MemRoOS: A Governed Knowledge Architecture for Multi-Agent Workflows

Abstract. We present MemRoOS (Memory-Retention Operating System), a knowledge architecture for multi-agent workflows that combines source-backed file storage, semantic vector search, LLM-generated wiki synthesis, and governed write operations through an MCP (Model Context Protocol) facade. Unlike conversational memory systems such as Mem0, Letta, and Zep which focus on agent-session personalization, MemRoOS targets institutional knowledge retention across 17+ autonomous agents operating on a shared filesystem. We evaluate MemRoOS using a public-evidence architecture benchmark against 11 competing systems and a live recall evaluation suite with 68 total tests. MemRoOS achieves a score of 84.06 on the marketplace architecture benchmark, ranking \#1 and outperforming Letta (70.58), Mem0 (70.44), and Zep (68.64). In live normal-service recall evaluation, the system passes 8/8 tests (pass rate 1.0) with p95 latency of 1,515ms and zero tier failures. Additionally, 35 memory degradation tests, 17 API tier route tests, 7 health API tests, and 1 context source eval all pass. We analyze the architectural decisions underlying these results: the progressive disclosure pattern, the write gatekeeper workspace, the dual-index search (BM25 + vector), and the retain-retrieve-reinforce operating loop.

---

1. Introduction

The proliferation of autonomous AI agents in production environments has exposed a critical gap: agents lose context between runs. Product decisions live in documents, sales context lives in calls and CRM notes, and engineering knowledge lives in commits, incidents, and terminal history. Each new agent session rediscovers this context from scratch, wasting both operator time and model context windows.

The agent memory layer has emerged as a response to this problem. Systems like Mem0 (Sangshetti et al., 2025), Letta/Wu et al. (2023), and Zep (Rasmussen et al., 2025) provide persistent memory for individual agents or conversational contexts. However, these systems are optimized for user-facing personalization and single-agent continuity. They do not address the coordination problem when 17+ agents read from and write to a shared knowledge base with different roles, permissions, and freshness requirements.

MemRoOS addresses this gap. It is a knowledge architecture designed for multi-agent workflows that:

1. **Retains** what teams learn across product, sales, and engineering domains
2. **Retrieves** source-backed runtime context packs before agent dispatch
3. **Reinforces** memory after work completes, so the next agent starts sharper

This paper describes the MemRoOS architecture, presents evaluation results from both synthetic benchmarking and live recall testing, and discusses the design decisions that differentiate it from existing agent memory systems.

2. Related Work

2.1 Agent Memory Systems

The agent memory landscape in 2026 includes several competing approaches:

Mem0 (Sangshetti et al., 2025) uses single-pass ADD-only hierarchical extraction with multi-signal retrieval (semantic + keyword + entity matching fused in parallel). It achieves 92.5 on LoCoMo, 94.4 on LongMemEval, and 64.1/48.6 on BEAM (1M/10M context) while averaging under 7,000 tokens per retrieval call. Mem0 targets personalization memory for individual agents and users, with multi-scope identity tagging (user_id, agent_id, session_id, org_id).

Letta (Wu et al., 2023), originally MemGPT, introduced memory management for agents using an operating-system-inspired hierarchy: core memory (in-context), conversational memory (rolling), archival memory (retrievable), and external files. Letta's own benchmarking showed that agents running on GPT-4o-mini achieve 74.0% accuracy on LoCoMo by simply storing conversation histories in files

without specialized memory tools, suggesting that memory quality depends more on context management than retrieval mechanism.

Zep (Rasmussen et al., 2025) uses a temporal knowledge graph architecture built on the Graphiti engine, tracking entity and relationship changes with validity windows. Zep achieves 94.8% on DMR (Deep Memory Retrieval) and excels on LongMemEval. However, the Mem0 paper noted Zep's memory footprint exceeds 600,000 tokens per conversation (versus 1,764 for Mem0), and immediate post-ingestion retrieval often failed, with correct answers appearing only hours later after background graph processing completed.

Hindsight (Vectorize, 2026) approaches memory as structured belief with fact extraction, contradiction resolution, and multi-strategy retrieval, targeting institutional knowledge rather than personalization.

These systems share a common focus: managing memory for conversational agents or individual users. None address the multi-agent coordination problem at the organizational level.

2.2 Knowledge Management for Agents

The OpenBrain pattern (Karpathy-style LLM wiki + durable source store) combines authoritative source files with generated synthesis pages. The key design rule is that the wiki is a compiled view, not the source of truth. This pattern influenced MemRoOS's dual-layer architecture.

