Semantic Memory State as KV Cache Replacement for Unbounded Context in Transformer Inference

İsa Kara

Unknown Evola Mind (UEM) Project

Abstract

Current transformer-based large language models (LLMs) face a fundamental scalability bottleneck: the Key-Value (KV) cache grows linearly with conversation length, consuming O(n) memory where n is the number of tokens. This paper proposes Semantic Memory State (SMS), a paradigm shift that replaces token-level KV caching with a high-level semantic representation of conversational context. By maintaining a structured, continuously-updated memory state instead of raw KV vectors, SMS achieves O(1) memory complexity while preserving contextual understanding. Compared to traditional KV caching, theoretical analysis suggests memory reduction of 50-99% versus INT8 quantization, and 0-98% versus INT4 quantization—with benefits increasing proportionally to conversation length. enabling effectively unbounded conversation length without degradation in inference speed. This approach draws inspiration from human cognitive architecture, where semantic meaning is retained while verbatim token sequences are discarded.

1. Introduction

Large language models based on the transformer architecture have achieved remarkable capabilities in natural language understanding and generation. However, their deployment at scale faces a critical challenge: the Key-Value (KV) cache mechanism, while essential for efficient autoregressive generation, creates a memory bottleneck that grows linearly with sequence length.

Consider a typical production scenario: a 70-billion parameter model serving concurrent users with 128,000-token context windows. Each token requires approximately 160KB of KV cache storage across all layers. A single conversation at maximum context consumes over 20GB of GPU VRAM solely for KV storage—memory that could otherwise serve additional users or enable larger batch sizes.

Current approaches to this problem—Grouped-Query Attention (GQA), Multi-Query Attention (MQA), PagedAttention, and sliding window mechanisms—optimize KV cache management but do not fundamentally alter its O(n) scaling behavior. This paper proposes a fundamentally different approach: replacing the KV cache entirely with a semantic memory state that maintains O(1) memory complexity regardless of conversation length.

2. Background

2.1 KV Cache Mechanics

In transformer-based autoregressive generation, each new token must attend to all previous tokens. Without caching, this would require recomputing the Key (K) and Value (V) projections for all previous tokens at each generation step, resulting in $O(n^2)$ computational complexity. The KV cache stores these projections, reducing generation to O(n) per token.

For a model with L layers, H attention heads, and head dimension d, the KV cache size per token is:

$$KV\_size = 2 \times L \times H \times d \times bytes\_per\_element$$

For a representative 70B parameter model (L=80, H=8 with GQA, d=128, INT8 quantization), this yields approximately 160KB per token. A 128,000-token context therefore requires over 20GB of VRAM solely for KV storage.

2.2 The Layer Hierarchy Problem

A critical aspect often overlooked in KV cache discussions is that each layer maintains separate K and V representations. This is not redundancy—each layer operates in a different representational space:

- Early layers (0-20): Capture syntactic structure, positional relationships, surface-level patterns
- Middle layers (20-50): Encode semantic relationships, entity recognition, contextual meaning
- Deep layers (50-80): Perform reasoning, abstraction, and task-specific computation

When generating a new token, the model must attend to previous tokens at each layer independently. The Query vector at layer 5 cannot meaningfully attend to Key vectors from layer 79—they exist in incompatible representational spaces. This necessitates storing the full KV hierarchy for all cached tokens.

2.3 Existing Optimization Approaches

Several techniques have been developed to mitigate KV cache memory pressure:

Grouped-Query Attention (GQA): Reduces KV heads while maintaining query heads, achieving 4-8× reduction in KV size.

PagedAttention: Manages KV cache as virtual memory pages, improving utilization from ~50% to >95%.

Sliding Window: Limits attention to recent tokens, bounding memory at cost of long-range context.

KV Cache Quantization: Reduces precision from FP16 to INT8/INT4, halving or quartering memory.

