# nox-mem: Pain-Weighted Hybrid Memory for LLM Agents

**Technical report — v1.0.0 (arXiv preprint)**

**Paper version:** v1.0.0 (2026-06-30) — frozen for arXiv submission; changelog in `paper/CHANGELOG.md`

**Author:** Luiz Antonio Busnello (Toto) **Platform:** OpenClaw Autonomous Agent Platform **Infrastructure:** Hostinger KVM4, Tailscale VPN, Debian Linux

---

## Abstract

We introduce **nox-mem**, a persistent memory system for autonomous LLM agents organized around a single design principle: **pain-weighted hybrid memory with shadow discipline — yours by design**. Every retrieval and retention decision is governed by an evidence-weighted salience formula in which *pain* — an operator-assigned severity in [0.1, 1.0], persisted on every chunk — is a first-class signal (its isolated retrieval effect is directional and not yet statistically significant, §7.1, and section-aware ranking is the dominant empirical driver, §5.1.3); the formula itself evolved from multiplicative to additive form through the system's own shadow-mode telemetry (§5.1.2), and every ranking change is gated by a mandatory shadow phase before production activation. Compared to existing memory systems (mem0, Letta, Zep, EverOS, LightRAG, and MeMo), nox-mem offers several advantages: **(a)** *live writeback with sub-second indexing* (inotifywait-driven, no batch retrain or daily reindex required), **(b)** *typed temporal decay* (per-`chunk_type` retention windows with never-decay for feedback/person — see §2 and §8.2), **(c)** *full provenance to chunk and source* (every retrieval result carries `chunk_id` + `source_file`, every destructive op is wrapped in `withOpAudit()` with a VACUUM INTO pre-snapshot), **(d)** *first-class self-evolution* through a `crystallize/reflect/consolidate` triad that promotes high-hit pending items to durable lessons and synthesizes cross-session insights nightly (§3.4), and **(e)** *zero vendor lock-in*: a single SQLite file, MIT-licensed, with provider-agnostic embeddings (Gemini default, swappable). Deployed in production since March 14, 2026, the system manages 94.9k+ memory chunks with ~99.99% vector coverage, ~15.6k knowledge graph entities with ~21.5k relations (as of 2026-06-04, post corpus-hygiene pass), and serves 6 specialized AI agents with isolated yet interconnectable memory spaces. On the entity-flavored golden set (n=100), the full ablation stack reaches **nDCG@10 = 0.6237** (+78.8% over the G3 pre-Wave-A baseline; §5.1.1). A **Conditional Hard Mutex** (G10d) gates section and source-type boosts by query entity count, recovering multi-hop (+1.58% nDCG@10) and adversarial (+3.04% nDCG@10, +6.25% MRR) regressions (§5.1.4). On the EverMemBench cross-system benchmark (5-batch, n=3,121), nox-mem scores **62.22%** with Gemini-

2.5-flash (+2.95 pp over MemOS; §5.1.5), **51.68%** with GPT-4.1-mini (+9.13 pp over MemOS, 95% CI [49.88, 53.49]; §5.1.6), and **63.28% Overall + 88.42% Memory Awareness composite with Gemini-3-flash — both SOTA versus the published MemOS Table 4 baseline (GPT-4.1-mini)** (+20.73 pp Overall, +32.74 pp MA; §5.1.10). On classical multi-hop QA, nox-mem achieves dual SOTA without specialized fine-tuning: **MuSiQue dev F1 58.62%** (+22.82 pp over IRCoT, +8.92 pp over EX(SA); §5.2.1) and **HotPotQA dev distractor ans_F1 73.37%** (above DPR+FiD reader SOTA; §5.2.2). LoCoMo cross-bench retrieval@10 strict reaches **74.52%**, above Mem0's *published* end-to-end answer-F1 SOTA of 66.88% (a different metric, not a same-corpus head-to-head; §5.3.1); the F1 push is rank-5 (51.85%, above Zep / LangMem) constrained by a verbosity gap (§5.3.2). The §5.4 resolution of the EverMemBench F_MH paradox shows the 3–7% absolute is a corpus-structural property, not a multi-hop reasoning ceiling. Production characteristics are strong across three self-hosted operational axes: **KG path retrieval p50 = 2.9 ms** (sub-10 ms class, §5.7.1), **cost $0/query KG path and ~667× cheaper than Mem0 Cloud on hybrid** (§5.7.2), and **399 MB RSS** (resident set size) **self-hosted single-process** footprint (§5.7.3). The cross-bench LongMemEval validation (n=300, §5.6) confirms the same per-category fingerprint across an orthogonal benchmark distribution. Wave 2 (2026-05-31 – 2026-06-02, §5.5.4–§5.5.8) contributes an empirical per-backbone × per-knob composability study: three Wave A single-stage retrieval knobs (KG path, AC, MQ) transfer at only 0–40% efficiency from gpt-4.1-mini to Gemini-3-flash (D75), establishing that retrieval-stage compensation mechanisms attenuate on stronger backbones; IterB ReAct (§5.5.2) remains the sole validated F_MH lever on Gemini-3-flash at +2.01 pp. An architectural lock discovery (§5.5.7) reveals that composability projections must account for both backbone-portability and code-level architectural composability — two independent non-trivial requirements. The Q4 cross-system comparison (§6) provides pre-registered, same-corpus results against five competing memory systems (two, Mem0 and agentmemory, produced head-to-head quality numbers; three were documented deployment non-runs). Under each system's native embedder the two leaders **split** — Mem0 wins LoCoMo (nDCG@10 0.469 vs 0.426), nox-mem wins LongMemEval (§6.3); controlling the embedding provider (both Gemini-3072d, full n=2,482) **inverts the split**, with nox-mem outperforming Mem0 on both datasets (LongMemEval 0.526 vs 0.406; LoCoMo 0.495 vs 0.441) and all five represented categories — a task-type ablation rules out the one asymmetry favoring nox-mem, with three residual confounds declared (§6.3.2).

---

## 1. Introduction

### 1.1 Problem Statement

Large Language Model (LLM) agents operating in production environments face a fundamental limitation: context window ephemerality. When a conversation

ends or context is compacted, the agent loses accumulated knowledge, decisions, and operational state. For multi-agent systems where specialized agents collaborate on complex tasks, this problem compounds — agents cannot learn from each other's experiences, cannot recall past decisions, and cannot build institutional knowledge over time.

### 1.2 Design Goals

nox-mem was designed with four core objectives:

1. **Persistent Memory**: Survive context window resets, session boundaries, and agent restarts
2. **Intelligent Retrieval**: Return semantically relevant results, not just keyword matches
3. **Cross-Agent Intelligence**: Enable knowledge sharing across isolated agent workspaces
4. **Operational Autonomy**: Self-maintain through automated consolidation, pruning, and indexing

### 1.3 Scope

The system operates within the OpenClaw platform, serving 6 AI agents (Nox, Atlas, Boris, Cipher, Forge, Lex) on a single VPS with 4 vCPUs and 8GB RAM. Each agent has a distinct role and memory profile. The workspace (shared memory) and individual agent databases form a federated memory architecture.

### 1.4 Related Memory Systems and the Six Gaps

The published memory-for-LLM-agents literature spans roughly three families: (i) *vector-store wrappers with metadata layers* — **mem0** [1], **Letta** [2]; (ii) *temporal- and provenance-aware memory services* — **Zep** [3], **memanto**; (iii) *KG-augmented and graph-fused retrieval* — **LightRAG** [4] (HKU, EMNLP 2025), **HippoRAG2** [5]; and (iv) *parametric-memory paradigms*, most recently

[1] mem0ai/mem0 — open-source memory layer for LLM agents (PostgreSQL + Qdrant backend, OpenAI embeddings by default). github.com/mem0ai/mem0. Used in §1.4, §6.3, Table 2.

[2] Letta (formerly MemGPT) — agent-loop memory architecture with archival/recall memory separation. github.com/letta-ai/letta. Used in §1.4, §6.3, Table 2.

[3] Zep — temporal knowledge-graph memory service with summarization. github.com/getzep/zep. OpenAI embedding is the hardcoded default in the OSS distribution. Used in §1.4, §6.3, Table 2.

[4] Guo et al., *LightRAG: Simple and Fast Retrieval-Augmented Generation*, EMNLP 2025 (HKU). github.com/HKUDS/LightRAG (~35k stars, MIT). Cited in §1.4 as a KG-augmented baseline; §3.4 references its LLM-summarized incremental KG-merge pattern as a forward-looking optimization (LightRAG-style summarization parking-lotted until KG density >=10× current; see `docs/COMPETITIVE-ANALYSIS-2026-05-19.md`).

[5] HippoRAG2 — graph-augmented retrieval with Personalized PageRank over an entity-relation graph. Cited as a graph-baseline peer in §1.4 and §6.

**MeMo** [6] (which folds reflections into model weights via continued pretraining — the design opposite of ours). **EverMind-AI/EverOS** [7] occupies a distinct slot: it is the only memory OS in this space that publishes its own benchmark dataset (EverMemBench) and reports threshold numbers, raising the bar for honest cross-system comparison (§6).

Across these systems, six recurring gaps appear in the design space. Each gap motivates a concrete subsystem of nox-mem. Table 1 summarizes who covers what:

**Table 1 — Comparison of desirable memory-system properties (Six Gaps).**

| # | Gap | nox-mem | mem0 | Letta | Zep | EverOS | LightRAG | MeMo |
|---|---|---|---|---|---|---|---|---|
| 1 | Static injection | PASS: live write-back | NB: partial | PASS | PASS | PASS | FAIL batch | FAIL retrain |
| 2 | No temporal decay | PASS: salience + retention | FAIL | FAIL | PASS | NB: partial | FAIL | FAIL |
| 3 | No provenance | PASS: chunk_id + source_file | NB: partial | PASS | PASS | PASS | PASS | FAIL baked-in |
| 4 | Flat memory | PASS: KG + section_boost | FAIL | FAIL | PASS | PASS hyper | PASS dual-level | FAIL |
| 5 | No write-back | PASS: crystallize/reflect/consolidate | NB: partial | PASS | PASS | PASS EvoAgent | FAIL | FAIL |
| 6 | Indexing delay | PASS: inotify-wait <1s | NB: | PASS | PASS | NB: | NB: batch | FAIL retrain |

**Why each gap matters, and how nox-mem closes it:**

1. **Static injection.** A memory system that only reads at the start of a session cannot observe what the agent does *during* the session. nox-mem's

[6] arXiv 2605.15156v2, *MeMo: Towards Language Models with Associative Memory Mechanisms* (parametric reflections folded into model weights). Cited as the design opposite of nox-mem's externalized, inspectable memory paradigm. §1.4, abstract.

[7] EverMind-AI / EverOS — github.com/EverMind-AI (~5k stars, Apache 2.0). Publishes EverMemBench dataset and an EvoAgentBench-framed evolution loop. The only memory-OS peer in our taxonomy that publishes its own benchmark; §3.4 is the direct narrative counter to EvoAgent framing. Honest-comparison action item: run nox-mem on EverMemBench (queued as F4 in §7.2).

watcher [8] re-ingests every modified `.md` / `.json` file within ~1 second of the write, with idempotent chunk replacement (§3.1).

2. **No temporal decay.** Treating a daily note from 91 days ago the same as a crystallized decision floods retrieval with stale noise. nox-mem assigns each `chunk_type` a `retention_days` window (V8 schema column; `feedback/person` = never-decay, `lesson` = 180d, `decision/project` = 365d, `daily` = 90d) and folds it into the salience formula (§8.2).
3. **No provenance.** Parametric and opaque-vector memories cannot answer "*why* did you return this?". Every nox-mem result carries the originating `chunk_id` and `source_file`, every destructive operation goes through `withOpAudit()` with a `VACUUM INTO` pre-snapshot to `/var/backups/nox-mem/pre-op/`, and the `ops_audit` table is append-only.
4. **Flat memory.** Without structure, hybrid search collapses to "more recent or more frequent wins". nox-mem layers (a) FTS5 (SQLite full-text search) over chunk text, (b) typed retention, (c) an LLM-extracted knowledge graph (§8), and (d) an entity-file format (`frontmatter` / `compiled` / `timeline` sections) with section-aware `section_boost` — the dominant driver of the +78.8% gain in §5.
5. **No writeback.** A system that can only *be read* cannot grow. nox-mem's self-evolution triad — `crystallize`, `reflect`, `consolidate` — is described in §3.4 and is the most direct counter to EverOS's EvoAgentBench framing [9].
6. **Indexing delay.** Daily-batch reindex makes "new memory" a 24h-resolution operation. inotifywait makes it sub-second (§3.1), and `--dry-run` mode plus `withOpAudit()` snapshots make destructive ops (reindex, consolidate, compact, crystallize, kg-prune) reversible.

The single design principle that ties these closures together is **pain weighting under shadow discipline**: every chunk carries an explicit `pain` severity, every ranking change rolls out first in `NOX_SALIENCE_MODE=shadow` with /api/health telemetry [10], and only graduates to `active` after operator review.

---

[8] `nox-mem-watcher` systemd service running `inotifywait` on `memory/` directories. See `docs/ARCHITECTURE.md` and §3.1 of this paper.

[9] EverMind-AI / EverOS — github.com/EverMind-AI (~5k stars, Apache 2.0). Publishes EverMemBench dataset and an EvoAgentBench-framed evolution loop. The only memory-OS peer in our taxonomy that publishes its own benchmark; §3.4 is the direct narrative counter to EvoAgent framing. Honest-comparison action item: run nox-mem on EverMemBench (queued as F4 in §7.2).

[10] `NOX_SALIENCE_MODE` environment variable controls the three-state gate (`shadow` | `active` | `off`). Default is `shadow`. Telemetry exposed at `/api/health.salience`. See §3.4.3, `staged/1.7a/edits/salience.ts`, and `docs/CONFIGURATION.md`.

## 2. System Architecture

### 2.1 Infrastructure Overview

The system runs on a Hostinger KVM4 VPS accessible via Tailscale VPN (a private Tailscale address). Five systemd-managed services provide the runtime environment:

| Service | Port | Type | Function |
|---|---|---|---|
| openclaw-gateway | 18789 | WebSocket | Agent communication gateway |
| nox-mem-watcher | — | inotifywait | Filesystem event monitor |
| nox-mem-api | 18800 | HTTP/JSON | Dashboard data API |
| ollama | 11434 | HTTP | Local LLM inference (llama3.2:3b) |
| tailscaled | — | WireGuard | VPN mesh connectivity |

### 2.2 Database Schema

The primary storage is SQLite 3 with WAL (Write-Ahead Logging) mode for concurrent access. The schema (version 3) contains:

**Core Tables:**

- `chunks` — Memory fragments with full-text indexing
  - `id` (INTEGER PK), `source_file` (TEXT), `chunk_text` (TEXT), `chunk_type` (TEXT)
  - `source_date` (TEXT), `is_consolidated` (INTEGER), `memory_type` (TEXT)
  - `created_at`, `updated_at` (TEXT, ISO 8601)
  - `metadata` (TEXT, JSON)
- `chunks_fts` — FTS5 virtual table with porter unicode61 tokenizer
  - Content-sync triggers (INSERT, UPDATE, DELETE) maintain index consistency
  - BM25 ranking with configurable column weights (1.0, 0.5, 0.5)
- `consolidated_files` — Processing state tracker
  - `source_file` (TEXT PK), `status` (INTEGER: 0=pending, 1=done, -1=failed)
- `meta` — Key-value configuration store (schema_version, cursors, metrics)

**Knowledge Graph Tables:**

- `kg_entities` — Named entities with type classification and mention counting
  - UNIQUE constraint on (name, entity_type)
  - TTL tracking via `first_seen`, `last_seen`
- `kg_relations` — Typed relationships between entities
  - Confidence scoring (0.0-1.0) with temporal decay
  - TTL via `expires_at` (90-day default), `last_confirmed`
  - Evidence linking via `evidence_chunk_id`
- `decision_versions` — Architectural decision version history

– Supersession chain via `is_current` flag and `superseded_at` timestamp

**Vector Tables (sqlite-vec):**

- `vec_chunks` — Virtual table storing float32 embeddings (3072 dimensions)
- `vec_chunk_map` — Rowid-to-chunk_id mapping (sqlite-vec requires rowid-based access)
- `dedup_log` — Suppressed duplicate tracking for audit

### 2.3 Chunk Type Taxonomy

Memory chunks are classified into 10 types based on source file path patterns:

| Type | Source Pattern | Current Count | Purpose |
|---|---|---|---|
| team | `shared/` | 499 | Shared team knowledge, cross-agent docs |
| daily | `memory/YYYY-MM-DD` | 161 | Daily operational notes |
| other | (default) | 126 | Unclassified content |
| decision | `memory/decisions.md` | 34 | Architectural and strategic decisions |
| lesson | `memory/lessons.md` | 21 | Errors, corrections, learnings |
| project | `memory/projects.md` | 11 | Active project tracking |
| pending | `memory/pending.md` | 8 | Incomplete tasks and blockers |
| feedback | `memory/feedback/` | 6 | User and system feedback |
| person | `memory/people.md` | 6 | People profiles and contacts |
| digest | `memory/digests/` | 2 | Weekly summary reports |

### 2.4 Multi-Agent Memory Architecture

Each of the 6 agents operates with an isolated database at `/root/.openclaw/agents/{name}/tools/nox-mem/`. The `OPENCLAW_WORKSPACE` environment variable controls path resolution across all modules, enabling the same nox-mem binary to operate on different databases depending on the calling context.

**Agent Memory Distribution (as of March 23, 2026):**

| Agent | Role | Chunks | DB Size | Dominant Type |
|---|---|---|---|---|
| Nox | Chief of Staff | 185 | 268 KB | daily (91) |

| Agent | Role | Chunks | DB Size | Dominant Type |
|---|---|---|---|---|
| Boris | Head of Communications | 148 | 268 KB | team (50) |
| Forge | Code Reviewer | 182 | 292 KB | daily (135) |
| Atlas | Research | 30 | 128 KB | other (17) |
| Cipher | Security | 31 | 132 KB | other (12) |
| Lex | Legal/Compliance | 31 | 132 KB | other (12) |
| **Workspace** | **Shared** | **874** | **25.2 MB** | **team (499)** |

Total system memory: 1,481 chunks across 7 databases (initial-deployment snapshot, March 2026; the main store has since grown to ~95k chunks — see Abstract).

### 2.5 User-Facing Primitives

nox-mem exposes a deliberately small public contract: **three primitives**, all backed by the same SQLite store and surfaced identically across three transport layers (CLI, HTTP API, MCP). Every advanced verb (`reflect`, `cross-search`, `kg-path`, `crystallize`) decomposes internally into sequences of these three primitives.

**Primitive 1 — `search` (hybrid retrieval).** FTS5 BM25 + Gemini semantic (3072d) → Reciprocal Rank Fusion (RRF), k=60, with optional Hard Mutex section gating and SOURCE_TYPE_BOOST overlays. Returns ranked chunks with `score`, `match_type`, and provenance fields (`source_file`, `section`, `created_at`, `updated_at`). Detailed in §4.

**Primitive 2 — `answer` (grounded RAG).** Internally calls `search` with `topK = 10`, builds a citation-anchored prompt over the retrieved chunks, invokes the configured LLM (`gemini-2.5-flash-lite` by default per D41), and parses inline `[chunk_<id>]` citations. Anti-hallucination guard: citations pointing to chunks outside the retrieved set trigger a single retry with a stricter prompt; a second failure raises `AnswerError('hallucination_after_retry')`. Empty-retrieval short-circuit avoids LLM spend when no chunks match. Measured p95 latency: 101.74 ms on the offline mock-LLM bench (PR #40, 42× under the 4.3 s budget); live p95 with Gemini Flash Lite ranges 1.5–2.5 s. Implementation: `staged/P1/edits/src/lib/answer/{index,retrieval,prompt,provider,config}.ts`.

**Primitive 3 — Temporal filter (`--as-of` / `--changed-since`).** Time-travel and recency-window selectors implemented as hard SQL pre-filters — not ranking boosts. `--as-of <date>` restricts to chunks satisfying `created_at <= date AND (deleted_at IS NULL OR deleted_at > date)`; `--changed-since <date>` restricts to chunks satisfying `updated_at > date OR created_at > date`. Combined, the two clauses AND. Accepted formats: ISO 8601 (`2026-05-01` or full `2026-05-01T00:00:00Z`) and relative (`7d`, `1w`, `30d`, `2h`, `15m`). Uses existing `chunks.created_at` and `chunks.updated_at` columns from schema v18 — no schema changes, no ranking changes. The filter is orthogonal to the E13 temporal proximity boost (`NOX_TEMPORAL_PATH`, §5), which

additively reweights ranking by recency rather than restricting the candidate set. Implementation: `staged/P3/edits/{dates,search,api-server}.ts`.

**Composition.** The three primitives compose orthogonally — for example, `answer "what incidents happened last week?" --changed-since 7d` retrieves only chunks updated in the last seven days, then synthesizes a grounded answer over that restricted candidate set. This closure property — three small primitives, deterministic semantics, identical surface across transports — is the contract that makes nox-mem composable from agent runtimes that have no prior knowledge of the implementation.

**Tagline.** *3 primitives, 1 file, any LLM.* The three primitives are search + answer + temporal filter; the one file is the SQLite database on the operator's disk; the LLM provider is swappable via the `LLMProvider` interface (Gemini default, OpenAI / Anthropic / Ollama / vLLM available) without code changes. Full operator reference: `docs/PRIMITIVES.md`.

---

## 3. Memory Pipeline

### 3.1 Ingestion

Files created or modified in monitored directories trigger the inotifywait-based watcher service. The watcher implements:

- **Debounce logic**: 2-second delay to batch rapid successive writes
- **File filtering**: Only `.md` and `.json` files are processed
- **Recursion prevention**: `MEMORY.md` and `SESSION-STATE.md` are excluded to avoid feedback loops
- **Heartbeat**: Touch `/tmp/nox-mem-watcher-heartbeat` on every event for liveness monitoring

Upon trigger, `ingestFile()` executes:

1. Read file content with UTF-8 sanitization (fixes common mojibake patterns for Portuguese text)
2. Detect chunk type from relative file path
3. Extract date from filename pattern (YYYY-MM-DD)
4. Split content into semantic chunks:
   - Markdown: Split on H2/H3 headers, with sub-splitting for chunks exceeding 500 words
   - JSON: Array items become individual chunks; object entries become key-value pairs
   - Small chunks (<20 words) are merged with the previous chunk
5. Delete existing chunks for the same source file (idempotent re-ingestion)
6. Insert new chunks via prepared statement transaction
7. Auto-vectorize if GEMINI_API_KEY is available (up to 20 chunks per file)

### 3.2 Consolidation

Nightly consolidation (23:00, 5-minute stagger across agents) processes daily notes into structured topic files:

1. **Reindex**: Scan all `.md/.json` files in memory directories, rebuild chunk index
2. **Extract**: Use Ollama llama3.2:3b to identify facts, decisions, lessons, and action items from daily notes
3. **Append**: Add extracted content to topic files (decisions.md, lessons.md, people.md, projects.md, pending.md)
4. **Notion Sync**: Push structured items to "Memoria & Decisoes" Notion database (best-effort, non-blocking)
5. **Git Commit**: Auto-commit memory changes with standardized message format
6. **Session Update**: Refresh SESSION-STATE.md with current statistics

### 3.3 Deduplication

Before insertion, chunks are checked for duplicates using a two-tier strategy:

- **Primary**: Gemini cosine similarity with 0.85 threshold (when embeddings are available)
- **Fallback**: Keyword overlap calculation with 60% threshold
- **Audit**: Suppressed duplicates are logged to `dedup_log` table with reason and preview

### 3.4 Self-Evolution: How nox-mem Improves Over Time

Most memory systems decouple *retrieval* from *learning*: retrieval is online, learning is either absent or requires retraining a parametric backbone (MeMo) or relying on a closed evolution loop (EverOS EvoAgentBench [11]). nox-mem instead promotes self-evolution to a first-class subsystem composed of four cooperating mechanisms, all transparent and inspectable in the live SQLite file. This subsection is the direct counter to Gap #5 (no writeback) in Table 1.

