workload Engineering (DCWE) team validated KV cache offload across 318 test configurations using vLLM with LMCache on an AI data-center scale AI processor with a Micron 9550 Pro 15.36TB NVMe SSD.[10]

The results are definitive:

- Zero throughput penalty: The full-hit phase (100% cache retrieval from NVMe) achieves 45,353 tokens/s — comparable to cold-start performance, proving NVMe SSD can serve as an effective KV cache backing store without degrading inference throughput.
- Linear I/O scaling: KV cache I/O scales linearly with context length — from 0.12GB at 1K tokens to 12.2GB at 100K tokens per full-hit phase.
- Read-dominated production workload: The full-hit phase — the critical path for production serving — is 100% read from the SSD perspective. The Micron 9550 Pro's 14.5 GB/s sequential read bandwidth provides substantial headroom.
- Highly sequential I/O: NVMe trace analysis shows 99% sequential I/O with a dominant block size of 64 KiB, ideal for NVMe SSD performance.

10. Based on Micron DCWE vLLM KV cache offload analysis, 318 test configurations, Llama-3.1-8B-Instruct using an integrated CPU-GPU data center superchip designed for enterprise-grade generative AI and high-performance computing workloads and a 15.36TB Micron 9550 PRO SSD (formatted capacity will be less).

SSD performance deep dive

The Micron 9550 Pro demonstrates excellent performance for KV cache-offload workloads. The I/O pattern is highly sequential with large block sizes, playing directly to the strengths of modern NVMe SSDs.

Table 6: SSD performance characteristics

Metric	Write phase	Full hit (read) phase
Avg. data per phase	5.63GB	5.69GB
Max. data per phase	13.96GB	12.21GB
Peak bandwidth (100ms)	10.9 GB/s	1.8 GB/s
Latency (P50)	864µs	20.4µs
Sequential I/O	99%	99%
Dominant block size	64 KiB	64 KiB

The Micron 9550 SSD is rated up to 14.5 GB/s[11] of sequential read bandwidth, providing 8x headroom above the measured peak read demand, indicating substantial capacity for serving multiple concurrent inference streams. The SSD operates at a fraction of its rated capacity, with an average queue depth of 7.8, indicating significant headroom for additional concurrent workloads.

Why a combined memory and storage hierarchy Is essential

No single tier of the memory and storage hierarchy can solve the KV cache challenge alone. Each tier has fundamental trade-offs between latency, bandwidth, capacity, and cost:

The measured P50 latency of 20.4µs for local NVMe reads is 5–25x faster than network storage and 250,000x faster than HDD. For the context memory storage tier — where multi-turn KV caches reside for minutes to hours — local NVMe SSD is the clear choice. But the full solution requires coordinating all tiers: HBM for active generation, DDR5 for context overflow, NVMe for persistent multi-turn context, and network storage for long-term RAG data.

The companies that optimize the entire memory and storage hierarchy — not just one tier — will define who can serve AI at scale, at speed, and at profit.

Micron's role across the memory and storage hierarchy

Micron is uniquely positioned to address the KV cache challenge because it provides products at every tier of the memory and storage hierarchy. No other company offers the full stack from HBM to NVMe SSD to network-attached storage.

Near memory: HBM4 and HBM3E

Micron produces HBM4 and HBM3E memory that provides extreme bandwidth for active KV cache during token generation. With bandwidth exceeding 1 TB/s per stack, HBM is essential for the attention computation that reads KV cache on every generated token. Micron's HBM3E is currently shipping in high volume for next-generation AI GPU platforms built on the most advanced current GPU architecture — supporting up to 8 TB/s of memory bandwidth per accelerator[13]

11. See the Micron 9550 SSD product page on micron.com to learn more.
12. Source: Micron DCWE measured performance data and tiering analysis.
13. See "Memory that matches the scale of human ambition" (micron.com) and "High-bandwidth memory (HBM)" (micron.com) to learn more about Micron products and performance metrics.

