AI Memory Architecture for Large Language Models

From Context Windows to Persistent Intelligence: A Comprehensive Technical Survey

Anjan Goswami

General Manager, SmartInfer LLC

January 2026

Abstract: Memory is the foundation enabling AI systems to retain, recall, and leverage information across interactions. This comprehensive survey examines memory architectures for large language models through the lens of the 3D-8Q taxonomy proposed by Wu et al. [1], which classifies memory along three dimensions: object (personal vs. system), form (parametric vs. non-parametric), and time (short-term vs. long-term). We analyze major architectural paradigms including virtual context management (MemGPT/Letta) [2], neural long-term memory (Titans) [3], retrieval-augmented generation [4], and graph-based memory systems [5]. The survey covers KV cache optimization techniques achieving 96%+ memory utilization through PagedAttention [6] and FlashAttention [7], production systems like Mem0 demonstrating 26% accuracy gains [8], and emerging neural memory modules from Meta [9] that add 128B parameters without proportional compute. We examine NVIDIA's open-source infrastructure including TensorRT-LLM, Dynamo, and disaggregated serving architectures processing 100+ billion tokens daily [10]. The analysis reveals that learned memory modules can outperform retrieval-based approaches for long-context reasoning while the convergence of neural memory with hardware-optimized KV management represents the next frontier in LLM serving infrastructure.

1. Introduction

Memory is the process of encoding, storing, and retrieving information, allowing humans to retain experiences, knowledge, skills, and facts over time. In the era of large language models (LLMs), memory refers to the ability of an AI system to retain, recall, and use information from past interactions to improve future responses [1]. The development of sophisticated memory architectures has become critical for enabling LLMs to maintain coherent identities across sessions, accumulate knowledge over time, and reason over million-token contexts.

This survey provides a comprehensive analysis of memory architectures for LLMs, covering foundational taxonomies, architectural paradigms, optimization techniques, and production systems. We examine both the theoretical foundations drawn from cognitive science and the practical engineering challenges of implementing memory at scale. The survey synthesizes research from academic institutions, major technology companies, and open-source communities to provide actionable insights for researchers and practitioners building memory-augmented AI systems.

The key contributions of this survey include: (1) a detailed analysis of the 3D-8Q memory taxonomy [1] with mappings to production systems; (2) comparative analysis of neural memory modules versus retrieval-augmented approaches; (3) comprehensive coverage of KV cache optimization techniques; (4) survey of NVIDIA's open-source memory infrastructure; and (5) identification of open problems and future research directions.

2. Foundational Taxonomy and Theoretical Framework

2.1 Human-AI Memory Parallels

The theoretical foundation for AI memory draws from the Atkinson-Shiffrin Multi-Store Model [11], which segments human memory into sensory register, short-term store (working memory), and long-term store. Each component finds direct analogs in modern LLM architectures.

Sensory memory in humans briefly buffers incoming perceptual data before processing. In LLMs, this corresponds to input tokenization and embedding—the initial conversion of text, images, or audio into machine-processable representations. Like human sensory memory, unprocessed tokens are discarded without additional attention.

Working memory operates through the transformer's attention mechanism [12], which functions as the "central executive" orchestrating information prioritization. The KV cache serves as an episodic buffer, integrating multimodal information during inference. Critically, working memory capacity constraints in both humans ($\sim 7 \pm 2$ items) and LLMs (finite context windows) create similar bottlenecks requiring sophisticated management strategies.

Explicit memory divides into episodic and semantic components. AI episodic memory stores user-specific interactions and preferences (ChatGPT Memory [13], MemoryBank [14]), while semantic memory encodes factual knowledge within model parameters through training. The key difference: human episodic memory degrades gracefully over time following the Ebbinghaus curve, while AI systems require explicit decay mechanisms to achieve similar behavior [14].

Implicit/procedural memory in AI encompasses learned task execution patterns—Voyager's skill library stores refined procedures for Minecraft tasks [15], while ReAct's thought-action-observation loops encode conditioned responses to environmental states [16].