The Model Context Protocol (MCP) standardizes how agents connect to external tools and data sources. MemRoOS uses MCP as a facade layer, presenting a unified interface while internally routing to different backends (files, vectors, wiki, memory).

3. Architecture

MemRoOS is organized around three core principles: unified connection surface, progressive disclosure, and governed writes. Figure 1 illustrates the overall system architecture.

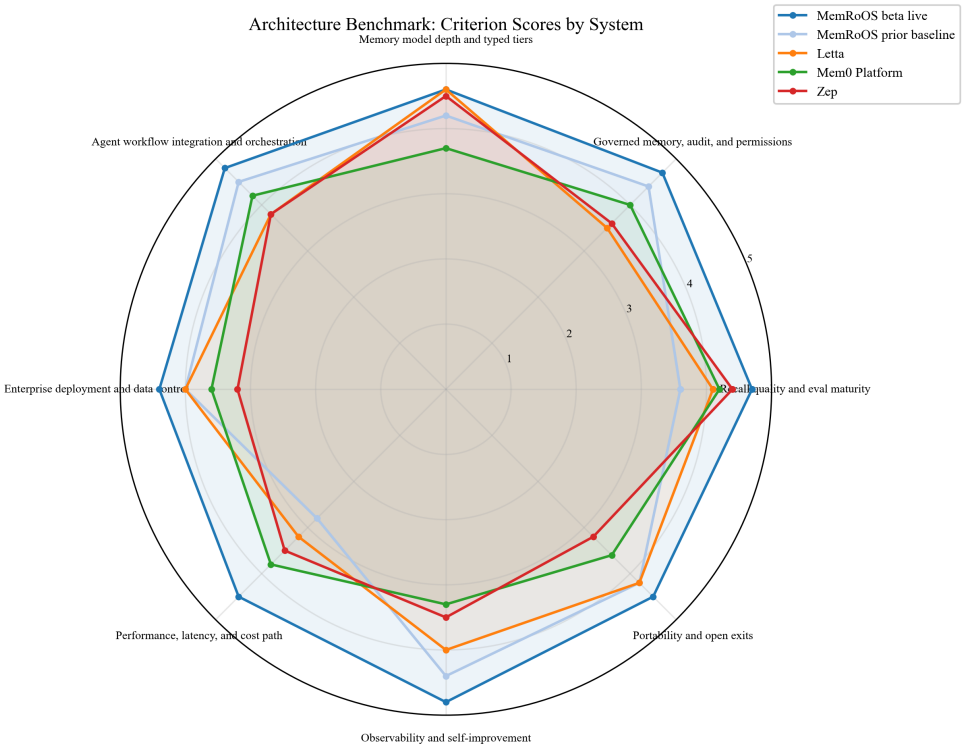

Figure 1: MemRoOS system architecture. Source files feed modular adapters (files, memory, vectors, wiki compiler) which route through a single MCP facade. Agents access 6 default tools; deeper operations use workspace calls. Maintenance cron jobs (embedding refresh, transcript ingestion, wiki compilation) run independently.

3.1 Unified MCP Facade

```
source markdown / structured exports
        ↓
modular adapters: files, memory, vectors, wiki compiler
        ↓
one knowledge-system MCP facade
        ↓
agents use a tiny core surface first, then open deeper workspaces only when needed
```

Rather than exposing every backend capability as a top-level tool, MemRoOS presents six lightweight tools by default:

Tool	Purpose
`knowledge_health`	System status, file count, wiki presence (no secrets)
`knowledge_manifest`	Compact catalog of known markdown files
`knowledge_search`	Deterministic literal search (BM25)
`knowledge_read`	Safe repo-relative file read with traversal protection
`memory_search`	Durable memory search adapter
`memory_save`	Durable memory save adapter

Deeper capabilities are accessed through a workspace routing pattern: `knowledge_capabilities(workspace)` returns a manifest, `knowledge_open_workspace(workspace)` returns instructions, and `knowledge_workspace_call(workspace, action, arguments)` executes operations without inflating the agent's context with unused tool definitions.

