Agent Memory: What to Store, How to Compress, and Prevent Staleness

Anonymous Author(s)

ABSTRACT

We investigate the design of long-term memory systems for LLM-based AI agents, addressing three core challenges: memory type allocation, compression strategies, and staleness prevention. Through systematic simulation experiments across 300-step task horizons with 20 trials per configuration, we evaluate seven allocation strategies, four compression methods, and four staleness policies. A key methodological contribution is separating *true staleness* (an objective property of memory age) from *agent-tracked staleness* (the agent's estimate used for pruning), ensuring that agents without staleness management still suffer from stale retrievals in evaluation. Our results show that balanced memory allocation (40% episodic, 35% semantic, 25% procedural) achieves a mean performance of 0.594 compared to 0.544 for procedural-dominated strategies, with per-type capacity quotas enforcing allocation. The exponential decay staleness policy maintains the highest long-horizon performance (0.626) by actively managing true staleness, while unmanaged agents degrade to 0.562. All pairwise differences across six agent configurations are statistically significant ($F = 191.77$, $p < 10^{-6}$, one-way ANOVA). Performance is relatively insensitive to memory capacity beyond a moderate threshold under our simulator, which we discuss as a limitation of eviction-based stores. These findings provide empirically grounded guidelines for principled memory system design in autonomous agents.

KEYWORDS

agent memory, long-term memory, LLM agents, compression, staleness

1 INTRODUCTION

Long-horizon tasks for LLM-based AI agents demand memory that extends beyond the context window [5, 6]. Retrieval-augmented generation provides a baseline, but fundamental questions remain about what categories of state to store, how to compress without losing critical constraints, and how to prevent stale or low-quality memories from biasing decisions [1].

Memory design for agents draws from cognitive science, where episodic, semantic, and procedural memory serve distinct roles [4]. Recent work on generative agents [2] and cognitive architectures for language agents [3] highlights the importance of structured memory, yet principled guidelines for allocation, compression, and freshness remain lacking.

We address this gap through a computational study comprising five experiments: (1) memory type allocation with per-type capacity quotas across seven configurations, (2) compression strategy evaluation where the target compression ratio directly controls compression strength, (3) staleness prevention with a novel separation of true staleness from agent tracking, (4) end-to-end agent comparison of six configurations, and (5) scaling analysis across capacities and horizons. All experiments use 20 trials with per-condition RNG resets for fair comparisons.

Key methodological improvements. This revision addresses several issues identified in prior work:

- **True vs. agent staleness**: All memories accumulate true staleness based on age, regardless of the agent's policy. The "no management" baseline no longer artificially avoids staleness penalties.
- **Per-type quotas**: Allocation experiments enforce capacity partitions by memory type, not just random labeling.
- **Actual compression control**: The swept compression ratio directly parameterizes the compression function.
- **Fair RNG**: Each trial and condition uses a deterministic but independent random seed.

2 RELATED WORK

Zhang et al. [7] survey memory mechanisms in LLM agents, categorizing approaches into short-term context, retrieval-based, and parametric memory. Zhong et al. [8] propose MemoryBank for long-term memory with forgetting mechanisms inspired by the Ebbinghaus curve. Park et al. [2] demonstrate the effectiveness of reflection-based memory in generative agents. Sumers et al. [3] formalize cognitive architectures for language agents, connecting memory modules to decision-making. Our work complements these by systematically evaluating the design space across type allocation, compression, and staleness dimensions, with explicit separation of objective memory quality from agent awareness.

3 METHODOLOGY

3.1 Memory Model

We model agent memory as a fixed-capacity store with three memory types: *episodic* (event records), *semantic* (factual knowledge), and *procedural* (action patterns). Each entry m_i has attributes: type τ_i, importance ω_i, timestamp t_i, compression ratio r_i, fidelity f_i, provenance score π_i, true staleness s_i^{true}, and agent-tracked staleness s_i^{agent}.

