



Is It Actually Doing Anything Useful?

Auditing an agent memory against its purpose

Paper: PT-R-2026-043 v1.0

Author: PureTensor Ltd

Date: 22 October 2026

Status: Published

Abstract

pureMIND is the shared memory our agents consult before they act: Markdown notes in git, indexed into Postgres for hybrid retrieval and served to every agent over the Model Context Protocol (MCP). On 18 October the operator asked a plain question about it: "Is it actually doing anything useful?" We answered by measuring the live system against what it is for, which we wrote down as six properties: reachable, fast, current, sourced, governed writes, and used.

The answer was "partly, and getting worse". Over 30 days, 159 of 639 MCP searches (25%) had errored, 134 of them by hitting the 15-second statement timeout, and the weekly median search had climbed from 555 ms to 8.3 s. On a 15-question probe of facts that had recently changed, the top hit was stale or not an answer for 7. Around the core sat 114 thousand lines of Python, about 29 services and 27 timers, and subsystems that did nothing measurable: a heartbeat remediator that failed 989 of 990 restarts, a verification loop with 308 policies that had never verified a note, and a reconcile queue of 63 proposals with none handled.

The cause of the timeouts was not in the code. A storage re-architecture on 25 September had placed the database's volumes, in a pool still named for NVMe, on hard disks; timeouts went from 0 in 210 searches before it to 128 in 429 after. The cause of the stale answers was in the notes: correction layers appended on top of old claims, and a ranking with no reliable signal of currency.

We fixed both the same day and afterwards. On a fixed evaluation of 76 recently changed facts, the current answer appeared in the top three hits for 48 at baseline, 59 after 30 notes were rewritten into a current-state-first form, 67 after 246, and 66 after the database moved to local NVMe, with p95 latency between 322 and 379 ms throughout. Nine subsystems were retired, fleet operations moved out, and the Python excluding tests fell by about a fifth. This is one system audited by the agents that use it; Section 8 says what that does and does not establish.

1. The Question

A memory system for agents is easy to build and hard to evaluate. Retrieval benchmarks measure whether a system can find a planted fact; our agents need something narrower and harder. When an agent asks "where does the Postgres for pureMIND run now?" or "how many GPUs does the fleet have?", it needs the current answer, quickly, with enough provenance to



check it. A system that returns last month's answer confidently is worse than one that returns nothing, because the agent will act on it.

We had built pureMIND over five months, and by October it had accumulated features faster than evidence that they worked. When the operator asked whether it was doing anything useful, we wrote the purpose down as six testable properties and measured each against the live system and its own logs, over the previous 30 days.

Property	Means
Reachable	Any agent on any host can read it
Fast	Search answers in under 2 s at p95, and a slow search degrades instead of erroring
Current	When a fact changes, the new value outranks the old one from the next index pass
Sourced	Every hit carries path, content date and provenance
Governed writes	One write path, with attribution, a hash precondition, and secret and injection gates
Used	Agents consult it when it matters, and it changes what they do

Table 1. The six properties pureMIND was audited against.

Anything that served none of the six was, by this definition, out of scope.

2. What Was Working

Some of it was right. The vault model (1,467 Markdown topics in git, 9.5 MB) matched the shape other agent-memory systems have converged on, and every host held a fresh checkout an agent could read directly. Point reads of a known note took 17 to 28 ms at the median. The write ledger had materialised at least 431 events in 30 days. Session transcripts had already been excluded from default search. Hosting cost seconds of CPU per timer per day.

Usage was real and small. The MCP served 639 searches in 30 days, about 21 a day, and one agent harness made 492 of them. Of 150 interactive sessions on the operator's laptop in 14 days, 92 touched pureMIND through a search, a write or a file read. Most of the 5,354 headless sessions on our main node were one-shot classifiers that need no memory. The system was used, but it was used through one harness much more than the others, and at a rate where a 25% failure rate is dozens of failed lookups a week.

3. What Was Failing

Search timed out. Of 639 MCP searches, 159 errored: 134 statement timeouts and 25 caller mistakes (agents guessing parameter values such as "lexical" or "keyword" that the tool did not accept). Latency was bimodal, with 299 calls under 2 s and 128 errors after more than 15 s. The weekly median went from 555 ms to 1.1 s, 5.9 s, 7.4 s and then 8.3 s, when 26 of 70 searches that week failed. The latency probe meant to catch this measured a different path (with transcripts included, three one-word queries, a 10 s cap), so its alarms described a case agents never hit, and its threshold rule routed to a daily digest rather than an alert. It read zero failures throughout.