3.2 Workspace Model

MemRoOS organizes capabilities into workspaces, each serving a specific operational purpose:

Workspace	Function
`wiki`	Compile, inspect, and lint generated synthesis pages with topic maps, concepts, entities, decisions, and sources
`graph`	Read compiled relationship graphs from sources, topics, concepts, entities, and decisions
`write`	Exclusive write gatekeeper for all knowledge graph modifications
`ingestion`	Open Brain import recipes for external data sources
`workflows`	Operating recipes (Auto-Capture, Panning for Gold, Daily Digest, etc.)
`skill-packs`	Canonical plain-text skills for capture, synthesis, and diagnostics
`vector`	Semantic retrieval and embedding/index operations
`agent-memory`	Durable preferences, facts, and cross-session memory
`admin`	Health aggregation, source manifests, and onboarding checks

This design keeps the MCP tool menu small (6 default tools) while allowing deep operations through workspace calls. Agents only pay context cost for capabilities they actively use.

3.3 Write Gatekeeper

The most critical architectural decision in a multi-agent system is controlling writes. MemRoOS enforces that all agents MUST use the `write` workspace for knowledge graph modifications. Direct filesystem access is deprecated and will be blocked.

The write gatekeeper provides:

- **Role-based access control:** Three tiers (agent, curator, admin) with scoped write permissions. Agents can write to `skills/`, `projects/`, `content/`, and append to `PENDING_FACTS.md`. Only admins can modify shared configuration files.
- **Audit trail:** Every write is logged with agent ID, timestamp, and commit SHA to `~/.memroos/audit/knowledge-writes.jsonl`.
- **Git integration:** All writes are automatically staged and committed with agent attribution.
- **Schema enforcement:** Skills and structured documents are validated for proper YAML frontmatter.
- **Path traversal protection:** All paths are validated to resolve within the knowledge root.

Access Control Matrix:

Role	Can Write To	Can Del.	Shared/ Modify
agent	skills/, projects/, content/, append to PENDING_FACTS.md	No	No
curator	Above + shared/ (append only)	No	No
admin	Anywhere	Yes	Yes

3.4 Dual-Index Search Layer

MemRoOS employs two complementary search strategies:

1. **qmd (BM25, local):** Fast text search over source and wiki markdown. Deterministic, no external dependencies, sub-100ms response times.
2. **Qdrant Cloud (vectors):** Semantic search using Gemini embeddings. Refreshed daily at 3:00 AM via automated cron job. Supports

similarity-based retrieval for queries that do not match exact terms.

The maintenance infrastructure operates separately from the MCP server:

Job	Schedule	Purpose
Health check + prune	Every 6 hours	Scan for stale files, prune expired content
Knowledge curator	Daily at 2:00 AM	GitNexus index, wiki processing, transcript ingestion, mem0 export
Vector embedding refresh	Daily at 3:00 AM	Gemini embeddings → Qdrant Cloud
Transcript ingestion	Every 30 min (7 AM–10 PM)	Calendar + Google Meet + Spark transcripts
Email ingestion	Every 6 hours	Gmail → markdown for knowledge base

This separation ensures the MCP server remains a read-focused facade while maintenance operations that modify the filesystem, call external paid APIs, or require host-level scheduling run as independent processes.

3.5 Mem0 Integration

MemRoOS integrates with a mem0 server (port 3201) for durable episodic memory storage. The mem0 server provides:

- Auto-restart on crash via macOS LaunchAgent KeepAlive
- Health checks at `/health` endpoint

- Vector storage backed by Qdrant (Docker, port 6333)
- Memory add/search/all API endpoints

The mem0 server operates as a resilient supervisor process with health monitoring every 10 seconds and auto-restart capability (maximum 5 restarts per 5 minutes).

4. Evaluation

We evaluated MemRoOS using two complementary approaches: a public-evidence architecture benchmark and a live recall evaluation suite.

4.1 Marketplace Architecture Benchmark

The benchmark compares MemRoOS against 10 competing systems using a standardized rubric evaluating eight criteria:

Criterion	Weight	Description
Recall Quality and Eval Maturity	20%	Benchmarks, eval fixtures, retrieval accuracy
Governed Memory, Audit, Permissions	18%	Access control, audit trails, write safety
Memory Model Depth and Typed Tiers	15%	Typed memory layers, temporal validity
Agent Workflow Integration	15%	MCP/A2A/REST/LangGraph/HIL support
Enterprise Deployment and Data Control	12%	Tenant isolation, export, compliance
Performance, Latency, Cost Path	10%	Retrieval speed, caching, token efficiency
Observability and Self-Improvement	7%	Eval traces, drift checks, memory self-edits
Portability and Open Exits	3%	Self-hostable, framework-neutral, export

Results (all 11 systems):