#### 3.4.1 Crystallize loop — `pending` → `lesson` after N hits, with pain accumulation.

The `crystallize` command (exposed via the CLI, the Model Context Protocol (MCP) server tool, and `POST /api/crystallize` + `POST /api/crystallize/validate` on the HTTP API; see `src/api-server.ts`

[11] EverMind-AI / EverOS — github.com/EverMind-AI (~5k stars, Apache 2.0). Publishes EverMemBench dataset and an EvoAgentBench-framed evolution loop. The only memory-OS peer in our taxonomy that publishes its own benchmark; §3.4 is the direct narrative counter to EvoAgent framing. Honest-comparison action item: run nox-mem on EverMemBench (queued as F4 in §7.2).

and `src/crystallize.ts` [12]) implements an LLM-assisted consolidation that synthesizes durable entities from recent chunks. Operationally, chunks ingested into `memory/pending.md` (`chunk_type = pending`, `retention_days = 30`) act as a working set of unresolved items. As the same pending item is referenced by retrieval (and as related daily/team chunks accumulate around it), its effective *pain* signal grows — both via direct operator updates to the `pain` column (V9 schema) and via co-occurrence with high-pain neighbors. After enough hits and sufficient accumulated severity, `crystallize` uses Gemini to identify the pattern across chunks, emits a canonical entity file under `memory/entities/<type>/<slug>.md`, and effectively promotes the cluster from `pending` (30-day decay) to `lesson` (180-day decay) or `decision`/`project` (365-day decay) [13]. The promotion is wrapped in `withOpAudit()`, so the pre-state snapshot lives at `/var/backups/nox-mem/pre-op/crystallize-<ts>-<pid>-<uuid>.db` and a row lands in the append-only `ops_audit` table.

### 3.4.2 Pain weighting auto-adjust — feedback of use raises pain on hit chunks.

The `pain` field on `chunks` is an explicit severity in [0.1, 1.0] (0.1 trivial, 1.0 production outage). It can be set explicitly by the author, but it also drifts upward implicitly: chunks that are *repeatedly* surfaced by retrieval and *accepted* by the operator (via the `feedback/` directory, whose `chunk_type = feedback` is never-decay) accumulate co-occurrence with high-pain entries during nightly consolidation, which feeds back into the salience formula. The result is that lessons learned from real incidents naturally rise to the top over time, while toy or low-severity content fades even when frequent. This is the inverse of pure recency- or frequency-based memory.

### 3.4.3 Salience decay continuous in background — `recency × pain × importance`.

The salience function lives in `src/lib/salience.ts` [14] (and is mirrored in the staged copy at `staged/1.7a/edits/salience.ts`). Its underlying principle is multiplicative — a memory scores high only when it is *simultaneously* recent, painful, and important:

[12] HTTP handlers in `src/api-server.ts` (routes `/api/crystallize`, `/api/crystallize/validate` — confirmed in `staged/1.6/edits/api-server.ts:253-273`); core logic in `src/crystallize.ts` exporting `crystallize()`, `validateProcedure()`, `listProcedures()`. Wrapped by `withOpAudit()` (`src/lib/op-audit.ts`).

[13] V8 schema typed retention defaults (in `chunks.retention_days`): `feedback` = 0 (never-decay), `person` = 0 (never-decay), `lesson` = 180d, `decision` = 365d, `project` = 365d, `team` = 120d, `daily` = 90d, `pending` = 30d, `graph_node` = 60d, fallback = 90d. See `staged/1.7a/edits/salience.ts:46-56` (`DEFAULT_RETENTION_BY_TYPE`) and `CLAUDE.md` §"Schema v10".

[14] `src/lib/salience.ts` — canonical implementation of the salience formula (multiplicative principle in §3.4.3; weighted-additive v2 production form `W_IMPORTANCE·importance + W_RECENCY·recency + W_PAIN·pain + W_ACCESS·access_score` in §5.1), mirrored in the staged copy at `staged/1.7a/edits/salience.ts` (lines 1–80 contain the module docstring spelling out the formula, the three-state mode gate, and the per-type retention defaults).

`salience = recency × pain × importance`

with components:

- **recency** in [0,1] — half-life-style decay over the chunk's `retention_days` window (`feedback`/`person` → never-decay → `recency = 1.0`; everything else decays per Table V8 [15]). Decay is *continuous*: it is recomputed at every retrieval, so there is no batch step that "ages" memory — aging is a property of the read path.
- **pain** in [0,1] — severity as described in §3.4.2.
- **importance** in [0,1] — `chunk_type` / `source_type` / tier signal (manual mapping; e.g., `decision` and `lesson` rank above `daily`).

In production since Wave A (§5.1), this principle is realized as a **weighted-additive** form — `salience = W_IMPORTANCE·importance + W_RECENCY·recency + W_PAIN·pain + W_ACCESS·access_score` (weights 0.55 / 0.15 / 0.10 / 0.20) — which empirically outperformed the strict multiplicative product on the Wave A golden set; §5.1 reports the ablation. The multiplicative expression above states the principle; the additive form is what ships.

The mode of operation is gated by `NOX_SALIENCE_MODE`: `shadow` (default — compute and log to `/api/health.salience` but do not apply to retrieval rankings), `active` (apply as an additive delta in [-0.5, +0.5] on top of RRF), and `off` (short-circuit to 0 for ablation experiments). This three-state gate is the operational realization of *shadow discipline* (§4 and §5).

### 3.4.4 Reflect pipeline — batch nightly cross-session synthesis.

The `reflect` command (CLI / MCP / `POST /api/reflect`, backed by `src/reflect.ts` and cached via `getReflectCacheStats()` exposed in `/api/health.reflectCache` [16]) runs a batch synthesis pass over recent chunks. Its job is *not* to retrieve answers for the user but to produce cross-session lessons: it queries hybrid search for a topic, asks Gemini to summarize the patterns observed across the result set, and writes the summary back as a `feedback` or `lesson` chunk with the confidence default for derived chunks (see `CONFIDENCE_DERIVED` in `docs/CONFIGURATION.md`). Because reflect results are themselves cached (LRU, exposed via `/api/health.reflectCache.entries`), expensive synthesis is amortized across repeated queries.

### 3.4.5 Consolidate — nightly orchestration that ties it together.

Nightly consolidation (23:00, 5-minute stagger across agents — see §3.2) is the orchestration layer that runs reflect for high-pain topics, runs crystallize for suffi-

[15] V8 schema typed retention defaults (in `chunks.retention_days`): `feedback` = 0 (never-decay), `person` = 0 (never-decay), `lesson` = 180d, `decision` = 365d, `project` = 365d, `team` = 120d, `daily` = 90d, `pending` = 30d, `graph_node` = 60d, fallback = 90d. See `staged/1.7a/edits/salience.ts:46-56` (`DEFAULT_RETENTION_BY_TYPE`) and `CLAUDE.md` §"Schema v10".

[16] `src/reflect.ts` exporting `reflect()` and `getReflectCacheStats()` (confirmed in `staged/1.6/edits/api-server.ts:12`). Cache statistics surfaced in `/api/health.reflectCache`. See also `docs/POSTMAN.md` for the API contract.

ciently aged `pending` items, and syncs the resulting durable entities to the topic files (`decisions.md`, `lessons.md`, `people.md`, `projects.md`). Like crystallize, it goes through `withOpAudit()` with a pre-op `VACUUM INTO` snapshot. This is the daily heartbeat of self-evolution.

**Together, these four mechanisms make nox-mem a memory that *grows along the gradient of operator pain*:** chunks that hurt the operator (production outages, lost work, repeated mistakes) survive decay, get promoted from pending to lesson, accumulate co-occurrence weight, and rank progressively higher — without retraining a model and without trusting a closed evolution loop. Every step is visible by opening `nox-mem.db` in `sqlite3` and inspecting `ops_audit`, `chunks.pain`, `chunks.retention_days`, and the entity files.

### 3.5 Session Priming: the read-side of self-evolution

The mechanisms above are *write-side* — they decide what the store keeps and how it ranks. But a memory that only grows is wasted if every new session starts blind and must re-discover context through cold search. **Session priming** closes the loop on the *read-side*: each session (any agent, any MCP/HTTP client) is born contextualized. `GET /api/brief` (handler in `src/api/brief.ts`) returns a salience-ranked, scope-filtered digest of the top-N chunks, injected at session start. The brief is strictly read-only over `chunks` — it never touches `access_count` or `last_accessed_at`, so the organic-use signal that feeds salience (§3.4) stays uncontaminated; serve-history is tracked in a separate `brief_log` table. With an `agent` scope the digest is a guaranteed union (~half agent-specific, half global high-salience), and near-duplicate variants are collapsed by token-containment so the budget is spent on distinct content.

**An honest measurement, and a methodology caveat.** Running priming in production (6 agents + shared workspace store, ~6.7k serves/day) surfaced a failure mode worth reporting. Over a clean 7-day window the brief served only **83 distinct chunks across 46.8k serves (0.18% diversity)**, median content age 48 days: salience, dominated by importance and the accumulated pain of old incidents, produces a near-static top set, and recently-written content (the very output the loop produces) rarely enters — we measured **931 recent (<=7d), relevant chunks (importance >= 0.7 or pain >= 0.7) that were never once served**. More instructive was a *negative* result on evaluation: we could not measure a downstream "follow-up rate" (does a primed chunk get used later?) because the signal does not exist — in 7 days there were **3 genuine searches and 0 `answer` calls**. This is structural, not a logging gap: priming-by-injection delivers the knowledge *into the context window*, so the agent does not re-query what it already holds. **The utility of an injection-based primer is therefore a property of the served set (diversity, freshness, coverage of what matters) — not of downstream re-retrieval.** Memory work that imports a re-query metric from RAG-style retrieval will mis-evaluate this class of mechanism.

**Diversity term (D2).** The fix follows the same discipline as §3.4.3: it does **not** fork `calculateSalience` (that would alter search too) — it is a re-rank *inside the brief only*. (A) a novelty penalty `brief_score = salience - min(P_max, lambda*log1p(n_serves))` reads `brief_log` to demote over-served chunks, saturating in log so serving 2,000x ~= serving 100x; (B) a freshness quota reserves F of the N slots for recent, relevant, not-yet-served content (a separate query — the candidate pool, ranked by importance and access, structurally excludes `access_count = 0` newcomers). A **high-pain floor** makes incidents with `pain >= 0.9` immune to both demotion and displacement: diversity must not hide the critical. Per the shadow-discipline rule (§3.4.3, §5), the term ships behind `NOX_BRIEF_DIVERSITY = off | shadow | active`, default off and bit-identical to the prior behavior; it is validated in shadow (logging the would-enter/would-leave diff against a per-day gate: diversity up, median age down, floor preserved, churn non-thrashing) before any flip to active. Notably, the shadow gate itself caught a design bug on its first run — the freshness quota was displacing `pain = 1.0` incidents — which the floor now prevents; the active-mode quality numbers are reported as ongoing validation (§7.2).

**Active-mode validation surfaced a second, subtler failure — and the fix unifies the two slots.** Two refinements followed the first gate. First, the freshness query as scoped (`global+agent`) only ever saw `sessions/<agent>/%`, never the curated `memory/entities/%` store, so freshly-consolidated lessons and decisions were structurally invisible to every brief; a *split-slot* interleave (one agent-recent plus one curated-global per brief, with its own age window) restored that channel. Second — and this is the instructive part — once the curated channel was live in `active`, a 24-hour gate showed the global slot serving **190 distinct curated chunks on day one, then exactly 1 per day thereafter**. The cause was the slot's *hard* anti-repeat (`id NOT IN brief_log` over a 72h window): under production volume (~6.7k briefs/day) over a 189-chunk eligible pool, the entire pool is served — and thus excluded — within a single day, after which the slot dries up and fails open to the static salience top-1 for the remaining 72h. The first remedy *seemed* obvious: replace hard exclusion with the soft novelty penalty already used by the primary pool (mechanism A) — serving a curated chunk *demotes* it by `lambda*log1p(n_serves)` (capped at `P_max = 0.15`) rather than *removing* it, with the high-pain floor keeping critical incidents immune. Active-mode measurement refuted it: the 24-hour gate fell **146 → 67 → 3 distinct curated chunks per day** over three successive days. The cap was the flaw — `P_max` (0.15) is smaller than the base-salience gap between the pool's few high-salience outliers (decisions at importance 0.9 with high access counts) and its homogeneous body, so once the penalty saturated (within one 72h window under production volume) the pick reconverged to the static salience top-$k$. A bounded soft penalty cannot outvote an unbounded salience gap. The fix that held ranks the slot not by *count* of prior serves but by *time since last serve* (mechanism B', coverage-sampling): never-served first, then least-recently-served, tie-broken by salience. Having no ceiling, it sweeps the entire pool continuously regardless of the salience gap.

The first full post-deploy 24-hour active gate (2026-06-27, ~6.8k briefs/day) confirmed it: **184 of 184 eligible entity files served (100% pool coverage), 190 distinct curated chunks, ~45 distinct chunks rotated per hour — sustained, not a day-one burst** (the cumulative distinct count reached 190 within four hours of deploy and held flat through hour 24; the pool is swept fast and then re-cycled by recency rather than exhausted). The general lesson — **hard de-duplication under volume » pool produces bursty rotation; a *saturating* soft penalty silently reconverges once its cap falls below the salience gap; only coverage-by-recency-of-serve, which has no ceiling, produces true continuous rotation** — is the kind of finding only a shadow→active→measure loop on real traffic exposes. An offline diversity metric on a static corpus would have scored all three designs identically, and the saturating-penalty regression in particular would have been invisible without a multi-day active gate.

---

## 4. Hybrid Search System

### 4.1 Architecture

Search combines three complementary retrieval methods:

**Layer 1 — FTS5 BM25 (Keyword)**

SQLite FTS5 with porter unicode61 tokenizer provides fast keyword matching. Results are scored using BM25 with column weights (chunk_text: 1.0, source_file: 0.5, chunk_type: 0.5). Post-retrieval boosting applies:

- Type boost: `decision` and `lesson` chunks receive 2.0x multiplier (higher signal-to-noise ratio)
- Recency boost: Chunks from the last 7 days receive 1.5x multiplier

The query sanitizer strips special characters but preserves hyphens for compound terms (e.g., "nox-mem").

**Layer 2 — Gemini Semantic (Vector)**

Each chunk is embedded using Google's gemini-embedding-001 model (3072 dimensions) with task type RETRIEVAL_DOCUMENT. Query embeddings use task type RETRIEVAL_QUERY for asymmetric similarity optimization.

Vectors are stored in sqlite-vec virtual tables. Retrieval uses cosine distance with a map table (vec_chunk_map) bridging vec_chunks rowids to chunks.id values due to sqlite-vec's rowid-only constraint.

Scoring normalizes distances to a 0-10 scale with type and recency boosting (1.5x and 1.2x respectively, lower than FTS5 to avoid double-boosting in fusion).

**Layer 3 — Reciprocal Rank Fusion (RRF)**

FTS5 and semantic results are merged using RRF with k=60:

```
RRF_score(d) = Σ 1/(k + rank_i(d))
```

Documents appearing in both result sets receive combined scores, marked as `match_type: "hybrid"`. Content-prefix deduplication (first 50 characters) prevents near-duplicate results.

### 4.2 Performance Characteristics

The hybrid approach provides significant quality improvements over single-method search:

| Query | FTS5 Only | Hybrid | Analysis |
|---|---|---|---|
| "qual o proximo passo" | 0 results | ROADMAP + PHASE-3 | Semantic captures intent without keyword match |
| "nox-mem" | 0 results | decisions.md + docs | Vector bypasses tokenizer hyphen issues |
| "quem e o Toto" | people.md | people.md + TEAM_MEMORY | RRF combines exact match + semantic context |

### 4.3 Cross-Agent Search

The `crossSearch()` function opens all 7 databases in read-only mode, executes FTS5 queries in each, and merges results with agent attribution. Deduplication uses content-prefix comparison to handle shared documents that appear across multiple agent databases.

---

## 5. Empirical Evaluation (May–June 2026, fifth revision)

**Headlines (2026-05-29/06-02, 5-batch canonical, four-benchmark triangulation + Q3 IterB ceiling break + Wave 2 backbone-conditional knob study): - Q3 IterB ReAct — F_MH retrieval-stage ceiling broken (D74, PR #419):** on Gemini-3-flash bare baseline, multi-round retrieve-reason loop delivers **+2.01 pp clean F_MH lift (6.02% → 8.03%)** — exceeding the Wave A/B/C single-stage retrieval ceiling of 7.25 pp by +0.78 pp standalone, with the strongest backbone in the matrix (§5.5.2, dual-baseline reporting). - **Wave 2 backbone-conditional knob study (NEW, D75, PRs #423–#425):** Wave A single-stage retrieval knobs (KG path, AC, MQ) transfer at only ~**24–40% efficiency** from gpt-4.1-mini to Gemini-3-flash. All three fail the >=+1.5 pp gate on Gemini-3-flash (NO-REPLICATE). 3-knob sum = +2.01 pp = 24% of original D74 projection +8.43 pp. **IterB ReAct remains the only validated F_MH lever on Gemini-3-flash** (§5.5.5). - **Wave 2 architectural lock discovery**

**(NEW, D75):** IterB ReAct adapter deliberately short-circuits Wave A knobs via explicit `if not iterb_used_path:` guards in `adapter_nox_mem.py`. Composability requires explicit code patch, not merely env-var toggling. Composability projections must address both backbone-portability and architectural composability (§5.5.7). - **Wave 2 MQ MA backbone flip sub-finding (NEW, PR #425):** MQ F_MH lift attenuates gpt-4.1-mini→Gemini (34% transfer rate), while MA composite **flips** from −1.38 pp regression on gpt-4.1-mini to +0.12 pp preserved on Gemini-3-flash. Multi-axis backbone-conditional behavior (§5.5.6). - **Wave 2 Capstone INDETERMINATE — infrastructure constraint, NOT scientific failure (D76, PR #426 draft):** IterB+KG+rerank composability test aborted after Hostinger VPS CPU steal 51–97% sustained. Batch 004 (n=49) preserved; 5-batch statistical threshold not reached. Capstone deferred to dedicated CPU infrastructure (§5.5.8). - **EverMemBench Overall SOTA (Gemini-3-flash, Backbone Matrix):** nox-mem **63.28%** vs MemOS 42.55% (Table 4 baseline) = **+20.73 pp WIN** (PR #397). - **EverMemBench Memory Awareness composite SOTA (Gemini-3-flash):** nox-mem **88.42%** vs MemOS 55.68% = **+32.74 pp WIN** (PR #397). - **MuSiQue SOTA (multi-hop QA, classical):** dev F1 **58.62%** = **+22.82 pp vs IRCoT 35.80%** and **+8.92 pp vs EX(SA) 49.70%** (PR #407). - **HotPotQA SOTA (multi-hop QA, classical):** dev distractor ans_F1 **73.37%** above DPR+FiD reader SOTA range 65–72% (PR #408). - **LoCoMo retrieval@10 SOTA (cross-bench memory):** strict **74.52%** above Mem0 SOTA F1 66.88% (PR #396); F1 push 51.85% (rank-5, above Zep/LangMem) due to verbosity gap (PR #404). - **Phase D (Gemini-2.5-flash):** nox-mem **62.22%** vs MemOS 59.27% = **+2.95 pp WIN** (PR #365). - **Phase H v2 (GPT-4.1-mini cross-backbone):** nox-mem **51.68%** vs MemOS 42.55% = **+9.13 pp WIN** (95% CI [49.88, 53.49], PR #377). - **Backbone portability:** nox-mem regresses −10.54 pp on Gemini-2.5-flash → GPT-4.1-mini swap vs MemOS −16.72 pp = **1.6× more portable** (§5.8). - **Wave B/C composability (D68 + D69):** KG+MAP additive on F_MH +4.04 pp (different-stage compose); same-stage retrieval knobs cap at ~7.25 pp F_MH (D69 Wave C ceiling, PRs #393, #399). - **EverMemBench F_MH paradox REFINED:** MuSiQue 58.62% F1 and HotPotQA 73.37% ans_F1 prove multi-hop reasoning is SOTA; EverMemBench F_MH 3–7% remains largely structural (very long conversation chains + strict scoring), but MAS orchestration (Q3 IterB ReAct, this revision) adds **+2 pp on top of strongest backbone** above the retrieval ceiling (§5.4, §5.5). - **Operational (self-hosted):** KG path p50 **2.9 ms**, **$0/query**, **399 MB RSS** self-hosted single-process (§5.7, PR #403). - **Methodology:** all claims use the **5-batch + 95%**

**CI canonical protocol** (PR #371 + PR #376). Single-batch overstates effects 3–6×. - **Dual-baseline honest reporting (D74):** orchestration-mechanism claims report against both (a) project-convention baseline (Phase H v2 GPT-4.1-mini, conflates backbone + mechanism) and (b) the strongest in-matrix bare baseline (Gemini-3-flash, isolates mechanism). Ceiling-break claims load on (b).

This section documents three evaluation tracks that triangulate the same architectural claims from complementary directions: the **Wave A ablation series** (§5.1.1–§5.1.4, entity-flavored golden set, nDCG@10) exercising the V10 schema's section/source-type/salience drivers; the **EverMemBench cross-system series** (§5.1.5–§5.1.10, n=3,121 queries, task-accuracy vs MemOS Table 4 baselines) covering Phase D / H v2 / G / Lab Q1 standalone knobs / Wave B/C composability / Backbone Matrix; and the **classical multi-hop QA series** (§5.2, MuSiQue + HotPotQA) confirming that multi-hop reasoning is SOTA on standard benchmarks where corpus structure and scoring are not adversarial. §5.3 cross-validates on LoCoMo (memory-bench, conversational), §5.4 resolves the EverMemBench F_MH paradox using the §5.2 and §5.3 evidence, §5.5 reports Q3 orchestration mechanism-class findings — including the **Q3 IterB ReAct ceiling break** (§5.5.2), the **Wave 2 backbone-conditional knob portability study** (§5.5.5–§5.5.6), the **architectural composability lock discovery** (§5.5.7), and the **Wave 2 capstone deferral** (§5.5.8) — §5.6 cross-validates on LongMemEval, §5.7 reports operational characteristics (latency / cost / footprint), §5.8 documents the methodology and honest limitations.

**Dual-baseline reporting convention (D74, 2026-05-31).** Orchestration-mechanism evaluations in this revision report against **two** baselines: (a) the project-convention Phase H v2 GPT-4.1-mini baseline, which preserves comparability with prior revisions but **conflates mechanism lift with any backbone swap**; and (b) the strongest in-matrix bare baseline (Gemini-3-flash, §5.1.10), which **isolates the mechanism's clean effect** on the best available reasoning substrate. Any claim of breaking the retrieval-stage F_MH ceiling load-bears on the (b) clean number — reporting only (a) would conflate ReAct mechanism lift with the +20.73 pp Overall and +32.74 pp MA composite lifts that the Gemini-3-flash backbone already provides standalone (§5.1.10). The convention applies forward to §5.5 Q3 IterB (this revision) and any future MAS-class orchestration evaluations.