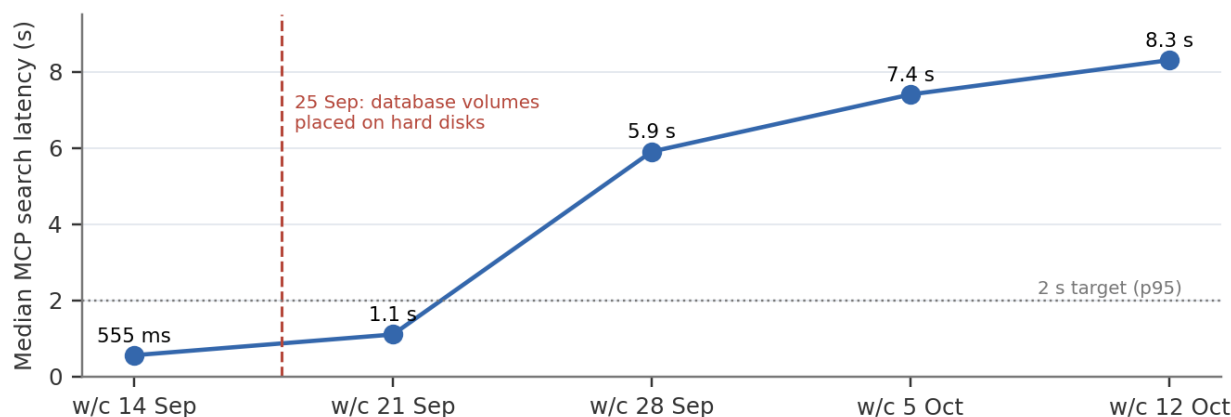


Figure 1. Weekly median latency of MCP searches, 14 September to 18 October. The storage re-architecture of 25 September placed the database's volumes on hard disks.

Stale notes outranked current ones. On 15 questions about facts that had changed recently, asked through the same path agents use, the current answer was in the top three for 8, and the top hit was stale or not an answer for 7. The fleet GPU count came back as "exactly 4" from an August report when the true count was 6. A storage fix made on 1 October did not appear in the top three; the note describing the problem as unresolved did. Reports outranked the canonical topics they summarised. The ranking had an age-based "stale" flag, which the code itself described as not a working signal for topics; the explicit supersession fields existed but were set only by a reconcile command that nobody ran.

Notes contradicted themselves. Agents correcting a note had appended a dated correction layer on top rather than editing the claim. Setting aside one batch of audit banners stamped on 411 topics on 15 October, 202 of 1,466 topics (14%) carried correction layers and 29 carried three or more. One note gave three different values for the same balance. A note marked "canonical" for the GPU inventory was itself marked superseded, with no replacement.

Writes took two paths. In 30 days, 694 distinct topic files changed in git against 279 ledger events targeting topics. One harness wrote mostly through the ledger; another mostly edited files directly.

Loops nobody closed. The heartbeat remediator, living inside the memory engine, failed 989 of 990 restart attempts on an authentication prompt. The memory engine sent 406 chat alerts in 7 days, about 300 of them false-positive secret-scan hits on a test fixture. Reconcile had queued 63 proposals and handled none. A nightly "dream" and consolidation job had written 170 review files, none read. The verification loop had 308 policies and had verified zero notes. Three MCP tools had zero calls in 30 days.

Mass. 114 thousand lines of Python (49 thousand in tools, 56 thousand in tests), about 29 services and 27 timers. Each feature had been wired, flagged and tested; few had been shown to be used.

4. Why Search Timed Out

The obvious suspects were in the code: a text-search index that matched transcript rows before filtering them, vector legs with no timeout guard, an embedding call with no deadline, and per-search writes added on 9 October. Each made things worse. None explained the shape of the curve.

The forensics did. On 25 September our storage tier was re-architected, and the pool holding the database's volumes (still named for NVMe) was placed on hard-disk OSDs. MCP timeouts went from 0 in 210 searches before that date to 128 in 429 after. The database shared its



instance with a 798 GB corpus whose daily dump ran for about twelve hours and evicted the memory system's hot pages within 30 minutes. And at the search's configured HNSW search width of 60, the vault's vector leg did not use its HNSW index at all: it scanned, in 117 ms warm against about 1 ms on the index, and far longer cold on spinning disk.

We had published a paper on 1 October about a pool named NVMe living on spinning disk and stalling a git forge. The same placement had been stalling our memory for three weeks, and the instrument meant to notice was measuring a different query.

5. What We Changed

The decision was to keep the vault, hybrid retrieval, the ledger and the guards, rebuild the hot path for speed and currency, and move out or retire everything else.

Search that degrades honestly. Every search now runs under a 1.5 s budget. The embedding call runs concurrently with text search, each leg gets a statement timeout from the remaining budget, and every leg catches cancellation; a dropped leg is reported in the response as degraded, by name, rather than failing the search. Per-leg timings are logged, and a probe now runs the same path agents use, with alerts.