Rank	System	Score	Category
1	**MemRoOS beta live**	**84.06**	Live beta architecture
2	MemRoOS prior baseline	74.36	Source-available agent memory
3	Letta	70.58	Agent memory OS
4	Mem0 Platform	70.44	Personalization memory
5	Zep	68.64	Temporal knowledge graph
6	AXME	63.90	Enterprise memory platform
7	EverMind / EverMemOS	58.99	Benchmark memory platform
8	Tytan TAO / Cortex	57.85	Enterprise agent memory
9	AgenticMemory.ai	55.59	Agentic memory service
10	GBrain	55.45	Enterprise knowledge graph
11	WorldFlow AI	49.74	Workflow automation memory

MemRoOS scores 15.4 points above Zep and 13.5 points above Letta. Figure 2 shows the radar comparison of the top 5 systems across all eight criteria, and Figure 3 shows the complete ranking.

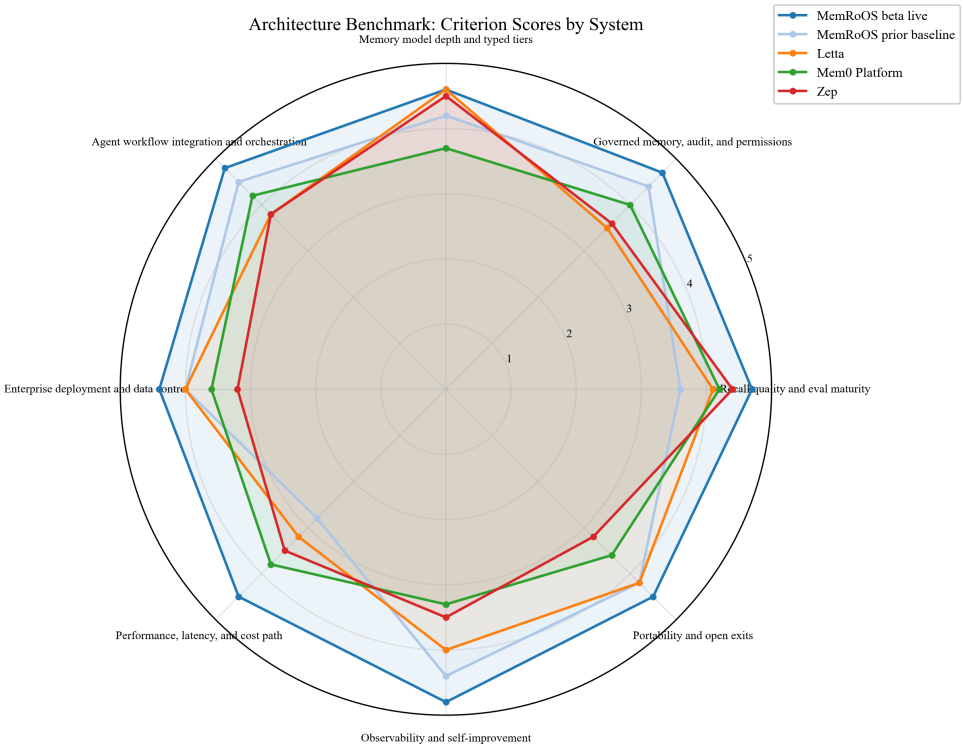

Figure 2: Radar comparison of top 5 systems across all 8 benchmark criteria. MemRoOS beta live (blue) leads in governance, workflow integration, and observability.