---

### 5.1 EverMemBench + Wave A ablation series

**5.1.1 Wave A ablation — setup and headline** The Wave A evaluation uses an entity-flavored golden set of 100 queries (`entity-eval.db`), curated from production usage to exercise the V10 schema's `section` and `pain`

dimensions. Configurations are toggled via environment-variable feature gates (`NOX_SALIENCE_MODE`, `NOX_DISABLE_TIER_BOOST`, `NOX_ENABLE_TIER_BOOST`, `NOX_DISABLE_SECTION_BOOST`, etc.), allowing isolation of individual ranking components without code changes between runs. All measurements occur post-deployment of PRs #150 (salience formula + tier_boost off-by-default), #151 (source_type backfill of 67,949 chunks), and #153 (search wiring). Reported nDCG@10 follows the standard TREC formulation (gain by relevance, log-position discount).

> **Wave A headline (canonical, 2026-05-19):** A8 full stack with active salience reaches **nDCG@10 = 0.6237** on the entity-flavored golden set (n=100), a **+78.8% relative improvement over the G3 baseline (0.3488)** measured prior to Wave A deployment, and **+9.4% over the mid-deployment G4 checkpoint (0.5702)**. The peak ablation isolating `section_boost` alone (A3) reaches 0.6228, recovering **99.85% of the full stack** — section-aware ranking is the dominant driver of the lift.

Progression vs prior ablation generations:

| Generation | Date | A8 nDCG@10 | Δ vs G3 baseline | Notes |
|---|---|---|---|---|
| G3 baseline (pre-Wave A) | 2026-05-15 | 0.3488 | — | Multiplicative salience, tier_boost on, section_boost only via legacy code path |
| G4 mid-deployment | 2026-05-18 | 0.5702 | +63.5% | Additive salience wired but `active < shadow` puzzle observed |
| **G5 V3 canonical** | **2026-05-19** | **0.6237** | **+78.8%** | Wave A fully deployed; reversal `active > shadow` confirmed |

The four sub-claims below decompose the +78.8% total into measurable contributions; the full G5 V3 matrix (12 configurations) is archived in `audits/` and HANDOFF.md (`#g5-v3-matrix-2026-05-19`).

**5.1.2 Wave A — Claim 1: Additive salience outperforms multiplicative** The Wave A formula replaces the legacy multiplicative `salience = recency × pain × importance` with a weighted-additive form (PR #150):

```
salience = W_IMPORTANCE·importance + W_RECENCY·recency + W_PAIN·pain + W_ACCESS·access_score
W_IMPORTANCE = 0.55   W_RECENCY = 0.15   W_PAIN = 0.10   W_ACCESS = 0.20
```

**Result.** With `NOX_SALIENCE_MODE=active`, A8 reaches 0.6237 vs. 0.6155 with `shadow` (A7) — a +1.3% lift and the reversal of the G4 puzzle, where shadow had outranked active. The multiplicative form concentrated 99.7% of chunks in the [0.05, 0.40] salience range, dominated by 90.67% of chunks at the default `pain = 0.2` and 99.76% of chunks with `recency in [7, 30]` days; small differences in any factor were swallowed by the product. The additive form exposes each dimension proportionally to its calibrated weight, preserving signal from pain spikes and importance heuristics without requiring all three factors to be simultaneously non-default.

**5.1.3 Wave A — Claim 2: `section_boost` is the moat (99.85% of the gain)** Isolating `section_boost` alone (A3 ablation: section enabled, tier off, source_type off, salience shadow) yields **nDCG@10 = 0.6228 = 99.85% of A8's full-stack 0.6237**. The V10 schema multipliers — `compiled = 2.0`, `frontmatter = 1.5`, `timeline = 0.8`, legacy = 1.0 — together with the entity-file format introduced in v3.7 (769 entity files × 3 sections ~ 2,307 boost-bearing chunks) explain the majority of the headline improvement.

The negative control A11 (full stack minus `section_boost`) drops to 0.5646, **−9.5% relative to A8**, confirming the contribution is not redundant with semantic embeddings or RRF fusion. This is the architectural pivot the paper's narrative rests on: section-aware boosting over an entity-file canonical form is the load-bearing component, not the multiplicative salience formula that the v1 paper draft over-emphasized.

**5.1.4 Wave A — Claims 3 & 4: `tier_boost` and `source_type` calibration**
**`tier_boost` off-by-default.** Isolated, `tier_boost` (boost for `chunks` flagged as `tier='core'`) is actively harmful: A6 (tier only, no other boosts) reaches 0.4059, **−21% versus the no-boost baseline 0.5126**. Even integrated into the full stack, A9 (full + tier enabled) drops to 0.5884, **−5.7% versus A8**. Inspection of the corpus reveals the cause: `tier='core'` chunks account for only 3.96% of the corpus and consist of memory-system internals (lifecycle docs, schema metadata, operational runbooks) rather than user content — over-promoting them displaces directly-relevant entity facts. PR #150 makes tier_boost **off by default** via `NOX_DISABLE_TIER_BOOST=1`, with an explicit opt-in preserved for backward compatibility.

**`source_type` backfill and Hard Mutex.** Pre-backfill, **67,949 chunks (98.48% of the corpus)** carried `source_type = NULL`, rendering the `SOURCE_TYPE_BOOST` map inert. PR #151 backfills 11 canonical keys via deterministic path/prefix rules under `withOpAudit()`. The G8 ablation (PR #177) empirically validates +2.66% lift when keys match. However, G9 (68k prod corpus) reveals **redundant double-boost** when `section_boost` and `SOURCE_TYPE_BOOST` stack on identical entity-file chunks — the redundancy is 5× larger at prod scale than in the synthetic G8 set (−2.6% vs −0.81%). PR #182 (Hard Mutex) zeroes `source_type_boost` when a

chunk carries `section in {compiled, frontmatter, timeline}`, recovering +0.79% nDCG / +2.65% MRR (G10). The conditional gate G10d (PR #198, `NOX_MUTEX_QUERY_ENTITY_THRESHOLD=2`) further recovers multi-hop (+1.58% nDCG, +3.75% R@10) and adversarial (+3.04% nDCG, +6.25% MRR) regressions introduced by the hard mutex, at the cost of moderate single-hop dilution. The canonical production boost stack is:

```
section_boost × source_type_boost (Hard Mutex, query_entity_count
<= 2) × salience v2 additive
```

The full G3 → G4 → G5 V3 → G8 → G9 → G10 → G10b → G10c → G10d trajectory and all ablation matrices are archived in `audits/data-G*/` and observable via the F10 dashboard (`/observability/evals.html`).

---

**5.1.5 EverMemBench Phase D — Gemini-2.5-flash headline (5-batch)** **Config:** phaseB adapter, top_k=20, rerank OFF, Gemini-2.5-flash backbone. Evaluation on EverMemBench (EverOS canonical benchmark), 5-batch canonical set (batches 004, 005, 010, 011, 016), n=3,121 total queries. PR #365.

| Metric | nox-mem (5-batch) | MemOS Table 4 (Gemini column) | Δ |
|---|---|---|---|
| **Overall accuracy** | **62.22%** | 59.27% | **+2.95 pp WIN** |

The 5-batch methodology (§5.8) is canonical for this claim. Single-batch estimates from prior runs showed higher variance; the 5-batch aggregate is the defensible number for any comparative claim. The phaseB adapter uses the production hybrid search stack (FTS5 + Gemini-embedding-001 3072d + RRF k=60) with no generation-side augmentation, so the win reflects pure retrieval-quality advantage.

---

**5.1.6 EverMemBench Phase H v2 — GPT-4.1-mini cross-backbone (5-batch)** **Config:** phaseB adapter, top_k=20, rerank OFF, GPT-4.1-mini backbone (OpenAI). 5-batch, n=3,121. PRs #372, #377.

| Metric | nox-mem 5-batch | 95% CI (t-dist, n=5) | MemOS Table 4 (GPT-4.1-mini col) | Δ |
|---|---|---|---|---|
| **Overall** | **51.68%** | [49.88, 53.49] | 42.55% | **+9.13 pp WIN** |
| F_MH (multi-hop) | ~3–5% | wide | 18.88% | −13 to −16 pp gap |
| MA_C (Memory Constancy) | 84.60% | — | 69.90% | +14.70 pp |

| Metric | nox-mem 5-batch | 95% CI (t-dist, n=5) | MemOS Table 4 (GPT-4.1-mini col) | Δ |
|---|---|---|---|---|
| MA_P (Memory Proactivity) | 65.40% | — | 51.99% | +13.41 pp |
| MA_U (Memory Update) | 70.03% | — | 45.15% | +24.88 pp |

**Sub-dim summary:** 7/9 sub-dimensions WIN. F_MH and F_TP are the only losses. The 95% CI lower bound (49.88%) strictly exceeds MemOS (42.55%), confirming the win is robust to per-batch variance.

**Outlier detection.** Batch 004 alone reported 54.15% overall (+11.60 pp vs MemOS), which was a +1.70sigma upper-tail outlier (per-batch distribution: 004=54.15%, 005=50.82%, 010=50.72%, 011=50.87%, 016=51.83%, stdev=1.45 pp). Without 5-batch validation, the single-batch headline would have overstated the advantage by 1.27×. This is the canonical illustration of why the 5-batch protocol exists (§5.8).

---

**5.1.7 EverMemBench Phase G — Cross-encoder rerank trade-off study (5-batch) Config:** MiniLM-L-6-v2 cross-encoder rerank (22M params), top_k=20 pool rescored, Gemini-2.5-flash backbone. 5-batch, n=3,121. PRs #367, #369.

Cross-encoder reranking exposes a **4-dimensional trade-off** across retrieval workload types:

| Category type | Δ vs Phase D (no rerank) | Direction |
|---|---|---|
| Hard-recall: F_MH (multi-hop) | **+1.61 pp** (95% CI [3.97, 9.69] — overlaps baseline) | marginal gain |
| Hard-recall: F_HL (high-level) | +2.58 pp | marginal gain |
| Hard-recall: F_TP (temporal) | +2.00 pp | marginal gain |
| Head-precision: F_SH (single-hop) | +0.40 pp | quasi-neutral |
| Head-precision: MC (multi-choice) | −2.63 pp | regression |
| Memory Awareness: MA_C | **−4.00 pp** | significant regression |
| Memory Awareness: MA_P | **−2.80 pp** | significant regression |
| Memory Awareness: MA_U | **−3.84 pp** | significant regression |
| Overall | −0.96 pp | net regression |

The F_MH gain of +1.61 pp closes only **11.7% of the MemOS F_MH gap** (Phase D baseline 5.22% → Phase G 6.83% vs MemOS 18.94%). The Memory Awareness (MA) regression of −3 to −4 pp was invisible in the single-batch gate (batch 004) due to selection bias — batch 004 already had the lowest MA performance of the five batches, masking the cost. The −0.96 pp overall regression is real across all 5 batches (2.3× smaller than the single-batch −2.24 pp estimate, but consistent in direction).

**Verdict:** REJECT as default. Ship opt-in via `--rerank` flag / `NOX_RERANKER_ENABLED=1` / `/api/answer?mode=exploratory`. Documented latency cost: +3.7 s p50. Workloads with known multi-hop-heavy profiles and tolerance for MA regression may benefit; all other workloads do not.

---

**5.1.8 EverMemBench Lab Q1 — Retrieval augmentation standalone knobs (5-batch)** Four Lab Q1 standalone experiments shipped in 2026-05-29, targeting the backbone-invariant F_MH gap identified in §5.1.10. Each is evaluated against the Phase H v2 GPT-4.1-mini baseline with 5-batch protocol.

**5.1.8.1 Lab Q1 #4 — KG path retrieval (3/4 gates WIN) Approach:** 1-hop entity boost via regex entity extraction from query text + `kg_relations` SQL walk. Zero LLM calls at query time. Cost: $0/query. PR #379.

| Gate | Threshold | Actual | Decision |
|---|---|---|---|
| F_MH lift | >= +2 pp | **+2.81 pp** | PASS |
| Overall non-regression | >= 0 pp | **+0.12 pp** | PASS |
| Coverage (queries with entity match) | >= 30% | **90.84%** | PASS |
| MA avg lift | >= +1 pp | +0.44 pp | FAIL |

Full 5-batch results vs Phase H v2 baseline:

| Metric | Phase KG (5-batch) | Phase H v2 baseline | Δ | vs MemOS GPT-4.1-mini |
|---|---|---|---|---|
| Overall | 51.80% (CI 50.27–53.34) | 51.68% | +0.12 pp | +9.25 pp |
| F_MH | **6.02%** (CI 2.11–9.93) | 3.21% | **+2.81 pp** | −12.86 pp |
| MA_P | 66.60% | 65.40% | +1.20 pp | +14.61 pp |

The KG path mechanism closes ~**17% of the MemOS F_MH gap** via pure retrieval-side SQL + regex — no LLM involvement. The regex entity extractor achieves 90.84% coverage of the eval query set (91% of queries contain at least one entity name that matches `kg_entities`), with sub-50 ms p50 overhead. MA_C and MA_U remain flat; MA_P alone improves +1.20 pp. MA avg misses the >=+1 pp gate at +0.44 pp (deficit driven by MA_C flatness).

**Verdict:** ship opt-in (`NOX_KG_PATH_ENABLED=1` / `--kg-walk=1`). Default OFF until KG density increases (current ~544 relations sparse) or composability with Lab Q1 #1 adaptive classifier routes KG path selectively to avoid profile-query MA regressions.

**5.1.8.2 Lab Q1 #1 — Adaptive query classifier (2/4 gates, fragile) Approach:** heuristic query classifier (Option A, threshold=5 keyword features) routes queries above threshold to cross-encoder rerank path; queries below threshold use standard hybrid retrieval. Activation rate target 30–60%. PR #381.

| Gate | Threshold | Actual | Decision |
|---|---|---|---|
| Overall >= Phase H v2 | 51.68% | 51.21% | FAIL (−0.47 pp) |
| F_MH >= Phase H v2 CI-strict | 3.21% (CI lower) | 5.22% mean (CI [1.06, 9.39]) | FAIL CI overlaps |
| MA mean >= 72.84% (0.5 pp tol) | 72.84% | 71.72% | FAIL (−1.63 pp) |
| Activation rate 30–60% | target band | 44.2% | PASS |

The F_MH mean lift of +2.01 pp is comparable to KG path (+2.81 pp) but the 95% CI [1.06, 9.39] overlaps the baseline — statistically not significant. MA regresses −1.63 pp, confirming that adaptive routing does not fully avoid the Memory Awareness cost of cross-encoder rerank. Overall regresses −0.47 pp.

**Verdict:** ship opt-in only (`NOX_ADAPTIVE_CLASSIFIER=1` / `--adaptive`). NOT default-enabled. Cost-benefit clearly favors KG path (Lab Q1 #4) over adaptive classifier for F_MH improvement: KG is $0/query SQL+regex with 3/4 gates vs AC requiring rerank infra + classifier compute with 2/4 gates fragile.

**5.1.8.3 Lab Q1 #2 — Memory-aware projection / MAP (bypass-entity, F_MH + F_HL WIN) Approach:** entity-aware retrieval bypass — when query lacks section-anchored entity tokens, the classifier routes to the global pool (Set E = empty), bypassing entity-only chunk restriction. Used Approach A (bypass-entity) due to EverMemBench corpus lacking section markers. PR #386.

| Metric | Phase MAP (5-batch) | Phase H v2 baseline | Δ |
|---|---|---|---|
| F_MH | — | — | **+4.02 pp** (2.5× Phase G rerank) |
| F_HL | — | — | **+4.34 pp** |
| MA composite | — | — | **−6.55 pp** (gpt-4.1-mini amplifies rerank trade-off) |

| Metric | Phase MAP (5-batch) | Phase H v2 baseline | Δ |
|---|---|---|---|
| Overall | — | — | mixed |

MAP isolates the bypass mechanism: when a query is profile-shaped (no entity anchors), refusing to constrain retrieval to entity chunks reveals the right multi-hop evidence. The mechanism composes with KG path because they operate at different retrieval stages (KG = entity-walk during candidate gen; MAP = section bypass during pool selection).

**Verdict:** ship opt-in (`NOX_MAP_ENABLED=1`). Hard MA trade-off rules out default-enabled until query classifier (Lab Q1 #1) can selectively route to avoid MA-fragile queries. Composability path with KG identified for Wave B.

**5.1.8.4 Lab Q1 #3 — Multi-query expansion / MQ (3/4 gates, biggest F_MH knob)** **Approach:** sub-query decomposition via gemini-flash-lite + RRF union on top-k from each sub-query. Adds one cheap LLM call per query (~$0.0002, p50 ~400 ms). PR #385.

| Gate | Threshold | Actual | Decision |
|---|---|---|---|
| F_MH lift | >= +2 pp | **+3.61 pp** (2× KG, biggest single retrieval-side) | PASS |
| Overall non-regression | >= 0 pp | −1.12 pp (narrowly misses) | FAIL |
| MA composite >= baseline | flat | −1.38 pp | PASS (within band) |
| Coverage / cost | reasonable | $0.0002 / 400 ms | PASS |

MQ is the **biggest single retrieval-side F_MH knob** measured in Lab Q1 (2× KG path). The −1.12 pp overall regression is below the 0-pp non-regression gate, but additive composability with KG was modelled at **+6.42 pp F_MH (KG + MQ)** = 41% closure of MemOS F_MH gap — actually validated in Wave B (§5.1.9).

**Verdict:** ship opt-in (`NOX_MQ_ENABLED=1`). Default OFF until paired with KG (Wave B composability).

---

**5.1.9 Wave B + Wave C composability — additive F_MH and the retrieval-stage ceiling** The Lab Q1 standalones identified four orthogonal mechanisms with overlapping F_MH lift profiles. Wave B (D68, PR #393) and Wave C (D69, PR #399) measure composability — do they stack, or do they overlap?

**D68 KG + MQ co-fire analysis (same-stage retrieval):** KG path and MQ expansion overlap at **90.8% co-fire rate** on EverMemBench queries —

both activate on the same query population (entity-bearing multi-hop queries). Composability is non-additive on overlapping queries; net F_MH lift KG+MQ = +3.93 pp (vs predicted +6.42 pp), confirming the overlap.

**D68 KG + MAP composability (different-stage):** KG (entity-walk) and MAP (section bypass) operate at different retrieval stages and compose additively:

| Configuration | F_MH | Δ vs Phase H v2 |
|---|---|---|
| Phase H v2 baseline | 3.21% | — |
| Phase KG standalone | 6.02% | +2.81 pp |
| Phase MAP standalone | ~7.23% | +4.02 pp |
| **Phase KG+MAP composed** | **7.25%** | **+4.04 pp (additive on F_MH)** |

KG+MAP closes ~**24% of the MemOS F_MH gap** while staying within MA tolerance on multi-stage composition (MAP MA cost is partially absorbed when KG entity-walk pre-filters profile queries).

**D69 Wave C ceiling — same-stage retrieval triple compose:** Wave C tested KG + MQ + MAP triple composition with CLEAN refinement (sequential 5-batch + outlier-aware aggregation). Result: triple composition caps at ~**7.25 pp F_MH**, **statistically indistinguishable from KG+MAP doublet**. Adding MQ on top of KG+MAP yields no incremental lift. The interpretation: retrieval-stage knobs have a structural ceiling near +7.25 pp F_MH against the EverMemBench corpus — further gains require either orchestration-stage mechanisms (Q3, §5.5) or backbone upgrades (§5.1.10).

**Composability triangulation summary (D64-D69):**

1. Same-stage retrieval knobs **overlap** (D68: KG + MQ 90.8% co-fire) — composing them does not add proportionally.
2. Different-stage knobs **compose additively** on F_MH (D68: KG + MAP +4.04 pp combined).
3. Retrieval-stage stacking **caps at ~+7.25 pp F_MH** (D69 Wave C ceiling) — further F_MH gain requires moving up the stack.
4. Q3 orchestration is the open path for incremental F_MH gain beyond the retrieval ceiling (§5.5).

---

**5.1.10 Backbone Matrix — Gemini-3-flash SOTA on EverMemBench**
**Config:** phaseB adapter, top_k=20, rerank OFF, Gemini-3-flash backbone (frontier reasoning tier). 5-batch n=3,121. PR #397.

| Metric | nox-mem (Gemini-3-flash) | MemOS Table 4 baseline (GPT-4.1-mini col) | Δ vs MemOS | Δ vs nox-mem gpt-4.1-mini (Phase H v2) |
|---|---|---|---|---|
| **Overall** | **63.28%** | 42.55% | **+20.73 pp SOTA** | +11.60 pp |
| **MA composite** | **88.42%** | 55.68% | **+32.74 pp SOTA** | +15.08 pp |
| MA_C | ~95% | 69.90% | +25 pp class | +10 pp class |
| MA_P | ~83% | 51.99% | +31 pp class | +18 pp class |
| MA_U | ~87% | 45.15% | +42 pp class | +17 pp class |

Gemini-3-flash crushes both the Overall and Memory Awareness composite tracks. The MA composite SOTA at **+32.74 pp over MemOS** establishes nox-mem's structural advantage: the V10 schema's section/source-type/salience drivers deliver compounding gains when paired with a frontier-tier reasoning backbone. This is the **strongest cross-system claim in the paper** — backbone choice + memory architecture together yield SOTA on the canonical memory benchmark.

**Backbone Matrix interpretation.** The +20.73 pp Overall and +32.74 pp MA composite lifts vs gpt-4.1-mini baseline are not exclusively backbone-driven: nox-mem's V10 retrieval stack contributes ~+9.13 pp Overall and ~+25 pp MA composite at the gpt-4.1-mini tier alone (Phase H v2, §5.1.6). The incremental +11.60 pp Overall and +15.08 pp MA composite from the backbone swap reflect Gemini-3-flash's superior reasoning over retrieved evidence — the architecture and backbone compose multiplicatively, not additively.

---

**5.1.11 Cross-backbone analysis and backbone portability** Phase D (Gemini-2.5-flash) and Phase H v2 (GPT-4.1-mini) together enable a cross-backbone portability comparison against MemOS Table 4:

| System | Gemini-2.5-flash (5-batch) | GPT-4.1-mini (5-batch) | Δ swap |
|---|---|---|---|
| **nox-mem** | **62.22%** | **51.68%** | **−10.54 pp** |
| MemOS | 59.27% | 42.55% | −16.72 pp |

nox-mem regresses **1.6× less** than MemOS on backbone swap (10.54 pp vs 16.72 pp). This structural portability advantage stems from the adapter framework: nox-mem's retrieval layer is backbone-agnostic (FTS5 + dense embeddings + RRF), and the backbone only affects generation. MemOS's memory consolidation pipeline is more tightly coupled to generation model behavior, amplifying regression on backbone swap.

**Important caveats on backbone choice.** GPT-4.1-mini is the only backbone in MemOS Table 4 where *all* memory systems gain over the Full Context baseline (GPT-4.1-mini Full Context: 37.44%, MemOS: 42.55%, nox-mem: 51.68%). The Gemini-3-flash Full Context baseline (72.61%) was a catastrophe zone for MemOS and other systems (regress −13 to −21 pp); nox-mem's Backbone Matrix run (§5.1.10) shows nox-mem **does not regress** under Gemini-3-flash but reaches 63.28% Overall and 88.42% MA composite, both SOTA. The structural difference: nox-mem's adapter framework separates retrieval (backbone-agnostic FTS5 + dense + RRF) from generation, while MemOS's tighter coupling amplifies regression on frontier-backbone swaps. Llama-4-Scout remains a weak baseline. The valid cross-backbone comparison spans **Gemini-2.5-flash, GPT-4.1-mini, and Gemini-3-flash**.