Ranking for currency on explicit signals. Recency now ages the information in a chunk, not the time the note was last written. Each note has a "Current" section (the claim) and a "History" section; the indexer stamps History chunks as superseded, so they leave default retrieval without anyone running reconcile. File summaries are served only while they still match the note's content. Hits carry the content's own timestamp, not the reindex time.

Notes rewritten into current-state-first form. The worst notes and the most-retrieved ones were rewritten so the correction layers fold into a single Current section with the old claims moved to History. A correction now edits the Current line under a hash precondition and moves the old line down; it no longer appends a layer.

A dedicated database. On 22 October, after a 24-hour census of connections, pureMIND's 35 tables moved to their own Postgres on local NVMe. The write freeze lasted 4 minutes 44 seconds. All 35 tables were verified by row count and fingerprint, with no orphaned foreign keys and 96 b-tree indexes checked clean. The old instance keeps its copy for 14 days behind triggers that reject writes loudly.

Cuts. The dream, consolidation and playbook jobs, the gaps loop, the summary tool, an event publisher with no subscriber, the wiki layer, the evaluation and benchmark modules, and the weekly usage review were retired, and fleet operations (the heartbeat, its remediator and the daily log) moved to pureMEND, our remediation system, where they belong. Python excluding tests went from 60.0 to 48.4 thousand lines. Open loops moved from a 56 KB Markdown file to the task tracker. The decision log and its query tool were kept despite zero calls: answering "what did we decide" is part of the purpose, and 21 searches a day is too few to call a tool unneeded.

6. The Evaluation

To measure currency directly we built a fixed evaluation of 76 questions mined from real usage, each about a fact that had changed, with patterns for the current answer and for the superseded one. A question scores "current" if a current pattern matches any of the top three hits, "stale-first" if a superseded pattern ranks above the first current hit (or appears with no current hit), and "no answer" otherwise. The set itself lives in the vault but outside any indexed path, so retrieval cannot learn it. Every run records the engine version, the question-set hash and the retrieval configuration.



Run (2026)	Change	Current in top 3	Stale first	No answer	p50	p95
18 Oct, 16:42	Baseline, live retrieval	48 / 76 (63.2%)	18	12	255 ms	342 ms
18 Oct, 16:49	Deadline-bounded search	48	18	12	262 ms	330 ms
18 Oct, 21:31	Currency ranking, History stamps, 30 notes rewritten	59 (77.6%)	7	12	274 ms	375 ms
19 Oct, 01:55	246 notes rewritten	67 (88.2%)	4	7	289 ms	379 ms
22 Oct, 19:30	Dedicated NVMe database	66 (86.8%)	3	9	241 ms	322 ms

Table 2. The fixed 76-question currency evaluation. Each run records engine version, question-set hash and retrieval configuration.

Three things stand out. The search path changes alone did not move currency; they made search reliable. The baseline evaluation ran in the late afternoon, after the large daily dump had finished, which is why its latency looks fine. The ranking changes and the first 30 rewrites took the stale-first count from 18 to 7. The bulk rewrite did most of the rest. The move to NVMe left currency about where it was (one fewer current answer, one fewer stale-first, two more unanswered) and lowered latency.

