# A 1.1M-token memory on an 8 GB laptop

KV-cache probing and splicing let a 4B model recall facts from a session 40× longer than its working context.



**Written by Azeruddin Sheikh**
INDEPENDENT R&D · 2026

LINKEDIN ↗

## Context

Long-context models are sold on a token number. The number is misleading. A model asked to hold a 1.1-million-token agent session in a single context either runs out of memory or pays to re-read the whole session on every turn.

Kamaloki is an independent research project that asks a different question: can a small model, on a laptop, keep a long session's *memory* without keeping its *tokens* in the context window?

The reference run used Gemma 4 E4B (4-bit) on Apple silicon through MLX. The test session was a real 1.1-million-token agent transcript with 94 exchanges, 15 planted facts, and 42 questions about the work in the session.

## The problem

The constraints were hard:

**Memory.** The target was an 8 GB machine. Holding the last 107K raw tokens alone used 8.25 GB and still forgot almost everything older than the recent window.

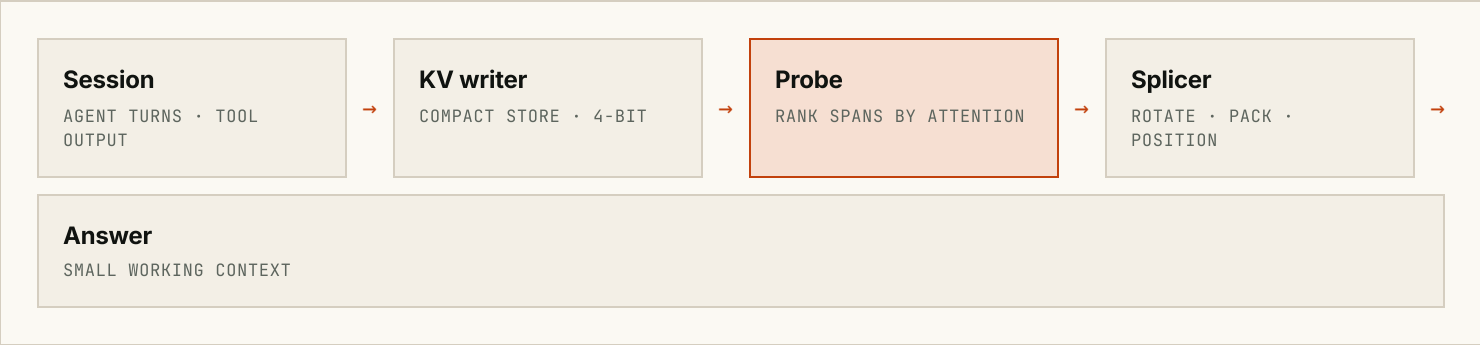
**Recall.** The model had to answer questions about specific facts scattered across the session, including facts that appeared exactly once.

**Cost.** Re-prefilling the full session for every question is slow and expensive. A memory system that takes 25 seconds to assemble history is not usable in an agent loop.

**Fidelity.** The spliced context had to behave like the original context. An approximation that changes the model's answer is worse than no memory at all.

## Architecture

The system writes the session's KV cache to disk as it runs, then probes that store to find the spans relevant to the current question. The selected spans are rotated back into the model's current position space and spliced into a small working context.




REFERENCE ARCHITECTURE. THE STORE IS WRITTEN INCREMENTALLY; ONLY GLOBAL-LAYER KEYS ARE KEPT FOR OLD BLOCKS.

The key decisions:

**Store the cache, not the text.** The memory is the model's own keys and values, written at the position they were computed. Retrieval is an attention probe against those keys, not an embedding search over chunks.

**Compact old blocks.** Blocks older than a threshold are rewritten with global-layer keys only, 4-bit, at a fraction of the original size. Sliding-window layers are dropped for those blocks because they cannot attend that far back anyway.

**Rotate, don't recompute.** The stored keys were written at their original positions. RoPE rotation moves every token by the same delta so the span can be placed at a new logical position without re-running the model.

