LAM: AN EFFICIENT AND CONTROLLABLE LOSSY AGENT MEMORY FRAMEWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent memory grows as agents read inputs, reason, and call tools. Longer histories increase inference cost and eventually exceed the context window. LLM-based summarization reduces this history but adds latency and provides no explicit bound on information loss. We propose LAM, a **L**ossy **A**gent **M**emory system with three components: a deterministic deduplication rule with a substitution bound on retrieval scores — a bound on score perturbation, not a certificate of unchanged ranking, a memory manager that preserves the cached prefix and overlaps compaction with inference, and a performance model that estimates compaction costs before deployment. On 600 agent trajectories, LAM removes 22.47% of observation tokens while retaining 99.984% of the measured gold-patch evidence. Evaluation on end-to-end latency shows that LAM achieves a 71.4×–91.6× end-to-end speedup from removing records before prefill, validating LAM's effectiveness and efficiency.

1 INTRODUCTION

AI agents retain the files they read, the commands they run, and the resulting outputs. This history supports task completion (Chhikara et al., 2025; Yu et al., 2026), but its growth raises inference cost and latency, and once it exceeds the context window some of it must go. Memory compaction decides what to retain, rewrite, compress, or discard (Chhikara et al., 2025; Fang et al., 2026; Chen et al., 2026); during long tasks it runs repeatedly and can block inference for tens of seconds per event (Cim et al., 2026; Kummer et al., 2026).

Existing methods select information for later reuse. LLM summarization and learned compressors process the context with a model (Mu et al., 2023; Ge et al., 2024; Cheng et al., 2024; Pan et al., 2024), adding inference cost; SemDeDup (Abbas et al., 2023) clusters embeddings with k-means, whose output can vary with its initialization; observation masking keeps recent tool observations and discards older ones (Lindenbauer et al., 2025). Each exposes a budget, cluster count, or window

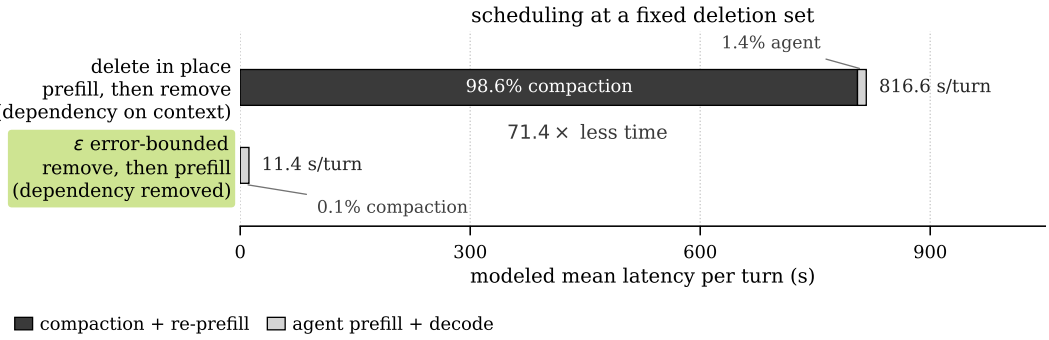

Figure 1: Modeled mean latency *per turn* at a fixed deletion set: same compressor, same removed records, only the schedule differs. Deleting from a prefilled context forces a re-prefill, so 98.6% of the average turn goes to compaction against 0.1% when removal precedes prefill. One 1,453-turn session; turn shape from 600 SWE-smith trajectories.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

size; none bounds the retrieval-score loss deletion causes. Furthermore their reproducibility depends on the model and execution configuration (Appendix D).

Tool outputs repeat: agents reread files, rerun tests, and list directories. Hashing (Quinlan & Dorward, 2002; Zhu et al., 2008) detects exact copies but misses outputs differing in small details. LAM therefore keeps exact hashing as its first layer — the mechanism serving systems already use to reuse identical KV blocks (Kwon et al., 2023) — and adds a second above it that merges near-duplicates whenever unit-normalized embeddings satisfy $\|a - b\|_2 \leq \delta$, a query-free test equivalent to a merge threshold on cosine. Thus every deleted record keeps a representative within δ, and for every unit query the maximum retrieval score drops by at most δ (Theorem 1) — a guarantee on scores, not ranking or task outcomes, which we evaluate separately (§3). With fixed embeddings and arrival order LAM is reproducible, and reports the maximum substitution distance it observed, $\hat{\delta}$.

Compaction scheduling. LAM tests each record on arrival, before prefill, so the prefilled prefix survives. Deleting from an existing context instead shifts subsequent positions and invalidates their KV entries. Specifically, re-prefilling a 360K context costs $\approx$538 s in our cost model against $\approx$12 s for the turn's inference. As a result, over a 1,453-turn session that removes the same records either way, deletion in place spends 98.6% of the run on compaction and re-prefill against 0.1% for LAM (Figure 1). The test can also run on the host CPU while the accelerator serves; encoding is charged when embeddings do not already exist.

Evidence retention. We score evidence retention against each SWE-smith (Yang et al., 2025) task's gold patch and count lines accessible only through reference stubs as lost. At a retention floor of 0.999, none of the evaluated observation-masking settings qualifies; SemDeDup removes 19.60% of tokens, compared with LAM's 22.47%. At a 0.99 floor, SemDeDup removes more (§3). LAM's removal rate rises from 0.80% at 1K tokens to 25.58% at 33K. On LHTB (Li et al., 2026), which contains 1,459 terminal-agent trajectories from 28 models under a different scaffold, the near-duplicate rule removes 33.39% of observation tokens, compared with 13.81% on SWE-smith. Near-duplicates are rare in the LoCoMo (Maharana et al., 2024) conversational-memory control.

Contributions.

- **A controllable, interpretable, reproducible compression algorithm for agent memory** (§2.3, §2.4). One knob, the merge threshold, sets the substitution radius δ that bounds the retrieval-score deficit; every deletion names its representative, and no model or randomized clustering is involved, so the deletion set does not move.
- **A memory manager for LAM Algorithm** (§2.5). Records are encoded once on arrival and matched through a band index, so removal precedes prefill, the cached prefix survives, and the test runs on the host CPU at 18.1 ms per turn.
- **A performance model that prices compaction ahead** (§2.6). Its inputs — device anchors, run counters, and whether the schedule forces a re-prefill — are properties of a method, so it prices summarization, masking, or any other compactor on the same axis.
- **A comprehensive evaluation of quality and end-to-end latency** (§3.2, 3.3). LAM removes 22.47% of observation tokens at 99.984% evidence retention, and at a fixed deletion set removing before prefill is worth 71.4×–91.6× end-to-end, separating LAM from masking, summarization, SemDeDup, and LLMLingua-2 by 9.4×–69.1×.

2 DESIGN

LAM combines bounded deduplication (§2.1–2.4), a memory manager that overlaps compaction with inference (§2.5), and a model of deployment cost (§2.6).

2.1 THE PROXIMITY RELATION

Let records be embedded as unit-normalized vectors in $\mathbb{R}^D$. Two records a, b are *ε-equivalent* when $\|a - b\|_2 \leq \delta$. This relation is not transitive, and for unit vectors it is equivalent to a threshold on cosine:

Algorithm 1 The ε error-bounded test for one turn, in which distance checks use full-precision embeddings while the byte codec compresses only the embeddings that are retained.