Main memory: MRDIMM and DDR5

Micron MRDIMMs and high-capacity DDR5 modules provide the main memory tier for KV cache overflow. With capacities of 200–400GB per socket and bandwidth of 100–300 GB/s, DDR5 serves as the first overflow tier when KV cache exceeds HBM capacity.[14] This tier holds minutes of KV cache for long-context queries and supports context switching operations.

Context memory storage: Micron 9650 SSD and 9550 SSD

The context memory storage tier is where multi-turn KV caches reside for minutes to hours. This is the workhorse tier for production inference, and Micron's high-performance NVMe SSDs[15] are purpose-built for this workload:

- 14.5 GB/s sequential read: Massive headroom for KV cache retrieval, even with bursty access patterns
- 10 GB/s sequential write: Handles KV cache writes at full speed during initial prompt processing
- 15.36TB capacity: Stores KV caches for hundreds of concurrent long-context requests per GPU
- PCIe® Gen6 (Micron 9650 SSD) and Gen5 (Micron 9550 SSD) x4 interface: Native bandwidth for current and next-generation GPU platforms
- 20.4µs P50 latency: Measured under real KV cache workloads[16]

Potential TCO Impact

Adding a high-performance NVMe SSD to a GPU server enables serving longer contexts without adding more GPUs. The economics are compelling:[17]

- GPU server cost: ~$30,000–$50,000 per GPU (current-generation, high-end AI accelerator with 192GB HBM3E).
- 15.36TB Micron 9550 PRO SSD: ~$2,000–$3,000.
- SSD cost as % of GPU: ~5–8%.

Without SSD offload, serving 100K contexts requires keeping all KV caches in GPU memory, which limits concurrent users or requires additional GPUs. With SSD offload, a single $2–$3K SSD enables 10–100x more concurrent long-context requests per GPU, dramatically improving GPU utilization and reducing cost per query.

Data center scale projections

At data center scale (1,000+ GPUs), the deployment model is straightforward: One high-performance NVMe SSD per GPU for mid-size models, scaling to 2–4 SSDs per GPU for larger 70B+ models.

Table 7: Data center scale SSD requirements[18]

Scale	GPUs	SSD capacity	SSD bandwidth	SSDs
Single Node	1	1.2TB	0.5 GB/s	1
Small Cluster	8	9.4TB	3.9 GB/s	8
Medium Cluster	64	75.0TB	31.2 GB/s	64
Large Cluster	1,000	1,171.9TB	488.3 GB/s	1,000
Hyperscale	10,000	11,718.8TB	4,882.8 GB/s	10,000

14. See the MRDIMM product page on micron.com to learn more about Micron MRDIMMs.
15. See the Micron 9550 SSD product page on micron.com and the Micron 9650 product page also on micron.com to learn more about these high-performance SSDs.
16. Source: Micron DCWE measured performance data and tiering analysis.
17. Prices noted are examples only and are based on open market estimates at the time of this document's publication. Actual prices may vary.
18. Source: Micron DCWE scaling projections based on measured single-GPU data.

Networked data lakes: Micron 6600 ION SSD

For the coldest tier, RAG contexts and persistent KV caches spanning days to weeks — Micron's 6600 ION SSD provides the capacity and cost-efficiency needed for network-attached data lakes. While bandwidth is lower than local NVMe, the 6600 ION's high capacity and endurance make it ideal for storing the vast volumes of context data that agentic AI systems accumulate over time.

The road ahead

The AI industry is converging on a clear conclusion: A multi-tier memory and storage hierarchy is essential for scaling AI inference. The frameworks are already responding: vLLM's LMCache, SGLang's RadixAttention, and TensorRT-LLM all support tiered KV cache management. High-performance memory and storage are becoming as essential to AI inference infrastructure as GPU compute itself.

Context lengths will continue to grow. Reasoning models will expand effective token counts. Agentic AI will demand persistent, reusable context stores. A multi-tier approach to scaling AI for general use requires significant growth in memory and storage capabilities, over and above software and hardware improvements.