2.2 The 3D-8Q Memory Taxonomy

Wu et al. [1] introduced the Three-Dimensional, Eight-Quadrant (3D-8Q) Memory Taxonomy, establishing the first comprehensive classification framework for LLM memory systems. The taxonomy spans three orthogonal dimensions:

Table 1: Three Dimensions of AI Memory Classification

Dimension	Axis A	Axis B	Distinguishing Factor
Object	Personal	System	User personalization vs. internal reasoning
Form	Parametric	Non-parametric	Model weights vs. external databases
Time	Short-term	Long-term	Session coherence vs. cross-session persistence

The intersection of these dimensions creates eight distinct quadrants, each representing a fundamentally different memory paradigm:

- **Quadrant I** (Personal + Non-parametric + Short-term): Working memory supporting multi-turn dialogue coherence. Representative systems include ChatGPT, Claude, and Gemini.

- **Quadrant II** (Personal + Non-parametric + Long-term): Episodic memory enabling personalization across sessions, including Mem0 [8], MemoryScope, and ChatGPT Memory [13].
- **Quadrant III** (Personal + Parametric + Short-term): Cached working memory for acceleration through Prompt Cache [17] and Contextual Retrieval [18].
- **Quadrant IV** (Personal + Parametric + Long-term): Semantic memory through personalized fine-tuning, including Character-LLM and MemoRAG.
- **Quadrant V** (System + Non-parametric + Short-term): Reasoning working memory storing intermediate outputs via ReAct [16], Reflexion [19], and Chain-of-Thought [20].
- **Quadrant VI** (System + Non-parametric + Long-term): Procedural memory capturing historical experience through Voyager [15], ExpeL [21], and Buffer of Thoughts [22].
- **Quadrant VII** (System + Parametric + Short-term): KV cache management for computational efficiency via vLLM [6], H2O [23], and PagedAttention.
- **Quadrant VIII** (System + Parametric + Long-term): Foundational knowledge encoded in parameters, including MemoryLLM [24], WISE [25], and Titans [3].

3. Architectural Paradigms

3.1 Context Window as Memory

The transformer attention mechanism operates as an associative memory block where key-value associations are stored and retrieved through pairwise similarity computation [12]. During inference, attention computes:

$$y_i = \sum_j \frac{\exp(Q_i \cdot K_j/\sqrt{d})}{\sum_l \exp(Q_i \cdot K_l/\sqrt{d})} \cdot V_j \tag{1}$$

The KV cache stores intermediate Key and Value tensors across all layers and attention heads as tensors with dimensions $[\text{batch}, \text{layers}, \text{heads}, \text{seq\_len}, d_k/d_v]$. Memory consumption follows: $2 \times L \times h \times t \times (d_k + d_v) \times \text{precision\_bytes}$ where L=layers, h=heads, t=tokens.

The quadratic scaling challenge ($O(n^2)$ complexity) emerges because self-attention computes attention values for every pair of tokens. While KV caching transforms per-step complexity from $O(n^2)$ to $O(n)$ for generation, total memory still grows linearly with sequence length, and the initial prefill remains quadratic [26].

3.2 Virtual Context Management: The MemGPT Paradigm

Packer et al. [2] introduced MemGPT by drawing direct inspiration from operating system virtual memory management. The architecture creates an illusion of unlimited context through hierarchical memory tiers and intelligent paging.

Main Context (Tier 1) functions as RAM—a fixed-size working context containing core memory blocks (persona describing agent identity, human describing user information) always passed to the LLM during inference. This tier is explicitly bounded by the model's context window.

External Context (Tier 2) functions as disk storage, comprising:

- **Recall Memory**: Conversation history stored for retrieval with pagination
- **Archival Memory**: Unlimited long-term storage using vector databases (Chroma/pgvector) with semantic search

The innovation lies in self-editing memory via function calls. The LLM actively manages its own context through tools like `core_memory_append()`, `archival_memory_insert()`, and `archival_memory_search()`. The heartbeat mechanism enables proactive behavior—when `request_heartbeat=true` in function output, the agent continues execution without user intervention, enabling multi-step autonomous reasoning.

Performance characteristics show document QA performance remains unaffected by increased context length, while multi-session chat enables persistent, evolving conversational agents. MemGPT operates 10-30× cheaper and 6-13× faster than iterative retrieval methods like IRCoT [2].