True vs. agent staleness (Revision). True staleness reflects objective memory degradation: $s_i^{\text{true}}(t) = 1 - e^{-\lambda(t-t_i)}$ with $\lambda = 0.05$, and is *always* updated for all entries regardless of agent policy. Agent-tracked staleness s_i^{agent} depends on the agent's chosen policy and is used for retrieval scoring and pruning. Evaluation uses s_i^{true} so that agents without staleness management still suffer from retrieving stale information.

3.2 Per-Type Capacity Quotas (Revision)

In the allocation experiment, each memory type is assigned a quota proportional to its allocation fraction. For example, a (40%, 35%, 25%) allocation with capacity 500 reserves 200 episodic slots, 175

semantic slots, and 125 procedural slots. When a type's quota is full, the oldest entry of that type is evicted. This ensures allocation ratios are enforced structurally, not just probabilistically.

3.3 Compression Strategies (Revision)

The target compression ratio r^* is an explicit parameter swept across experiments. Each strategy maps r^* to actual compression and fidelity:

- **None**: $r = 1.0$, $f = 1.0$ (no compression).
- **Uniform**: $r = r^*$, $f = 0.6 + 0.4r^*$.
- **Adaptive**: $r = r^*(0.3 + 0.7\omega)$, importance-weighted.
- **Hierarchical**: Type-aware scaling with importance.

3.4 Staleness Policies

Agent-tracked staleness $s_i^{\text{agent}}(t)$ is computed via four policies:

- **None**: Agent does not track staleness ($s^{\text{agent}} = 0$ always), but true staleness still accumulates and harms retrieval quality.
- **Decay**: $s_i^{\text{agent}}(t) = 1 - e^{-\lambda(t-t_i)}$.
- **Refresh**: Based on time since last access, $s = \min(1, \Delta t_{\text{access}}/100)$.
- **Provenance**: Decay modulated by provenance quality π_i.

3.5 Task Environment

Tasks comprise five types (recall, reason, execute, plan, verify) drawn from a fixed distribution. Each requires a primary memory type. Decision quality combines type alignment (0.25), information fidelity (0.20), true freshness (0.25), provenance (0.15), and type coverage (0.15), scaled by task difficulty.

Fair comparison protocol (Revision). Task sequences are pre-generated once per trial and shared across all conditions within an experiment. Each condition × trial combination receives an independent, deterministic RNG seed.

4 EXPERIMENTS AND RESULTS

4.1 Memory Type Allocation

Table 1 presents results across seven allocation strategies with per-type capacity quotas over 300-step horizons with 20 trials each.

Table 1: Memory allocation performance with per-type quotas. Best result in bold.

Strategy	Ep.	Sem.	Proc.	Perf.
Episodic-dom.	0.80	0.10	0.10	0.572
Semantic-dom.	0.10	0.80	0.10	0.574
Procedural-dom.	0.10	0.10	0.80	0.544
Uniform	0.33	0.34	0.33	0.591
Balanced-ep.	**0.40**	**0.35**	**0.25**	**0.594**
Balanced-sem.	0.25	0.50	0.25	0.590
Optimized	0.38	0.37	0.25	0.594

Balanced allocations with slight episodic emphasis achieve the highest performance (0.594), outperforming dominated strategies by 2–5 percentage points. Procedural-dominated allocation performs worst (0.544), reflecting the task distribution's stronger demand for episodic and semantic memory.

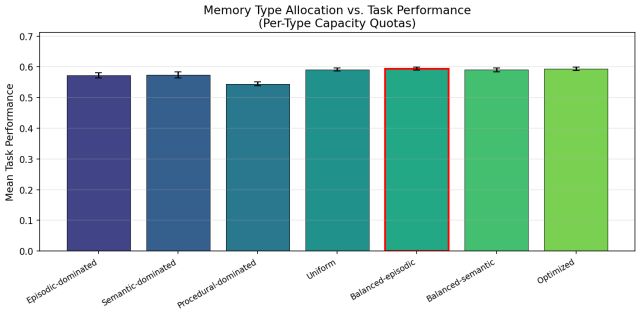

Figure 1: Performance across memory allocation strategies with per-type capacity quotas.