The five optimization directions for KV cache

The industry is pursuing five complementary directions to address the KV cache challenge:

- KV size reduction: Quantization, compression, and pruning to reduce the bytes stored per token
- KV bandwidth reduction: Grouped-query attention, flash-style kernels, and better memory layouts to reduce HBM bandwidth pressure
- KV placement and tiering: GPU ↔ CPU ↔ NVMe offload, unified memory, and paged attention to extend effective KV capacity
- KV sharing and reuse: Prefix caching, prompt caching, and session continuity to avoid redundant KV computation
- KV-aware scheduling: Continuous batching, admission control, and eviction policies to optimize KV utilization

These directions are not mutually exclusive. A production inference deployment will likely combine multiple approaches — using quantization to reduce per-token KV size, tiered memory/storage to extend capacity, prefix caching to avoid redundant computation, and intelligent scheduling to maximize GPU utilization.

Open problems

- Standardized KV formats: Interoperability between runtimes and accelerators could enable more tooling and portability
- Beyond vanilla transformers: Architectures that reduce attention's dependence on full-history KV (hybrids, recurrence, state-space components)
- Quality-aware KV compression: Adaptive precision based on layer, head, or token importance
- Operational maturity: Better observability for KV cache — hotspot detection, paging rates, fragmentation, and cache hit rates
- KV for agentic workloads: Long-running tool loops need better eviction, summarization, and state handoff
- Cross-tier orchestration: Intelligent policies for when and how to move KV cache data between HBM, DDR5, NVMe, and network storage tiers

Conclusion

KV cache is no longer a behind-the-scenes implementation detail, it's a first-class systems problem that shapes cost, latency, and product features like long context and chat continuity. A good optimization plan starts by identifying whether you're constrained by memory capacity, bandwidth, or serving dynamics, then choosing the smallest set of changes that move your p99 and cost-per-token in the right direction.

The evidence is clear: A tiered memory and storage hierarchy is essential for production AI inference. Micron's DCWE team has validated NVMe SSD-based KV cache offload across 318 test configurations with zero throughput penalty. But the SSD tier is just one part of the solution — optimizing the full hierarchy from HBM to network storage is what enables AI at scale.

Algorithmic improvements like TurboQuant and MLA continue to reduce KV cache per token, but these gains are continuously reinvested in longer contexts and more concurrent users. The demand for memory and storage capacity is not diminishing — it is accelerating.

In a world where inference is the business, memory and storage don't just support inference — they drive inference economics. The companies that optimize the entire memory and storage hierarchy will define who can serve AI at scale, at speed, and at profit.

Micron is building that hierarchy, from HBM4 at the GPU, to DDR5 and MRDIMM for main memory, to the Micron 9650 and Micron 9550 for context memory storage, to the Micron 6600 ION for networked data lakes. Every tier of the inference memory hierarchy is a Micron product.

Author Bio

Karthik Ganesan is a Technical Fellow at Micron Technology, leading storage solutions architecture for the Core Data Center Business Unit. He is a seasoned technologist with deep expertise in storage systems, performance optimization, and systems design. At Micron, he focuses on next-generation storage innovations that power AI, cloud, and enterprise workloads.

Additional references

"Training Compute-Optimal Large Language Models" (arxiv.org, Chinchilla scaling laws).

"Algorithmic Progress in Language Models" (epoch.ai).

Epoch AI. (2025). "Trends in AI Inference" epochai.org/research/trends-in-ai-inference

"The Future of AI Workloads" (mckinsey.com).

"TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate." (arxiv.org).

"An Empirical 100 Trillion Token Study with OpenRouter" (arxiv.org) discusses public usage statistics for reasoning model token ratios.

micron.com/ai

©2026 Micron Technology, Inc. All rights reserved. All information herein is provided on an "AS IS" basis without warranties of any kind, including any implied warranties, warranties of merchantability or warranties of fitness for a particular purpose. Micron, the Micron logo, and all other Micron trademarks are the property of Micron Technology, Inc. All other trademarks are the property of their respective owners. Products are warranted only to meet Micron's production data sheet specifications. Products, programs and specifications are subject to change without notice. Rev. A 10/2026 CCM004-1681249710-11908