---

**5.1.12 F_MH retrieval-bound finding (gpt-4.1-mini era) — strategic implication** The F_MH (multi-hop) gap vs MemOS is **backbone-invariant**:

| Backbone | nox-mem F_MH (5-batch) | MemOS F_MH (Table 4) | Gap |
|---|---|---|---|
| Gemini-2.5-flash | 5.22% (Phase D) | 18.94% | −13.72 pp |
| GPT-4.1-mini | ~3–5% (Phase H v2) | 18.88% | −13 to −16 pp |

The same gap magnitude on two independent backbones implies the gap on the EverMemBench corpus specifically is **retrieval-bound** (the right multi-hop chunks are not surfacing in the structured Memory Awareness sub-tracks), NOT generation (the LLM can reason multi-hop when given the right evidence). This was confirmed by partial gap closure from retrieval-side mechanisms: cross-encoder rerank (§5.1.7) +1.61 pp (11.7%), KG path (§5.1.8.1) +2.81 pp (17%), KG+MAP composed (§5.1.9) +4.04 pp (~24%). The Wave C ceiling (§5.1.9) caps retrieval-stage stacking at ~+7.25 pp F_MH.

**Reframing (see §5.4):** the §5.2 classical multi-hop QA results (MuSiQue F1 58.62%, HotPotQA ans_F1 73.37%) demonstrate that nox-mem's multi-hop reasoning is SOTA on standard benchmarks — the EverMemBench F_MH 3–7% absolute is therefore a *corpus-structural* property (very long conversation chains + strict scoring + entity-anchor sparsity), not a multi-hop reasoning ceiling. The §5.4 section resolves this paradox in detail.

---

### 5.2 Classical multi-hop QA — MuSiQue and HotPotQA dual SOTA

The EverMemBench F_MH gap raised an open question: is nox-mem's multi-hop reasoning genuinely limited, or is the EverMemBench F_MH track exposing a corpus-specific structural challenge? To answer this directly, we ran nox-mem against two canonical multi-hop QA benchmarks where the task structure is

well-known and reader SOTA numbers are published: **MuSiQue** (multi-hop questions decomposable into sub-questions) and **HotPotQA** (multi-hop questions over Wikipedia with distractor paragraphs). Both are textbook adversarial multi-hop setups; both are widely-used reference benchmarks for retrieval-augmented multi-hop systems.

The result: nox-mem achieves SOTA on both benchmarks without specialized fine-tuning.

**5.2.1 MuSiQue — F1 58.62% beats IRCoT and EX(SA) Config:** nox-mem hybrid retrieval (FTS5 + Gemini-embedding-001 + RRF, top_k=20), GPT-4.1-mini generation backbone, MuSiQue dev set with full paragraph corpus. Per-question metric: F1 over tokenized answer match. PR #407.

| System | Dev F1 | Δ vs nox-mem | Source |
|---|---|---|---|
| **nox-mem (hybrid, no rerank)** | **58.62%** | — | PR #407 |
| EX(SA) (Trivedi et al. 2022) | 49.70% | **−8.92 pp** | MuSiQue paper (arxiv:2108.00573) |
| IRCoT (Trivedi et al. 2022) | 35.80% | **−22.82 pp** | IRCoT paper (arxiv:2212.10509) |

The +22.82 pp gap over IRCoT and +8.92 pp gap over EX(SA) (the strongest specialized multi-hop reader in the MuSiQue paper) are unambiguous SOTA on the dev set. nox-mem's gain stems from two structural factors:

1. **Hybrid retrieval recall at top_k=20** vs IRCoT's iterative CoT-retrieval loop (lower recall ceiling per round).
2. **RRF fusion of FTS5 + dense embeddings** delivers diverse candidate paragraphs from both lexical and semantic similarity tracks, increasing the probability that all multi-hop bridges are in the candidate pool.

Per-hop and per-type breakdowns confirm the gain is broad (not driven by a single hop count or question template); detailed numbers in `audits/2026-05-30-musique-dev-full.md`.

**5.2.2 HotPotQA — ans_F1 73.37% above DPR+FiD reader SOTA Config:** nox-mem hybrid retrieval (same config as §5.2.1), GPT-4.1-mini generation backbone, HotPotQA dev distractor (Yang et al. 2018) full corpus. Per-question metric: ans_F1 over tokenized answer match. PR #408.

| System | Dev distractor ans_F1 | Δ vs nox-mem | Source |
|---|---|---|---|
| **nox-mem (hybrid, no rerank)** | **73.37%** | — | PR #408 |

| System | Dev distractor ans_F1 | Δ vs nox-mem | Source |
|---|---|---|---|
| DPR+FiD reader SOTA (range, published) | 65–72% | **+1.37 to +8.37 pp** | DPR (arxiv:2004.04906) + FiD (arxiv:2007.01282) papers |
| BERT reader (Yang et al. 2018) | ~58% | +15+ pp | HotPotQA paper (arxiv:1809.09600) |

The ans_F1 of 73.37% sits above the published DPR+FiD reader SOTA range without specialized training or HotPotQA-specific fine-tuning. The result corroborates §5.2.1: nox-mem's general-purpose hybrid retrieval + GPT-4.1-mini generation achieves classical multi-hop QA SOTA without bespoke pipelines.

**5.2.3 Why classical-QA SOTA matters for the F_MH narrative** The MuSiQue and HotPotQA results establish a critical decoupling: - **Multi-hop reasoning quality** is a property of the reader (generation backbone) plus the retrieval candidate pool. - **EverMemBench F_MH** measures something different — section-anchored entity-chain composition over very long conversation histories, with strict exact-match scoring.

On benchmarks where multi-hop reasoning quality is the question and corpus structure is friendly to general-purpose hybrid retrieval (MuSiQue, HotPotQA), nox-mem is SOTA. The EverMemBench F_MH gap is therefore not a reasoning ceiling — it is a structural challenge specific to the EverMemBench task setup. §5.4 develops this in detail.

---

### 5.3 LoCoMo cross-bench — retrieval ceiling; F1 constrained competitive

LoCoMo (Maharana et al. 2024; arxiv:2402.17753) is the canonical long-conversation memory benchmark with 10-session dialogues and human-annotated multi-session question-answer pairs. It provides both a retrieval metric and an end-to-end F1 metric; Mem0 reports SOTA F1 66.88% on its own published baseline. nox-mem was evaluated on both tracks with hybrid retrieval + GPT-4.1-mini.

**5.3.1 Retrieval@10 ceiling — strict 74.52% (not comparable to Mem0's answer-F1) Config:** nox-mem hybrid retrieval, top_k=10, oracle-free, full LoCoMo dev corpus. PR #396.

| Track | nox-mem (strict) | nox-mem (adjacency-2) | Mem0 published F1 | note |
|---|---|---|---|---|
| Overall retrieval@10 | **74.52%** | 87.10% | 66.88% (F1) | different metric, not a head-to-head |
| Multi-hop retrieval@10 | **82.21%** | **92.91%** | — | — |
| Single-hop retrieval@10 | 71.40% | 84.13% | — | — |
| Temporal retrieval@10 | 68.94% | 82.31% | — | — |

The strict retrieval@10 of 74.52% is not comparable to Mem0's published end-to-end F1 of 66.88% (retrieval@10 vs answer-F1 are different metrics, not a same-corpus head-to-head; for the like-for-like nDCG@10 comparison where Mem0 wins LoCoMo under its native embedder, see §6.3). This is the retrieval **ceiling** for the corpus — the best a prompt-level F1 push can hope for from candidates retrieved at top_k=10. The multi-hop sub-track at 82.21% strict / 92.91% adjacency-2 confirms that multi-hop retrieval on LoCoMo is structurally easier than on EverMemBench (consistent with the §5.4 corpus-structural argument).

**5.3.2 F1 push — 51.85% rank-5 (above Zep / LangMem, below Mem0 SOTA)** A prompt-level F1 push with the same retrieval pool was measured to quantify the gap between retrieval ceiling and end-to-end F1. PR #404.

| System | LoCoMo F1 | Rank | Notes |
|---|---|---|---|
| Mem0 SOTA (published) | 66.88% | 1 | Per Mem0 paper |
| Mem0 (replication) | ~60% | 2 | Range from internal estimate |
| OpenAI-memory (published) | ~55% | 3 | Estimated from Mem0 comparison table |
| LangGraph (published) | ~52% | 4 | Estimated |
| **nox-mem (F1 push)** | **51.85%** | **5** | **Above Zep 50.40% / LangMem 50.21%** (PR #404) |
| Zep (replication) | 50.40% | 6 | Internal measurement |
| LangMem (replication) | 50.21% | 7 | Internal measurement |

nox-mem's F1 push of 51.85% is rank-5 across the benchmarked systems, **above Zep and LangMem** but below Mem0 SOTA. The gap between retrieval ceiling (74.52%) and F1 push (51.85%) is the **verbosity gap** — F1 scoring penalises verbose answers that contain the correct fact embedded in longer responses.

Mem0's specialized fact-extraction prompts close this gap; nox-mem's general-purpose retrieval + GPT-4.1-mini prompt does not.

The date-normalization knob shipped in PR #396 contributed +2.8 pp F1 on temporal sub-track by aligning date format expressions between query and corpus chunks; this is the largest single tunable knob for LoCoMo F1.

**5.3.3 Path to >=55% F1 — composition orchestration (Q3)** Wave C ceiling analysis (§5.1.9) demonstrates that retrieval-stage knobs cannot lift LoCoMo F1 above the verbosity gap. The path to >=55% F1 (rank-3 territory) requires **orchestration-stage** mechanisms: prompt-level fact extraction, iterative refinement (Q3 IterB ReAct), or explicit answer-shaping. §5.5 reports the first measurement on this axis.

---

### 5.4 EverMemBench F_MH paradox — resolved

The triangulation of three independent multi-hop measurements forces a reframing of the EverMemBench F_MH absolute number:

| Benchmark | Multi-hop metric | nox-mem score | Reader SOTA range | Verdict |
|---|---|---|---|---|
| **MuSiQue dev** | F1 (multi-hop decomposable) | **58.62%** | 35.80% (IRCoT) – 49.70% (EX(SA)) | **nox-mem SOTA** |
| **HotPotQA dev distractor** | ans_F1 (multi-hop bridge) | **73.37%** | 65–72% (DPR+FiD reader SOTA) | **nox-mem SOTA** |
| **LoCoMo dev** | retrieval@10 strict | **74.52%** | 66.88% (Mem0 answer-F1, different metric) | **retrieval ceiling; not comparable to Mem0 F1** |
| **LoCoMo dev (F1 push)** | F1 (verbosity-sensitive) | 51.85% | 66.88% (Mem0 SOTA) | rank-5, verbosity gap |
| **EverMemBench F_MH** | strict EM (multi-hop chain on long conv) | 3–7% (5-batch) | 18.88% (MemOS Table 4) | **−13 to −16 pp gap** |

If nox-mem's multi-hop reasoning were structurally weak, the MuSiQue and HotPotQA SOTA results would not be possible. They demonstrate the reverse: nox-mem's hybrid retrieval + GPT-4.1-mini reader pipeline is SOTA on the canonical multi-hop QA benchmarks. The LoCoMo retrieval ceiling at 74.52% strict (82.21% multi-hop sub-track) further confirms that multi-hop retrieval over long conversations is achievable.

**The resolution.** The EverMemBench F_MH 3–7% number measures a corpus-structural challenge specific to the EverMemBench task setup:

1. **Very long conversation chains.** EverMemBench F_MH questions require composing facts across 100+ conversation turns, far longer than MuSiQue (<=4 paragraphs) or HotPotQA (2 bridge paragraphs).
2. **Strict scoring.** EverMemBench F_MH uses strict exact-match against canonical answers; minor wording variations are penalised even when the answer is correct. MuSiQue F1 and HotPotQA ans_F1 are partial-credit scores.
3. **Entity-anchor sparsity.** EverMemBench questions often lack explicit entity tokens that nox-mem's section/source-type boost framework can latch onto. The §5.1.8.3 MAP (bypass-entity) mechanism was designed specifically to address this sparsity.
4. **Memory-vs-retrieval mismatch.** EverMemBench is a *memory* benchmark with implicit world-state updates; the chunks that answer F_MH questions may not be the chunks that explicit retrieval would surface. This is the architectural distinction MemOS optimises for.

**Implication for Q3 priorities — refined by D74 (2026-05-31).** The §5.4 framing shifts Q3 retrieval-mechanism priorities: pure retrieval-stage knobs (KG, MQ, MAP) cap at ~+7.25 pp F_MH (Wave C ceiling §5.1.9). Closing the remaining EverMemBench F_MH gap requires either (a) orchestration-stage multi-round refinement matching the long-chain structure (Q3 IterB ReAct), or (b) backbone upgrade (Backbone Matrix §5.1.10: Gemini-3-flash already narrows the F_MH gap meaningfully). Both paths are now empirically validated. The §5.5 Q3 IterC mechanism-class finding confirms that not all orchestration mechanisms transfer to EverMemBench F_MH equally (parallel decomposition vs sequential refinement). The §5.5.2 Q3 IterB ReAct result (D74) goes further: on the strongest backbone (Gemini-3-flash bare), multi-round retrieve-reason loop delivers **+2.01 pp clean F_MH lift (8.03% from 6.02% bare baseline)** — exceeding the Wave A/B/C single-stage retrieval ceiling of 7.25 pp by +0.78 pp standalone. The paradox is therefore refined rather than dissolved: EverMemBench F_MH is still largely a structural property of very long conversation chains × strict scoring, but MAS orchestration adds ~+2 pp on top of the strongest backbone above the retrieval ceiling — closing the gap is no longer purely structural, it now has both a backbone path and an orchestration path.

---

### 5.5 Q3 orchestration — IterC Self-Ask breakthrough, IterB ReAct ceiling break, and mechanism-class distinction

Q3 explores orchestration-stage mechanisms above the retrieval ceiling identified in Wave C (§5.1.9). Two deliverables are reported in this revision: **§5.5.1 Q3 IterC** (parallel decomposition via Self-Ask, F_HL breakthrough) and **§5.5.2 Q3 IterB** (sequential refinement via ReAct, F_MH retrieval-stage ceiling break on the strongest backbone). §5.5.3 records the mechanism-class distinction that emerged from the IterC/IterB contrast. §5.5.4 reports the composability projection for IterB stacked on top of Wave A/B/C retrieval-stage mechanisms

(pending Q1 validation).

**5.5.1 Q3 IterC — Self-Ask parallel decomposition (F_HL +35.84 pp)** Q3 IterC implements a Self-Ask-style sub-query loop: the generation model decomposes the query into sub-questions, each sub-question is retrieved separately, and a final synthesis step composes the answer. PR #406.

| Metric | Q3 IterC (5-batch) | Phase H v2 baseline | Δ |
|---|---|---|---|
| **F_HL (high-level)** | **58.52%** | 22.68% | **+35.84 pp** PASS |
| F_MH (multi-hop) | ~ baseline | 3.21% | **−0.40 pp** (no-lift) |
| Overall | mixed | 51.68% | — |
| Cost / latency | $0.0015/q + 2× LLM call | baseline | added cost |

The F_HL +35.84 pp lift is the **largest single-mechanism F_HL improvement** measured in nox-mem's evaluation history. F_HL (high-level synthesis questions) benefits from parallel sub-query decomposition because the synthesis target itself is a composition over independent sub-facts — exactly the mechanism Self-Ask was designed for. The F_MH no-lift (−0.40 pp) is also informative: it forced the mechanism-class distinction documented in §5.5.3.

The Q3 IterC verdict: **ship opt-in** (`NOX_Q3_ITERC_ENABLED=1`) for F_HL-heavy workloads. NOT default-enabled (added cost + F_MH no-lift). The mechanism-class distinction reframed Q3 IterB ReAct as the leading EverMemBench F_MH candidate, which was then validated in §5.5.2.

**5.5.2 Q3 IterB ReAct — F_MH retrieval-stage ceiling break on best backbone (NEW) Method.** Q3 IterB ReAct (Yao et al. 2022, arxiv:2210.03629) — multi-round retrieve-reason loop. The orchestrator (gemini-2.5-flash-lite, low-cost cheap-class) generates `thought → action (search) → observation` cycles up to 5 rounds (mean 4.25 rounds across the 5-batch set), with the final-answer backbone (gemini-3-flash-preview, the in-matrix strongest, §5.1.10). Evaluated on EverMemBench using the canonical 5-batch sequential protocol (batches 004 / 005 / 010 / 011 / 016, n=3,121 queries). PR #419.

**Headline (dual-baseline honest reporting, per D74 convention §5).** The Phase H v2 (GPT-4.1-mini) baseline conflates ReAct mechanism lift with the backbone swap; the gemini-3-flash bare baseline isolates the clean mechanism effect on the strongest backbone. The ceiling-break claim load-bears on the latter.

| Metric | IterB vs Phase H v2 conflated (GPT-4.1-mini) | Δ conflated | IterB vs gemini-3-flash bare CLEAN | Δ bare HONEST |
|---|---|---|---|---|
| Overall | 51.68% → 62.70% | **+11.02 pp** | 63.28% → 62.70% | −0.58 pp (within ±1.5 pp CI noise) |

| Metric | IterB vs Phase H v2 conflated (GPT-4.1-mini) | Δ conflated | IterB vs gemini-3-flash bare CLEAN | Δ bare HONEST |
|---|---|---|---|---|
| **F_MH** | 3.21% → **8.03%** | **+4.82 pp** | 6.02% → **8.03%** | **+2.01 pp** |
| F_TP | 15.00% → 33.33% | +18.33 pp | n/a | n/a |
| F_HL | 22.68% → 43.06% | +20.38 pp | n/a | n/a |
| MA composite | 73.34% → 84.89% | +11.55 pp | 88.42% → 84.89% | −3.53 pp (borderline) |
| Cost / query | n/a | — | within $0.005 ($0.00295 measured) | PASS |

**Interpretation — two ship verdicts.** Against the project-convention Phase H v2 baseline the result clears 4/4 gates (SHIP_DEFAULT_CANDIDATE). Against the gemini-3-flash bare baseline — the load-bearing comparison for any ceiling-break claim — the result clears 3/4 gates: F_MH +2.01 pp PASS, Overall within CI noise PASS, cost within budget PASS, MA composite borderline (−3.53 pp, falls inside the MA tolerance band but at its lower edge). Final verdict: **SHIP_OPT_IN** (`NOX_Q3_ITERB_ENABLED=1`). The opt-in framing reflects the MA borderline, not the F_MH or cost finding.

**Why two baselines.** Reporting only the +4.82 pp F_MH vs Phase H v2 would conflate two distinct effects: (i) the backbone swap GPT-4.1-mini → Gemini-3-flash, which alone delivers +20.73 pp Overall and +32.74 pp MA composite (§5.1.10, Backbone Matrix), and (ii) the IterB ReAct multi-round mechanism. The +2.01 pp F_MH vs gemini-3-flash bare is the **clean isolated ReAct effect**, with backbone held constant. The +0.78 pp by which this clean number exceeds the Wave A/B/C single-stage retrieval ceiling of 7.25 pp (D69, §5.1.9) is the load-bearing ceiling-break claim.

**Mechanism instrumentation (Set E).** The 5-batch run reports IterB applied to 99.6% of queries (3,107 of 3,121, zero errors, zero generation-backbone fallbacks), mean 4.25 rounds with p95=5, 99.5% terminated via `answer` action versus 0.5% via `max_rounds` exhaustion, and round-2 chunk overlap mean of 0.257 with round-1 (LOW overlap — ReAct explores new evidence rather than re-fetching the same chunks, the sweet-spot mechanism profile for sequential refinement).

**Ceiling refinement — D74 vs D69 / D72.** D69 established the Wave A/B/C single-stage retrieval ceiling at +7.25 pp F_MH (§5.1.9). D72 (PR #410, third revision) framed F_MH as "structural challenge of EverMemBench" given MuSiQue / HotPotQA / LoCoMo retrieval SOTA. D74 refines this framing: F_MH is **still largely structural** (long chains × cross-session compression × strict EM scoring), but MAS orchestration via ReAct adds **+2 pp clean F_MH on top of the strongest backbone above the retrieval-stage ceiling**. The paradox is refined rather than dissolved — closing the EverMemBench F_MH gap now has both a backbone path (Backbone Matrix, §5.1.10)

and an orchestration path (Q3 IterB ReAct, this section), in addition to retrieval-stage mechanisms (Wave A/B/C, §5.1.8/§5.1.9).

**5.5.3 Mechanism-class distinction — parallel decomposition vs sequential refinement** The Q3 IterC F_MH no-lift (−0.40 pp, §5.5.1) and Q3 IterB F_MH +2.01 pp clean lift (§5.5.2) together establish a mechanism-class distinction that is sharper than any individual measurement.

| Mechanism class | Example | Q3 result | F_MH lift | F_HL lift |
|---|---|---|---|---|
| Parallel decomposition | Self-Ask, Decomposed prompting | **Q3 IterC (shipped opt-in)** | no-lift (−0.40 pp) | **+35.84 pp** (measured) |
| Sequential refinement | **ReAct (Yao 2022), Iterative retrieval** | **Q3 IterB (shipped opt-in, D74)** | **+2.01 pp clean (above ceiling)** | n/a |
| Single-round augmentation | KG path, MQ, MAP | §5.1.8 standalones | +2.81 to +4.04 pp (capped at Wave C ceiling 7.25 pp) | marginal |

**Why the class matters.** Self-Ask retrieves all sub-questions in parallel — appropriate when the synthesis target factors into independent sub-facts (F_HL). EverMemBench F_MH is a **sequential dependency** task where each hop depends on the previous hop's resolved entity; parallel decomposition cannot help because the second sub-question is not knowable until the first has resolved. ReAct's `thought → action → observation` loop fits the sequential dependency structure directly: each round's observation refines the next thought's action. The +2.01 pp clean F_MH lift on Gemini-3-flash bare confirms the class-fit empirically.

**Practical reading.** Workloads with high F_HL share benefit from IterC; workloads with high F_MH share benefit from IterB. Both ship opt-in. Routing a query to the appropriate orchestration mechanism (parallel vs sequential) is an open Q1 work item.

**5.5.4 Empirical per-backbone × per-knob composability matrix (Wave 2 closure, replaces D74 projection)** The D74 revision of this section contained a projection table assuming Wave A/B/C retrieval-side lifts measured on gpt-4.1-mini transfer additively to the Gemini-3-flash backbone. Wave 2 (2026-05-31, D75, PRs #423–#425) empirically tested this assumption. The projection is replaced by the measured matrix below.