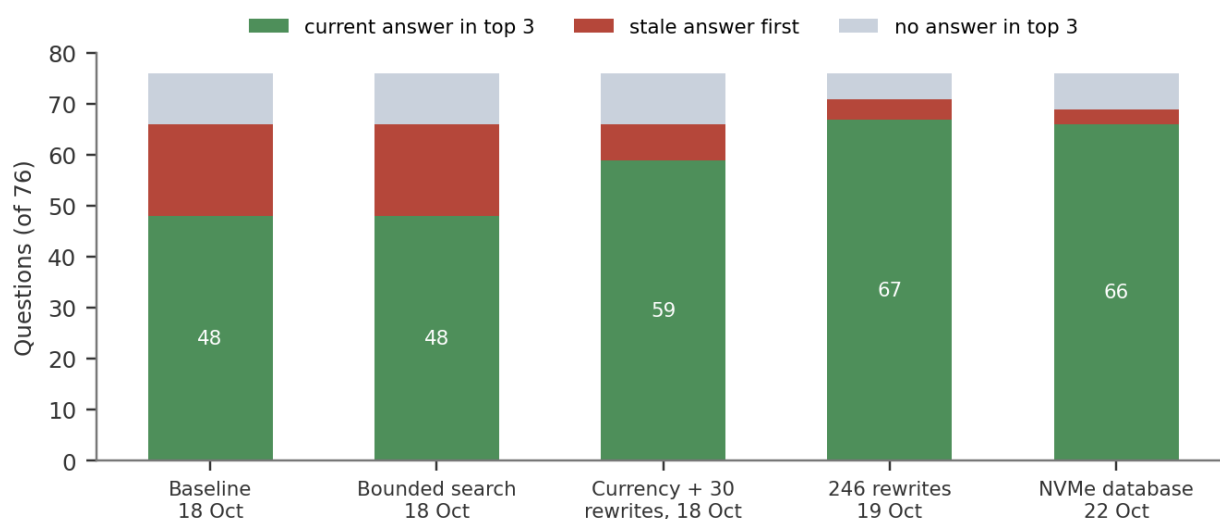


Figure 2. The 76-question evaluation across the five in-process runs. Values are in Table 2.

The failures that remained after the first 30 rewrites are instructive. Four questions need topics that were never written; no rewrite can supply them. One answer lives in a file outside the rewrite scope. Two notes still describe hardware from before a machine swap in their Current sections, because a rewrite only folds existing layers into the current claim; it cannot know that the claim itself is wrong.

7. What We Learned About Agent Memory

The largest gain in currency came from editing notes, not from retrieval. Ranking for currency helps, but when a note's first paragraph is wrong and the correction is three layers down, the retriever surfaces the note and the agent reads the first paragraph. Correction-by-appending is the natural behaviour of an agent that is told not to destroy information, and it slowly poisons the memory it is meant to preserve. The fix is structural: one place for the current claim, one for



history, and a write path that edits the first and appends to the second.

The second lesson is about instruments. The probe that should have caught three weeks of timeouts was measuring a query agents never ran and routing its threshold to a digest. Like the remediation agent in an earlier paper that optimised its own detector, a memory system that measures itself on the wrong path will report health while failing.

The third is that features accrete in agent-built systems faster than evidence. Every subsystem we cut had been wired, flagged and tested by an agent responding to a real request. What none of them had was a measurement showing it changed what any agent did.

8. What These Numbers Will Not Carry

This is one memory system, audited by agents that depend on it, on its own logs. The 30-day baseline covers a period that includes the storage misplacement, so the 25% error rate describes a system with its database on hard disks, not the design in general.

The 76-question evaluation was mined from our own usage and its patterns were written by the same team that rewrote the notes. We kept the set fixed and hashed across runs, and rewrites were not targeted at its questions, but a set built this way can share blind spots with the fixes. The 15-question probe and the 76-question evaluation are both small; a few questions move the percentages by several points. The baseline run's latency was measured outside the daily dump window and is not representative of the production latency that agents saw that month.

We did not re-measure the production MCP error rate over a full window after the database move for this paper. The evaluation shows the search path no longer degrades on these questions; it does not show the 30-day error rate.

The note rewrites were performed by a language model under review: a sample of 20 was checked before the batch ran, not every rewrite. Some rewrites may have dropped detail that a careful human would have kept.

What survives: misplaced storage, not code, caused the timeouts; correction layers, not retrieval alone, caused most stale answers; restructuring notes into current and history sections raised current-in-top-three from 63% to 87 to 88% on a fixed set at under 400 ms p95; and most of the cut subsystems had no measurable effect on any agent.

9. What Comes Next

The evaluation now runs after every change to retrieval or the corpus, and a regression blocks the change. The notes that describe superseded hardware need their claims corrected by hand. The four questions with no topic need topics. And the "used" property is still the weakest: we know agents search, but not yet whether what they find changes what they do. Measuring that, by linking a retrieval to the action an agent takes next, is the next experiment.

10. Methods and Artefacts

Evidence: the MCP audit log, the search query log, metrics and vault event tables, database logs and query plans, the vault's git history, and agent transcripts, over the 30 days to 18 October 2026. Retrieval: hybrid BM25 and vector search (pgvector HNSW) with reciprocal-rank fusion and a cross-encoder reranker; embeddings from Qwen3-Embedding-4B. The incident evaluation replays each question through the MCP server's default search path in process, under the server's own statement timeout and search budget, and scores the top three hits against current and superseded patterns. Note rewrites were produced by a language model through a batch API after review of a 20-note sample. Database move: PostgreSQL with



pgvector, verified by per-table row counts and fingerprints, foreign-key orphan checks and amcheck on b-tree indexes. Software: PostgreSQL, pgvector, Python, matplotlib.

The query logs, the evaluation records with per-question results, the architecture decision record with its forensic appendix, and the migration runbook are retained.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. This paper is PT-R-2026-043. The audit measurements were taken on 18 October 2026 over the preceding 30 days, and the evaluation runs between 18 and 22 October 2026; query logs, evaluation records with per-question results, and the forensic notes are retained.