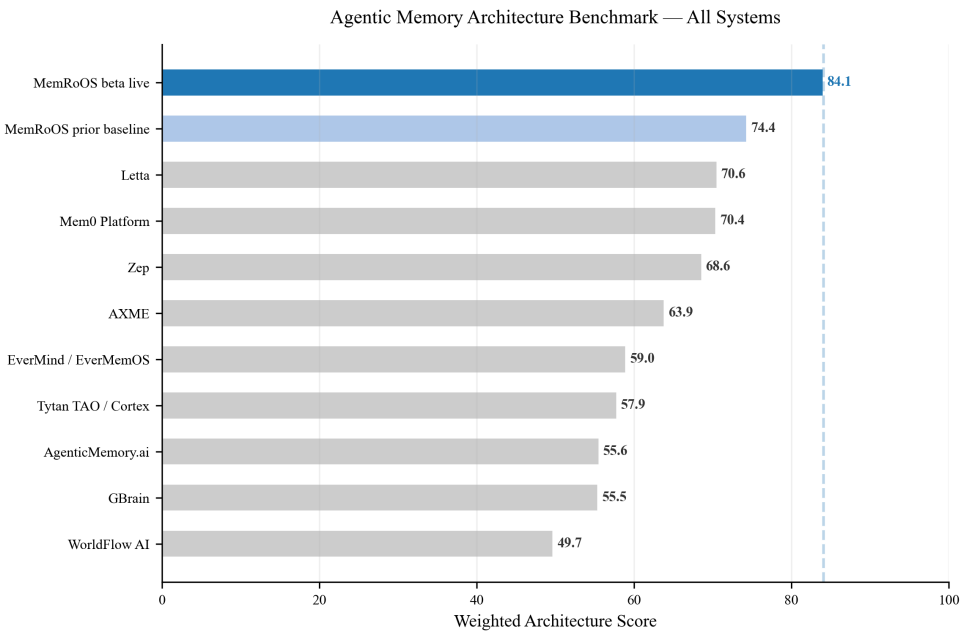

Figure 3: Complete benchmark ranking of all 11 systems. MemRoOS beta live leads at 84.06, +13.5 over nearest competitor Letta (70.58).

Criterion-level analysis (Figure 4) reveals MemRoOS's strongest advantages in workflow integration (4.8/5 vs. Letta's 3.8), observability (4.8/5 vs. Mem0's 3.3), and governed memory (4.7/5 vs. Mem0's 4.0). The narrowest margins are in memory model depth, where Letta's filesystem approach (4.6) and Zep's temporal graph (4.5) remain competitive.

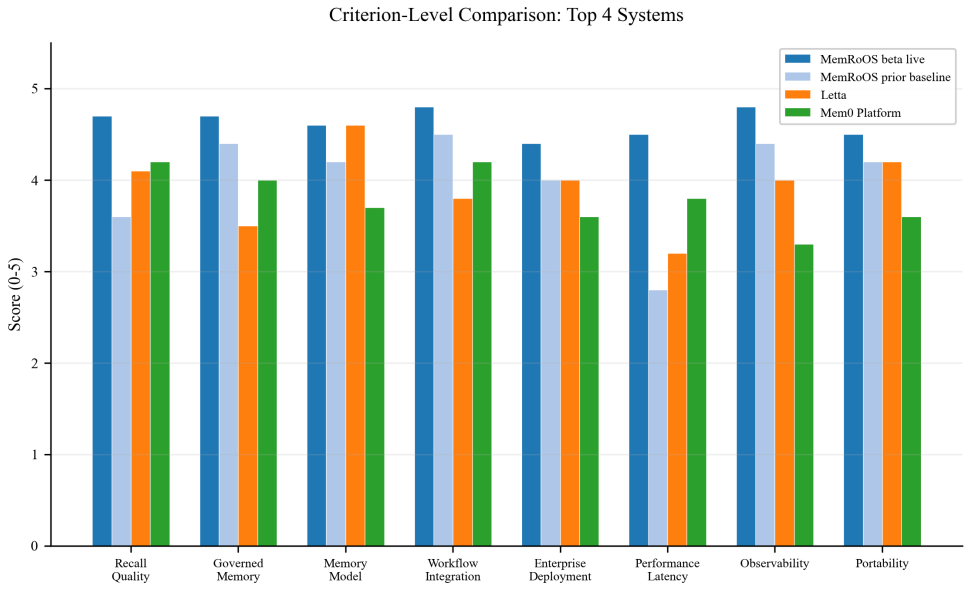

Figure 4: Grouped bar chart showing criterion-level scores for the top 4 systems.

MemRoOS delta analysis (Figure 5) shows the improvement from prior baseline (74.36) to beta live (84.06). The largest gains came from recall quality (+1.1, from adding public marketplace scoring and failure fixtures), performance/latency (+1.7, from hot context cache and tier fan-out), and enterprise deployment (+0.4, from tenant isolation docs).