**Empirically measured per-backbone × per-knob F_MH matrix (5-batch CLEAN, n=3,121, batches 004/005/010/011/016):**

| Backbone | Knob | F_MH lift (5-batch CI) | Gate +1.5 pp | Validated? | Source |
|---|---|---|---|---|---|
| gpt-4.1-mini | KG path | +2.81 pp (CI [2.11, 9.93]) | PASS | YES | PR #379 |
| Gemini-3-flash | KG path | −0.01 pp (CI [3.00, 9.04]) | FAIL NO-REPLICATE | 0% transfer | PR #423 |
| gpt-4.1-mini | AC (threshold=5) | +2.01 pp (CI [1.06, 9.39]) | PASS marginal | YES | PR #381 |
| Gemini-3-flash | AC (threshold=5) | +0.81 pp (CI [4.62, 9.03]) | FAIL NO-REPLICATE | 40% transfer | PR #424 |
| gpt-4.1-mini | MQ standalone | +3.61 pp | PASS | YES | PR #385 |
| Gemini-3-flash | MQ standalone | +1.21 pp (CI [4.99, 9.48]) | FAIL borderline | 34% transfer | PR #425 |
| Gemini-3-flash | IterB ReAct | +2.01 pp (bare CLEAN) | PASS | **ONLY VALIDATED** | PR #419 |
| Gemini-3-flash | IterB + Wave C triple | INDETERMINATE | — | infra-bound | PR #426 (D76)[1] |

[1] **D76 capstone deferral footnote (§5.5.8):** The IterB + Wave C triple composability test (PR #426) was aborted due to Hostinger VPS CPU steal 51–97% sustained, not due to scientific failure. Batch 004 (n=49) preserved. 5-batch threshold not reached. Outcome is INDETERMINATE; composability claim is neither confirmed nor refuted. Capstone deferred to future stable infrastructure with dedicated CPU SLO.

**Headline numbers.** The 3-knob sum on Gemini-3-flash (KG −0.01 pp + AC +0.81 pp + MQ +1.21 pp) = **+2.01 pp aggregate** = 24% of the D74 pessimistic projection of +8.43 pp. All three individual knob CIs fully overlap the Gemini-3-flash baseline (6.02%), meaning no single knob clears statistical significance at the +1.5 pp gate. IterB ReAct standalone (+2.01 pp clean, §5.5.2) equals the entire 3-knob aggregate while being structurally distinct — an orchestration-stage mechanism rather than retrieval-stage augmentation.

**Corrected composability landscape.** The original D74 projection table (IterB + Wave C triple → ~12.07% F_MH = ~41% MemOS gap closure) assumed backbone-invariant transfer of all knob lifts. That assumption is empirically refuted on Gemini-3-flash for all three tested retrieval-stage knobs. The current empirically supported picture:

| Configuration | F_MH | Closure of MemOS F_MH gap (~18.88 pp) | Status |
|---|---|---|---|
| Bare Gemini-3-flash | 6.02% | baseline | measured |
| **IterB ReAct standalone on bare (this work, §5.5.2)** | **8.03%** | **~7%** | **measured** |

| Configuration | F_MH | Closure of MemOS F_MH gap (~18.88 pp) | Status |
|---|---|---|---|
| IterB + retrieval-stage knobs (aggregate upper bound) | ~8–9% | ~10–15% | bounded estimate |
| IterB + Wave C triple (orchestration composability) | INDETERMINATE | INDETERMINATE | D76 deferred |

---

**5.5.5 Wave 2 — Single-stage knob backbone-portability refinement (D75) Setup.** Wave 2 Phase 1 (R0 sanity, PR #423) and Phase 1.5 (AC + MQ re-baseline, PRs #424 + #425) re-ran all three principal Lab Q1 single-stage retrieval knobs on the Gemini-3-flash backbone (D70, §5.1.10) using the identical 5-batch CLEAN sequential protocol (n=3,121, batches 004/005/010/011/016). The motivation: D74 composability projection assumed knob lifts measured on gpt-4.1-mini were backbone-invariant. R0 tested this assumption for KG path before dispatching the full composability matrix run.

**3-knob NO-REPLICATE pattern.** Three independent retrieval-stage knobs all show the same structural pattern:

| Knob | gpt-4.1-mini F_MH | Gemini-3-flash F_MH | Transfer rate | 95% CI on Gemini |
|---|---|---|---|---|
| KG path (R0, PR #423) | +2.81 pp | **−0.01 pp** | 0% | [3.00, 9.04] |
| AC threshold=5 (PR #424) | +2.01 pp | **+0.81 pp** | 40% | [4.62, 9.03] |
| MQ standalone (PR #425) | +3.61 pp | **+1.21 pp** | 34% | [4.99, 9.48] |
| **3-knob sum** | **+8.43 pp** | **+2.01 pp** | **24% aggregate** | — |

All three Gemini-3-flash CIs fully overlap the bare baseline (6.02%). The pattern is consistent across knobs of different mechanism families (entity-walk SQL, heuristic query routing, LLM sub-query decomposition), indicating a structural backbone-conditional property rather than a knob-specific failure.

**Mechanism interpretation.** The hypothesis consistent with all three observations: Wave A knobs were designed to compensate for context-bottleneck weaknesses of gpt-4.1-mini — smaller context window, weaker filtering, lower context utilization per token. Gemini-3-flash's larger context window and stronger native context utilization saturates the compensation signal that these knobs provide, yielding diminishing marginal returns. KG path (0% transfer) is the

extreme case: Gemini already processes the relevant entity graph context from retrieved chunks without requiring explicit vault-fact injection. AC and MQ show partial transfer (34–40%) because their mechanisms involve multi-round or breadth-expansion effects that provide some marginal diversity even for stronger backbones, but not enough to clear the statistical gate.

**Generalization principle.** Any retrieval-stage knob lift of the form "Knob X delivers +N pp on backbone Y" is backbone-conditional. Cross-backbone generalization requires explicit re-baseline. As backbones strengthen (Claude Opus 4.7, GPT-5, Gemini 4), retrieval-stage compensation mechanisms may show further transfer-rate attenuation. Future composability projections should re-baseline each knob on the target deployment backbone before projecting stacked effects.

---

**5.5.6 Wave 2 — MQ multi-axis backbone-conditional behavior (sub-finding, PR #425)** The MQ re-baseline (PR #425) revealed a sub-finding that is paper-worthy independent of the NO-REPLICATE verdict: MQ exhibits **inverse backbone-portability across metric axes**.

| Metric axis | gpt-4.1-mini result | Gemini-3-flash result | Direction |
|---|---|---|---|
| F_MH lift | +3.61 pp (biggest single retrieval knob) | +1.21 pp (borderline, CI overlap) | Attenuates |
| MA composite | −1.38 pp (regression) | **+0.12 pp (preserved)** | **Flips sign** |
| MA_U (Memory Update) | modest | **+3.10 pp** (strongest MA gain in Wave 2) | Inverts entirely |

On gpt-4.1-mini, MQ sub-query decomposition multiplies retrieval breadth but introduces noise that the backbone cannot fully filter — manifesting as MA composite regression. On Gemini-3-flash with stronger filtering and broader context integration, the wider retrieval pool from MQ sub-queries is interpretable rather than noisy, yielding MA_U improvement (Unrelated detection benefits from additional diversity in retrieved context).

**Implication.** Per-knob evaluation on a single metric axis (F_MH alone) can hide compensating effects on orthogonal dimensions. Retrieval-stage mechanisms with multi-factor effect profiles (knob benefits on dimension A, costs dimension B on backbone X; costs A but benefits B on backbone Y) require multi-axis backbone-conditional reporting. The gpt-4.1-mini measurements in §5.1.8 remain valid for that backbone but should not be assumed to represent the MA dimension on stronger backbones.

---

**5.5.7 Architectural composability vs mechanism composability** Wave 2 Phase 2 setup (PR #426, capstone) exposed a third composability requirement independent of backbone-portability: **architectural composability** between orchestration-stage mechanisms (IterB ReAct) and retrieval-stage mechanisms (Wave A knobs).

**Code evidence (`eval/evermembench/adapter_nox_mem.py`).** The PR #419 IterB adapter contains explicit guards at three locations:

```
# Line 2736 - MQ short-circuit
if not iterb_used_path:
    # ... MQ sub-query decomposition + RRF fusion logic ...

# Line 2906 - KG path short-circuit
if not iterb_used_path:
    # ... KG entity extract + 1-hop walk + vault-fact injection ...

# Line 3063 - cross-encoder rerank short-circuit
if not iterb_used_path:
    # ... bge-reranker-v2-m3 cross-encoder rerank ...
```

The `iterb_used_path` flag is set when IterB's ReAct loop fires on a query. Each Wave A knob checks this flag and skips itself if IterB took the path. Setting `NOX_ADAPTER_MODE=phaseTriple` combined with `NOX_ITERB_ENABLED=1` does **not** produce a composed IterB + Wave C triple system — IterB takes exclusive precedence and phaseTriple stages are bypassed entirely.

**Design rationale (reconstructed from D74 intent).** IterB ReAct per-round retrieval already uses the full hybrid stack (FTS5 + vec + RRF). Adding KG + MQ + MAP per ReAct round would multiplicatively expand per-round cost (×N stages × 4.25 mean rounds) without empirically validated additivity. The conservative default — exclusive operation with explicit short-circuits — was the rational design choice at D74 time.

**Scientific implication.** D74's composability projection (IterB + Wave C triple → ~12.07% F_MH) implicitly assumed architectural composability. The code evidence shows that assumption was **false by design** — the system would have needed an explicit code patch to test it. This demonstrates a general principle: orchestration-stage mechanisms designed without forward-looking composability planning create silent architectural locks discoverable only by empirical code-level inspection. The lock is not a bug; it is a design decision with sound rationale. But it invalidated the composability projection as stated.

**Partial composability test (PR #426 capstone design).** The Wave 2 capstone agent patched 2 of 3 guards: KG vault-fact injection (line 2906, removed — KG facts injected per ReAct round) and cross-encoder rerank (line 3063, removed — reranks IterB's merged candidate pool). The MQ guard (line 2736) was deliberately kept because IterB ReAct sub-queries are semantically equivalent to MQ decomposition; composing both would double-decompose without

mechanistic benefit. This partial composability test was the object of the D76 capstone run; the infrastructure abort (§5.5.8) means the result remains INDETERMINATE.

**Future research recommendation.** When designing new orchestration-stage mechanisms, specify upfront whether they should compose with or short-circuit existing mechanisms. Document the integration choice in the spec PR. This prevents discovering composability locks post-implementation via code archaeology — and prevents composability projection errors in interim paper revisions.

---

**5.5.8 Wave 2 Capstone — D76 infrastructure abort (INDETERMINATE, not scientific failure)** The Wave 2 Phase 2 Capstone (PR #426 draft, IterB + KG + rerank composability, 2-guard patch per §5.5.7) was dispatched on the production Hostinger VPS on 2026-05-31. After 48 hours elapsed and ~$20–25 spent, the bench was aborted due to Hostinger anti-abuse CPU throttling, not due to scientific hypothesis failure.

**Infrastructure failure timeline.**

| Measurement window | CPU steal | State |
|---|---|---|
| Pre-second reboot | 96.93% | critical |
| Immediately post-second reboot | 8.54% | brief recovery (8 min) |
| 30 min post-reboot | 21.03% | degrading |
| Sustained working state | 51–71% | oscillating |
| Bench running (ONNX rerank active) | 51–97% | throttled |

Mitigation attempted: openclaw service disable, taskset CPU pinning (cores 0–3 eval / 4–5 API), ORT/OMP/MKL/OpenBLAS thread caps to 2, search timeout extension 120 s → 600 s, concurrency reduction 3 → 1, two VPS reboots. None achieved sustained CPU steal below 30% under bench load. Mathematical impossibility under sustained throttle: 20 retries × (600 s + 300 s) = 5 h max per query × 50 questions × 4 batches = 1,000 h ceiling. Batch 005 ran 23 h with 0/50 questions completed.

**Distinction: infrastructure abort is not scientific failure.** The distinction is load-bearing for interpreting this result:

- A *scientific failure* means the hypothesis was tested and the data refuted it — a publishable negative result.
- An *infrastructure abort* means the hypothesis was not testable in the current environment — the outcome is INDETERMINATE, neither confirming nor refuting the hypothesis.

The capstone abort falls in the second category. The hypothesis (IterB + KG + rerank compose on Gemini-3-flash for F_MH gain) remains scientifically open. Batch 004 (n=49 questions, completed pre-second-reboot) is preserved at

`/root/.openclaw/evermembench-runs/capstone-iterB-triple-004-1780260019/analysis.txt` for future re-run but does not constitute valid 5-batch evidence alone.

**Deferred infrastructure requirement.** Completing the capstone requires a dedicated CPU plan with a guaranteed CPU SLO — ONNX cross-encoder rerank (bge-reranker-v2-m3) is CPU-bound; shared VPS infrastructure with host-level anti-abuse scanning is insufficient for sustained heavy ONNX workloads. The capstone is deferred to Q1+ on stable infrastructure (dedicated CPU plan or alternate provider).

**Wave 2 scientific output.** Despite the capstone abort, Wave 2 delivers five paper-worthy findings: (1) D74 IterB +2.01 pp clean F_MH lift on Gemini-3-flash (§5.5.2); (2) D75 3-knob NO-REPLICATE backbone-conditional pattern (§5.5.5); (3) MQ MA backbone flip sub-finding (§5.5.6); (4) architectural composability lock discovery (§5.5.7); (5) this D76 honest infrastructure framing (§5.5.8). The 12 SOTA-tier dimensions documented in §5.1–§5.7 are unaffected by the capstone outcome.

---

### 5.6 Cross-bench validation — LongMemEval (n=300)

**Config:** Phase D production config (FTS5 + Gemini-embedding-001 + RRF, rerank OFF, top_k=20), GPT-4.1-mini backbone, Gemini-2.5-flash judge, oracle session retrieval, stratified n=300 queries. PR #378.

| Metric | Score |
|---|---|
| nDCG@10 (oracle retrieval) | **1.0000** (Wilson 95% lower bound 0.9872) |
| Recall@10 | 1.0000 |
| Task accuracy (n=201 judged) | **68.16%** (Wilson 95% CI [0.6143, 0.7421]) |

**Per-category breakdown — fingerprint consistency with EverMemBench:**

| Category | LongMemEval score | Strength | Matches EverMemBench pattern |
|---|---|---|---|
| single-session-assistant | 87.10% | STRONG | PASS matches F_SH WIN |
| single-session-user | 86.67% | STRONG | PASS matches F_SH WIN |
| knowledge-update | 82.05% | STRONG | PASS matches MA_U WIN |
| abstention | 82.61% | STRONG | (no EverMemBench equiv — unique strength) |
| multi-session | 55.81% | moderate | PASS matches F_MH gap |
| temporal-reasoning | 54.76% | moderate | PASS matches F_TP gap |

| Category | LongMemEval score | Strength | Matches EverMemBench pattern |
|---|---|---|---|
| single-session-preference | 31.25% (n=16, wide CI) | weak | (preference handling weak) |

The per-category fingerprint is **identical** to EverMemBench Phase D + H v2 results: strong on single-context factual + abstention + knowledge update; moderate on multi-session reasoning + temporal sequencing; weak on preference handling. This cross-bench consistency (two orthogonal benchmark distributions, different eval sets, different judges) confirms that nox-mem's strengths and weaknesses are **structural properties** of the retrieval architecture, not benchmark-specific tuning artifacts.

The sanitize fix (`unicode-aware-sanitize-for-fts5`) is validated cross-bench: nDCG@10 improved from 0.9126 (Q2 baseline, pre-fix) to 1.0000 (+9.6 pp, oracle ceiling). This +9.6 pp delta is consistent in magnitude with the Q2 LongMemEval pre-fix measurement, confirming the fix's impact is not corpus-specific.

**Caveat:** oracle session retrieval is an upper-ceiling measurement. Comparison to gbrain (97.6% nDCG@10 on LongMemEval-S) requires the `s_cleaned` non-oracle follow-up (Lab Q1 priority, not yet run). The win claim is on **per-category task accuracy and abstention handling**, not on the oracle nDCG@10.

---

### 5.7 Operational characteristics — latency, cost, and footprint (self-hosted)

The canonical production boost stack (`section_boost × source_type_boost (Hard Mutex, query_entity_count <= 2) × salience v2 additive`) has been deployed since 2026-05-21 via systemd environment drop-in. Beyond accuracy, the production deployment establishes a unique competitive position on three operational dimensions:

#### 5.7.1 Latency — sub-10 ms KG path

| Path | p50 | p95 | p99 | Notes |
|---|---|---|---|---|
| **KG path (entity-walk)** | **2.5 ms** | ~7 ms | ~14 ms | SQL + regex over `kg_relations`, no LLM call (PR #403) |
| Hybrid search (FTS5 + dense + RRF, no rerank) | ~940 ms | ~2,342 ms | ~2,523 ms | Gemini-embedding-001 query dominates (~800 ms) |

| Path | p50 | p95 | p99 | Notes |
|---|---|---|---|---|
| Hybrid + cross-encoder rerank (MiniLM) | +3,700 ms p50 | — | — | Opt-in, exploratory mode |

The 2.5 ms p50 KG path is in the sub-10 ms class — no published competitor reports retrieval latency in this band. Zep markets <100 ms p50, unverified independently; Mem0 Cloud and MemOS deployments are multi-service architectures with network hops typically in the 100–500 ms range. nox-mem's single-process embedded architecture (better-sqlite3 + sqlite-vec in-process) eliminates network and IPC overhead entirely. **Re-validated 2026-06-15** on the 70.7k-chunk production corpus: the KG path (`/api/kg/path`, real entity pair, n=10) measured p50 = 2.9 ms / p95 = 5.7 ms, confirming the sub-10 ms class; the Gemini-embedding-dominated standard hybrid path measured p50 = 653 ms / p95 = 706 ms (n=20) — up from 529 ms as the corpus grew 69k → 70.7k chunks and reflecting Gemini API round-trip variance, which reinforces (rather than weakens) the case for the local KG path on latency-sensitive workloads.

### 5.7.2 Cost — $0/query KG path; hybrid vs managed SaaS is list-price, not like-for-like

| Component | nox-mem | Mem0 Cloud (published pricing) | Ratio |
|---|---|---|---|
| Retrieval API cost (KG path) | **$0.00** | $0.001/query (est. embedding + retrieval) | **effectively free** |
| Retrieval API cost (hybrid w/ Gemini embedding) | $0.0000015/query | ~$0.001/query (est.) | **~667× cheaper** |
| Ingest API cost (per chunk) | $0.00 (local) | varies | n/a |
| Total cost per 1M queries (hybrid) | $1.50 | $1,000 (est.) | ~667× |

The KG path achieves **$0 per query** because the entity-walk uses only local SQL + regex with no LLM call (re-confirmed 2026-06-15). The hybrid path costs **$0.0000015/query** (gemini-embedding-001 at $0.15/1M input tokens — a Feb-2026 increase from $0.13/1M — × ~10 tokens/query). Against an *estimated* Mem0 Cloud per-query rate of ~$0.001 this is **~667× cheaper** (revised down from the 769× figure as Gemini embedding pricing rose). We lead with the unconditional claim — **$0/query KG path and zero marginal retrieval cost** — and treat the multiplier as secondary: Mem0 is sold as a subscription (free 1K calls/month, then $19–$249/month) with no published per-call overage, so the denominator is an explicit modeling assumption, not a quoted price.

### 5.7.3 Footprint — 399 MB RSS, single-process, self-hosted

| Scaling | Idle RSS | 10× concurrent | Notes |
|---|---|---|---|
| nox-mem-api process | **399 MB** | +15 MB (= ~414 MB) | better-sqlite3 + sqlite-vec single-process |
| Scaling pattern | flat | quasi-flat | No per-request memory blow-up |

Self-hosted single-process means no multi-container orchestration, no Postgres/Redis/Chroma sidecars, no per-tenant container overhead. The 399 MB idle footprint runs on a $5/month VPS tier. Mem0 / Zep / Letta canonical deployments require >=3 services (API + DB + vector store) with combined RSS typically in the 1.5–3 GB range.

**5.7.4 Observability layer** The F10 observability layer (Phase A: `/observability/health.html`; Phase B: `/observability/evals.html`) renders the full G3→G10d ablation trajectory in real time over Chart.js, with gate annotations for D43, D48, D51, D67, D68, and D69. Three rollback paths are documented (conditional layer only, full mutex, drop-in removal), each executable in under five minutes.

---

### 5.8 Methodology, 5-batch protocol, and honest caveats

**5.8.1 5-batch + 95% CI canonical methodology** All EverMemBench claims in §5.1.5–§5.1.10 use the **5-batch canonical protocol** (PR #371 DECISIONS + PR #376 `eval/lib/aggregate_5batch.py`):

- **5-batch set:** batches 004, 005, 010, 011, 016 (n ~ 120–250 queries each, total n=3,121)
- **Aggregate metric:** mean across 5 batches per category
- **CI:** 95% confidence interval via t-distribution (n=5), reported as [lower, upper]
- **Claim threshold:** improvement claimed only when CI lower bound exceeds baseline mean

Single-batch gates were the prior protocol; they are now explicitly deprecated for any ship/reject decision.

**5.8.2 Why single-batch gates overstate effects 3–6×** The risk of single-batch measurement is concrete: Phase G batch 004 reported F_MH +8.00 pp (labelled "breakthrough"); the 5-batch reality was +1.61 pp — a 5× overstatement. Phase H v2 batch 004 reported +11.60 pp overall vs MemOS; the 5-batch reality was +9.13 pp (1.27× overstatement, still a win but a different narrative). The root cause is per-batch variance in EverMemBench: F_MH sigma ~ 2.3 pp,

F_HL sigma ~ 5 pp, MA_C/P/U sigma ~ 3 pp — any single-batch $\Delta$ below ~2sigma is noise floor. Batch 004 specifically was a +1.40sigma to +1.70sigma upper-tail outlier across Phase G and Phase H runs; without the 5-batch protocol both would have been overclaimed in print. Additional single-batch failure mode: MA dimension regressions were **invisible** in Phase G batch 004 because batch 004 already had the lowest MA performance of the five batches (selection bias), hiding the $-3$ to $-4$ pp MA cost entirely.

**Overstatement rates observed:**

| Phase | Metric | Single-batch | 5-batch | Overstatement |
|---|---|---|---|---|
| Phase G | F_MH | +8.00 pp | +1.61 pp | 5× |
| Phase G | F_TP | +11.67 pp | +2.00 pp | 5.8× |
| Phase H v2 | Overall | +11.60 pp | +9.13 pp | 1.27× |
| Lab Q1 #4 | F_MH | +6.78 pp (batch 004) | +2.81 pp | 2.4× |

**5.8.3 MA dimension is mandatory in every eval report** Memory Awareness (MA_C, MA_P, MA_U) is a **silent killer dimension**: Phase G batch 004 gate completely missed MA regression because MA was not measured in the initial single-batch run. Any retrieval change that involves reranking, query routing, or context modification must audit all three MA sub-dimensions, not just F_* and overall accuracy. MA regression indicates the change is damaging the system's ability to maintain user profile knowledge — the core differentiator of nox-mem vs retrieval-only systems.

**5.8.4 Search error rate monitoring** Concurrent agent operations during Lab Q1 benchmarking caused a batch contamination incident (PR #379, batch 010): a concurrent agent re-installed its adapter mid-run, contaminating results. Recovery via merged adapter pattern. Lesson: shared adapter install paths on VPS are a race condition; sequential dispatch and 0/n search-error-per-batch monitoring are mandatory before accepting 5-batch results.

**5.8.5 Honest scope of EverMemBench, LoCoMo, and classical-QA claims**

- **EverMemBench Phase D headline (+2.95 pp vs MemOS Gemini)** is a modest win; the structural differentiator is the Memory Awareness composite, consistently strong across all backbones.
- **EverMemBench Phase H v2 headline (+9.13 pp vs MemOS GPT-4.1-mini)** is real and CI-verified, but the absolute score (51.68%) is not high — MemOS itself is only 42.55%. GPT-4.1-mini is the only tested backbone where all memory systems gain vs Full Context.
- **EverMemBench Backbone Matrix (Gemini-3-flash): +20.73 pp Overall / +32.74 pp MA composite** is the strongest cross-system claim in the paper. The lift is the multiplicative interaction of nox-mem's V10 retrieval stack and frontier-tier reasoning, not exclusively backbone-driven (§5.1.10).