Input: arrivals $x_1 \ldots x_n$; retained set R and band index I; threshold t; step ε
Output: updated R, I; stubs; measured bound $\hat{\delta}$

$\delta \leftarrow \sqrt{2-2t}$; $\quad \hat{\delta} \leftarrow 0$
for $j = 1$ **to** n **do** ▷ arrival order
  $e_j \leftarrow \textsc{Encode}(x_j)$ ▷ encode each arrival once
  $B \leftarrow \textsc{BandHash}(Q_\varepsilon(e_j))$
  $\mathcal{C} \leftarrow \bigcup_{b \in B} I[b]$ ▷ candidate records
  $y^* \leftarrow \min\{\, y \in \mathcal{C} : \|e_j - e_y\|_2 \le \delta \,\}$ ▷ verify distance; choose lowest index
  if y^* exists **then**
    emit stub $\langle x_j \equiv y^* \rangle$; $\quad \hat{\delta} \leftarrow \max(\hat{\delta},\ \|e_j - e_{y^*}\|_2)$
  else
    $R \leftarrow R \cup \{x_j\}$; $\quad I[b] \leftarrow I[b] \cup \{j\}\ \forall b \in B$; $\quad$ store $\textsc{cuSZp}(e_j)$
return $R,\ I,\ \hat{\delta}$

$$\|a-b\|_2^2 = 2 - 2\cos(a,b), \qquad \text{so merging at } \cos \ge t \iff \|a-b\|_2 \le \delta = \sqrt{2-2t}. \tag{1}$$

We call t the *merge threshold* and δ the *substitution bound*; ε denotes the error budget generically and δ its value under ℓ_2. Each fixes the other, and §2.4 shows the same δ bounds the retrieval-score loss: the radius of the query-free test is the size of the guarantee. We use ℓ_2 because a coordinatewise tolerance $\|a-b\|_\infty \le 2\varepsilon$ would yield a bound proportional to $\sqrt{D}$ instead. The quantization step in Q_ε only proposes candidates and does not enter the bound (Appendix K.1).

2.2 Three layers

L0, byte-exact hashing. This layer is carried over from LLM serving: vLLM hashes fixed-size blocks so a repeated prefix reuses one copy of the KV cache instead of recomputing it (Kwon et al., 2023). We apply the same mechanism one level up, to the text an agent accumulates rather than the cache it produces — xxHash3 over individual lines, repeats replaced by stubs — lines being the granularity that removes the most under exact matching (Appendix B). L0 thus inherits the serving cache's need for an exact match, which L1 removes, and runs after L1 to preserve the blocks near-duplicate matching uses (Figure 4). **L1, ε-equivalence candidates.** We quantize each coordinate with $Q_\varepsilon(x) = \mathrm{round}(x/2\varepsilon)$, divide the vector into bands, and hash each band; records sharing a band digest are candidates and every candidate pair is checked in full-precision ℓ_2. Hashing is $O(N)$ at fixed dimension and band settings, while verification cost scales with the candidate count. Missed candidates change removal but never invalidate the substitution bound on accepted merges. **L2, error-bounded byte compression.** We compress retained embeddings with cuSZp (Huang et al., 2023) under a relative *codec error bound*, separate from the substitution bound. Deduplication and byte compression multiply their storage reductions. Byte compression does not reduce the text in the context (Appendix E.7).

2.3 The ε error-bounded test

LAM applies the redundancy test before each arriving record is prefilled. The test is the bound: an arrival is dropped only when some retained record lies within ℓ_2 distance $\delta = \sqrt{2-2t}$ of it, the radius that equation 1 assigns to the merge threshold t. Every merge is therefore a δ-substitution (Definition 1), and the error it introduces is at most δ for every query rather than on average (Theorem 1); at $\cos \ge 0.95$, $\delta = 0.316$. The operator also reports the largest distance it used, $\hat{\delta} \le \delta$. Algorithm 1 describes the rule.

The index contains only retained records, and decisions are never revised. The rule has three properties. **Prefix stability:** replacing an arriving duplicate with a stub leaves all prefilled tokens in place. Later eviction can invalidate the prefix and is accounted for separately (Appendix F.2). **Single encoding:** each chunk is embedded once. Encoding, which accounts for 99.989% of compaction cost, processes only the arriving tokens Δ rather than the full context C. **Measured bound:** the maximum substitution distance $\hat{\delta}$ is updated with each merge and reported for the turn; the maximum

across turns gives the run's bound. Each dropped chunk is replaced by a reference stub identifying its retained representative. The stub supports later lookup and inspection. We report both gross removal and net removal after the 0.69% input-token cost of stubs.

2.4 THE SUBSTITUTION BOUND

Each merge deletes a record and retains a nearby representative. This can change retrieval scores and rankings (Appendix C). The following bound limits the decrease in the maximum score for any unit query.

Definition 1 (δ-substitution)**.** (R, π) is a *δ-substitution* of S if $R \subseteq S$, $\pi : S \to R$ with $\pi|_R = \mathrm{id}$, and $\|x - \pi(x)\|_2 \leq \delta$ for every $x \in S$.

Theorem 1 (Score deficit)**.** *Let (R, π) be a δ-substitution of S. Then for* every *unit query q,* $0 \leq \max_{x \in S} q \cdot x - \max_{z \in R} q \cdot z \leq \delta$*, and both constants are tight.*

The proof uses Cauchy–Schwarz (Appendix C). Both endpoints are attainable. The bound holds for all unit queries in the same embedding space without assumptions about their score distribution. It also implies a distribution-free rate–distortion bound; Appendix C relates admissible removal to the corpus's covering number.

Scope of the bound. The theorem does not guarantee unchanged rankings or task outcomes. The rank-invariance test in Appendix K.1 requires a sufficient score margin. On LoCoMo, the bound $\delta = 0.3162$ exceeds the median top-1 margin of 0.0185 by 17×; even at $\cos \geq 0.99$, only 0.46% of queries pass the test. At the shipped threshold the bound therefore certifies score perturbation, not rank: it is not a retrieval guarantee, and we do not claim one. We therefore measure retrieval quality and evidence retention separately (§3).

The substitution map. For each deleted x, $\pi(x)$ is the lowest-indexed retained candidate within δ. With fixed embeddings and arrival order, this choice is independent of candidate traversal order. Each substitution therefore satisfies Theorem 1. Direct comparison with a retained representative is necessary because the distance relation is not transitive. At $\cos \geq 0.95$, this rule removes 13.81% of observation tokens.

The measured bound. Each run reports its maximum substitution distance alongside the configured bound: 0.268 versus 0.316 at $\cos \geq 0.95$. The smaller value bounds the substitutions made in that run. Fixed embeddings give deterministic decisions under the tested arithmetic implementation. Across machines, encoder outputs vary slightly: measured cosine differences are 2.4×10^{-7} and the spread in $\hat{\delta}$ is 4.9×10^{-4}. No tested pair crossed the threshold (Appendix D); this is an empirical result, not a guarantee for records arbitrarily close to the threshold.

2.5 SYSTEM DESIGN

The system uses three design choices (Figure 2), the first panel of which follows an arrival through candidate generation by band hashing and exact distance verification. **Placement.** The CPU test runs concurrently with GPU inference. Appendix F.3 measures this configuration and finds little serving interference at the selected thread count. Encoding and byte-codec costs are accounted for separately when required. **Locality.** The band index stores retained records by digest, so each arrival checks its candidates rather than scanning the full context. The byte codec reduces embedding storage; text remains available for reference lookup. **Granularity.** Encoding processes only the turn's new chunks. Search cost still depends on the retained index size, which we measure in Appendix E.11.