3.3 Neural Long-Term Memory: The Titans Architecture

Behrouz et al. [3] from Google Research introduced Titans, a paradigm shift where the memory module is a multi-layer perceptron whose parameters *are* the memory. Unlike static models, Titans updates these weights during inference using a surprise-based memorization mechanism—events that violate expectations (measured by high gradients) are more memorable.

The memory update formulation:

$$M_t = (1 - \alpha_t) \cdot M_{t-1} + S_t \tag{2}$$

$$S_t = \eta_t \cdot S_{t-1} - \theta_t \cdot \nabla \ell(M_{t-1}; x_t) \tag{3}$$

$$\ell(M; x) = \|M(k_t) - v_t\|_2^2 \tag{4}$$

Where S_t represents the surprise metric combining past surprise (momentum) and momentary surprise (current gradient), η_t controls surprise decay, θ_t sets the learning rate for momentary surprise, and α_t implements the forgetting gate.

Titans offers three architectural variants:

- **MAC (Memory as Context)**: Segments sequences into chunks, retrieves from long-term memory as a context prefix, then applies attention.
- **MAG (Memory as Gate)**: Parallel processing through sliding window attention (short-term) and neural memory (long-term), combined via learned gating with output: $o = y \otimes M(\tilde{x})$.
- **MAL (Memory as Layer)**: Sequential processing where memory compresses context before attention.

Performance results demonstrate 2M+ token context scaling with higher accuracy than competitors in needle-in-haystack tasks. On the BABILong benchmark [27], Titans outperforms GPT-4 and Llama 3.1-70B despite having $\sim$70$\times$ fewer parameters.

3.4 Retrieval-Augmented Memory

Retrieval-Augmented Generation [4] combines information retrieval with generative models through a four-stage pipeline: indexing (documents $\rightarrow$ chunks $\rightarrow$ embeddings $\rightarrow$ vector database), retrieval (query embedding $\rightarrow$ similarity search $\rightarrow$ top-K documents), augmentation (documents injected into prompt), and generation (LLM produces answer using augmented context).

Dense retrieval using transformer encoders (BGE, E5, sentence-transformers) captures semantic similarity via cosine distance in embedding space—handling synonyms and conceptual matching but requiring substantial storage and potentially retrieving "semantic lookalikes." Sparse retrieval (BM25/TF-IDF) provides exact keyword matching via inverted indexes—fast, interpretable, and memory-efficient but blind to semantic relationships.

Hybrid retrieval combines both approaches:

$$\text{score} = \alpha \times \text{dense\_score} + (1 - \alpha) \times \text{sparse\_score} \tag{5}$$

with typical $\alpha = 0.5$–0.7. IBM research demonstrates three-way retrieval (BM25 + dense + sparse vectors) achieves 10–30% precision improvements over single-method approaches.

3.5 Graph-Based Memory Systems

HippoRAG [5] draws from hippocampal indexing theory, orchestrating LLMs, knowledge graphs, and Personalized PageRank (PPR) to enable multi-hop reasoning. The architecture comprises three components: an artificial neocortex (LLM for language processing and knowledge extraction), a parahippocampal region (embedding model for entity detection and synonymy linking), and an artificial hippocampus (open knowledge graph storing entity-relation-entity triples).

The key innovation: PPR propagates relevance through entity relationships, enabling single retrieval steps to achieve multi-hop reasoning that standard RAG cannot accomplish. Results show up to 20% improvement over state-of-the-art RAG on multi-hop QA while being 10–30$\times$ cheaper and 6–13$\times$ faster than iterative approaches [5].

Mem0's graph extension [8] represents memory as a directed labeled graph $G = (V, E, L)$ where nodes represent entities with types, embeddings, and metadata, while edges encode relationships as triplets. This hybrid datastore combines vector databases for semantic similarity with graph backends (Neo4j, Memgraph, Neptune) for relational structure.

4. Memory Operations and Lifecycle Management

4.1 Memory Encoding and Construction

Modern LLM memory systems extract facts through LLM-driven extraction—using language models to identify relevant facts, preferences, and contextual information from conversations [8]. The process generates structured outputs (subject-predicate-object triples), automatically scores importance, and classifies memory types (preference, personal, skill, goal, opinion, experience).