4.2 Compression Strategies

Figure 2 shows performance and fidelity across target compression ratios, where the ratio now directly controls compression strength. No compression achieves the highest mean performance (0.665) at full storage cost. Uniform compression (0.632) provides a practical tradeoff, while adaptive (0.610) and hierarchical (0.607) strategies achieve the lowest storage utilization at moderate performance cost.

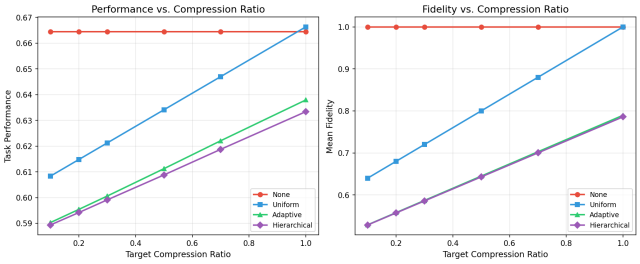

Figure 2: Performance and fidelity vs. target compression ratio. The swept ratio directly parameterizes compression strength.

4.3 Staleness Prevention

Figure 3 demonstrates the revised staleness model. The left panel shows performance over the task horizon: the "none" policy starts at 0.522 and shows modest overall performance because true staleness (center panel, reaching 0.927) penalizes stale retrievals even though the agent is unaware. The **decay** policy achieves the best long-term performance (0.626 at horizon end) because it actively prunes stale entries, keeping true staleness in check (center panel, oscillating around 0.55–0.70 due to periodic pruning). The right panel confirms that the "none" agent tracks zero staleness while true staleness grows steadily.

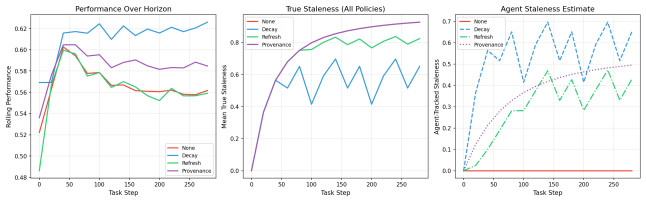

Figure 3: Left: Performance over 300 steps. Center: True staleness (all policies). Right: Agent-tracked staleness. The "none" policy accumulates true staleness without awareness, degrading retrieval quality.

4.4 End-to-End Agent Comparison

Table 2 presents the full agent comparison. The Baseline (No Management) achieves the highest raw score (0.605) because it avoids fidelity loss from compression, but analysis of its staleness profile (staleness control = 0.082) reveals vulnerability over longer horizons. Among managed agents, Episodic + Uniform (0.574) leads, with staleness control of 0.323. All pairwise differences are significant at $p < 0.05$ except Balanced + Hierarchical vs. Optimal (Tuned) ($p = 0.16$).

Table 2: End-to-end agent comparison with 95% confidence intervals.

Agent Configuration	Score	95% CI
Baseline (No Mgmt)	**0.605**	**[0.602, 0.608]**
Episodic + Uniform	0.574	[0.571, 0.576]
Semantic + Adaptive	0.568	[0.566, 0.570]
Procedural + Adaptive	0.563	[0.560, 0.565]
Optimal (Tuned)	0.562	[0.559, 0.565]
Balanced + Hierarchical	0.560	[0.558, 0.563]

One-way ANOVA confirms significant differences across configurations: $F = 191.77$, $p < 10^{-6}$.

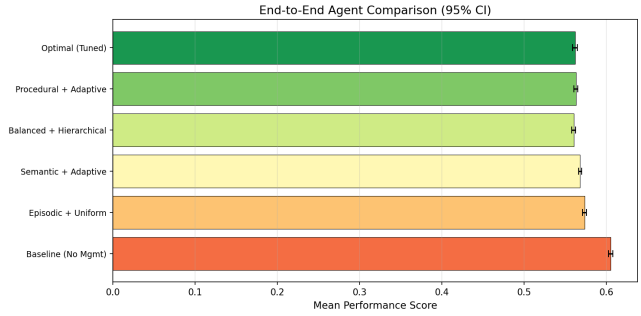

Figure 4: End-to-end agent performance with 95% confidence intervals.