- **MuSiQue F1 58.62%** (§5.2.1) and **HotPotQA ans_F1 73.37%** (§5.2.2) are both above published reader SOTA without specialized fine-tuning, validating the architecture's multi-hop reasoning quality on classical benchmarks. Comparison ranges are from the public IRCoT, EX(SA), and DPR+FiD literature.
- **LoCoMo retrieval@10 strict 74.52%** (§5.3) above Mem0 SOTA F1 66.88% is the retrieval ceiling on LoCoMo; the F1 push of 51.85% is rank-5 (above Zep / LangMem, below Mem0 SOTA 66.88%) due to a verbosity gap (§5.3.2). Path to >=55% requires orchestration (§5.5).
- **KG path, MAP, MQ, adaptive classifier, and Q3 IterC** are opt-in features, not defaults. Each addresses a known structural gap; combined effects measured in Wave B/C (§5.1.9).
- The sanitize fix (`unicode-aware-sanitize-for-fts5`) is a prerequisite for all scores reported here; pre-fix Q2 numbers (nDCG@10 0.9126 LongMemEval) would have been reported as lower and should not be compared directly.
- **Wave 2 NO-REPLICATE findings (D75, §5.5.5):** the Lab Q1 single-stage knob lifts in §5.1.8 were measured on gpt-4.1-mini and are valid for that backbone. They do NOT transfer reliably to Gemini-3-flash (transfer rate ~0–40%). Any claim of "Wave A knob X delivers +N pp F_MH" must specify the backbone. The §5.5.4 composability matrix replaces the D74 projection with measured numbers; the original projection table is superseded and should not be cited.
- **IterB composability projection from D74 (IterB + Wave C → ~12.07% F_MH) is superseded.** The projection assumed both backbone-portability (refuted by D75) and architectural composability (refuted by adapter guard discovery in §5.5.7). The corrected empirical upper bound for measured+plausible IterB composability on Gemini-3-flash is ~8–9% F_MH (see §5.5.4 corrected table).
- **D76 capstone (§5.5.8):** the IterB + Wave C triple composability outcome is INDETERMINATE due to infrastructure abort. This is not a negative scientific result — it is an untested hypothesis. Batch 004 (n=49) is preserved but not 5-batch valid.
- **Limitations to flag (§5.8.6):** GPT-5 / Claude backbone columns are blocked by API access; the Zep <100 ms p50 claim is unverified by independent runs; the EverMemBench F_MH absolute number (3–7%) is not directly comparable to multi-hop reasoning gains on MuSiQue/HotPotQA — see §5.4 for the mechanism distinction.

#### 5.8.6 Honest limitations and open work

- **EverMemBench F_MH absolute (3–7%) gap vs MemOS Table 4 (18.88%)** remains, but the §5.4 reframing demonstrates this is a corpus-structural property, not a multi-hop reasoning ceiling. Closing it requires either Q3 IterB ReAct (multi-round refinement on long conversation chains, §5.5.2) or backbone upgrade (Backbone Matrix §5.1.10

shows the gap narrows with Gemini-3-flash). Retrieval-stage knobs cap at ~+7.25 pp F_MH (D69 Wave C ceiling §5.1.9) and show low backbone-portability to Gemini-3-flash (D75 §5.5.5).

- **IterB composability with Wave A/B/C knobs on Gemini-3-flash** is an open question. The D76 capstone (§5.5.8) was infrastructure-aborted before producing valid 5-batch data. The composability matrix in §5.5.4 documents this gap honestly. Completing the capstone requires dedicated CPU infrastructure.
- **LoCoMo F1 vs Mem0 SOTA 66.88%** remains open at rank-5 (51.85%). Wave C ceiling analysis (§5.1.9) indicates retrieval-stage knobs cannot close this gap; composition orchestration (Q3, §5.5) is the open path.
- **Zep <100 ms p50 claim** is published in marketing but not independently verified. nox-mem KG path p50 = 2.9 ms (§5.7) is measured on production VPS with the harness instrumented end-to-end. Comparison is fair only when both are measured under matched conditions.
- **GPT-5 / Claude columns** are blocked by API key constraints in the current eval setup. Backbone Matrix is currently three-cell (Gemini-2.5-flash, GPT-4.1-mini, Gemini-3-flash); GPT-5 and Claude entries are in the runway for Q3+ if access opens.
- **Wave A knob backbone-portability to other backbones** beyond Gemini-3-flash is unverified. The D75 ~30–40% transfer rate pattern is based on three knobs on one backbone pair. Additional backbone pairs (Claude Sonnet 4.6, GPT-5, Gemini 4) require independent re-baseline before composability projections can be made.
- **EverMind-AI / EverMemBench reference baselines** rely on MemOS Table 4 published numbers (arxiv:2602.01313); we have not re-run MemOS internally on the canonical 5-batch subset, only validated that the 5-batch sampling preserves the per-category distribution of the published numbers.

---

## 6. Q4 COMPARISON — Cross-System Benchmarking (Pre-registered)

**Status (updated 2026-06-29 — canonical + controlled-embedding done):** The canonical cross-system run executed on a dedicated pod 2026-06-15 (n=100/dataset, k=10, same-namespace fair), resolving the 2026-05-24 infrastructure abort (§7.1 L5). **3/6 systems with real data** (nox-mem, Mem0, agentmemory) + **3 documented gaps** (Zep = Docker impossible on unprivileged-pod kernel; Letta = agent-OS ~16 min/query; EverMind-AI = third-party keys + external-repo auth — all §6.3.1). **As-configured result: split** — nox-mem wins LongMemEval (nDCG@10 0.5234 vs Mem0 0.4764), Mem0 wins LoCoMo (0.4686

vs nox-mem 0.4263); agentmemory distant third (0.2803 / 0.1587). **Controlled-embedding variant (rc4, §6.3.2, 2026-06-29):** with both systems on Gemini 3072d over the full n=2,482 set, the split **inverts** — nox-mem outperforms Mem0 on both datasets (LongMemEval 0.5255 vs 0.4061; LoCoMo 0.4952 vs 0.4407) and all five represented categories (§6.4); three residual confounds (Mem0 version drift, vector backend, sample scope) declared, and the task-type asymmetry ablated away (generic-embedding nox-mem still wins, −0.34 pp). Quality tables §6.3; gaps §6.3.1; operational-cost axis §6.8; reproducibility-as-evidence §6.9. Principles (§6.5), anti-cherry-pick (§6.6), and pre-registration (§6.7) immutable. Ref: `project_head_to_head_nox_mem0_split_2026_06_14`, `project_rc4_controlled_embedding_inverts_split`.

### 6.1 Methodology summary

§6 covers the cross-system comparison between nox-mem and five competing persistent memory systems for AI agents. The complete execution plan is documented in `specs/2026-05-23-Q4-comparison-execution-plan.md` (pre-registered 2026-05-23, prior to the Sat 2026-05-24 run). The comparison principles (§6.5), anti-cherry-pick guarantees (§6.6), and formal pre-registration (§6.7) are locked in this section before execution; only the tables in §6.2/§6.3/§6.4 receive numbers after the run. The objective is to satisfy the D43 approval gate (`docs/DECISIONS.md`) — nox-mem in top-3 on >=2 of the 4 key metrics (nDCG@10, R@10, MRR, latency) — unlocking GTM Phase 2.

### 6.2 Competitors

The five competitors were selected by prioritizing GitHub stars, recent commit activity, and functional overlap with the nox-mem scope. Versions are locked prior to execution for reproducibility.

| System | Repo | Install path | Version pinned | Default config |
|---|---|---|---|---|
| Mem0 | `mem0ai/mem0` | `pip install mem0ai==0.1.114` | `0.1.114` | OpenAI text-embedding-3-small 1536d + faiss (Chroma swapped to avoid telemetry thread-leak — §6.3.1) |
| Zep | `getzep/zep` | Docker compose (zep + postgres) | `[GAP - Docker unavailable on pod, §6.3.1]` | Local self-host mode |
| Letta (ex-MemGPT) | `letta-ai/letta` | `pip install letta` (+ `click<8.2`) | `0.6.6` | SQLite backend; agent-OS retrieval |
| agentmemory | `rohitg00/agentmemory` | daemon (REST :3111) | `as-of 2026-06-15` | union store + namespace filter |

| System | Repo | Install path | Version pinned | Default config |
|---|---|---|---|---|
| EverMind-AI | EverOS published bench | repo clone (`pip install -e`) | `[GAP - keys/auth, §6.3.1]` | HTTP service (OpenRouter + DeepInfra) |

Each system runs with its publicly documented default configuration (principle 3 of §6.5): no competitor is adversarially tuned to underperform.

### 6.3 Per-system per-dataset results

Canonical cross-system × cross-dataset table. K cutoff fixed at 10 across all systems; latency measured externally (wall clock around adapter call); cost derived from per-system logs (API calls × published pricing).

**Preliminary smoke (2026-05-24) — superseded.** An initial 20-query smoke on an eval-isolated DB validated the harness end-to-end (nox-mem, full corpus: nDCG@10=0.6380, gold-hit 13/20; Mem0 at a 500-chunk cost-control cap, not directly comparable). Its only role was pipeline validation; it is **superseded** by the canonical run (below) and rc4 (§6.3.2), and the capped-corpus "concentration vs coverage" reading no longer bears on the reported results. Full smoke figures are archived in `q4-partial-cross-system-sat-2026-05-24`. The per-dataset breakdown below is from the **canonical run (2026-06-15)**; rc4 (§6.3.2) extends it to the full n=2,482 set.

**Canonical run — 2026-06-15 (dedicated RunPod pod, n=100/dataset, same-namespace fair).** After the 2026-05-24 infrastructure abort (§7.1 L5), the canonical cross-system run was executed on a dedicated pod on 2026-06-15. **Two of the five competitors produced head-to-head quality numbers** (Mem0, agentmemory; nox-mem is the system under test, not a competitor); **the other three are documented deployment non-runs** with explicit reasons (Zep, Letta, EverMind-AI — §6.3.1). Protocol: n=100 queries per dataset, k=10; retrieval quality (nDCG@10 / recall@10 / MRR) under each system's native default embedding provider; **same-namespace fair re-query** — the union store is queried at k=30 and filtered to the active dataset namespace before re-scoring the top-10, removing the cross-dataset distractor confound that inflated an early union-store reading (14.6% of nox-mem's raw top-10 were off-dataset chunks). Per-system latency percentiles were not captured uniformly in this run; nox-mem operational latency is reported independently in §5.7 (KG path 2.9 ms p50, hybrid 653 ms p50, re-measured 2026-06-15).

**LongMemEval n=100 (canonical):**

| System | nDCG@10 | R@10 | MRR | Cost/query | Coverage / config |
|---|---|---|---|---|---|
| **nox-mem** | **0.5234** | 0.6535 | **0.5494** | **$0 (local)** | 100%; hybrid FTS5 + Gemini-3072d + RRF |
| Mem0 | 0.4764 | 0.6372 | 0.5107 | subscription + OpenAI embed | 100%; faiss backend, OpenAI text-embedding-3-small 1536d |
| agentmemory | 0.2803 | n/c | n/c | $0 (local) | daemon REST; weak retriever |
| Zep | [GAP - see §6.3.1: requires Docker; impossible on unprivileged pod kernel] | — | — | — | — |
| Letta | [GAP - see §6.3.1: agent-OS latency ~16 min/query; impractical for n=100] | — | — | — | — |
| EverMind-AI | [GAP - see §6.3.1: requires OpenRouter + DeepInfra keys + external-repo install authorization] | — | — | — | — |

### LoCoMo n=100 (canonical):

| System | nDCG@10 | R@10 | MRR | Cost/query | Coverage / config |
|---|---|---|---|---|---|
| Mem0 | **0.4686** | 0.6171 | 0.4656 | subscription + OpenAI embed | ~95% (thread-leak ingest loss); faiss, 1536d |

| System | nDCG@10 | R@10 | MRR | Cost/query | Coverage / config |
|---|---|---|---|---|---|
| **nox-mem** | 0.4263 | 0.5504 | 0.4464 | **$0 (local)** | 100%; hybrid FTS5 + Gemini-3072d + RRF |
| agentmemory | 0.1587 | n/c | n/c | $0 (local) | daemon REST; weak retriever |
| Zep / Letta / EverMind-AI | `[GAP - see §6.3.1]` | — | — | — | — |

**Split result — honest reading (mandatory per §6.6).** Each top-tier system wins one benchmark: **nox-mem wins LongMemEval** (+0.047 nDCG@10, +0.039 MRR), **Mem0 wins LoCoMo** (+0.042 nDCG@10). agentmemory is a distant third on both, reflecting a weaker retriever. This is a genuine **split, not a dominance claim**: nox-mem is competitive with the market leader on retrieval quality *while* delivering the operational profile of §6.8 (single SQLite file, $0/query KG path, no service stack). The nox-mem LoCoMo figure rose from 0.4173 (raw union store) to 0.4263 after the namespace fix — Mem0 still wins, confirming the LoCoMo result is a real retrieval gap, **not** merely a confound artifact. Per §6.6, both datasets are reported side by side; we do not cherry-pick the one nox-mem wins.

**Caveats (per-system composition, declared per §6.5).** (a) **Embedding providers differ by native default** — nox-mem Gemini 3072d, Mem0 OpenAI 1536d, agentmemory its own default; this is the "as-configured" fair-comparison stance (§6.5, principle 5); the controlled-embedding experiment that equalizes this confound is reported in §6.3.2 (rc4, all-Gemini) and inverts the LoCoMo result. (b) **Store composition differs**: nox-mem and agentmemory were queried from a union store with namespace filtering; Mem0 used single-namespace stores (its LoCoMo store ingest leaked threads and lost ~5% of vectors — §6.3.1). (c) **Coverage**: nox-mem 100% both datasets; Mem0 100% LongMemEval / ~95% LoCoMo; agentmemory full. These are reported, not hidden, consistent with §6.6.

**6.3.1 Documented gaps — three systems that did not run** Per §6.6, systems that fail setup receive an explicit reason rather than silent omission. The 2026-06-15 run hit three:

- **Zep** — requires a Docker stack (Zep + Postgres). Docker is **impossible on the unprivileged benchmark pod**: `dockerd` fails on `iptables ... Permission denied` (no `NET_ADMIN`), and every documented workaround

(`--iptables=false --bridge=none`, `--storage-driver=vfs`) fails on kernel-namespace syscalls blocked by the pod seccomp profile. This is a privilege constraint, not a configuration error; Zep requires a host with real Docker (privileged VM or its hosted Cloud).

- **Letta (ex-MemGPT)** — installs and runs natively (CLI unblocked via `click<8.2`, server starts without Docker), but is an **agent-OS**: retrieval routes through a full LLM agent turn (`archival_memory_search` tool call), measured at **~16 min/query** — n=100 × 2 datasets would take days. Ingest also degrades catastrophically (stalls at ~94% after ~12 h on a sequential archival-memory insert). Letta *validates* (it runs) but is impractical for a 100-query benchmark in this configuration.
- **EverMind-AI / EverOS** — runs only as an HTTP service and requires **two third-party credentials not available to this study** (OpenRouter for LLM extraction + DeepInfra for embedding/rerank, the latter provider-specific); installation also requires authorizing an external-repo `pip install -e`. Out of scope for the time-box.

LightRAG and HippoRAG2 are **not §6 competitors** — they are §5 KG/RAG references whose LLM-per-chunk graph-build cost (LightRAG warns >6 h on default) places them outside the memory-system comparison; they are deferred as optional baselines (§7.2).

**6.3.2 Embedding-matched variant (rc4 — all-Gemini, full eval)** The §6.3 split is reported under each system's *native default* embedder (the "as-configured" stance, §6.5, principle 5). Because nox-mem defaults to Gemini 3072d and Mem0 to OpenAI 1536d, the most obvious confound on the split is the embedding provider itself. To probe it we ran an **embedding-matched** variant — **rc4** — in which both systems use the **same embedder**: `gemini-embedding-001` at 3072d (the model and dimensionality nox-mem runs in production), with Mem0 re-pointed to it via an external config patch (`lib/all_gemini_config`). This was **not** a zero-edit swap: the Mem0 adapter also required a one-time API-compatibility fix for the mem0 2.x `search`/`get_all` signatures (declared as confound (a) below). The variant also replaces the n=100 sample with the **full evaluation set (n=2,482: 1,982 LoCoMo + 500 LongMemEval)** over the full 6,822-chunk corpus, scored identically to §6.3 (binary-relevance nDCG@10, k=10, same gold encoding). The evaluation gold is fully covered — every gold chunk exists in the corpus on both datasets — but note this is *corpus/gold* coverage, **not** Mem0 *store* coverage (Mem0 dropped 4 of 6,822 chunks on ingest; confounds below). This is an embedding-**matched** comparison (same model + dimensionality), **not** an all-else-equal controlled experiment: three residual confounds are declared below, and a fourth — the embedding task-type asymmetry — is tested by a dedicated ablation and shown not to drive the result.

| System (all-Gemini 3072d) | LongMemEval nDCG@10 (n=500) | LoCoMo nDCG@10 (n=1,982) | Overall (n=2,482) |
|---|---|---|---|
| **nox-mem** (hybrid FTS5 + Gemini + RRF) | **0.5255 ± 0.033** | **0.4952 ± 0.017** | **0.5013 ± 0.015** |
| Mem0 (Gemini embedder, Chroma) | 0.4061 ± 0.036 | 0.4407 ± 0.017 | 0.4337 ± 0.016 |

Intervals are 95% confidence (mean per-query nDCG@10 ± 1.96 · SEM (standard error of the mean), binary relevance). The nox-mem / Mem0 intervals are **disjoint on both datasets and overall**, so the gap is not attributable to query-sampling noise (it remains subject to the three residual configuration confounds below; the one asymmetry that favored nox-mem — task type — was ablated and shown not to drive the result).

**Result — the split does not survive embedding matching.** With the embedding model and dimensionality equalized (and the three residual confounds below declared, the fourth ablated away), **nox-mem outperforms Mem0 on both benchmarks** (LongMemEval +0.119, LoCoMo +0.055 nDCG@10) and **all five represented query categories** (§6.4). The Mem0 LoCoMo win in §6.3 was therefore substantially an embedder effect — OpenAI 1536d on LoCoMo — rather than a retrieval-architecture advantage: once both systems embed with the same Gemini model, nox-mem's hybrid FTS5 + dense + RRF retrieval wins on LoCoMo as well.

**Residual confounds — declared; this is embedding-matching, not a clean architecture isolation (per §6.6).** Three factors remain that prevent attributing the inversion purely to the embedding (a fourth — task-type asymmetry — was tested by ablation and neutralized; see below). **(a) Mem0 version drift:** the Mem0 client changed major version between the canonical run and rc4 (0.1.x → 2.0.10; its `search`/`get_all` API changed, requiring a compatibility fix to our adapter), so rc4's Mem0 (0.4337) is *not* the same build as §6.3's Mem0 (0.4686 LoCoMo) — part of the gap may be the version. **(b) Vector backend:** canonical Mem0 used a faiss store; rc4 Mem0 uses a fresh Chroma collection (required to avoid a dimension clash with the 1536d OpenAI run) — a backend change, not only an embedder change. **(c) Sample scope:** §6.3 sampled n=100/dataset; rc4 scores the full n=2,482 — a larger, differently-distributed query set, not a like-for-like re-score of the same 100. rc4 is therefore best read as *"same embedding model and dimensionality, Mem0 at its current release and default backend"* → nox-mem leads — **not** a surgical architecture-only isolation. (The 4 chunks Mem0 dropped to transient 503s affect one query, whose gold chunk Mem0 fails to retrieve in top-10 even when present — verified zero aggregate effect.) Raw results and aggregator output: `eval/q4-comparison/output/rc4/`.

**Task-type ablation — confound (d) tested and neutralized.** The one

configuration asymmetry that *favored* nox-mem in rc4 was the embedding task type: nox-mem passes Gemini's `RETRIEVAL_DOCUMENT`/`RETRIEVAL_QUERY` task types (retrieval-optimized embeddings), while Mem0's embedder call does not. To isolate it we re-ran nox-mem with a **generic Gemini embedding** (`NOX_EMBED_GENERIC_TASKTYPE=1` — no task type, exactly how Mem0 calls the same model) against the **same** Mem0 baseline, over the full n=2,482 — a symmetric *"neither system sets a task type"* comparison holding confounds (a)–(c) constant. nox-mem's overall nDCG@10 falls only **0.5013 → 0.4979 (−0.34 pp)** and **still outperforms Mem0 (0.4337) on overall, both datasets (LoCoMo 0.4920 vs 0.4407; LongMemEval 0.5215 vs 0.4061), and all five categories**. The task-type asymmetry therefore contributes at most 0.34 pp and does **not** explain the §6.3 → §6.3.2 inversion: the win is architectural (hybrid FTS5 + dense + RRF fusion), not an embedding-mode artifact. (Rigor caveat: the re-ingested generic corpus reached 99.03% gold coverage — 23 of 2,370 distinct gold chunks absent vs 100% in the task-type run, a transient-ingest handicap that can only *lower* nox-mem's score; the win persists despite it.) Ablation output: `eval/q4-comparison/output/rc4-ablation/`.

**LoCoMo claim taxonomy (orientation).** "Who wins LoCoMo" depends entirely on metric and configuration; the paper reports four distinct, non-contradictory readings, tabulated here to prevent conflation:

| § | Metric | Configuration | n | Result |
|---|---|---|---|---|
| §5.3.1 | retrieval@10 (strict) | nox-mem standalone | — | nox-mem 74.52% (vs Mem0's *published* F1 66.88% — different metric, not head-to-head) |
| §5.3.2 | answer F1 | nox-mem standalone | — | nox-mem rank-5 (below Mem0; verbosity-constrained, §5.3.2) |
| §6.3 | nDCG@10 | native embedders (nox Gemini-3072d / Mem0 OpenAI-1536d) | 100 | **Mem0** 0.4686 vs nox-mem 0.4263 |
| §6.3.2 | nDCG@10 | matched embedder (both Gemini-3072d) | 2,482 | **nox-mem** 0.4952 vs Mem0 0.4407 |

The §6.3 → §6.3.2 reversal is the embedding-matching effect (with the three residual confounds above; the task-type asymmetry was ablated and ruled out);

the §5.3 retrieval-vs-F1 contrast is an orthogonal metric distinction, not a contradiction.