Hierarchical summarization through Reflective Memory Management (RMM) [28] creates multi-level memory structures: prospective reflection dynamically summarizes interactions across granularities (utterances, turns, sessions), while topic-based memory management extracts dialogue snippets with summaries based on distinct topics.

4.2 Memory Update and the Impossible Triangle

Wang et al. [25] identified the impossible triangle in lifelong model editing—three competing objectives that cannot be simultaneously achieved:

- **Reliability**: Model remembers both current and previous edits after sequential editing
- **Locality**: Editing doesn't influence irrelevant pretrained knowledge
- **Generalization**: Model understands edits and generalizes to different query forms

WISE (Rethinking Knowledge Memory for Lifelong Model Editing) [25] bridges this gap through dual parametric memory:

$$\text{WISE} = \text{Main Memory (pretrained)} + \text{Side Memory (edited)} \tag{6}$$

The architecture employs: (1) Knowledge Sharding—different edit sets reside in distinct parameter subspaces using orthogonal subspace projection; (2) Router Mechanism—trained classifier directs queries to appropriate memory; (3) Knowledge Merging—shards periodically merge into shared memory via TIES merging.

4.3 Forgetting Mechanisms

H2O (Heavy-Hitter Oracle) [23] exploits a key observation: accumulated attention scores follow power-law distribution—approximately 20% of tokens contribute most attention value. With 20% heavy hitters retained, H2O achieves up to 29× throughput versus DeepSpeed Zero-Inference while maintaining generation quality.

StreamingLLM [29] discovered the attention sink phenomenon—initial tokens receive disproportionate attention regardless of semantic importance due to softmax normalization requiring attention scores sum to 1. The solution: always retain 4 attention sink tokens plus a sliding window of recent tokens, enabling processing of 4M+ tokens with constant memory usage and 22.2× speedup over sliding window recomputation.

5. KV Cache Optimization and GPU Memory Hierarchies

5.1 PagedAttention and Memory Utilization

vLLM's PagedAttention [6] revolutionized KV cache management by borrowing operating system virtual memory concepts. Instead of contiguous allocation, the KV cache is partitioned into small fixed-size blocks (typically 16 tokens, ∼12.8 KB for a 13B model). A block table maps logical (contiguous) blocks to non-contiguous physical locations in GPU memory, with blocks allocated on-demand as new tokens are generated.

The results are transformative: memory utilization reaches 96%+ (versus 40% previously), throughput improves 2–24× over HuggingFace Transformers, and copy-on-write sharing reduces memory overhead by up to 55% for beam search [6].

5.2 FlashAttention Memory Optimization

FlashAttention [7] fundamentally changed attention computation by exploiting the GPU memory hierarchy. Rather than materializing the $N \times N$ attention matrix in HBM, FlashAttention tiles the computation to fit entirely in SRAM, computing attention block-by-block with online softmax normalization. Memory complexity drops from $O(N^2)$ to $O(N)$—enabling 64K+ token contexts without approximation.

FlashAttention-3 [30] leverages Hopper-specific features: WGMMA tensor core instructions, hardware-accelerated TMA memory transfers, warp-specialization for overlapping compute and data movement, and FP8 support reaching 1.2 PFLOPS. GPU utilization reaches ∼75%, roughly doubling FlashAttention-2 performance.

5.3 Quantization and Compression

KIVI [31] achieves 2.6× peak memory reduction with INT4 per-channel key quantization and per-token value quantization, enabling 4× larger batch sizes. The insight: Key cache requires per-channel quantization (outliers concentrated in specific channels) while Value cache requires per-token quantization (no consistent outlier pattern).

SqueezeAttention [32] introduces layer-wise budget allocation—measuring layer importance via cosine similarity before/after self-attention, then categorizing layers into groups with different KV budgets. This achieves 30–70% memory reduction with up to 2.2× throughput improvement.