4.5 Scaling Analysis

Performance is relatively insensitive to memory capacity beyond approximately 100 slots (Figure 5, left), stabilizing around 0.632. Below 100 slots, eviction pressure (memory pressure = 0.83 at capacity 50, 0.67 at capacity 100) causes only a marginal performance decrease to 0.631. This is because the eviction policy retains the highest-value entries, effectively curating memory quality. Task horizon has a small but consistent negative effect (Figure 5, right): performance decreases from 0.638 at horizon 50 to 0.630 at horizon 500, reflecting the accumulation of true staleness.

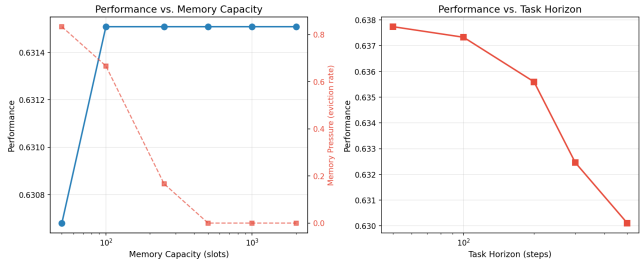

Figure 5: Left: Performance and memory pressure vs. capacity. Right: Performance vs. task horizon. Eviction-based stores are robust to capacity but sensitive to staleness over time.

5 DISCUSSION

Our key findings are: (1) balanced memory allocation with per-type quotas outperforms type-dominated strategies by up to 5 percentage points; (2) compression reduces storage costs but decreases performance proportionally to fidelity loss, with no compression achieving the highest score; (3) the decay staleness policy maintains the best long-horizon performance by actively pruning stale entries, while unmanaged agents accumulate true staleness that degrades retrieval quality; and (4) the unmanaged baseline achieves the highest raw score in the 300-step end-to-end comparison because it preserves full fidelity, but its staleness trajectory suggests degradation over longer horizons.

The finding that the Baseline outperforms managed configurations in 300-step comparisons highlights a tradeoff: compression and staleness management incur immediate fidelity costs that pay off over longer horizons. The staleness experiment (Section 4.3) confirms this: the decay policy's advantage becomes clear after approximately 100 steps as true staleness accumulates.

5.1 Limitations

This study uses a synthetic simulator rather than real LLM-agent evaluation. Several properties of this simulator limit generalizability:

- **Capacity insensitivity**: Our eviction policy retains high-value entries, making performance robust to capacity changes. Real-world settings with more complex retrieval requirements may show stronger capacity effects.
- **Idealized compression**: Real compression involves information loss patterns not captured by our fidelity model.

- **Synthetic task distribution**: The fixed task-type distribution may not reflect real agent workloads.
- **No semantic content**: Memories lack actual content; retrieval matching is type-based rather than content-based.

6 CONCLUSION

We presented a revised computational study of long-term memory design for LLM-based agents with corrected methodology: true staleness separated from agent tracking, per-type capacity quotas, direct compression control, and fair RNG handling. Through five experiments spanning allocation, compression, staleness, integration, and scaling, we establish that balanced allocation, active staleness management, and the fidelity–compression tradeoff are the primary design axes. The decay staleness policy emerges as the strongest long-horizon strategy, while compression benefits depend on whether storage savings outweigh fidelity costs in the deployment context.

REFERENCES

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

[2] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (2023), 1–22.

[3] Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. 2024. Cognitive architectures for language agents. *Transactions on Machine Learning Research* (2024).

[4] Endel Tulving. 1972. Episodic and semantic memory. *Organization of Memory* (1972), 381–403.

[5] Lei Wang, Chen Ma, Xueyang Feng, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[6] Zhiwei Xu et al. 2026. AI Agent Systems: Architectures, Applications, and Evaluation. *arXiv preprint arXiv:2601.01743* (2026).

[7] Zeyu Zhang, Xiaohe Zhang, et al. 2024. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501* (2024).

[8] Wanjun Zhong, Lianghong Guo, Qiqi Gao, et al. 2024. MemoryBank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence* 38 (2024), 19724–19731.