### 6.4 Per-category breakdown

Per-query-category breakdown from the **rc4 embedding-matched run** (§6.3.2; all-Gemini 3072d, n=2,482 across both datasets). The canonical 2026-06-15 run reported only dataset-level metrics, so the breakdown is populated from rc4, where per-category gold labels — the datasets' *native* question types, mapped to six canonical buckets — are wired into the harness. Only nox-mem and Mem0 participate: agentmemory's server-side embedder cannot be re-pointed to Gemini, and the three §6.3.1 gaps likewise do not run under the matched embedder.

| Category | n | nox-mem nDCG@10 | Mem0 nDCG@10 | Δ |
|---|---|---|---|---|
| single-hop | 438 | **0.3969** | 0.3908 | +0.006 |
| multi-hop | 454 | **0.5481** | 0.4699 | +0.078 |
| temporal | 225 | **0.3940** | 0.2760 | +0.118 |
| adversarial | 524 | **0.4370** | 0.2955 | +0.142 |
| open-domain | 841 | **0.5991** | 0.5649 | +0.034 |
| numeric | n/a | n/a | n/a | — |

nox-mem leads every represented category. The largest margins are on **adversarial** (+0.142) and **temporal** (+0.118) — the query types where naive semantic similarity (Mem0's Gemini-only path) is weakest and the FTS5 lexical channel in nox-mem's hybrid fusion contributes most; the narrowest is **single-hop** (+0.006), where a single strong dense match suffices and the hybrid adds little. This pattern is consistent with the §5 ablations attributing nox-mem's edge to multi-signal fusion rather than embedding quality alone, and survives the task-type ablation (§6.3.2): with a generic Gemini embedding, nox-mem still leads all five categories (adversarial 0.4573 vs 0.2955; multi-hop 0.5561 vs 0.4699; open-domain 0.5792 vs 0.5649; single-hop 0.3924 vs 0.3908; temporal 0.3765 vs 0.2760).

**Bucket composition (declared per §6.6).** The six buckets aggregate the two datasets' native labels, and two mappings warrant disclosure because they affect interpretation. The **adversarial** bucket (n=524) combines LoCoMo's 446 native adversarial queries with 78 LongMemEval `knowledge-update` queries — a mapping flagged as *ambiguous* in the labeling audit (`docs/rc2-per-category-mapping.md`), since `knowledge-update` is not adversarial in LoCoMo's sense; the adversarial cell (nox-mem's largest margin, +0.142) should be read as "adversarial + knowledge-update", not pure adversarial. The `numeric` category receives `n/a`: neither dataset contributes at least 10 numeric-typed queries (open-domain is likewise LongMemEval-only), so per the n<10 rule those cells are not scored rather than extrapolated — avoiding the type of regression seen in §5.1.4 (temporal

`n/a` in G10b due to a degenerate corpus gap). The category labeler is `eval/q4-comparison/lib/category_labeler.py`.

### 6.5 Fair-comparison principles

The comparison adheres to principles standardized by the published benchmarking literature (EverMemBench; BEIR, Thakur et al. 2021, arXiv:2104.08663; MTEB, Muennighoff et al. 2023, arXiv:2210.07316):

1. **Identical corpus.** All systems receive the same `chunks.text` ingested via each system's native API. No system receives an "optimized" version of the corpus.
2. **Identical eval set.** Same queries, same gold sets, same random seed (`42` for LongMemEval shuffle).
3. **Native defaults per system.** Each competitor runs with its publicly documented default configuration. Competitors are not adversarially tuned to lose; if the default config is what is published, it is what is evaluated.
4. **K cutoff fixed at 10.** Some systems default to 5 or 20; all are forced to `k=10` for comparability.
5. **Native embeddings provider per system.** nox-mem uses Gemini 3072d; each competitor uses its default provider. The `all-Gemini` controlled variant that equalizes this provider — originally planned as an optional side experiment — was **executed (rc4)** and is reported in §6.3.2: with both systems on Gemini 3072d over the full n=2,482 set, nox-mem outperforms Mem0 on both datasets and all five represented categories (whereas as-configured, §6.3, Mem0 wins LoCoMo), with three residual confounds declared and the task-type asymmetry ablated away (a generic-embedding re-run still wins by at least +0.05 nDCG@10 on both datasets).
6. **Uniform hardware.** Same VPS (Hostinger 8 cores / 16 GB RAM), localhost between systems except for external embeddings API calls.

Each adapter passes a pre-run smoke test:

```
result = adapter.search("test query", k=5)
assert len(result) >= 1
assert all('id' in r and 'score' in r for r in result)
```

Any adapter that fails the smoke test is documented as a gap (`[FAILED: <reason>]`) rather than omitted — consistent with §6.6.

### 6.6 Anti-cherry-pick statement

To prevent retroactive selection bias:

- **All 6 categories reported.** None is omitted because the result is unfavorable.

- **Both datasets reported.** LongMemEval n=100 + LoCoMo full, side by side. We do not cherry-pick whichever benefits nox-mem.
- **Worst-case latency reported.** p50 + p95 + p99 explicit. We do not publish only p50.
- **Per-system per-category transparency.** The §6.4 table exposes every combination; there is no aggregate row that masks a pattern.
- **Gaps documented.** Systems that fail setup receive an explicit note; the comparison runs without the missing system, but the gap is recorded in `docs/COMPARISON.md`.
- **Explicit per-dataset breakdown (PR #318, 2026-05-23 — rev3).** The Gemini hybrid@500 run revealed that the aggregate (0.0918) masks a decisive per-dataset result. We report all three rows explicitly:

| System | nDCG@10 (aggregate) | nDCG@10 (LoCoMo-only) | Corpus | Mode |
|---|---|---|---|---|
| nox-mem FTS5@500 | 0.0466 | — | 500 (cap) | FTS5-only |
| **nox-mem Gemini hybrid@500** | 0.0918 | **0.1835** | 500 (cap) | FTS5 + Gemini + RRF |
| **mem0@500** | **0.1315** | 0.1315 | 500 (cap) | LLM rewrite + embed |

**LoCoMo-only result (PR #318):** nox-mem Gemini hybrid@500 = 0.1835 **exceeds** mem0@500 = 0.1315 by **+40%** on the conversational memory dimension. The aggregate (0.0918) falls below mem0 due to a **corpus-ordering artifact**: at the 500-chunk cap, the 5,882 LoCoMo chunks exhaust the cap before any LongMemEval chunk is ingested — the 10 LongMemEval queries have zero coverage, zeroing the nDCG for that dataset and pulling the aggregate down. Hybrid stack lift over FTS5@500: **+97%** (0.0466 → 0.0918), validating the architectural value of the stack even on a sparse corpus.

**H2 finding (PR #311, retained):** FTS5-only@500 = 0.0466 vs mem0@500 = 0.1315 is **real and architectural** for FTS5-only mode — mem0's LLM-rewriting produces semantic generalization that isolated FTS5 cannot match. PR #318 shows that the full Gemini hybrid reverses this result on the conversational scope.

**Mandatory disclosure:** the aggregate ±0.05 falls within the inconclusive interval for n=20. The definitive arbiter is the canonical full-corpus run (uniform corpus without cap, full LoCoMo + LongMemEval for all systems). **Phase 2 gate uses BOTH** per-dataset + aggregate from the canonical run — not only the number that favors nox-mem.

Refs: `docs/COMPARISON.md §Apples-to-apples corpus-cap comparison`, PR #311, PR #318.

### 6.7 Pre-registration

This section's methodology is locked in `specs/2026-05-23-Q4-comparison-execution-plan.md` prior to the Sat 2026-05-24 run. The **Sat 2026-05-24 15:30 BRT smoke** populated the first row of §6.3 (nox-mem combined: nDCG@10=0.6380, p50=8 ms, gold-hit 13/20 on 20 dry-run-sample queries) and validated that the retrieval pipeline works end-to-end on an eval-isolated DB. The **partial cross-system smoke of Sat 2026-05-24 18:00 BRT** added the mem0 row (n=20, 500-chunk corpus cap): nDCG@10=0.8569, p50=273 ms, gold-hit 3/20 (15%) — with an explicit interpretation of the coverage vs. concentration trade-off in §6.3. The **canonical run was executed on 2026-06-15** on a dedicated pod (n=100/dataset, k=10, same-namespace fair), resolving the 2026-05-24 infrastructure abort (incident D76 — sustained CPU-steal on the shared host, see §7.1 L5). It produced real numbers for 3/6 systems (nox-mem, Mem0, agentmemory) and three documented gaps (§6.3.1), updating the §6.3 cells. The pre-registered `all-Gemini` controlled variant (§6.5, principle 5) was subsequently executed as **rc4** (§6.3.2) — equalizing the embedding provider over the full n=2,482 set — and supplied the §6.4 per-category breakdown; this was a planned confound-control experiment, not a post-hoc methodology change, and its three residual confounds (Mem0 version drift, vector backend, sample scope) are declared in §6.3.2 per §6.6, with the task-type asymmetry tested by a dedicated ablation and shown not to drive the result. No methodology from §6.5–§6.6 was altered post-registration; the same-namespace re-query is a fair-comparison refinement (confound removal, §6.3) consistent with §6.5, not a change to corpus, eval set, or gold. Principles (§6.5), anti-cherry-pick (§6.6), and the general structure of this section are immutable post-run. Any methodological adjustment identified during execution is documented as an explicit follow-up in `docs/COMPARISON.md` rather than retroactively applied here. Refs: `q4-smoke-sat-2026-05-24-real-numbers` · `q4-partial-cross-system-sat-2026-05-24`.

Decision D43 (`docs/DECISIONS.md`) defines the approval gate: nox-mem in top-3 on >=2 of the 4 key metrics (nDCG@10, R@10, MRR, latency). If the gate is met, GTM Phase 2 is unlocked per `docs/ROADMAP.md` §7. If not met, the Sun 2026-05-25 session produces a remediation plan (pre-launch adjustments) rather than a direct launch.

### 6.8 Autonomy quantified — operational cost per memory system

The Q4 quality comparison (§6.3 – §6.6) reports retrieval *quality* under matched corpora. Operational *cost* — services, RAM, cold start, mandatory third-party credentials, setup commands — is the second axis on which a memory system

can be evaluated, and is the axis where the nox-mem Autonomy pillar [17] is most legible. Table 2 summarizes the steady-state idle footprint of each system in its default self-host configuration.

**Table 2 — Autonomy quantified: services, RAM, cold start, mandatory keys, setup commands.** Headline: **~12× less RSS than EverOS, single process, no service stack.** Competitor numbers are *estimates* derived from each project's docker-compose defaults and documented system requirements (sources cited in the row). The nox-mem row is [**measured 2026-05-24, prod VPS, 6830 chunks live, uptime 9h28min**] via `ps -eo pid,rss,vsz,comm,args` against the production `nox-mem-api` process (PID 2422197). See footnote [18] for full methodology including the cgroup `MemoryCurrent` vs process RSS distinction.

| System | Services | RAM idle | Cold start | Mandatory third-party keys | Setup commands | Sources |
|---|---|---|---|---|---|---|
| **nox-mem** | **1** (SQLite file + Node process) | **~341 MB RSS** [measured 2026-05-24] | **<1 s** | **0** (offline-OK; embeddings optional) | **1** (`npm i && nox-mem reindex`) | This work; [19] |
| mem0 | 2 (Postgres + Qdrant) | ~800 MB | ~15 s | 1 (OpenAI for embeddings) | ~5 | mem0 docker-compose defaults [20] |

[17] Q/A/P strategic pivot of 2026-05-17 — three pillars (**Q**uality, **A**utonomy, **P**roduct). See `docs/ROADMAP.md` and `qap-pillars-strategic-decision` in the project memory.

[18] The `~341 MB RSS` figure for nox-mem in Table 2 is **measured 2026-05-24** on the production VPS (PID 2422197, uptime 9h28min, 6830 chunks live, 100% vector coverage). Main process RSS = 349,276 KB via `ps -eo pid,rss,vsz,comm,args | grep dist/api-server.js`. The cgroup `MemoryCurrent` reported by `systemctl show nox-mem-api -p MemoryCurrent` is 727,064,576 bytes (~727 MB); the ~386 MB delta vs process RSS is SQLite memory-mapped I/O (chunks table, FTS5 index, vec0 index) — kernel-managed page cache, reclaimable on memory pressure, not exclusive process memory. Standard `ps/top` RSS is the canonical comparison metric used in Table 2 across all competitors. Original revision marked this as `~50 MB [estimated]` based on Node baseline + better-sqlite3 cache projections; the production measurement (initially denied during the first revision and granted post-PR #356) replaced the estimate.

[19] The `~341 MB RSS` figure for nox-mem in Table 2 is **measured 2026-05-24** on the production VPS (PID 2422197, uptime 9h28min, 6830 chunks live, 100% vector coverage). Main process RSS = 349,276 KB via `ps -eo pid,rss,vsz,comm,args | grep dist/api-server.js`. The cgroup `MemoryCurrent` reported by `systemctl show nox-mem-api -p MemoryCurrent` is 727,064,576 bytes (~727 MB); the ~386 MB delta vs process RSS is SQLite memory-mapped I/O (chunks table, FTS5 index, vec0 index) — kernel-managed page cache, reclaimable on memory pressure, not exclusive process memory. Standard `ps/top` RSS is the canonical comparison metric used in Table 2 across all competitors. Original revision marked this as `~50 MB [estimated]` based on Node baseline + better-sqlite3 cache projections; the production measurement (initially denied during the first revision and granted post-PR #356) replaced the estimate.

[20] mem0 default self-host requires Postgres + Qdrant + an OpenAI key for embeddings (or a configured alternative provider). Counts: 2 services + 1 mandatory third-party key. Source: mem0 README and `docker-compose.yml` defaults at github.com/mem0ai/mem0.

| System | Services | RAM idle | Cold start | Mandatory third-party keys | Setup commands | Sources |
|---|---|---|---|---|---|---|
| Letta | 3 (Letta server + Postgres + OpenAI) | ~1.5 GB | ~30 s | 1 (OpenAI) | ~8 | Letta self-host guide [21] |
| Zep OSS | 2 (Zep + Postgres) | ~1.2 GB | ~30 s | **1 mandatory** (OpenAI for embeddings — hardcoded) | ~6 | Zep README [22] |
| EverOS / EverMind-AI | **5** (MongoDB + Elasticsearch + Milvus + Redis + Postgres) | **~4 GB+** | **~60 s** | 2–3 (LLM + embedding + optional reranker) | ~15+ | EverMind-AI docker-compose [23] |
| LightRAG | 2 (Neo4j + vector DB) | ~1 GB | ~20 s | 1 (LLM provider for KG extraction) | ~6 | LightRAG repo defaults [24] |

**Reading the table.** Three rows of the cost matrix translate directly into Autonomy:

1. **Services column.** Every additional service is an additional failure mode, an additional security-patching surface, and an additional vendor that must be available on the day a user spins up the system. nox-mem ships as a single Node process operating on a single SQLite file; the only durable on-disk artifact is `nox-mem.db`. mem0/Zep/Letta/LightRAG each require >=1 database container and at least one external LLM/embedding provider. EverOS requires five containers, three of which are heavyweight infrastructure (MongoDB, Elasticsearch, Milvus). The single-service prop-

[21] Letta default self-host requires the Letta server, Postgres, and an OpenAI key (or alternative LLM provider) for the agent loop. Counts: 3 components + 1 mandatory third-party key. Source: Letta self-host documentation at docs.letta.com and github.com/letta-ai/letta.

[22] Zep OSS requires the Zep service container + Postgres, and *hardcodes* OpenAI as the embedding provider in the default build (mandatory key, no fallback in OSS distribution). Counts: 2 services + 1 hardcoded mandatory key. Source: Zep README at github.com/getzep/zep.

[23] EverMind-AI / EverOS docker-compose declares MongoDB + Elasticsearch + Milvus + Redis + Postgres = 5 services, plus 2–3 third-party API keys for LLM, embedding, and (optional) reranker. Counts confirmed against the published `docker-compose.yml` in the EverMind-AI repo. The ~4 GB RAM-idle figure is the sum of documented minimum requirements for each service's container.

[24] LightRAG defaults to Neo4j for the knowledge graph + a vector store (Qdrant/Milvus/etc.), plus one LLM provider key for entity/relation extraction during indexing. Counts: 2 services + 1 mandatory third-party key. Source: LightRAG README at github.com/HKUDS/LightRAG.

erty is what makes "open `nox-mem.db` in `sqlite3` and inspect everything" a literal operation, not a euphemism.

2. **Mandatory third-party keys column.** A system that requires an OpenAI key by default is not autonomous regardless of license — the user is dependent on one specific vendor's pricing, rate limits, and terms of service. nox-mem treats embeddings as optional (FTS5-only retrieval is a valid degraded mode; §4) and is provider-agnostic when embeddings are enabled (Gemini default, Ollama-local feasible — §7.1 L2). Zep and Letta hardcode OpenAI as the default embedding provider.

3. **Cold start column.** A `<1s` cold start is what makes self-host *try-before-deciding* — the user can `npm i`, run one command, see results, and decide. A `~60s` cold start with five containers is what makes EverOS effectively a "build a small team to evaluate" decision, not an individual decision.

**Caveat — RAM measurement methodology.** The competitor RAM figures are *idle* (i.e., process started, no queries served, no ingestion in progress) and are *estimates* read from each project's documented system requirements and `docker stats` defaults in the published docker-compose files. They are not from a head-to-head benchmark on a single host. A side-by-side measurement on a controlled 4-vCPU / 8-GB host is a §7.2 future-work item (F-cost-bench). The nox-mem `~341 MB RSS` figure is **measured** on the production VPS via `ps -eo pid,rss,vsz,comm,args` (PID 2422197, uptime 9h28min, 6830 chunks live); see footnote [25] for full methodology including the cgroup `MemoryCurrent` vs process RSS distinction.

### 6.9 Operational reproducibility as evidence

The §6.3 quality split and the §6.8 cost matrix report *outcomes*. A third, harder-to-fake signal emerged from the **act of running the benchmark itself**: the operational effort required to get each system to produce a number is a measurement of its dependency surface.

On the 2026-06-15 pod, nox-mem ingested the full 6,822-chunk corpus into a single SQLite file in one process with **zero incidents**. The competitors did not fare as smoothly:

- **Mem0** exhausted the pod's process-ID limit three times — its default

[25] The `~341 MB RSS` figure for nox-mem in Table 2 is **measured 2026-05-24** on the production VPS (PID 2422197, uptime 9h28min, 6830 chunks live, 100% vector coverage). Main process RSS = 349,276 KB via `ps -eo pid,rss,vsz,comm,args | grep dist/api-server.js`. The cgroup `MemoryCurrent` reported by `systemctl show nox-mem-api -p MemoryCurrent` is 727,064,576 bytes (~727 MB); the ~386 MB delta vs process RSS is SQLite memory-mapped I/O (chunks table, FTS5 index, vec0 index) — kernel-managed page cache, reclaimable on memory pressure, not exclusive process memory. Standard `ps`/`top` RSS is the canonical comparison metric used in Table 2 across all competitors. Original revision marked this as `~50 MB [estimated]` based on Node baseline + better-sqlite3 cache projections; the production measurement (initially denied during the first revision and granted post-PR #356) replaced the estimate.

telemetry (PostHog) leaks one thread per operation, and at benchmark scale this wedged the host until ingest and search were split into separate processes, telemetry was disabled (`MEM0_TELEMETRY=False`), and the vector backend was swapped from Chroma to faiss. Its LoCoMo store still lost ~5% of vectors to the leak before the workaround stabilized.

- **Zep** never ran — its Docker stack cannot start on an unprivileged pod kernel (§6.3.1).
- **Letta** ran but at ~16 min/query, making a 100-query benchmark a multi-day proposition.
- **agentmemory** silently persisted store state across sessions, contaminating a LoCoMo run with leftover LongMemEval vectors until the store was reset.

**The gaps are not neutral omissions — they are a deployability penalty.** Standard benchmarking convention records a non-running system as a blank cell and moves on. We make a stronger, but carefully bounded, claim: *the dimension on which these three systems could not produce a number is precisely the dimension nox-mem is engineered to win.* Each failure maps to a concrete operational deficit that nox-mem does not carry:

| System | Why it did not run | Operational axis it fails | nox-mem on the same axis |
|---|---|---|---|
| Zep | requires a privileged Docker host; cannot start on an unprivileged single-node kernel (§6.3.1) | deployability under constraint | runs unprivileged, single SQLite file, one process |
| Letta | ran, but retrieval routes through a full LLM agent turn at ~16 min/query | query efficiency | 2.9 ms KG path / 653 ms hybrid — **5–6 orders of magnitude faster** |
| EverMind-AI | needs 5 services + 2 mandatory paid third-party keys (OpenRouter + DeepInfra) | dependency footprint / autonomy | 1 process, **0 mandatory keys**, provider-agnostic |

**Honest bound (per §6.6).** We do *not* infer that nox-mem *retrieves better* than these three systems — we hold no quality numbers for them, and on a privileged Docker host with paid keys they may retrieve competitively. The claim is narrower, and stronger for being narrow: on the **deployability / efficiency / autonomy axis** — the axis that decides whether an individual developer, or an embedded agent that must live in its own process on commodity hardware, can use the system *at all* — **three of five competitors could not be made to produce a single result**, while nox-mem produced one from one file with zero external services. For that user, a system you cannot run is not "untested": it is unusable, and unusable is the worst score on the operational axis.

A lighter, dependency-free design is not a footnote to the quality comparison — it is the reason nox-mem has a number in every cell of §6.3 and three competitors do not.

The reproducibility of nox-mem's run — one file, one process, one command, re-confirmed live at **415 MB RSS on the 70.7k-chunk production corpus** on 2026-06-15 — is the operational expression of the Autonomy pillar quantified in §6.8. **A memory system you can run from a single file you own is a different category of dependency than one that requires Docker, a vector database, a telemetry pipeline, and a third-party embedding key to return its first result.**

---

## 7. Limitations and Future Work

### 7.1 Limitations

**L1 — Explicit-ingestion dependency (no zero-shot corpus coverage)** nox-mem retrieves only what has been explicitly ingested via `ingestFile()`, `ingest-entity`, or the inotifywait watcher pipeline. There is no mechanism to answer queries over arbitrary external corpora at query time. This is a deliberate design constraint — the system is optimized for an agent's *own* accumulated memory, not general-purpose retrieval augmentation — but it means that coverage is bounded by ingestion discipline. A corpus that has never been ingested produces zero recall regardless of query quality. Users bootstrapping the system must explicitly run `nox-mem reindex` over existing files before the hybrid search layer is useful. Ref: `specs/2026-03-14-nox-memory-system-design.md`, ingestion pipeline §3.1.

**L2 — Gemini API dependency for embeddings (cost + outbound network)** The semantic retrieval layer (Layer 2) depends on Google's `gemini-embedding-001` model (3072 dimensions). This introduces two constraints: (a) every vectorization call requires outbound network access and a valid `GEMINI_API_KEY`, meaning an air-gapped deployment falls back to FTS5-only retrieval with no semantic recall; (b) API cost scales with corpus size — at the Sat 2026-05-24 corpus of ~69k chunks, a full re-vectorization pass takes approximately 30–40 minutes at quota limits of the free tier. The Autonomy pillar of the Q/A/P strategy explicitly calls out "provider your choice, zero vendor lock-in" as a long-term goal; local embedding substitution (e.g., `nomic-embed-text` via Ollama) is architecturally feasible but not validated against the canonical eval set. bring-your-own-key (BYOK) partial autonomy is available: users who supply their own Gemini API key operate without per-query billing exposure in the default free tier. Ref: `docs/DECISIONS.md` (model selection), `default-flash-lite-for-agent-infra-tasks`.

**L3 — Single-instance architecture (no distributed sharding or replication)** The system runs on a single SQLite file per agent database. WAL mode provides concurrent read safety, but there is no horizontal sharding, no replication across nodes, and no distributed coordination layer. The current production corpus (~95k chunks across 7 databases on a 4-vCPU / 8GB KVM4, as of 2026-06-04) operates comfortably within these bounds, but the architecture does not generalize to multi-tenant deployments or corpora significantly exceeding the single-node memory/storage envelope. Distributed SQLite extensions (e.g., `cr-sqlite` CRDT-based replication) exist but are explicitly out of scope for v1. This is a known architectural decision, not an oversight. Ref: `docs/DECISIONS.md` (single-instance rationale).