While effective, all these approaches preserve the fundamental O(n) scaling. They optimize the constant factor but do not change the asymptotic behavior.

3. The Core Insight

3.1 What KV Cache Actually Stores

The KV cache stores mathematical representations of tokens—not their meaning. Consider a conversation where the user states "My operating system is Windows 11." The KV cache stores:

- 80 layers × K vectors × V vectors for "My"
- 80 layers × K vectors × V vectors for "operating"
- 80 layers × K vectors × V vectors for "system"
- ... and so on for each token

The semantic content—"user's OS is Windows 11"—is distributed across these vectors in a form that requires the full transformer computation to decode. The KV cache is not a knowledge store; it is a computational shortcut.

3.2 Human Memory as Alternative Model

Human conversational memory operates fundamentally differently. When a person says "My operating system is Windows 11," a human listener does not store the acoustic waveform or phoneme sequence. Instead, they extract and store the semantic content: "This person uses Windows 11."

This semantic compression is lossy but intelligent—it discards verbatim representation while preserving meaning. Critically, this enables O(1) memory scaling: the semantic state "user's OS is Windows 11" occupies the same memory whether the conversation has 10 or 10,000 prior exchanges.

3.3 The Redundancy Problem

Consider two sequential user messages:

> Message 1: "What is 2+2?"
>
> Message 2: "What is 56+56?"

In the current paradigm, the KV cache for Message 1 includes full 80-layer representations for the "+" token. When Message 2 arrives, the model computes another complete 80-layer KV representation for another "+" token. Yet the semantic content—"addition operator"—is identical and already encoded in model weights.

The KV cache is storing redundant computational artifacts rather than unique contextual information. The model "knows" what "+" means from training; the KV cache merely records that it appeared at specific positions.

4. Proposed Method: Semantic Memory State

4.1 Architecture Overview

Semantic Memory State (SMS) replaces the KV cache with a structured, natural-language representation of conversational context. The core principle: instead of caching token-level computational artifacts, maintain a high-level semantic summary that captures all information necessary for coherent continuation.

System Architecture:

1. User message arrives
2. Memory Update Module processes: [Previous Memory State] + [New Message] → [Updated Memory State]
3. Response Generation receives: [System Prompt] + [Updated Memory State] + [Current Message]
4. KV cache from previous turn is discarded
5. New KV cache built only for current context window (~750 tokens)

4.2 Memory State Structure

The Memory State is a structured text representation containing mandatory fields that evolve but never disappear:

- **Interaction Stage:** Current phase (exploration/diagnosis/solution/execution/feedback)
- **User Intent:** Accumulated understanding of user's goals
- **Main Topic:** Subject matter of conversation
- **Problem/Request:** Specific issue being addressed
- **Context:** Relevant background information established
- **Current Known State:** Confirmed facts and decisions
- **Expected Next Step:** Anticipated continuation

The memory state is not a transcript—it never contains verbatim dialogue. It is a semantic compression: a higher-level representation that captures meaning without preserving exact wording.

4.3 Update Mechanism

Memory update follows a strict cumulative protocol:

1. Preserve entire existing memory state
2. Integrate new information into appropriate fields
3. Revise fields as needed without deleting prior meaning
4. Output complete updated state

Critical invariant: The memory state must pass a "standalone comprehension test"—if someone reads only the memory state, they should understand the full conversation arc from beginning to current point.

4.4 Implementation Considerations

The Memory Update Module can be implemented via several strategies:

Option A - Main Model: Use the primary model (e.g., 70B) with a separate inference call before response generation. Higher quality, additional latency.

Option B - Auxiliary Model: Use a smaller model (0.6B-8B range) specifically for memory updates. Memory update is a relatively simple task—semantic compression rather than generation—potentially suitable for smaller, efficient models.

Option C - Hybrid: Auxiliary model for routine updates; main model for complex context shifts.