2.6 PERFORMANCE MODEL

We estimate deployment cost by dividing a turn into the part that scales with context length C (KV reads during decode, prefill of arrivals against the existing context) and the part that does not (streaming weights once per decode step). Let φ be the share of turn latency in the first part. Eliminating that part gives an Amdahl ceiling of $1/(1 - \varphi)$ for speedup from context reduction. Without overlap, modeled end-to-end time is

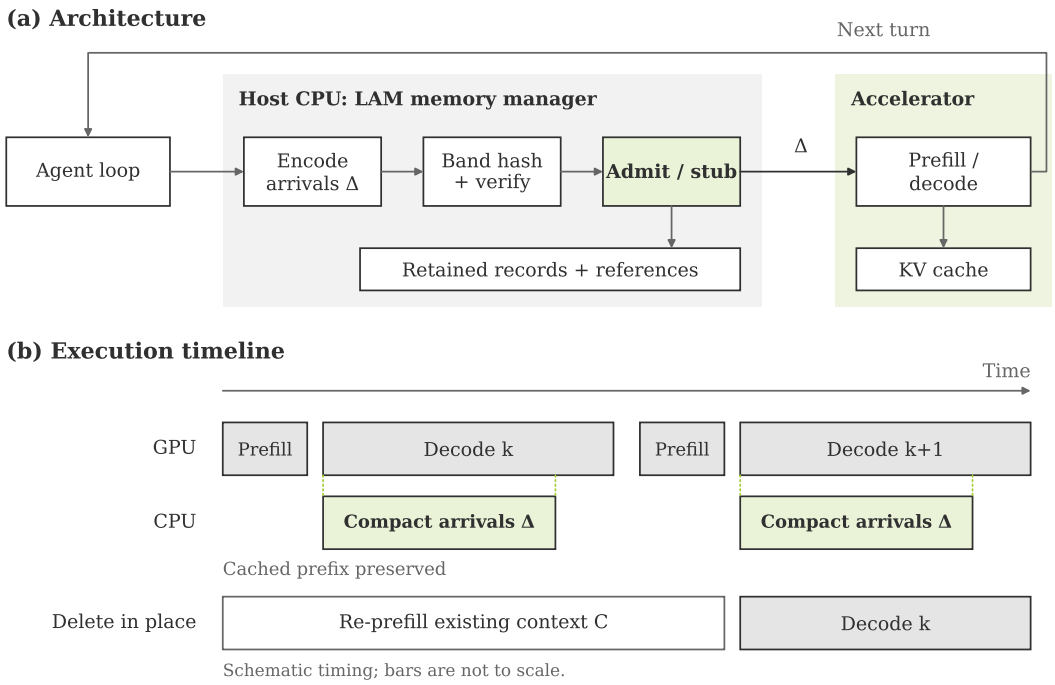

Figure 2: System design, where **(a)** traces one arriving record through the host memory manager to either a retained record or a reference stub, and **(b)** contrasts the two schedules, the upper one voiding the shaded cache suffix that must then be prefilled again and the lower one leaving the prefilled prefix intact.

Table 1: Inputs to equation 2, split into offline terms measured once per device by kernel microbenchmark and online terms read from the run itself.

	symbol	quantity	value / source
offline	R_p, t_d	prefill rate, decode step time	GB10 anchors, Table 25
	R_e	encoder throughput	71,708 tok/s
	R_h	xxHash3	38.2 GB/s
	R_v	exact ℓ_2 verify	13.0 M pairs/s
	R_z	cuSZp	53.3 GB/s
	β	bytes per token	3.68 B/tok
online	C, Δ, G_a	context, arrivals, decoded tokens	637, 39 per turn
	$\lvert\mathcal{C}\rvert$	candidate pairs verified	run
	φ	context-scaling share of a turn	21.6% (b1, 32K), 76.6% (b32)
	$\mathbb{1}[\cdot]$	requires re-prefill	0 test first, 1 delete-in-place

$$t_{\text{e2e}} = \underbrace{\sum_{\text{turns}} \left[\frac{\Delta}{R_p} + G_a\, t_d(C) \right]}_{\text{serving}} + \underbrace{\sum_{\text{events}} \left[\Delta\left(\tfrac{1}{R_e} + \tfrac{1}{R_h} + \tfrac{1}{R_z} \right) + \frac{|\mathcal{C}|}{R_v} \right]}_{\text{compaction}} + \underbrace{\sum_{\text{events}} \frac{C}{R_p}}_{\text{re-prefill}}. \tag{2}$$

Table 1 separates device measurements (Appendix G) from run-dependent inputs, and its indicator is what decides whether a schedule pays re-prefill at all. The context-dependent share φ varies with batch size, context length, and model architecture. At batch 1 and $C = 32$K, $\varphi = 21.6\%$, giving a context-reduction speedup ceiling of 1.28×; at batch 32 the ceiling is 4.28×. These deployment parameters must therefore accompany speedup comparisons. A prefix-preserving test removes the re-prefill term. In the model, deletion from a $C = 360$K context adds 538 s per event to a turn with 12 s of inference. The overlap experiment in Appendix F.3 measures how much of the compaction term can run concurrently with serving.

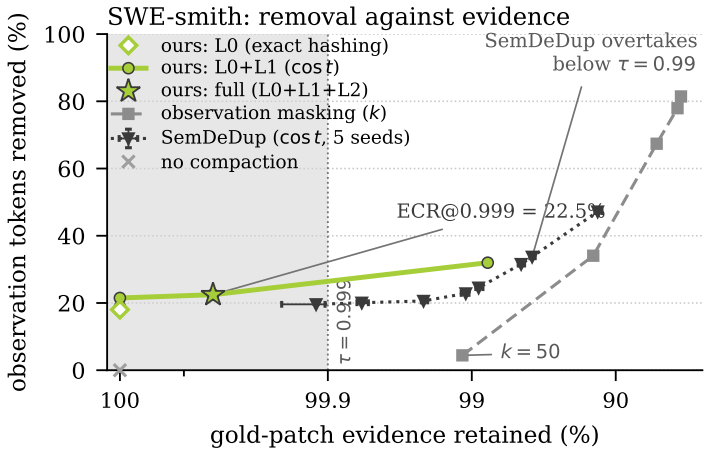

Figure 3: Token removal against evidence retention, one marker per measured setting and up-and-left better, where the three accented series are LAM's cumulative layers — L0, L0+L1, and the full pipeline — and the error bars give SemDeDup's spread across seeds.

Candidate generation. We use standard banded hashing and evaluate its operating points on agent observations (§3.4; Appendix E.10). Full-precision verification preserves the substitution bound at every setting. We report comparison savings at full pair recall so the candidate filter matches exhaustive greedy deduplication. Lower recall can retain additional records; it preserves the substitution bound but need not preserve the packing property behind the removal ceiling.

3 Experiments

3.1 Experimental setup: benchmarks and baselines

Benchmarks. The primary corpus contains 6,026 SWE-smith (Yang et al., 2025) agent trajectories (361,802 records, 136.3M tokens); unless stated otherwise we use a fixed 600-trajectory subset with 15,169 observation records, 27,986 chunks, and 9.07M observation tokens, encoding 512-token windows with `bge-base-en-v1.5` (Xiao et al., 2024) ($D = 768$). The determinism study uses 27,986 of those windows. LHTB (Li et al., 2026) adds 1,459 trajectories from 28 models and 131,357 steps, with roughly an order of magnitude more steps per trajectory (Appendix E.9); LoCoMo (Maharana et al., 2024) provides a conversational-memory control, in which we find only 2 near-duplicate pairs among 17,296,021. Everything runs on two GPU systems: an NVIDIA GB10 for the main measurements and a two-node NVIDIA A100-SXM4-40GB cluster, four NVIDIA A100-SXM4-40GB on its serving node, for the rest.

Baselines. We report LAM in three cumulative configurations: L0 alone, L0+L1, and the full pipeline. L0, the exact-hashing layer inherited from serving-side prefix caching (§2.2), is therefore an ablation of our own design rather than a competing system. Evidence retention is measured by searching for gold-patch lines in the residual text, which applies to deletion methods: observation masking, keeping the last k observations (Lindenbauer et al., 2025), and SemDeDup (Abbas et al., 2023). A relaxed variant also covers token dropping (Appendix E.4), admitting LLMLingua-2 (Pan et al., 2024), the intake compressor of LightMem (Fang et al., 2026) and thus that component rather than the full system. Neither variant assesses paraphrases or latent representations, so MemAgent (Yu et al., 2026), Gisting (Mu et al., 2023), ICAE, and xRAG are compared on latency cost and a Mem0-style deletion prompt separately. Removal ratios are token-weighted. Finally, LAM deduplicates within each trajectory while the main SemDeDup configuration clusters across them (Appendix E.8); we compare at matched removal where applicable.