5.4 GPU Memory Hierarchy

Modern GPUs present a multi-tiered memory system critical for LLM inference:

Table 2: GPU Memory Hierarchy for LLM Inference (H100 Reference)

Tier	Capacity	Bandwidth	Primary Use
Registers	∼256 KB/SM	∼8 TB/s	Active operands
Shared Memory/L1	128–256 KB/SM	15–20 TB/s	FlashAttention tiling
L2 Cache	50 MB (H100)	∼3 TB/s	Hot KV entries
HBM3	80 GB	3.35 TB/s	Model weights, KV cache
CPU DRAM	256–512+ GB	32–900 GB/s	Overflow storage
NVMe SSD	Terabytes	5–14 GB/s	Large context offload

NVIDIA's generational improvements address the memory wall: H200 increases HBM capacity by 76% (80GB → 141GB) and bandwidth by 43% (3.35 → 4.8 TB/s) [33]. Blackwell B100/B200 introduces dual-die design with 10 TB/s chip-to-chip interconnect, native FP4 precision for KV cache, and 5th-generation Tensor Cores with 2× attention acceleration [34].

6. Neural Memory as Alternative to RAG

6.1 Memory Layers at Scale

Meta's Memory Layers at Scale [9] takes a different approach: trainable key-value lookup mechanisms that add massive parameter counts without proportional compute increases. Using product-key quantization to avoid $O(N)$ key comparisons, these layers achieve 128 billion memory parameters with custom CUDA kernels delivering 3 TB/s memory bandwidth—nearly 8× faster than baseline PyTorch implementations.

The results are striking: a 1.3B parameter model with 64M memory keys achieves 168% improvement on Natural Questions (7.76% → 20.78% accuracy) and approaches Llama2-7B performance with 10× fewer FLOPs. An 8B model trained on just 1T tokens matches models trained on 15T tokens in factual recall tasks.

6.2 MemoryLLM Self-Updating Models

MemoryLLM [24] embeds a fixed-size memory pool within the transformer's latent space, enabling true self-update of model parameters. The architecture combines a 7B parameter Llama2 backbone with approximately 1B parameters dedicated to memory (30 blocks × 256 tokens). Critically, the system maintains no degradation after ∼1 million memory updates, with exponential decay ensuring old knowledge is gracefully forgotten while new information is absorbed.

The 2025 extension M+ [35] combines a co-trained retriever with latent memory, extending effective context to 160K tokens—8× the original capacity—while maintaining the self-updatable property.

6.3 Comparison: RAG versus Learned Memory

Table 3: RAG vs. Learned Memory Modules Comparison

Dimension	RAG	Learned Memory (Titans/Meta)
Latency	Variable (retrieval + LLM)	Consistent (single forward pass)
Factual accuracy	Retriever-dependent	Up to 168% improvement
Maximum context	Limited by retriever + LLM	2M+ tokens demonstrated
Update cost	Low (index new docs)	High (requires training)
Multi-hop reasoning	Weak (retriever bottleneck)	Strong (BABILong results)
Interpretability	High (can cite sources)	Lower (weights encode knowledge)

Learned memory excels when: the same facts are queried repeatedly (amortizing training cost), long-context reasoning is required (where RAG struggles with multi-hop), latency is critical (no retrieval overhead), or reasoning must be deeply integrated.

7. NVIDIA Open-Source Memory Infrastructure

7.1 TensorRT-LLM

TensorRT-LLM [36] provides the foundational inference engine with sophisticated memory management. Key features include paged KV cache (configurable block sizes from 8–128 tokens), in-flight batching for continuous request processing, FP8/INT8/INT4 quantization support, and host memory offloading via configurable parameters.

Memory is managed through three major contributors: weights (fixed based on model/precision/parallelism), activation tensors (pre-computed at engine build time with TensorRT's liveness-based optimization), and KV cache (default 90% of remaining GPU memory).

7.2 NVIDIA Dynamo

NVIDIA Dynamo [37] addresses distributed inference across GPU clusters through four key components:

- **KV Cache Manager (KVBM)**: Implements cost-aware offloading across memory hierarchies (HBM → DRAM → SSD → network storage) with intelligent eviction policies.
- **Smart Router**: Tracks KV cache location across clusters, calculating overlap between new requests and cached blocks to route requests for maximum cache hit rate—achieving 3× improvement in time-to-first-token and 2× reduction in average latency.
- **NIXL**: NVIDIA Inference Transfer Library providing unified API for high-throughput, low-latency communication across NVLink, InfiniBand, RoCE, and Ethernet.
- **GPU Resource Planner**: Uses SLA-based planning to determine optimal prefill/decode configurations with dynamic scheduling.