**Keep the tail real.** The most recent turns are computed fresh. The splice only supplies the distant memory.

**Measure recall on facts that appear once.** Transcript questions that recur in the session can be answered from the recent tail. The planted-fact set is the clean test of long-range memory.

## What I did

Built the KV store, probe, ranker, and splicer for a 42-layer Gemma 4 E4B model, including the model-specific RoPE and sliding-window assumptions.

Built a replay harness that runs a 1.1M-token session through the engine as if it were live, with the model's replies teacher-forced.

Built the memory evaluation: 15 planted facts, 42 transcript facts, and a grader.

Fixed the memory peak on large turns by loading and rotating only the sliding entries that survive, bringing the worst case from 9.21 GB to 7.85 GB in the step rebuild.

Ran the comparison against a frontier model reading the raw session, and against the same small model holding the last 107K tokens raw.

## Results

| SYSTEM | OVERALL | PLANTED FACTS | OLD FACTS (≥20 EXCHANGES) | PEAK MEMORY | PER QUESTION |
|---|---|---|---|---|---|
| DeepSeek Flash, raw session (~981K tokens) | 96.5% | 13/15 | 11/12 | cloud | — |
| **Kamaloki, E4B, ~28K working context** | **89.5%** | **15/15** | **12/12** | **4.93 GB** | **3.4 s** |
| Kamaloki, two-level probe | 87.7% | 13/15 | 10/12 | 4.93 GB | 3.3 s |
| E4B, last 107K raw tokens, no retrieval | 50.9% | 2/15 | 0/12 | 8.25 GB | — |

The clean measure is the planted-fact set: each fact appears exactly once. Kamaloki recalled **15/15** from a session 40× longer than its working context. A frontier model reading the raw session recalled **13/15**.

History assembly — the work between “question arrives” and “model can start generating” — took **0.78 s** with splicing versus **24.6 s** for a full gold re-prefill: **31× faster**.

A separate classification experiment scored a closed set of labels by probing cached label keys instead of generating an answer. On a 58-item product set it reached **96.6% top-1** at **0.1 s per item**, with no training. On DBpedia’s 219 classes it beat generative prompting **80.4% vs 78.2%**.

| 15/15 | 31× | 4.93 GB | 3.4 s |
|---|---|---|---|
| PLANTED FACTS | FASTER ASSEMBLY | PEAK MEMORY | PER QUESTION |

**What did not work**

The same engine scored **0/14** on a separate agentic task suite where DeepSeek Flash scored 14/14. The harness worked; the 4B model could not do the tasks. That result is in the record because it defines the boundary of the claim: this is a memory architecture, not a general capability upgrade.

The two-level probe, added for sessions above ~2M tokens, cost one planted fact at 1.1M tokens and saved no meaningful time. It is disabled by default at this scale.

## What it proves

Long-document and long-session work usually fails on memory architecture, not on model intelligence. Once the failure is framed as a cache, retrieval, and positioning problem, it becomes an engineering problem with measurable tradeoffs — and a cost you can model before you buy a bigger model.

The public benchmark harness and technical note are in kamaloki-bench.

---

AT A GLANCE

HEADLINE RESULT

15/15

planted facts recalled from 1.1M tokens

LABEL

Independent R&D

PERIOD

2026

CONTEXT

Local inference · long-context memory

STACK

MLX · Gemma 4 E4B 4-bit | KV-cache store | Attention probing | RoPE rotation and splicing | Evaluation harness

---

**Download PDF →**

**Discuss a similar problem →**

| | |
|---|---|
| 1.1M<br>TOKEN SESSION | 15/15<br>PLANTED FACTS |
| 31×<br>FASTER HISTORY ASSEMBLY | 4.93 GB<br>PEAK MEMORY |

NEXT CASE STUDY

A browser-native agent with 3–10× token savings

**All work →**