Table 2: Quality of the residual memory. "Removal" is the share of observation tokens deleted, "evidence" the share of the 6,276 gold-patch lines still present. Resolve-rate Ceiling (RSC) is an upper bound a rerun could attain, rather than a rerun. The shaded row means LAM outperforms others. LLMLingua-2 has no RSC, represented by a dash, due to non-determinism.

method	rule	removal	evidence	RSC
no compaction	—	0.00%	100.000%	75.83%
observation masking ($k = 50$)	recency	4.42%	99.140%	75.17%
LAM, L0 only	exact	18.02%	**100.000%**	**75.83%**
LAM, full ($\cos \geq 0.95$)	δ-substitution	22.47%	**99.984%**	**75.67%**
SemDeDup ($\cos 0.99$)	centroid	24.55%	98.885%	71.53%
LLMLingua-2 (rate 0.80)	token	25.60%	71.383%	—
LAM, full ($\cos \geq 0.90$)	δ-substitution	31.99%	**98.709%**	**70.33%**
SemDeDup ($\cos 0.95$)	centroid	33.74%	97.374%	68.33%
observation masking ($k = 20$)	recency	34.06%	93.021%	66.33%

3.2 Quality

Important Memory Preservation. Evidence-constrained removal, $\mathrm{ECR@}\tau$, is the maximum token removal a method achieves while retaining at least a fraction τ of the 6,276 gold-patch evidence lines across 598 trajectories; lines reachable only through stubs count as lost and L2 compresses stored embeddings rather than context tokens, so it moves neither axis. At the shipped threshold LAM loses 1 line, and at $\tau = 0.999$ it removes 22.47% against 19.60% for SemDeDup and no qualifying masking setting (Figure 3, Table 17); SemDeDup removes more at lower floors, 33.74% versus 31.99% at 0.95.

We selected the 0.999 floor after examining 35 settings from four families (Appendix E.5): there every admissible setting retains all evidence in at least 99.3% of trajectories and the largest ceiling decrease is 0.17 points, against 1.83 at 0.99. SemDeDup's cross-trajectory search contributes to its higher removal: restricting it to each trajectory drops removal from 33.74% to 13.69% at $\cos t = 0.95$. Masking removes 4.42% at $\tau = 0.99$, 5.1× less than LAM. We have extensive experiments in Appendix E.5.

Task-level quality. Retention is a property of the residual memory; what a practitioner asks is whether the agent could still finish the task from it. We therefore score each method by its resolve-rate ceiling (RSC): the share of 600 trajectories that were resolved *and* retain every witnessed evidence line, a bound on what any rerun over the compacted memory could reach rather than a measurement, since an intact trajectory can still fail for reasons unrelated to compaction. Uncompacted the ceiling is 75.83%; LAM gives up 0.16 points of it for 22.47% removal, where SemDeDup falls to 71.53% and masking reaches 66.33% by only 34.06%. Furthermore the separation widens with removal: at 31.99% LAM holds 70.33% against 68.33% for SemDeDup at 33.74%. LLMLingua-2 rewrites tokens instead of deleting records, so no ceiling is defined for it; its 71.383% retention already places it far below every method at comparable removal.

3.3 System performance

With the compressor, corpus, and deletion set fixed, LAM achieves 71.4× speedup at batch 1 and 91.6× at batch 32 from removing records before prefill instead of deleting them from a prefilled context (Figure 1; Appendix F.2). The benefit comes from the schedule, not the rule: any prefix-preserving test receives it, L0 included. Table 3 puts the published methods on that axis, granting each its best measured removal — all but L0 remove more than LAM — and the cheapest schedule its rule permits; uncompacted, the run exhausts the KV cache turn 1,341. They separate by 9.4×–69.1× at batch 1, an ordering that does not track removal — LLMLingua-2 removes least and pays most — because none of the four is prefix-stable, so each pays a re-prefill that LAM's schedule never incurs. The rollouts pay that re-prefill: vLLM serves 97% of LAM's prompt tokens from cache against 48% for masking, 100 re-prefilled tokens per turn against 1,760. Seconds do not follow, because at a mean context of 6,236 tokens compaction is never forced and the shortest prompt wins;

Table 3: End-to-end latency per turn, with the ratio to LAM in parentheses, where "wall" defers compaction to the context budget and "admit" removes before prefill; the shaded row is the shipped configuration, and accented cells are the best in their column.

method	schedule	r_{eff}	batch 1	batch 32
observation masking ($k = 5$)	wall	0.75	107.7 (9.4×)	105.1 (11.8×)
LLM summarization ($\rho = 10$)	wall	0.90	498.0 (43.6×)	495.5 (55.8×)
SemDeDup (global k-means, 0.95)	wall	0.32	647.9 (56.7×)	645.4 (72.7×)
LLMLingua-2 (rate 0.80)	wall	0.24	790.5 (69.1×)	787.9 (88.7×)
LAM, L0 only	admit	0.17	12.0 (1.1×)	9.5 (1.1×)
LAM, full	admit	0.22	**11.4**	**8.9**

Table 4: Merge-threshold sweep, in which $\delta = \sqrt{2 - 2t}$ is what the theorem permits against the $\hat{\delta}$ the run attained, "evidence" is gold-patch retention and "resolve" the resolve-rate ceiling, and the shaded row is the shipped default.

$\cos t$	δ	mean $\hat{\delta}/\delta$	keep	R@10	tokens removed	evidence	resolve
0.99	0.1414	0.350	0.916	0.9978	18.89%	100.000%	75.83%
0.98	0.2000	0.409	0.896	0.9947	19.64%	100.000%	75.83%
0.96	0.2828	0.532	0.856	0.9881	21.53%	100.000%	75.83%
0.95	0.3162	0.559	0.837	0.9846	22.50%	99.984%	75.67%
0.90	0.4472	0.706	0.696	0.9518	32.02%	98.709%	70.33%

Table 5: System-performance breakdown.

	GB10	A100
Bandwidth	237 GB/s	1,365 GB/s
GPU Utilization	0.87	0.88
Achieved Throughput in fp16	90.3 TFLOP/s	288.5 TFLOP/s
Encoder Throughput	71.7 K tok/s	99.9 K tok/s
ratio, bandwidth		5.7×
ratio, encoder		**1.4**×

the rollouts establish the mechanism, not the separation. The extended evaluation of performance model is shown in Appendix F.

3.4 Ablation Study

Algorithm ablation. Table 4 sweeps five merge thresholds over 600 trajectories: no substitution exceeds the configured bound, mean realized displacement being 0.559 of it at the shipped $\cos \geq 0.95$, and retention and the resolve-rate ceiling degrade slowly until 0.90, so we ship 0.95. Layer order matters there because the layers overlap: assuming independent removal overstates their joint rate by 4.0–11.7 points (Table 12), and running L0 first costs 2.45 points at $\cos \geq 0.95$ and 4.08 at 0.90, since line deletion fragments the blocks L1 would match. Thus we run L1 first, leave the base layer last, and configure L2 for the bound rather than for peak ratio (Appendix E.7). Candidate generation keeps the test cheap at full pair recall, examining 31.6% of 283,980 within-trajectory pairs; a band match needs agreement in all r coordinates, so looser thresholds need smaller bands and $r = 32$ reaches only 0.314 recall at 0.95. Furthermore, the yield grows with the horizon: scoring at each input size T only the trajectories reaching it, removal rises from 0.80% at $T = 1K$ to 25.58% at $T = 33K$, so short contexts understate it. The rule is insensitive to arrival order — online and batch residuals are byte-identical on 600/600 trajectories — but not to the encoder, four of which differ in removal by 2.7× and recover 75.8%–90.9% of the default deletion set at matched removal, so thresholds need per-encoder calibration (Appendix B.4). Search is not the bottleneck: at most 112 chunks are retained and search is 0.46% of test time (Appendix E.11).

System performance breakdown. The cost model uses measured, not advertised, device numbers (Table 5, left): advertised A100 bandwidth would overstate it by 14%, and bandwidth differs by 5.7× across the two devices while encoder throughput differs by 1.4×, so encoder cost does not scale with bandwidth. The re-prefill term the schedule avoids also holds on a server: cold prefill of 30,720 tokens takes 1.12 s against 0.09 s once 99.7% is cached, a 12.1× difference; quadratic and linear fits extrapolate to 64 s and 13 s at 360K, beyond the measured range.