7.3 Disaggregated Serving Architectures

DistServe [38] separates LLM inference into distinct GPU pools—prefill instances generate KV cache from prompts while decode instances handle autoregressive token generation. Results show 7.4× more requests served within the same SLO and 12.6× tighter SLO achievable versus vLLM baseline.

Mooncake [10] from Moonshot AI powers the Kimi service, processing over 100 billion tokens daily across thousands of nodes. Its KV-cache-centric architecture features a disaggregated cache pool spanning CPU DRAM, SSDs, and remote RDMA storage; a Conductor scheduler routing requests based on KV cache distribution; and custom transfer engines achieving 2.4× faster RDMA transfers.

8. Production Systems and Benchmarks

8.1 Mem0 Production Performance

Chhikara et al. [8] introduced Mem0 as a production-ready memory layer for AI agents. The architecture implements a two-phase pipeline: an extraction phase processing message pairs to identify salient candidate facts, and an update phase comparing each candidate to existing memories via vector similarity with an LLM determining ADD, UPDATE, DELETE, or NOOP operations.

Benchmark results on LOCOMO [39] demonstrate 26% relative accuracy improvement over OpenAI's memory implementation, with Mem0's graph variant achieving ∼2% additional improvement. The system delivers 91% lower p95 latency versus full-context baselines with >90% token cost savings.

8.2 A-MEM Zettelkasten-Inspired Memory

A-MEM [40] implements the Zettelkasten note-taking methodology for LLM memory. Each memory contains content, timestamp, LLM-generated keywords, tags, context descriptions, dense embedding, and links. Results demonstrate 85–93% token reduction compared to baselines with particularly strong multi-hop reasoning performance.

8.3 Evaluation Benchmarks

- **LOCOMO** [39]: Long-context conversational memory with 10–50 conversations spanning up to 35 sessions. Key finding: even best systems lag human levels by 56%, with temporal reasoning gap reaching 73%.
- **BABILong** [27]: Extends bAbI benchmark to 20 reasoning tasks scalable to 50M tokens. Critical finding: LLMs utilize only 10–20% of context effectively.
- **MemoryBench** [41]: Evaluates memory and continual learning in LLM systems.

9. Open Problems and Future Directions

9.1 Current Challenges

Key limitations remain across memory architecture research:

- **Scalability**: Million+ token contexts with acceptable latency
- **Accuracy-Efficiency Trade-off**: Compression without quality degradation
- **Cross-Session Persistence**: Maintaining coherent long-term user models
- **Privacy and Security**: Protecting sensitive information in memory systems
- **Multimodal Integration**: Unified memory across text, images, audio, video
- **Memory Hallucination**: Preventing false memory injection and retrieval errors

9.2 Emerging Research Directions

Wu et al. [1] identify six evolutionary directions:

1. **Unimodal → Multimodal Memory**: Cross-modal memory encoding and retrieval
2. **Static → Stream Memory**: Real-time, continuous memory updates
3. **Specific → Comprehensive Memory**: Integrated multi-system architectures
4. **Exclusive → Shared Memory**: Multi-agent and cross-domain knowledge transfer
5. **Individual → Collective Privacy**: Group-level privacy frameworks
6. **Rule-based → Automated Evolution**: Self-improving memory systems

9.3 Memory Scaling Laws

Lu et al. [42] found that LLM fact knowledge capacity scales linearly with model size but negative exponentially with training epochs. Critical implication: memorizing all Wikidata facts would require 1000B parameters for 100 epochs—practically infeasible with current approaches. Unlike compute scaling laws, no comprehensive memory-specific power laws exist relating memory capacity $\times$ retrieval accuracy $\times$ compute.

10. Conclusion

Memory architecture for LLMs has evolved from simple context windows to sophisticated multi-tier systems that increasingly mirror human cognitive organization. The 3D-8Q taxonomy [1] provides essential conceptual clarity, revealing that effective memory must span personal/system objects, parametric/non-parametric forms, and short/long-term persistence. Production systems like Mem0 [8] demonstrate these principles yield measurable gains (26% accuracy improvement, 91% latency reduction), while architectural innovations like Titans [3] prove that 2M+ token contexts are achievable through test-time memorization.