5. Case Study: Reference Model Analysis

To illustrate the practical implications of SMS, we present a concrete analysis using a representative modern LLM. The following specifications reflect typical architecture choices in current open-weight models:

- Parameters: 70 billion
- Layers: 80
- KV Heads: 8 (Grouped-Query Attention)
- Head Dimension: 128
- Quantization: INT8 (1 byte per value)
- Maximum Context: 128,000 tokens

5.1 KV Cache Calculation

Per-token KV cache requirement:

$$KV\_per\_token = 2 \times L \times H \times d \times bytes$$
$$KV\_per\_token = 2 \times 80 \times 8 \times 128 \times 1 = 163{,}840\ bytes \approx 160\ KB$$

At various conversation lengths:

- 1,000 tokens: 160 MB
- 8,000 tokens: 1.28 GB
- 32,000 tokens: 5.12 GB
- 128,000 tokens (max): 20.48 GB

5.2 Inference Speed Degradation

Measured inference speeds on datacenter hardware (NVIDIA A100 80GB) with the reference 70B model:

- 1,000 tokens context: ~45 tokens/second
- 8,000 tokens context: ~35 tokens/second
- 32,000 tokens context: ~20 tokens/second
- 64,000 tokens context: ~12 tokens/second
- 128,000 tokens context: ~6-8 tokens/second

Speed degradation exceeds 80% from short to maximum context. This degradation stems from increased memory bandwidth requirements as the KV cache grows, eventually becoming the dominant bottleneck even on high-bandwidth HBM memory.

5.3 SMS Applied to Reference Model

With Semantic Memory State, the context window becomes bounded (though not strictly fixed):

- System prompt: ~300 tokens
- Memory state: ~200-500 tokens (adaptive based on task complexity)
- Current user message: ~50-500 tokens (variable)
- **Typical total: ~750-1500 tokens (bounded)**

Adaptive Memory State: The memory state size is not strictly constant. Complex tasks (debugging code, multi-step reasoning, detailed analysis) may require larger memory states to preserve critical context. However, this growth is bounded by design constraints and remains dramatically smaller than linear KV cache growth.

KV cache requirement with SMS (typical case):

$$1000\ tokens \times 160\ KB = 160\ MB\ (bounded)$$

Comparison at 128K conversation length:

- Traditional (INT8): 20.48 GB KV cache, ~6-8 t/s
- Traditional (INT4): 10.24 GB KV cache, ~10-12 t/s
- SMS (typical): 160 MB KV cache, ~40+ t/s
- SMS (complex task): 240 MB KV cache, ~35+ t/s
- **Memory reduction vs INT8: 98.4% (typical) to 97.6% (complex)**

- **Memory reduction vs INT4: 96.8% (typical) to 95.3% (complex)**

5.4 Speculative Decoding Consideration

When using speculative decoding with a draft model (typically 0.5B-4B parameter range), both models maintain KV caches. The draft model's KV cache is smaller due to fewer layers and heads, but still scales with O(n). SMS benefits both models equally, maintaining constant memory for the entire speculative decoding pipeline.

In typical speculative decoding configurations, the draft model adds 5-15% overhead to total KV cache requirements. With SMS, both main and draft model KV caches remain constant regardless of conversation length, preserving the speed benefits of speculative decoding even in extended conversations where traditional approaches would see severe degradation.

6. Theoretical Analysis

6.1 Memory Complexity

Traditional KV Cache:

$$Memory = O(n) \text{ where } n = \text{total conversation tokens}$$

For a conversation of 10,000 tokens with 160KB per token (INT8): 1.6GB

With INT4 quantization: 800MB (still O(n) scaling)

Semantic Memory State:

$$Memory = O(1) \text{ — bounded regardless of conversation length}$$

Bounded context window of ~750-1500 tokens (system prompt + adaptive memory state + current message): 120-240MB

Reduction factor vs INT8: 50-99% depending on conversation length.

Reduction factor vs INT4: 0-98% — minimal benefit for short conversations, significant for long.