Placing the compactor on that serving node costs the server almost nothing (Table 5, right) and keeps up at 4.7× the GPU's output-token rate; the node has 64 cores, so past 16 compactor threads both sides lose. Isolating the compactor does not help: confining it to the two of four NUMA domains furthest from the GPUs costs more bandwidth than it saves the server, so we ship unrestricted placement (Appendix F.3).

4 Related work

Context compression, model-based and model-free. Gisting learns soft tokens for a prefix (Mu et al., 2023), ICAE encodes context into memory slots (Ge et al., 2024), xRAG projects a document embedding into a token (Cheng et al., 2024), and LLMLingua-2 selects text with a token-level classifier (Pan et al., 2024): all require model inference, none bounds substitution error, and reproducibility turns on weights and settings. Model-free, min-hash (Broder, 1997), LSH (Indyk & Motwani, 1998) and SimHash (Charikar, 2002) underlie LAM's banded candidate generation, error-bounded HPC compression (Huang et al., 2023) motivates its distortion control, and SemDeDup clusters embeddings offline (Abbas et al., 2023). LAM instead substitutes under a bound as records arrive: across five seeds only 50.9% of SemDeDup's deletions recur, while LAM's are identical and name their representative (Appendix E.8).

Agent memory and off-path compaction. Mem0 issues LLM ADD/UPDATE/DELETE operations (Chhikara et al., 2025), observation masking retains the last k observations (Lindenbauer et al., 2025), LightMem filters intake and consolidates offline (Fang et al., 2026), MemAgent learns a rewriting policy over fixed memory (Yu et al., 2026), and Omri et al. (2026) characterize memory growth — all without a substitution bound. Slipstream (Chen et al., 2026) and Parallel Context Compaction (Cim et al., 2026) move LLM summarization off the critical path, but rewriting a prefix still forces re-prefill; LAM removes before prefill, so the prefix survives and the test runs on the CPU.

Serving systems, performance models, and rate–distortion. PagedAttention supports block-level KV reuse (Kwon et al., 2023), CachedAttention preserves caches across turns (Gao et al., 2024), Preble schedules for prefix sharing (Srivatsa et al., 2025), and SMetric balances agent sessions (Wang et al., 2026); LAM preserves prefixes and adopts their exact-match hashing as L0. Vidur simulates inference latency (Agrawal et al., 2024) and KVServe models KV compression under service constraints (Liu et al., 2026); LAM's adds the error threshold and compaction and re-prefill cost, while Kolmogorov ϵ-entropy (Kolmogorov & Tikhomirov, 1959; Posner & Rodemich, 1971) frames its removal bounds. Against prompt compression studied in expectation (Nagle et al., 2024), decision-preserving (Zou et al., 2026) and for KV caches (Colaco & Lahjouji, 2026), our worst-case bound implies one for every source on the record set (Corollary 2), complementing evaluations of task success, cost and retrieval (Zheng et al., 2025; Kummer et al., 2026; Johnson & Lee, 2026; Chai et al., 2026).

5 Conclusion

LAM removes redundant agent observations with a query-independent test that bounds the retrieval-score deficit. Because removal precedes prefill the prefix survives, worth 71.4×–91.6× in the cost model — a gain any prefix-preserving rule receives, LAM's exact-hashing floor included. The near-duplicate layer costs 2,883× more per event, yet together they remove 22.47% at 99.984% retention, auditably.

AI USE STATEMENT

Aid or polish writing. We used generative AI tools to polish the manuscript: tightening prose, improving readability, and checking notation and cross-references for consistency. The authors approved the final wording and are responsible for every claim it makes.

Retrieval and discovery. We used the same tools to develop search utilities, to crawl the literature and public code repositories for related systems, and to summarize the candidates returned. They located work; they did not judge it. The authors read each cited paper in the original and checked every claim attributed to it against the primary source.

Execution. The tools assisted with coding, debugging, and code review for the measurement harness, the baselines, and the plotting code. The authors specified what each program had to compute and verified its output against the recorded results. The research question, methodology, experimental design, analyses, and conclusions are the authors', and so are the substitution bound and its corollaries. No reported measurement was generated, imputed, or evaluated by an AI tool.

ETHICS STATEMENT

This work studies agent execution traces from public research datasets. It involves no human subjects or personal data. Deduplication can remove information an agent later needs. Section 3 measures evidence loss; live task success remains unmeasured.

REPRODUCIBILITY STATEMENT

With fixed embeddings, arrival order, and arithmetic implementation, the rule produces deterministic deletion indices and representatives. Choosing the lowest-indexed eligible representative removes dependence on candidate traversal order. Appendix D tests byte-codec repeatability and deletion-set stability across encoder batch sizes and hardware. Encoder outputs are not bit-exact across configurations, but no tested decision crossed the merge threshold. Hardware constants and benchmark sizes are in Appendix G; proofs are in Appendix C and Appendix I. Anonymized code and the measurement harness are provided as supplementary material for review and will be released on acceptance.

REFERENCES

Amro Abbas, Kushal Tirumala, Dániel Simig, Surya Ganguli, and Ari S Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav S. Gulavani, Ramachandran Ramjee, and Alexey Tumanov. Vidur: A large-scale simulation framework for llm inference. In P. Gibbons, G. Pekhimenko, and C. De Sa (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 351–366, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/b74a8de47d2b3c928360e0a011f48351-Paper-Conference.pdf.

Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pp. 21–29, 1997. doi: 10.1109/SEQUEN.1997.666900.

Di Chai, Leye Wang, Zeshen Su, Zhiguo Xia, and Zhihang Yu. KVMem: Virtualizing million-token agent workspaces on a consumer GPU. *arXiv preprint arXiv:2609.04852*, 2026.

Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pp. 380–388, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134959. doi: 10.1145/509907.509965. URL https://doi.org/10.1145/509907.509965.

Zhuofu Chen, Rui Pan, Yinwei Dai, and Ravi Netravali. Slipstream: Trajectory-grounded compaction validation for long-horizon agents. *arXiv preprint arXiv:2605.08580*, 2026.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xRAG: extreme context compression for retrieval-augmented generation with one token. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Musa Cim, Burak Topcu, Chita Das, and Mahmut Taylan Kandemir. Parallel context compaction for long-horizon LLM agent serving. *arXiv preprint arXiv:2605.23296*, 2026.

Ashwin Gerard Colaco and Nada Lahjouji. What to keep, what to forget: A rate–distortion view of memory compaction in llms and agents, 2026. URL https://arxiv.org/abs/2607.08032.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.18866.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-efficient large language model serving for multi-turn conversations with cachedattention. In *Proceedings of the 2024 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC'24, USA, 2024. USENIX Association. ISBN 978-1-939133-41-0.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uREj4ZuGJE.

Yinghan Hou and Zongyou Yang. Control under compression: Reliability frontiers for tool-using agents. *arXiv preprint arXiv:2608.01056*, 2026.

Yafan Huang, Sheng Di, Xiaodong Yu, Guanpeng Li, and Franck Cappello. cuszp: An ultra-fast gpu error-bounded lossy compression framework with optimized end-to-end performance. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701092. doi: 10.1145/3581784.3607048. URL https://doi.org/10.1145/3581784.3607048.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pp. 604–613, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919629. doi: 10.1145/276698.276876. URL https://doi.org/10.1145/276698.276876.

Warren Johnson and Charles Lee. Prompt compression in production task orchestration: A pre-registered randomized trial. *arXiv preprint arXiv:2603.23525*, 2026.

Andrey N. Kolmogorov and Vladimir M. Tikhomirov. ε-entropy and ε-capacity of sets in function spaces. *Uspekhi Matematicheskikh Nauk*, 14(2):3–86, 1959. URL https://www.scilit.com/publications/c6f6a85b58bb3d0c68fc5f5e70621c53. English translation: *American Mathematical Society Translations*, Series 2, vol. 17, pp. 277–364, 1961.