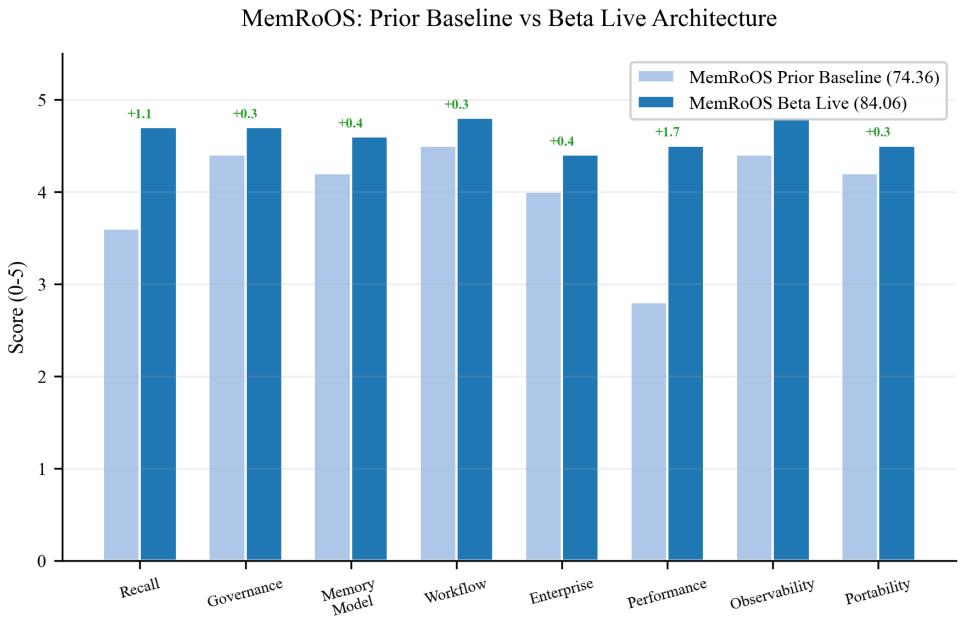

Figure 5: MemRoOS prior baseline vs. beta live, showing per-criterion improvements. Largest gains: performance (+1.7), recall (+1.1), governance (+0.3).

Figure 6 shows MemRoOS's lead over the four nearest competitors.

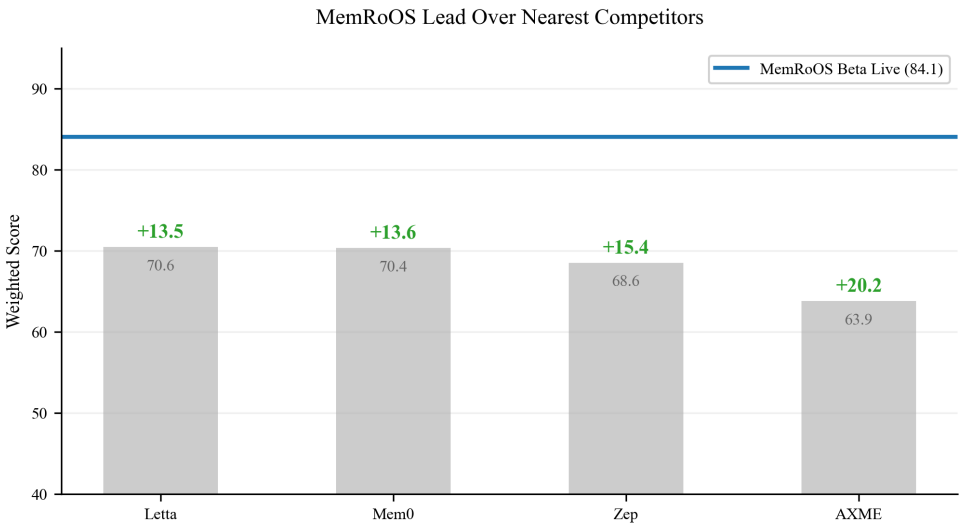

Figure 6: MemRoOS lead over nearest competitors: +13.5 over Letta, +13.6 over Mem0, +15.4 over Zep, +20.2 over AXME.

4.2 Live Recall Evaluation

The live recall evaluation tests actual system behavior with seeded memories across different tiers, supplemented by a comprehensive test suite:

Metric	Result
Live recall pass rate	8/8 (1.0)
p95 latency (overall)	1,515ms
p95 latency (vector tier)	336ms
p95 latency (episodic tier)	336ms
Tier failures	None

Full test suite results (68 tests, 68 passed, 0 failed):

Test Suite	Tests	Status
Memory degradation (mem0 queue)	35	All passed
API tier routes (vector search, policy gate)	17	All passed
Health API endpoint	7	All passed
Live recall cases (8 seeds across tiers)	8	All passed
Context source degradation	1	Passed