Note: Even with aggressive KV cache quantization (INT4), traditional approaches maintain O(n) scaling. SMS provides bounded memory regardless of quantization applied to either approach.

6.2 Inference Speed

In traditional systems, inference speed degrades as conversation grows because:

- Each new token attends to all cached tokens: O(n) attention computation
- Full KV cache must be read from VRAM: O(n) memory bandwidth

With SMS, the effective context window remains bounded (~750-1500 tokens), yielding:

- Constant attention computation: O(1)
- Constant memory bandwidth: O(1)

Inference speed no longer degrades with conversation length. A 1000-message conversation runs at the same speed as a 10-message conversation.

6.3 Scalability Implications

Enterprise Scale (1 million concurrent users, 10K avg tokens):

Traditional (INT8): 1M × 10K tokens × 160KB = 1.6PB VRAM

Traditional (INT4): 1M × 10K tokens × 80KB = 800TB VRAM

SMS (typical): 1M × 1000 tokens × 160KB = 160TB VRAM

SMS (complex tasks): 1M × 1500 tokens × 160KB = 240TB VRAM

Infrastructure cost reduction vs INT8: ~85-90%

Infrastructure cost reduction vs INT4: ~70-80%

Note: These estimates assume average conversation length. In practice, many conversations are shorter (higher savings) while some complex tasks require extended context (lower savings). SMS provides the greatest advantage for long, multi-turn conversations.

API Cost Implications:

Current API pricing models charge per input token. Long conversations become expensive because the full history is transmitted and processed each turn. With SMS, effective input length remains bounded, potentially enabling more predictable pricing for extended conversations.

7. Related Work

Several lines of research address the memory and efficiency challenges of transformer inference. We position SMS relative to these approaches.

7.1 KV Cache Optimization

Significant effort has focused on optimizing KV cache management within the existing paradigm:

- **PagedAttention (vLLM):** Applies virtual memory concepts to reduce fragmentation, achieving >95% memory utilization.
- **GQA/MQA:** Reduces the number of KV heads, shrinking per-token cache size by 4-8×.
- **Sliding Window:** Bounds memory by attending only to recent tokens, but loses long-range context.
- **KV Cache Quantization:** INT8 reduces memory 2× vs FP16; INT4 reduces 4×. Widely deployed in production systems.

These approaches are valuable and complementary—SMS can be combined with GQA and quantization for additional gains. However, all preserve O(n) scaling. At 128K tokens, even INT4 with GQA requires gigabytes of memory. SMS achieves O(1) by replacing the paradigm entirely, though with the trade-off of lossy semantic compression.

7.2 Alternative Architectures

State Space Models, particularly Mamba, achieve O(1) memory through fixed-size hidden states that compress context selectively. Mamba demonstrates competitive performance with transformers while eliminating the KV cache entirely. However, this requires architectural change—existing transformer models cannot benefit. SMS preserves the transformer architecture, applying to any existing model without retraining or modification. The approaches are complementary: SMS could potentially be applied to hybrid Transformer-Mamba architectures.

7.3 Context Compression

Context compression methods like AutoCompressors and In-Context Former (IC-Former) learn to compress token sequences into compact vector representations. These operate at the token/embedding level, producing 'digest vectors' that replace original context. While effective, compression still scales with input length and requires specialized training. SMS operates at the semantic level using natural language, requiring no additional training—the LLM itself performs compression through its existing capabilities.

7.4 Virtual Context Management

MemGPT (Memory-GPT) introduces OS-inspired virtual context management, using hierarchical memory tiers with paging between main context and external storage. The system maintains full conversation history in archival storage, retrieving relevant segments on demand. This is the closest existing work to SMS, but differs fundamentally:

- **MemGPT:** Lossless storage with selective retrieval. Full history preserved.
- **SMS:** Lossy compression into semantic state. History discarded after extraction.
- **MemGPT:** Retrieval latency for accessing archived context.
- **SMS:** No retrieval—all context is immediately available in memory state.
- **MemGPT:** Storage requirements grow with conversation length.
- **SMS:** Constant storage regardless of conversation length.