Cornelius Kummer, Lena Jurkschat, Michael Färber, and Sahar Vahdati. Prompt compression in the wild: Measuring latency, rate adherence, and quality for faster llm inference. In *Advances in Information Retrieval: 48th European Conference on Information Retrieval, ECIR 2026, Delft, The Netherlands, March 29 – April 2, 2026, Proceedings, Part I*, pp. 257–271, Berlin, Heidelberg, 2026. Springer-Verlag. ISBN 978-3-032-21288-7. doi: 10.1007/978-3-032-21289-4_17. URL https://doi.org/10.1007/978-3-032-21289-4_17.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, Lichao Sun, Haotao Mi, and Leowei Liang. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading, 2026. URL https://arxiv.org/abs/2607.08964. Trajectories taken from the public leaderboard dump IntelligenceLab/LHTB-leaderboard.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as llm summarization for agent context management. In *4th Workshop on Deep Learning for Code (DL4C), NeurIPS*, 2025. URL https://arxiv.org/abs/2508.21433. arXiv:2508.21433.

Zedong Liu, Xinyang Ma, Dejun Luo, Hairui Zhao, Bing Lu, Wenjing Huang, Yida Gu, Xingchen Liu, Zheng Wei, Jinyang Liu, Dingwen Tao, and Guangming Tan. Kvserve: Service-aware kv cache compression for communication-efficient disaggregated llm serving. In *Proceedings of the ACM SIGCOMM 2026 Conference*, SIGCOMM '26, pp. 1–15, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400724671. doi: 10.1145/3789240.3829139. URL https://doi.org/10.1145/3789240.3829139.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.747/.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Alliot Nagle, Adway Girish, Marco Bondaschi, Michael Gastpar, Ashok Vardhan Makkuva, and Hyeji Kim. Fundamental limits of prompt compression: A rate-distortion framework for black-box language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=TeBKVfhP2M.

Yasmine Omri, Ziyu Gan, Zachary Broveak, Robin Geens, Zexue He, Alex Pentland, Marian Verhelst, Tsachy Weissman, and Thierry Tambe. Agent memory: Characterization and system implications of stateful long-horizon workloads. *arXiv preprint arXiv:2606.06448*, 2026.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.57. URL https://aclanthology.org/2024.findings-acl.57/.

Edward C. Posner and Eugene R. Rodemich. Epsilon entropy and data compression. *The Annals of Mathematical Statistics*, 42(6):2079–2125, 1971. ISSN 00034851, 21688990. URL http://www.jstor.org/stable/2240137.

Sean Quinlan and Sean Dorward. Venti: A new approach to archival data storage. In *Conference on File and Storage Technologies (FAST 02)*, Monterey, CA, January 2002. USENIX Association. URL https://www.usenix.org/conference/fast-02/venti-new-approach-archival-data-storage.

Vikranth Srivatsa, Zijian He, Reyna Abhyankar, Dongming Li, and Yiying Zhang. Preble: Efficient distributed prompt scheduling for LLM serving. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=meKEKDhdnx.

Jiahao Wang, Kaizhan Lin, Kaixi Zhang, Jinbo Han, Xingda Wei, Sijie Shen, Chenguang Fang, Wenyuan Yu, Rong Chen, and Haibo Chen. SMetric: Rethink LLM scheduling for serving agents with balanced session-centric scheduling. *arXiv preprint arXiv:2607.08565*, 2026.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. URL https://arxiv.org/abs/2309.07597. The bge encoder family; we use bge-base-en-v1.5.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2504.21798. arXiv:2504.21798.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=k5nIOvYGCL.

Zhang Zheng, Jinyi Li, Yihuai Lan, Xiang Wang, and Hao Wang. An empirical study on prompt compression for large language models. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=lbFVTPv4s6.

Benjamin Zhu, Kai Li, and Hugo Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. In *6th USENIX Conference on File and Storage Technologies (FAST 08)*, San Jose, CA, February 2008. USENIX Association. URL https://www.usenix.org/conference/fast-08/avoiding-disk-bottleneck-data-domain-deduplication-file-system.

Mingxi Zou, Zhihan Guo, Langzhang Liang, Zhuo Wang, Qifan Wang, Qingsong Wen, Irwin King, Lizhen Qu, and Zenglin Xu. Remember the decision, not the description: A rate-distortion framework for agent memory. *arXiv preprint arXiv:2605.10870*, 2026.

APPENDIX ROADMAP

The appendix contains no new claims. Every section below expands one result already stated in the body: it gives the measurement the body summarizes, the proof the body cites, or the supplementary experiment that tests the same claim under a different corpus, encoder, or setting. Table 6 is the map from the body to the section that expands it.

Table 6: Each appendix section and the main result it expands.

main result	expanded in	what the appendix adds
§2.5 system design	App. A	default configuration, the batch form of the operator, and its equivalence to the online one
§2.2 layer structure	App. B	the full redundancy census, granularity sweep, and encoder sensitivity
§2.4 substitution bound	App. C	proofs, ranking margins, and the rate–distortion corollary
§2.3 online operator	App. D	byte- and deletion-set determinism across runs, batch sizes, and devices
§3.2 retention and removal	App. E	per-floor breakdowns, the ratio ledger, the second corpus (LHTB), and the within-SemDeDup comparison
§3.3 end-to-end latency	App. F	the φ decomposition, the prefix-stability table, the measured re-prefill probe, and CPU/GPU overlap
§2.6 performance model	App. G, App. M	the measured device anchors and the cost class each method falls into

A IMPLEMENTATION DETAILS

This section expands the system design of §2.5: what the shipped configuration is, and what the batch form of the operator does.

Table 7 lists the default configuration. Algorithm 2 applies the rule to a complete trajectory. It processes records in the same order as Algorithm 1 and produces identical residuals on 600/600 tested trajectories (Appendix E).

Table 7: Default configuration. Parameter sweeps are in Appendix E.10 (band width, quantizer step), Appendix E (threshold) and Table 25 (device).

stage	setting
chunking	fixed 512-token windows over tool observations, no overlap
encoder	`bge-base-en-v1.5`, unit-normalized, $D = 768$
threshold	$\cos t = 0.95$, i.e. $\delta = 0.316$ by equation 1
candidates	banded LSH over Q_ε, verified by exact ℓ_2
byte codec	cuSZp, relative error bound 1% of $\max \lvert x \rvert$
stubs	one line per drop, charged as input (0.69%)
devices	NVIDIA GB10 and NVIDIA A100-SXM4-40GB (Table 25)

Both algorithms choose the lowest-indexed eligible representative and process records in trajectory order. They therefore agree for fixed embeddings and candidate settings when no record is evicted. Eviction is handled separately because it can remove a representative and invalidate cached context.

Reproducibility and transfer. The deletion set is byte-identical across six batch sizes and across two machines sharing neither PyTorch version, CUDA version nor GPU architecture (digest `8e788921d75c7e76` at $\cos \geq 0.95$ on both NVIDIA GB10 and NVIDIA A100-SXM4-40GB), where SemDeDup agrees with itself on only 50.9% of its drop set across five seeds. On LHTB the rule removes 33.39% of observation tokens against 13.81% here (Figure 7), showing that near-duplicate removal also applies to a second scaffold.

Algorithm 2 Batch form. Same relation, same first-occurrence witness, applied to a finished trajectory instead of to arrivals.