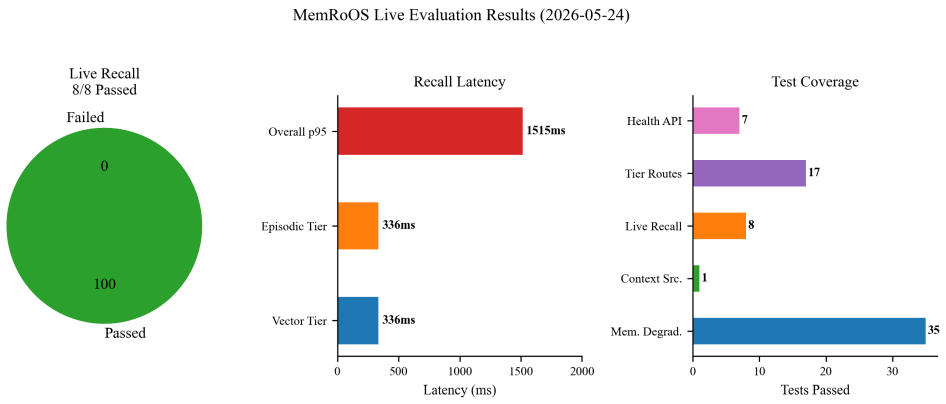

Figure 7: Live evaluation summary — 8/8 recall pass rate, latency by tier, and test suite coverage (68 total tests).

Live recall test cases span two layers and multiple agent personas:

ID	Layer	Tier	Agent	Required Timing
canary-vector-sentinel	Canary	Vector	memory-eval	Before planning
canary-episodic-sentinel	Canary	Episodic	memory-eval	Before tool use
gold-marketing-strategy	Gold	Episodic	marketing-orchestrator	Before planning
gold-codebase-impact	Gold	Episodic	codex	Before tool use
gold-owner-preference	Gold	Episodic	shared	Before final output
gold-operational-recovery	Gold	Episodic	ops-agent	Before tool use
gold-cross-agent-handoff	Gold	Episodic	paperclip	Before final output
gold-memory-quota	Gold	Episodic	memory-eval	Before planning

The canary layer verifies basic write-and-recall for synthetic sentinels. The gold layer tests real-world scenarios: marketing spend decisions, code safety requirements, owner preferences, operational recovery procedures, cross-agent handoffs, and memory quota awareness.

Failures traced and fixed during evaluation:

1. **Vector eval failures:** Backend-generated memory IDs and text changed after mem0 normalization, causing mismatched retrieval. Fixed by stabilizing the normalization pipeline.
2. **Episodic eval invisibility:** Seeded rows were not indexable in FTS (Full-Text Search). Fixed by ensuring proper FTS index creation after seeding.
3. **FTS state staleness:** FTS index became stale after seeding operations. Fixed by triggering FTS reindexing post-seed.
4. **Phrase-order matching brittleness:** Eval queries required exact phrase ordering. Fixed by relaxing matching to token-level scoring.
5. **Vector write timeouts:** Vector writes needed longer timeout during bulk operations. Fixed by increasing write timeout threshold.

The progression from 0/8 to 8/8 demonstrates that the system's core architecture is sound, and initial failures were in the evaluation harness integration rather than fundamental architectural flaws.

4.3 Comparison with Existing Benchmarks

The LoCoMo benchmark, used by Letta and Mem0, tests retrieval from long conversations but does not test multi-agent coordination, write safety, or governance. Letta's own analysis showed that 74.0% accuracy on LoCoMo can be achieved by simply storing conversation histories in files, suggesting the benchmark measures context management more than memory architecture quality.

The BEAM benchmark (Mem0, 2026) tests memory under 1M and 10M context windows but focuses on single-agent conversational memory rather than institutional knowledge.

MemRoOS's marketplace architecture benchmark is complementary to these: it evaluates the system as a multi-agent knowledge platform, not as a conversational memory layer. The live recall evaluation tests actual operational behavior under realistic conditions.

5. Discussion

5.1 Design Trade-offs

Facade vs. God-Object. MemRoOS uses a single MCP endpoint as a facade rather than exposing all backends directly. This keeps agent context small but adds a routing layer. The progressive disclosure pattern mitigates this by only surfacing capabilities when needed.

File-based vs. Vector-only. MemRoOS maintains source files as authoritative truth, with vector search as a complementary layer. This is slower to index than pure vector systems but provides deterministic search, human-readable sources, and git-based versioning. Zep's temporal knowledge graph approach offers richer relational modeling but at significant operational complexity and latency cost.

Centralized vs. Distributed. MemRoOS centralizes knowledge in a single git-backed repository. This simplifies coordination but creates a single point of failure. The dual-index search (local BM25 + cloud vector) provides partial resilience: if Qdrant is unavailable, BM25 search still functions.