SMS trades perfect recall for bounded memory, drawing inspiration from human cognition where semantic gist is retained while verbatim content is lost.

7.5 Positioning of SMS

SMS occupies a unique position in this landscape:

- Unlike KV optimizations: Changes the paradigm rather than optimizing within it
- Unlike Mamba: Works with existing transformer architectures
- Unlike context compression: Uses natural language, not learned vectors
- Unlike MemGPT: Eliminates storage growth through lossy semantic extraction

The core insight—that conversations can be represented as evolving semantic states rather than token sequences—has not been previously explored as a KV cache replacement strategy.

8. Discussion

8.1 Limitations and Trade-offs

Information Loss: Semantic compression is inherently lossy. Verbatim recall ("What exactly did I say in message 47?") is impossible. The system trades perfect recall for bounded memory.

Update Quality: Memory state quality depends on the update module's ability to accurately extract and integrate semantic content. Failure modes include: information omission, incorrect summarization, loss of nuance.

Task Suitability: Some tasks require token-level fidelity: code debugging with exact line references, mathematical proofs with step-by-step derivations, legal document analysis. These may require hybrid approaches or task-specific memory structures.

Adaptive Memory Growth: Memory state is not strictly constant. Complex tasks (multi-step debugging, detailed code analysis, intricate reasoning chains) require larger memory states to

preserve critical context. While bounded, this variability means memory savings depend on task complexity. Nevertheless, even in worst-case scenarios, SMS maintains O(1) scaling compared to O(n) for traditional approaches.

Comparison with Quantized KV Cache: Modern KV cache implementations support INT8 and INT4 quantization, reducing per-token memory by 2-4×. This significantly impacts SMS's relative advantage:

- **Short conversations (≤2K tokens):** SMS provides ~50% reduction vs INT8, but near-zero benefit vs INT4.
- **Medium conversations (10K tokens):** SMS provides ~90% reduction vs INT8, ~80% vs INT4.
- **Long conversations (128K tokens):** SMS provides ~99% reduction vs INT8, ~98% vs INT4.

The breakeven point versus INT4 occurs around 2K tokens—below this threshold, SMS offers minimal memory advantage over aggressively quantized KV cache. However, the fundamental O(1) vs O(n) scaling difference means SMS benefits grow unboundedly with conversation length, making it increasingly valuable for long-context applications.

8.2 Comparison with Existing Approaches

SMS differs fundamentally from existing KV cache optimizations:

PagedAttention: Optimizes memory allocation; SMS eliminates the cached content.

Sliding Window: Discards old context entirely; SMS preserves semantic content.

GQA/MQA: Reduces per-token KV size; SMS maintains constant total size.

Retrieval-Augmented Generation: Retrieves from external store; SMS maintains live semantic state.

Unique SMS Advantage — Human-Readable Memory: Unlike KV cache (opaque tensor data), SMS uses natural language that users can inspect, understand, and edit. This transparency enables user correction of errors, addition of missing context, and removal of sensitive information. No existing KV cache optimization offers this level of user control and interpretability.

8.3 Relationship to Human Cognition

The SMS approach mirrors aspects of human memory architecture:

Working Memory: Limited capacity, current focus—analogous to current context window

Semantic Memory: Extracted meaning, compressed representation—analogous to Memory State

Memory Consolidation: Active process of extracting meaning—analogous to Memory Update Module

Humans do not store verbatim transcripts of conversations. They extract meaning, discard surface form, and reconstruct details as needed. This "lossy but intelligent" compression enables effectively unlimited conversational memory in a fixed-capacity system. SMS applies this principle to transformer inference.