Input: chunks $x_1 \ldots x_N$ in trajectory order; threshold t; step ε
Output: retained set R; map π; measured bound $\hat{\delta}$

$\delta \leftarrow \sqrt{2-2t}$; $\quad R \leftarrow \emptyset$; $\quad \hat{\delta} \leftarrow 0$
$e_i \leftarrow \textsc{Encode}(x_i)$ for all i ▷ one pass, batched
for $i = 1$ **to** N **do**
  $\mathcal{C} \leftarrow \{\, y \in R : \textsc{BandHash}(Q_\varepsilon(e_y)) \cap \textsc{BandHash}(Q_\varepsilon(e_i)) \neq \emptyset \,\}$
  $y^* \leftarrow \min\{\, y \in \mathcal{C} : \|e_i - e_y\|_2 \leq \delta \,\}$
  if y^* exists **then** $\pi(x_i) \leftarrow y^*$; $\hat{\delta} \leftarrow \max(\hat{\delta}, \|e_i - e_{y^*}\|_2)$
  else $R \leftarrow R \cup \{x_i\}$; $\pi(x_i) \leftarrow x_i$

Table 8: Byte-exact duplicate tokens by granularity, within-trajectory. The line-level figure, not the record- or chunk-level one, is the floor any near-duplicate contribution must be measured above.

granularity	unit	removable
record	one whole tool observation	1.61%
paragraph	blank-line separated block	1.61%
chunk-512	fixed 512-token window	2.61%
line	a single line	**18.14%**

B Redundancy in agent trajectories

This section expands the layer structure of §2.2, measuring how much each layer has to work with before any of them is applied.

We measure exact and near-duplicate observations in the SWE-smith `tool` split: 361,802 records, 6,026 trajectories, and 136.3M tokens. All matches are within a trajectory. We tokenize tool observations with `cl100k_base`.

B.1 L0 granularity: byte-exact duplication

Removal varies by an order of magnitude across granularities (Table 8). Paragraph and whole-record hashing give nearly the same rate because blank lines rarely divide these observations into useful units. Line hashing removes 18.14%, compared with 2.61% for fixed chunks. When an agent rereads an edited file, a few changed lines can alter a whole-record or chunk hash while leaving most line hashes unchanged. We therefore use line hashing for L0.

B.2 Near-duplicate headroom

We embed 512-token chunks with `bge-base-en-v1.5` and apply the greedy rule in Section C.4. On 600 trajectories with 15,169 records, 27,986 chunks, and 9.07M tokens, the near-duplicate layer removes 13.81% at $\cos \geq 0.95$. Since L0 alone removes 18.14% on the full corpus, we evaluate both layers and their composition on a shared subset in Section 3.

Conversational memory. On LoCoMo, a conversational memory benchmark, the same procedure finds 2 near-duplicate pairs out of 17,296,021 ($\approx 10^{-5}$). Redundancy at this level is a property of *agent trajectories*, not of memory benchmarks in general.

Accounting. Removal is measured as $\mathrm{tok}(\text{original}) - \mathrm{tok}(\text{residual})$ over whole texts, never as a sum of per-piece token counts: the tokenizer merges across newlines, so summing the parts overstates the whole by roughly 5%.

B.3 Composition of the two layers and the effect of their order

The two layers can remove overlapping content, so their combined removal must be measured. At $\cos \geq 0.95$, their composition removes 4.47 percentage points more than L0 alone after accounting

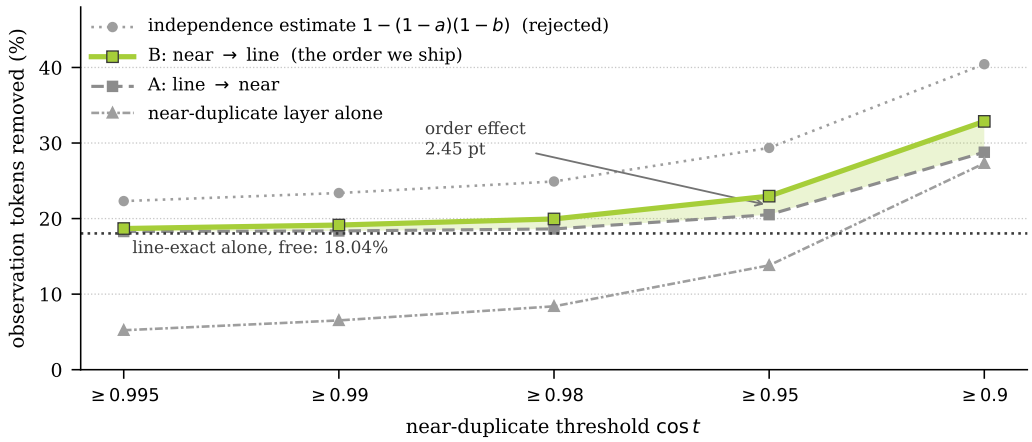

Figure 4: Layer composition over 600 trajectories, weighted by tokens. Running L0 before L1 reduces joint removal by 2.45 percentage points at $\cos \geq 0.95$. Assuming independence overstates removal by 4.0–11.7 points. Net joint removal is 22.50%, or 4.47 points above L0 alone.

for stubs (Figure 4). Individually, L1 removes 13.81% and L0 removes 18.04%. Table 15 reports the additional processing cost.

Layer order. L0 followed by L1 removes 20.52% at $\cos \geq 0.95$; the reverse order removes 22.97%. The difference is 2.45 percentage points and increases to 4.08 at $\cos \geq 0.90$. Removing repeated lines first fragments the chunks L1 matches on. LAM therefore runs L1 before L0. The independence estimate $1 - (1 - a)(1 - b)$ overstates measured joint removal by 4.0–11.7 percentage points.

B.4 Encoder sensitivity

The main evaluation uses one embedding model. To test sensitivity, we repeat the near-duplicate experiment with four encoders on 14,756 chunks from 300 trajectories. This ablation applies L0 first in all arms to isolate incremental near-duplicate removal. The encoders differ in training family (bge, e5, gte) and size: MiniLM-L6 has one-third the width and half the depth of the others. The substitution bound holds in each space; removal and retained evidence depend on the encoder.

At $t = 0.95$, removal ranges from 11.26% to 30.57%, a spread of 2.7×, compared with 13.79% for bge. Merge thresholds are not directly comparable across embedding spaces. At this threshold, e5 and gte recover 99.6–99.7% of bge's deletion set and remove additional chunks. Matching bge's removal rate instead requires thresholds $t^\star = 0.975$ for e5 and gte and 0.935 for MiniLM. The resulting Jaccard similarities are 0.85, 0.76, and 0.65, with deletion-set recall 90.9%, 85.9%, and 75.8%, respectively. The overlap indicates similar redundant content across encoders, but each encoder requires its own threshold calibration.

C A query-independent substitution bound

This section expands the substitution bound of §2.4: the proofs behind Theorem 1, and what the bound does and does not say about ranking.

C.1 Ranking margins

Top-k invariance requires the gap between ranks k and $k+1$ to exceed the relevant score perturbation bound. On 1,536 LoCoMo queries against 5,882 records, the median margin at $k = 3$ is 0.0056, well below the substitution bound at useful merge thresholds (Appendix K.1). We therefore bound the score deficit of a substitution without assuming a ranking margin. Retrieval quality is evaluated separately in Appendix E.2.

C.2 THE BOUND

Definition 1 and Theorem 1 are stated in Section 2; here we prove them and develop what else they imply.

Proof of Theorem 1. The left inequality is immediate from $R \subseteq S$. For the right, let x^* attain the maximum over S. Since $\pi(x^*) \in R$,

$$\max_{z \in R} q \cdot z \;\geq\; q \cdot \pi(x^*) \;=\; q \cdot x^* - q \cdot (x^* - \pi(x^*)) \;\geq\; q \cdot x^* - \|x^* - \pi(x^*)\|_2 \;\geq\; \max_{x \in S} q \cdot x - \delta,$$

using Cauchy–Schwarz and $\|q\|_2 = 1$. For tightness, take $S = \{a, b\}$ with $\|a - b\|_2 = \delta$, $R = \{b\}$, $\pi(a) = b$, and $q = (a - b)/\delta$, a unit vector for which the deficit is exactly δ; hence no smaller constant is valid. Tightness of 0 follows from any π that retains the maximiser. □

Theorem 1 requires no assumption about score distributions or ranking margins.