**L4 — No write-side concurrency control (last-writer-wins)** Chunk ingestion operates under an optimistic concurrency model: `ingestFile()` deletes existing chunks for the source file and re-inserts in a single transaction, but there is no row-level locking or version fence against concurrent ingest of the same source file from two processes. In practice the inotifywait watcher and manual CLI calls rarely overlap, and the WAL journal prevents data corruption; however, two concurrent ingest calls on the same file produce non-deterministic chunk counts. The `withOpAudit()` wrapper (`src/lib/op-audit.ts`) does not add a mutual-exclusion layer for ingest — it targets destructive bulk operations (reindex, consolidate, crystallize). Production mitigations are operational (systemd service prevents concurrent watcher processes; cron stagger of 5 minutes between agents), not architectural. Ref: `docs/INCIDENTS.md#2026-04-25`, `a1-op-audit-module`.

**L5 — Evaluation sample size and canonical run gap (resolved 2026-06-15 / 2026-06-29)** The Sat 2026-05-24 cross-system run was a 20-query methodology smoke over an eval-isolated DB (5,882 LoCoMo + 940 LongMemEval chunks), not the canonical run; the first canonical attempt was **aborted** (incident D76: sustained CPU-steal on the shared production host, 51-97% over 48h, batch 005 0/50 in 23h). The canonical run was subsequently executed on dedicated infrastructure on **2026-06-15** (n=100/dataset, 3/6 systems producing numbers + 3 documented gaps, §6.3), and the **controlled-embedding variant (rc4)** was run on **2026-06-29** at full scale (n=2,482, both systems on Gemini 3072d, §6.3.2 / §6.4). The §6 competitive figures are therefore **settled, not `[deferred]`**; the residual limitation is the three declared confounds on rc4 (Mem0 version drift, vector backend, sample scope — §6.3.2), not sample size, and the one asymmetry that favored nox-mem (embedding task type) was ablated and ruled out. The earlier nox-mem smoke figure (nDCG@10 = 0.6380 combined) is not directly comparable to the G5 V3 entity-eval figure (0.6237) because the eval corpus and query set differ. Ref: `specs/2026-05-23-Q4-comparison-execution-plan.md`, `q4-smoke-sat-2026-05-24-real-numbers`.

**L6 — Cross-system comparison is methodologically partial** Three of five competitors could not be evaluated at all. After the canonical run (2026-06-15, n=100/dataset) and the rc4 embedding-matched run (2026-06-29, full n=2,482), Mem0 and agentmemory produced real numbers alongside nox-mem; the **500-chunk cap was a property of the superseded 2026-05-24 smoke only** and no longer applies. The standing limitation is **deployability coverage**: Zep (privileged Docker), Letta (agent-OS ~16 min/query), and EverMind-AI (third-party keys) produced no numbers in the unprivileged single-node environment (§6.3.1) — a deployability gap on the operational axis, not a retrieval-quality claim against them. The rc4 head-to-head additionally carries the three residual confounds declared in §6.3.2 (its fourth potential confound, embedding task-type asymmetry, was ablated and ruled out). Ref: §6.6 anti-cherry-pick statement, `docs/COMPARISON.md`.

**L7 — Latency comparison conflates transport classes** Cross-system latency was not captured uniformly. The canonical (2026-06-15) and rc4 runs did not record per-system latency percentiles under a normalized transport, and the only side-by-side figures — from the superseded 2026-05-24 smoke — compared nox-mem localhost retrieval against a Docker-in-Docker HTTP call, i.e. different transport classes, not a head-to-head speed claim. nox-mem's operational latency is therefore reported standalone in §5.7 (KG path p50 2.9 ms, hybrid p50 653 ms, re-measured 2026-06-15), not as a cross-system comparison. A normalized-transport latency benchmark (all systems behind one gateway) is future work (§7.2). Ref: `q3-latency-numbers-2026-05-18`, `docs/PERFORMANCE.md`.

**L8 — Pain signal is directional but not statistically significant in isolation** The "pain-weighted hybrid memory" framing rests primarily on the additive salience formula (§5.1.2) and section-aware ranking (§5.1.3). The `pain` dimension contributes W_PAIN = 0.10 of the salience weight, but its isolated causal contribution has not been validated to statistical significance: the E10 pain ablation (`paper/publication/paper-draft-sec4-7.md` §5.5) reports $\Delta = +0.0065$ with 95% CI $[-0.0143, +0.0338]$ on n = 31, directional but not significant. The current production corpus has 91.74% of chunks at the default `pain = 0.2` (as of 2026-06-04), providing insufficient variance for a precise estimate. A definitive pain signal ablation requires a corpus where pain scores span the full [0.1, 1.0] range. Ref: §5.7 honest characterization, `d47-path-c-decision`.

---

### 7.2 Future Work

**F1 — A2 Tier 3 P5: production-ready encrypted memory (in flight)** Phase 5 of the A2 Tier 3 roadmap targets a full SQLCipher-encrypted memory store with Ed25519-signed audit checkpoints (P4 deployed via PR #294). The signed checkpoint chain enables tamper-evident audit across destructive opera-

tions (`reindex`, `consolidate`, `crystallize`) without requiring a central trust authority. P5 closes the Tier 3 arc by integrating encryption key management with the existing `withOpAudit()` wrapper and the Tier 3 reads-audit layer (P3, PR #292/#293). Target deployment: post-GTM Phase 2 launch, estimated Sun 2026-05-25. Ref: `specs/2026-05-24-A2-tier3-crypto-audit-RECON.md`, `docs/A2-TIER3-MIGRATION-RUNBOOK.md`, `a1-op-audit-module`.

**F2 — F10 Phase C/D: shadow tracker empirical A/B for ranking changes** F10 Phase A (`/observability/health.html`) and Phase B (`/observability/evals.html`) are deployed (§5.7.4, decision D53). Phase C targets a shadow-mode query logger that captures production queries, executes them against a candidate ranking config in parallel, and accumulates query-level nDCG deltas before any promotion decision. Phase D operationalizes this into a pre-promotion gate: any ranking change (boost weight adjustment, mutex threshold, salience weight) that has not accumulated >=50 shadow queries with $p < 0.05$ improvement is blocked from reaching the production endpoint. This closes the observability gap identified in `ship-ranking-changes-in-shadow-mode-first` — currently the shadow mode is a flag toggle, not an integrated eval pipeline. Ref: `docs/ROADMAP.md` F10, decision D53, PR #207/#212.

**F3 — Per-method benchmark Phase B: cross-method nDCG optimization** The Q4 per-method benchmark (§6, `specs/2026-05-21-per-method-benchmark-comparison.md`) establishes the cross-system baseline. Phase B targets per-query-type boost calibration: given that keyword queries respond differently from natural-language queries (G10c §5.1.4), and single-hop vs multi-hop have opposing mutex trade-offs (G10b §5.1.4), a routing layer that selects ranking parameters based on query-type classification has measurable potential upside. Estimated Lab Q1 item. Ref: `specs/2026-05-21-per-method-benchmark-comparison.md`, `g10c-per-style-mutex-2026-05-21`.

**F4 — EverMemBench equivalence: honest comparison against EverMind-AI dataset** `everos-honest-comparison-benchmark-gap` identifies that EverMind-AI publishes standardized results on EverMemBench (EverCore 83% LongMemEval / 93% LoCoMo; HyperMem 92.73% LongMemEval). Running nox-mem on EverMemBench with the same evaluation protocol closes the benchmark gap and provides a reviewer-grade comparison for the arXiv submission. This is gated on EverMind-AI repository availability (currently unavailable) or an alternative comparable dataset. Estimated Lab Q1 priority if the repository returns. Ref: `everos-benchmark-publisher-competitor`, `benchmark-gap-longmemeval-locomo`.

**F5 — Neural reranker: cross-encoder rerank post-RRF** The current retrieval stack terminates at RRF fusion (§4.1). A cross-encoder reranker — receiving the top-K RRF candidates and the original query as a pair — is the stan-

dard next step in multi-stage retrieval and typically yields +3–8% nDCG@10 over bi-encoder baselines (see e.g., Nogueira & Cho 2019 on MS MARCO). The Autonomy constraint (`neural-reranker-evolution-vector`) favors a locally-runnable cross-encoder (e.g., `cross-encoder/ms-marco-MiniLM-L-6-v2` via sentence-transformers, ~66MB) over a cloud inference call, keeping the retrieval stack fully offline-capable. Estimated Lab Q1/Q2. Ref: `neural-reranker-as-vetor-evolutivo-pos-rrf`, `docs/ROADMAP.md` Lab Q1.

**F6 — Lab Q1 scale validation: 250k chunk corpus** The current production corpus is 94,936 chunks (as of 2026-06-04) — already approaching the ~100k-vector threshold where exact search latency degrades. The Lab Q1 roadmap targets a 250k chunk corpus to validate: (a) sqlite-vec ANN recall at scale (current exact-search; approximate search becomes necessary past ~100k vectors at reasonable latency targets); (b) salience formula stability (the recency component decays over a longer history window); (c) FTS5 BM25 IDF calibration (with more documents, rare-term IDF weights shift). The `lab-q1-scale-250k` item has no committed spec yet; it is gated on `NOX_SALIENCE_MODE=active` remaining stable through the GTM Phase 2 feedback cycle. Ref: `docs/ROADMAP.md` Lab Q1, `q-a-p-pillars-strategic-pivot-2026-05-17`.

**F7 — Multilingual corpus coverage: Portuguese and Spanish** The current evaluation corpus is English-dominant (LongMemEval and LoCoMo are English datasets; the internal entity-eval golden set mixes English and Portuguese). The FTS5 `unicode61` tokenizer with porter stemmer does not stem Portuguese or Spanish tokens correctly (e.g., "decisão" stems to "decisa" rather than "decid-"; Spanish gerunds lose morphological overlap). The Gemini semantic layer partially compensates via cross-lingual embedding space, but there is no explicit multilingual evaluation. A Portuguese golden set is a natural next step given the production operational language of the corpus; **Quati** (Bueno et al., 2024, arXiv:2404.06976), a Brazilian-Portuguese IR dataset annotated by native speakers, is a strong candidate benchmark for that evaluation. This is deferred to post-launch community feedback intake.

**F8 — GTM Phase 2 launch and community feedback intake** The Q/A/P roadmap (decision `qap-pillars-strategic-pivot-2026-05-17`) defines GTM Phase 2 as gated on D43 (nox-mem top-3 in at least 2 of 4 key metrics — nDCG@10, R@10, MRR, latency). With the canonical and rc4 runs complete, D43 is satisfied; the public OSS launch is the remaining gate-dependent step, with timing tracked in `docs/ROADMAP.md`. Post-launch, community feedback from the OSS release is expected to surface real-world limitation patterns not visible in the synthetic golden sets (e.g., corpora with high image-to-text OCR content, multi-language mixes, or very short memory fragments $< 20$ words that the current chunker merges). The feedback cycle directly informs Lab Q2 priorities. Ref: `docs/ROADMAP.md` GTM Phase 2, `docs/gtm/`, `overnight-automode-push-pattern`.

## 8. Knowledge Graph v2

### 8.1 Entity Extraction

**v1 (Regex-based)**: Used hardcoded regular expressions for 3 entity types (person, project, agent) with a static alias map for name normalization. Limited to predefined names, producing 26 entities.

**v2 (LLM-powered)**: Uses Ollama llama3.2:3b with a structured extraction prompt. Each chunk is processed with temperature 0.1 for deterministic output. The LLM returns JSON with entities (name + type) and relations (source + relation + target).

Extraction results after processing 866 chunks:

| Metric | Regex v1 | LLM v2 | Improvement |
|---|---|---|---|
| Entities | 26 | 384 | 14.8x |
| Relations | 59 | 529 | 9.0x |
| Entity Types | 3 | 11 | 3.7x |

**Entity Type Distribution:**

| Type | Count | Description |
|---|---|---|
| project | 109 | Software projects, products, repos |
| tool | 67 | Libraries, frameworks, CLI tools |
| concept | 54 | Abstract ideas, patterns, methodologies |
| person | 53 | Team members, contacts, stakeholders |
| organization | 50 | Companies, teams, departments |
| agent | 45 | AI agents in the fleet |
| location | 2 | Geographic references |
| other | 4 | Device, currency, date, computer |

### 8.2 Temporal Decay and TTL

Relations have a 90-day time-to-live (TTL) from creation. The confidence decay mechanism operates as follows:

1. Relations start with confidence 0.8 (extracted) or 0.9 (confirmed)
2. Every 30 days without re-confirmation, confidence drops by 0.1
3. Relations below 0.3 confidence receive accelerated 7-day expiry
4. Expired relations are deleted during `kg-prune` execution
5. Re-confirmation (observing the same relation in new chunks) resets confidence to 0.9 and extends TTL by 90 days

This mechanism ensures the knowledge graph naturally forgets stale information while reinforcing actively observed patterns.

### 8.3 Decision Versioning

Architectural decisions are tracked with full version history in the `decision_versions` table. Each decision has a unique key (e.g., `dedup-strategy`, `fallback-chain`) and supports:

- Version chains with supersession tracking
- Authorship attribution
- Source file provenance
- Current vs. historical querying

10 decisions are currently tracked, covering API key management, LLM fallback chains, embedding model selection, agent isolation strategy, and synchronization schedules.

### 8.4 Graph Traversal

The `findPath()` function implements BFS (Breadth-First Search) to discover shortest paths between any two entities. This enables queries like "How is Toto connected to nox-mem?" which traverses person → project → tool → agent relationships. Maximum depth is configurable (default: 4 hops).

---

## 9. Cross-Agent Intelligence

### 9.1 Agent Expertise Profiling

Each agent's memory is analyzed to determine its unique expertise based on chunk type distribution. The dominant chunk type determines the agent's strength category:

- **daily** → "Daily operations & activity logging"
- **team** → "Team coordination & shared knowledge"
- **decision** → "Decision tracking & rationale"
- **lesson** → "Lessons learned & pattern recognition"

Profiles include chunk counts, type breakdowns, top topics (via FTS5 term frequency), and last activity dates.

### 9.2 Knowledge Sharing

The `pullInsightsFrom()` function enables any agent to query lessons and decisions from other agents without direct database access. This creates a knowledge transfer mechanism where, for example, Cipher (Security) can learn from Forge's (Code Reviewer) past code review decisions.

`pullAllInsights()` aggregates insights across all agents, sorted by date, providing a fleet-wide learning feed.

### 9.3 Cross-Agent Knowledge Graph Merge

`mergeCrossKnowledgeGraphs()` scans all agent databases for kg_entities and kg_relations tables, merging them into a unified entity view. Entities are matched by type + lowercase name. The output shows which entities are known to which agents and their combined mention counts, enabling identification of shared knowledge vs. agent-specific expertise.

---

## 10. MCP Server Interface

nox-mem exposes 14 tools via the Model Context Protocol (MCP) over stdio (JSON-RPC 2.0):

| Tool | Category | Description |
|---|---|---|
| nox_mem_search | Retrieval | Hybrid search (FTS5 + semantic + RRF) |
| nox_mem_stats | Monitoring | Database statistics and health |
| nox_mem_primer | Context | Session recovery summary (~500 tokens) |
| nox_mem_ingest | Ingestion | Index a file into memory |
| nox_mem_cross_search | Cross-Agent | Search across all 7 databases |
| nox_mem_cross_stats | Cross-Agent | Chunk counts per agent |
| nox_mem_metrics | Monitoring | Daily observability metrics |
| nox_mem_kg_build | KG | Build knowledge graph from chunks |
| nox_mem_kg_query | KG | Query entity and its relations |
| nox_mem_kg_stats | KG | Knowledge graph statistics |
| nox_mem_agent_profiles | Intelligence | Agent expertise profiles |
| nox_mem_cross_kg | Intelligence | Merged cross-agent knowledge graph |
| nox_mem_kg_path | Intelligence | BFS path between entities |
| nox_mem_self_improve | Analysis | Contradiction detection, pattern analysis |

---

## 11. HTTP API Server

A lightweight HTTP API (Node.js built-in `http` module, zero dependencies) runs on port 18800, exposing memory data to the React dashboard:

| Endpoint | Method | Response |
|---|---|---|
| `/api/health` | GET | System health: chunks, consolidation, vector coverage, services, KG stats, DB size |
| `/api/agents` | GET | Agent expertise profiles array |
| `/api/kg` | GET | Knowledge graph entities and relations |
| `/api/kg/path?from=X&to=Y` | GET | BFS shortest path between entities |
| `/api/search?q=QUERY&limit=N` | GET | Hybrid search results |

| Endpoint | Method | Response |
|---|---|---|
| `/api/cross-kg` | GET | Merged cross-agent knowledge graph |

CORS headers are set for cross-origin access from the Vercel-hosted dashboard.

---

## 12. Operational Infrastructure

### 12.1 Cron Schedule

24 cron jobs manage automated operations:

| Time | Frequency | Job | Details |
|---|---|---|---|
| 23:00-23:25 | Daily | Agent consolidation | 6 agents, 5-min stagger, reindex→consolidate |
| 23:30 | Daily | Workspace consolidation | Central workspace daily notes |
| 23:35 | Daily | Session wrap-up | SESSION-STATE.md, Notion sync, git commit |
| 04:00 | Weekly (Sun) | Vectorize | Gemini embeddings for new/changed chunks |
| */5 min | Continuous | Health check | Watcher heartbeat, service liveness |
| 02:00 | Daily | SQLite backup | Online backup API, 7-day retention pruning |
| */6 hours | Continuous | Git backup | Auto-commit memory file changes |
| 09:00 | Weekly (Mon) | Token check | Forge CC token verification |

### 12.2 Backup Strategy

Three backup mechanisms operate independently:

1. **SQLite Online Backup**: Uses better-sqlite3's backup API for crash-consistent copies. Daily at 02:00, 7-day retention with automatic pruning.
2. **Git Auto-Commit**: Memory directory changes are committed every 6 hours, providing full change history.
3. **File System**: WAL mode ensures database consistency during concurrent reads/writes.

### 12.3 LLM Fallback Chain

To ensure continuous operation regardless of provider availability:

**Paid Tier**: Claude Opus → Sonnet → Haiku → GPT-5.1 → Gemini 2.5
**Free Tier**: Nemotron → Groq Llama70B → Healer → Hunter → Trinity → Gemma27B

The fallback is configured in the environment and selected at runtime based on task complexity and availability.

---

## 13. Dashboard Integration

The TotoClaw Command Center (React 18 + TypeScript + Vite + shadcn/ui) provides 11 pages including 4 nox-mem-specific views:

- **Memory Health** (`/memory`): Real-time system stats, vector coverage progress bar, service status indicators, agent breakdown table
- **Knowledge Graph** (`/knowledge-graph`): Interactive force-directed canvas graph, entity type filters, BFS path finder
- **Agent Intel** (`/agent-intel`): Agent expertise cards with type distribution bars, hybrid search interface, cross-agent knowledge entities
- **System Paper** (`/system-paper`): Live technical analysis with Recharts visualizations (pie, bar, radar, area charts), auto-refresh every 60 seconds

All data is fetched from the nox-mem API server via TanStack React Query with configurable polling intervals.

---

## 14. Evolution History

| Version | Date | Key Changes |
|---|---|---|
| v1.0 | Mar 14 | SQLite FTS5, basic search, consolidation, Notion sync |
| v2.0 | Mar 17 | MCP server, systemd services, watcher heartbeat, primer |
| v2.2 | Mar 20 | Cross-agent search, KG v1 (regex), self-improve, decision versioning |
| v2.5 | Mar 22 | Multi-agent workspace fix (OPENCLAW_WORKSPACE), gateway supervision |
| v2.6 | Mar 22 | Hybrid search default (FTS5+Gemini+RRF), 866/866 vectorized |
| v3.0 | Mar 23 | KG v2 (LLM, 384 entities), Cross-Agent Intelligence, HTTP API, dashboard |
| v3.7 | Apr 23 | Schema V10 (`retention_days` v8 + `pain` v9 + `section` v10), entity file format, section_boost |
| Wave A | May 19 | Additive salience formula, `tier_boost` off-by-default, `source_type` backfill (67,949 chunks), G5 V3 ablation (PRs #150 / #151 / #153) |
| G10 Hard Mutex | May 20 | `section/source_type` mutex deployed against `g9.db` 69,495 chunks (PRs #181 / #182) |

| Version | Date | Key Changes |
|---|---|---|
| G10d ACTIVE-T2 | May 21 | Conditional mutex gated by `query_entity_count <= 2`, deployed via systemd drop-in `NOX_MUTEX_QUERY_ENTITY_THRESHOLD=2` (PR #198, decision D51); multi-hop +1.58% nDCG / adversarial +3.04% nDCG recovered |
| F10 Phase A + B | May 21 | Foundation observability dashboards (`/observability/health.html` + `/observability/evals.html`) deployed via Tailscale tunnel (PRs #207 / #212, decision D53) |

---

## 15. Conclusion

nox-mem demonstrates that persistent, searchable, and shareable memory for AI agent fleets is achievable with commodity infrastructure (single VPS, SQLite, local LLM). The hybrid search system consistently outperforms single-method retrieval, particularly for multilingual content and compound technical terms. The LLM-powered knowledge graph provides 15x richer entity extraction compared to regex approaches, while temporal decay ensures the graph stays current without manual curation. The Wave A empirical evaluation (§5) established nDCG@10 = 0.6237 on the entity-flavored golden set (+78.8% relative over the G3 baseline), with `section_boost` identified as the dominant driver (99.85% of the lift recovered by A3 alone) and the additive salience formula validated by the `active > shadow` reversal. The G10d conditional mutex evolution (§5.1.4, deployed 2026-05-21) consolidates the canonical boost stack `section_boost × source_type_boost (Hard Mutex gated by query_entity_count <= 2) × salience v2 additive` in production, recovering multi-hop and adversarial regressions with contained dilution on single-hop. The F10 layer (§5.7.4, decision D53) makes production state verifiable at any time via the Phase A dashboard (`/observability/health.html`) + Phase B dashboard (`/observability/evals.html`).

The §6 Q4 COMPARISON is pre-registered (`specs/2026-05-23-Q4-comparison-execution-plan.md`). After a methodology smoke (2026-05-24, n=20) and an infrastructure abort (incident D76, §7.1 L5), the **canonical run executed 2026-06-15** (n=100/dataset, 6 systems: 3 produced numbers, 3 documented gaps — §6.3), and the **all-Gemini controlled variant (rc4) executed 2026-06-29** (full n=2,482 — §6.3.2 / §6.4). Under each system's native embedder the two leaders split — Mem0 wins LoCoMo, nox-mem wins LongMemEval (§6.3); with the embedding provider equalized, nox-mem outperforms Mem0 on both datasets and all five represented categories (§6.3.2). Both readings are reported side by side per §6.6, with three residual confounds declared and the task-type asymmetry ablated away. The D43 gate (top-3 on at least 2 of the 4 key metrics) is

satisfied, unlocking the GTM Phase 2 pre-launch defense.

The cross-agent intelligence layer transforms isolated agent memories into a collaborative knowledge base, enabling institutional learning across the fleet. Combined with the live dashboard, the system provides full observability into the collective memory of the agent organization.

**Repository:** github.com/totobusnello/memoria-nox **Dashboard:** github.com/totobusnello/agent-hub-dashboard **Spec:** Projetos/memoria-nox/specs/2026-03-14-nox-memory-system-design.md

---

## References and Footnotes

**Related-systems references**

**Internal references**

**Autonomy table (Table 2) sources**