Three insights emerge as particularly significant: (1) The impossible triangle is navigable through dual parametric memory with knowledge sharding [25]; (2) Intelligent forgetting proves as important as remembering—H2O's heavy-hitter eviction [23] and StreamingLLM's attention sinks [29] enable efficient scaling; (3) Hybrid architectures combining neural memory with paged allocation, learned compression with hardware-aware tiling, and disaggregated serving with intelligent memory tiering consistently outperform single-paradigm systems.

The convergence of learned memory modules with hardware-optimized KV management represents the next frontier in LLM serving infrastructure. Systems treating memory as a first-class schedulable resource—unified across learned, cached, and retrieved modalities—will define the next generation of memory-efficient AI systems.

Author's Note

This survey synthesizes research from the rapidly evolving field of AI memory architecture. The landscape continues to change quickly, with new architectures and optimizations emerging regularly. Readers are encouraged to consult the cited papers for the latest developments and implementation details.

About the Author

Anjan Goswami is Head of AI and Data at Microsoft PowerPoint, leading Copilot AI strategy and applied science initiatives. He has 20+ years of AI leadership experience across major technology companies including Adobe, Salesforce, Walmart, Amazon A9, and eBay. He holds a PhD in Computer Science from UC Davis and has 10+ patents in ranking and search

systems.

References

[1] Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *arXiv preprint arXiv:2504.15965*, 2025. Huawei Noah's Ark Lab.

[2] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

[3] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024. Google Research.

[4] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[5] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *arXiv preprint arXiv:2405.14831*, 2024.

[6] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[7] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[8] Prateek Chhikara et al. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

[9] Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, et al. Memory layers at scale. *arXiv preprint arXiv:2412.09764*, 2024. Meta AI Research.

[10] Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: A kvcache-centric disaggregated architecture for llm serving. *arXiv preprint arXiv:2407.00079*, 2024.

[11] Richard C Atkinson and Richard M Shiffrin. Human memory: A proposed system and its control processes. *Psychology of Learning and Motivation*, 2:89–195, 1968.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[13] OpenAI. Memory and new controls for chatgpt. https://openai.com/blog/memory-and-new-controls-for-chatgpt, 2024.

[14] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38:19724–19731, 2024.

[15] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[16] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[17] In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.

[18] Anthropic. Introducing contextual retrieval. https://www.anthropic.com/news/contextual-retrieval, 2024.

[19] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[20] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[21] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. 38:19632–19642, 2024.

[22] Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. *arXiv preprint arXiv:2406.04271*, 2024.

[23] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

[24] Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, et al. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*, 2024.

[25] Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *arXiv preprint arXiv:2405.14768*, 2024.

[26] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624, 2023.

[27] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *arXiv preprint arXiv:2406.10149*, 2024.

[28] Zhen Tan, Jun Yan, I Hsu, Rujun Han, Zifeng Wang, Long T Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, et al. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. *arXiv preprint arXiv:2503.08026*, 2025.

[29] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[30] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*, 2024.

[31] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *Proceedings of Machine Learning Research*, 2024.

[32] Zihao Wan et al. Squeezeattention: 2d management of kv-cache in llm inference via layer-wise optimal budget. *arXiv preprint arXiv:2404.04793*, 2024.

[33] NVIDIA Corporation. Nvidia h200 tensor core gpu architecture. https://www.nvidia.com/en-us/data-center/h200/, 2024.

[34] NVIDIA Corporation. Nvidia blackwell architecture technical brief. https://www.nvidia.com/en-us/data-center/technologies/blackwell-architecture/, 2024.

[35] Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending memoryllm with scalable long-term memory. *arXiv preprint arXiv:2502.00592*, 2025.

[36] NVIDIA Corporation. Tensorrt-llm: High-performance inference for large language models. https://github.com/NVIDIA/TensorRT-LLM, 2024.

[37] NVIDIA Corporation. Nvidia dynamo: A low-latency distributed inference framework. https://github.com/ai-dynamo/dynamo, 2024.

[38] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 193–210, 2024.

[39] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[40] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

[41] Zeyu Zhang et al. Memorybench: A benchmark for memory and continual learning in llm systems. *arXiv preprint arXiv:2510.17281*, 2024.

[42] Xingyu Lu et al. Scaling laws for fact memorization of large language models. *Findings of EMNLP*, 2024.