8.4 Relationship to Production Memory Systems

Production LLMs such as Claude (Anthropic) and ChatGPT (OpenAI) implement memory features for cross-session user context. These systems maintain persistent user information (preferences, facts, prior interactions) that carries across conversations. However, a critical distinction must be made:

- **Production Memory:** Operates alongside traditional KV caching, not as a replacement.
- **KV Cache Growth:** Still grows linearly within each conversation session.
- **Memory Scope:** Focused on user profile and preferences, not full conversation state.
- **Update Frequency:** Periodic background updates, not every turn.

SMS differs fundamentally: it replaces KV cache entirely with semantic state, achieving bounded memory for arbitrarily long single conversations. The approaches are complementary—production memory handles cross-session continuity while SMS could handle within-session efficiency. A combined system might use production memory for user context and SMS for conversation state, eliminating both cross-session amnesia and within-session memory growth.

9. Future Work

Several directions merit investigation:

1. Empirical Validation: Benchmark SMS against traditional KV caching across diverse conversation types, measuring both memory efficiency and response quality.

2. Memory Update Optimization: Investigate optimal model size and architecture for the Memory Update Module, including potential for specialized fine-tuning.

3. Hybrid Approaches: Develop task-aware systems that dynamically choose between SMS and traditional KV caching based on task requirements.

4. Memory State Structures: Explore domain-specific memory schemas optimized for particular use cases (technical support, creative writing, code development).

5. Long-term Persistence: Investigate memory state persistence across sessions, enabling true long-term user context.

6. User-Editable Memory State: Since SMS uses natural language representation, users can directly view, edit, and correct the memory state. This enables: (a) transparency—users see exactly what the model "remembers"; (b) error correction—users can fix misunderstandings or outdated information; (c) personalization—users can add context the model couldn't infer; (d) privacy control—users can remove sensitive information. This human-in-the-loop approach may yield significantly better results than fully autonomous memory management.

10. Conclusion

This paper introduced Semantic Memory State (SMS), a paradigm for transformer inference that replaces token-level KV caching with high-level semantic representation. By maintaining a structured, continuously-updated memory state instead of raw KV vectors, SMS achieves bounded O(1) memory complexity regardless of conversation length.

The approach draws inspiration from human cognitive architecture, recognizing that effective memory is not verbatim recording but intelligent semantic compression. Current KV caching stores computational artifacts; SMS stores meaning.

Theoretical analysis suggests memory reduction of 50-99% versus INT8 KV cache, and 0-98% versus INT4 KV cache, with benefits scaling proportionally to conversation length. For short conversations (≤2K tokens), aggressive quantization may match SMS efficiency; for long conversations (10K+ tokens), SMS provides substantial improvements. For enterprise-scale deployment, this translates to significant reductions in infrastructure cost.

Perhaps more significantly, SMS removes the concept of "context window" as a hard limit. Conversations can extend indefinitely without unbounded memory growth or speed degradation. The model no longer "forgets"—it maintains semantic continuity across arbitrary conversation lengths, though with the inherent trade-off of lossy compression.

The question is not whether AI should remember like humans—it is whether AI should remember better than the current brute-force approach allows. SMS suggests that the answer lies not in storing more tokens, but in storing meaning.

Acknowledgments

This paper was developed as a result of a dialogue with Claude AI (Claude Opus 4.5 + Extended Thinking). The core concept of Semantic Memory State as a KV cache replacement—including the insight that KV cache stores positional/computational artifacts rather than semantic information, and the idea of user-editable memory states—originated from the author. Technical analysis, mathematical formulations, literature review, related work comparison, and manuscript preparation were conducted with AI assistance. The author reviewed, directed, and takes full responsibility for all claims presented.

The author acknowledges that this work represents a new form of human-AI collaboration in research, where human intuition and creative insight combine with AI capabilities in technical elaboration and writing. This transparency is provided in the interest of academic integrity and to contribute to evolving norms around AI-assisted research.