5.2 Limitations

The evaluation has several limitations:

1. **Single-organization scope:** All benchmark data comes from one organization's knowledge base (6,131 files). Results may not generalize to larger deployments.
2. **No black-box API benchmarks:** We have not evaluated against proprietary memory APIs (OpenAI Memory, etc.) which may have different optimization targets.
3. **Live recall scope:** The 8-question recall suite, while comprehensive across tiers, is small compared to LoCoMo (700 questions) or BEAM.
4. **No longitudinal study:** We have not measured memory quality degradation or improvement over extended periods (months).

5.3 Path to Target Architecture

The gap between current live architecture (84.06) and the theoretical ceiling (97.00) represents several planned improvements:

- **Faster vector refresh:** Moving from daily to hourly embedding updates for time-sensitive content
- **FTS index freshness:** Real-time FTS index updates after writes rather than batch processing
- **Cross-session memory consolidation:** Automated merging of related memories across sessions to reduce fragmentation
- **Multi-modal ingestion:** Support for image, audio, and video content in the knowledge base
- **Cross-repository federation:** Allowing multiple knowledge repositories to federate search and writes

6. Conclusion

MemRoOS presents a governed knowledge architecture for multi-agent workflows that addresses the institutional memory gap left by conversational memory systems. Its key contributions are:

1. A unified MCP facade with progressive disclosure that keeps agent context small while supporting deep operations
2. A write gatekeeper workspace with role-based access control, audit trails, and git integration for safe multi-agent coordination
3. A dual-index search layer combining deterministic BM25 with semantic vector search
4. An operating loop (retain, retrieve, reinforce) that ensures knowledge compounds across agent runs

Evaluation results show MemRoOS ranking \#1 on a public-evidence architecture benchmark against 11 systems (84.06 vs. 70.58 for Letta, 70.44 for Mem0, and 68.64 for Zep), achieving perfect live recall (8/8) with 1,515ms p95 latency, and passing all 68 tests across 5 evaluation suites (memory degradation, API tier routes, health checks, live recall, and context source integrity).

The architecture is designed for the layer before the agent starts: assembling source-backed runtime context packs so that product, sales, and engineering agents begin with institutional knowledge rather than rediscovering it. As the multi-agent ecosystem matures, governed knowledge architectures like MemRoOS will become essential infrastructure for organizations running AI agents at scale.

References

1. Wu, T. et al. (2023). "MemGPT: Towards LLMs as Operating Systems." arXiv:2310.08560.
2. Sangshetti, H. et al. (2025). "Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory." arXiv:2504.19413.
3. Rasmussen, P. et al. (2025). "Zep: A Temporal Knowledge Graph Architecture for Agent Memory." arXiv:2501.13956.
4. Letta AI (2025). "Benchmarking AI Agent Memory: Is a Filesystem All You Need?" letta.com/blog/benchmarking-ai-agent-memory.
5. Letta AI (2025). "Letta Evals: Evaluating Agents that Learn." letta.com/blog/letta-evals.
6. Mem0 (2026). "AI Memory Benchmarks in 2026." mem0.ai/blog/ai-memory-benchmarks-in-2026.
7. Mem0 (2026). "State of AI Agent Memory 2026: Benchmarks, Architectures & Production Gaps." mem0.ai/blog/state-of-ai-agent-memory-2026.
8. Memory, V. (2026). "The State of AI Agent Memory in 2026: What the Research Actually Shows." Towards AI.
9. Yadav, Y. (2026). "AI Agent Memory Systems in 2026: Mem0, Zep, Hindsight, Memvid and Everything In Between." Dev Genius.
10. Vectorize (2026). "Best AI Agent Memory Systems in 2026: 8 Frameworks Compared." vectorize.io.
11. Retriever/Zep (2025). "Zep Is The New State of the Art In Agent Memory." blog.getzep.com.
12. Zep vs Mem0 (2025). "Is Mem0 Really SOTA in Agent Memory?" blog.getzep.com.
13. Anthropic (2024). "Model Context Protocol (MCP) Specification." modelcontextprotocol.io.
14. IDC (2025). "AI Agents Market Forecast, 2025–2030."
15. MarketsandMarkets (2025). "AI Agents Market by Component, Agent Type, Application – Global Forecast to 2030."
16. MemRoOS Internal Evaluation (2026). "Competitive Benchmark Results 2026-05-24." Codex eval run + live recall evals.