Corollary 1 (top-k δ-cover)**.** *For every unit q and every k, each member of the true top-k of S is either retained in R or has its representative in R within ℓ_2 distance δ; its score is reproduced to within δ, uniformly over q.*

Proposition 1 (rank-wise deficit)**.** *Let s_j and r_j be the j-th largest scores in S and in R. If π is injective on the true top-k, then $0 \leq s_j - r_j \leq \delta$ for all $j \leq k$.*

Proof. π maps the top-j set to j distinct elements of R, each scoring at least $s_j - \delta$, so $r_j \geq s_j - \delta$; and $r_j \leq s_j$ since $R \subseteq S$. □

If π is not injective on the top-k, two collapsed records $x \neq y$ with $\pi(x) = \pi(y)$ satisfy $\|x - y\|_2 \leq 2\delta$: they are near-duplicates of each other, which is the deletion the method is designed to perform.

The bound read as a rate. Let $X \sim P$ be a random chunk supported on S. A δ-substitution (R, π) is a *deterministic test channel* $X \mapsto \pi(X)$ whose distortion $\|X - \pi(X)\|_2 \leq \delta$ holds almost surely, hence also in expectation, under every P at once.

Corollary 2 (distribution-free rate–distortion bound)**.** *Let (R, π) be a δ-substitution of S, and let $R_P(D)$ be the rate–distortion function of a source $X \sim P$ supported on S under $d(x, \hat{x}) = \|x - \hat{x}\|_2$. Then for* every *such P,*

$$R_P(\delta) \;\leq\; I(X; \pi(X)) \;=\; H(\pi(X)) \;\leq\; \log_2 |R| \quad \textit{bits}, \tag{3}$$

and the same chain holds under the almost-sure distortion constraint.

Because π is deterministic, $H(\pi(X) \mid X) = 0$, and its output entropy is at most $\log_2 |R|$ (Appendix I). The pointwise substitution bound implies equation 3 for every source on S. An expected-distortion bound alone does not imply a guarantee for each query on a finite record set. The same δ here bounds both embedding distortion and retrieval-score deficit. Prior work applies Shannon rate–distortion formulations to prompt compression (Nagle et al., 2024) and agent memory (Zou et al., 2026). We also use metric entropy (Kolmogorov & Tikhomirov, 1959; Posner & Rodemich, 1971) to bound the size of a retained set.

Corollary 3 (measured bracket on the rate and on the δ-entropy)**.** *Let $H_\delta(S) = \log_2 N(S, \delta)$ be the Kolmogorov δ-entropy of S (§C.3) and let $R_P^{\text{as}}(\delta)$ be the rate at distortion δ required* almost surely*. Two runs of greedy first-occurrence, at radius δ and at radius 2δ, bracket both:*

$$\log_2 |R_{\text{greedy}}(2\delta)| \;\leq\; \sup_P R_P^{\text{as}}(\delta) \;\leq\; H_\delta(S) \;\leq\; \log_2 |R_{\text{greedy}}(\delta)|\,. \tag{4}$$

The two greedy passes bound both the worst-case rate and metric entropy; they do not establish equality between them. At $\delta = 0.3162$, the measured interval is $[13.41, 14.52]$ bits, compared with 14.77 for the lossless index. The achieved entropy $H(\pi(X))$ is 14.35, below $\log_2 |R|$ because cluster masses are unequal. At $\delta = 0.1414$, the interval has width 0.09 bits. The interval widens with δ as the second pass merges more records.

From bits to tokens. At the default threshold, the index bound decreases from 14.77 to 14.52 bits, a reduction of 0.25 bits (1.69%), while the same run removes 22.50% of tokens. These quantities differ because index rate is logarithmic in retained-set size, whereas context cost scales with token count. The corresponding token objective is

$$T(\delta) \triangleq \min_{(R,\pi)\ \delta\text{-sub. of } S} \sum_{z\in R} w(z), \qquad w(z) = \text{tokens of } z, \tag{5}$$

a minimum-weight dominating-set problem on the δ-graph of S, which is NP-hard. Coverings of equal size can have different token weights. Theorem 2 therefore bounds chunk counts in Table 9; its token columns report measured removal only.

C.3 A METRIC-ENTROPY CEILING ON REMOVAL

Theorem 1 bounds substitution error. Covering and packing numbers bound how many records a substitution can remove.

Smallest admissible memory. Let $N(S,\delta)$ be the *internal covering number*, the least $|C|$ over $C \subseteq S$ such that every $x \in S$ has some $c \in C$ with $\|x-c\|_2 \le \delta$, and $M(S,\delta)$ the *packing number*, the greatest $|P|$ over $P \subseteq S$ with $\|p-p'\|_2 > \delta$ for distinct p, p'. Unwinding Definition 1, (R,π) is a δ-substitution of S for a suitable map π exactly when R is an internal δ-covering. Thus the maximum admissible removal depends on the geometry of S:

$$\rho^*(\delta) \triangleq \max_{(R,\pi)\ \delta\text{-sub. of } S} 1 - \frac{|R|}{|S|} = 1 - \frac{N(S,\delta)}{|S|}. \tag{6}$$

No compactor allowing error δ under Theorem 1 can remove more than $\rho^*(\delta)$, however it is implemented.

Proposition 2 (greedy is simultaneously a cover and a packing)**.** *Let R be produced by greedy first-occurrence at radius δ (§C.4): chunks arrive in order, and a chunk is appended to R iff it is at distance $> \delta$ from every element already in R. Then $N(S,\delta) \le |R| \le M(S,\delta) \le N(S,\delta/2)$.*

Proof. Cover. A chunk is dropped only when some element of R at that moment is within δ of it, and elements are never removed; so R is an internal δ-covering and $|R| \ge N(S,\delta)$. *Packing.* A chunk enters R only if it is at distance $> \delta$ from every earlier element, so all pairs in R are separated by more than δ and $|R| \le M(S,\delta)$. *Right-hand side.* Let C be a minimal internal $\delta/2$-covering and send each p in a δ-packing to a $c \in C$ within $\delta/2$. Two distinct points sharing a c would be at distance $\le \delta$, contradicting separation, so the map is injective and $M(S,\delta) \le |C| = N(S,\delta/2)$. □

Both properties hold online without knowing S in advance. We verify the packing property directly: at $t = 0.99$ the largest cosine between any two surviving chunks across the corpus is 0.9899, and at every other threshold it likewise falls strictly below t.

Theorem 2 (removal ceiling)**.** *Let P be any 2δ-packing of S. Then $\rho^*(\delta) \le 1 - |P|/|S|$. In particular, running greedy first-occurrence at radius 2δ yields such a P, so*

$$\rho^*(\delta) \le \rho_{\text{greedy}}(2\delta), \tag{7}$$

a ceiling that is computable in one extra pass. *Under the cosine parameterization $\delta = \sqrt{2-2t}$, radius 2δ is the threshold $t' = 4t - 3$.*

Proof. Let C be a minimal internal δ-covering, $|C| = N(S,\delta)$, and map each $p \in P$ to some $c \in C$ with $\|p-c\|_2 \le \delta$. If distinct p, p' shared a c then $\|p-p'\|_2 \le 2\delta$, contradicting the 2δ-packing condition; the map is injective, so $|P| \le N(S,\delta)$ and equation 6 gives the claim. Greedy at radius 2δ returns a 2δ-packing by Proposition 2. For the parameterization, $\sqrt{2-2t'} = 2\sqrt{2-2t}$ gives $t' = 4t - 3$. □

Bounding M by volume is too weak: a δ-packing of the unit sphere in $\mathbb{R}^D$ has $\log_2 M \le D \log_2(1 + 2/\delta)$, which at $D = 768$ and $\delta = 0.316$ permits $\approx 2^{2200}$ survivors. The binding constraint is the intrinsic geometry of what agents observe, and Theorem 2 makes measuring it a one-pass computation.

Table 9: The removal ceiling, measured. For each operating point t we run greedy first-occurrence a second time at $t' = 4t - 3$ (radius 2δ); by Theorem 2 its removal upper-bounds what *any* δ-bounded compactor could achieve here. Cardinality over 27,986 chunks is the bounded quantity. Token columns report observed removal in the two passes and are not upper bounds on token removal.

		chunks removed (cardinality)			tokens removed	
t	δ	attained	ceiling (t')	% of ceiling	at t	at t'
0.99	0.141	8.42%	14.16%	59.48%	6.16%	11.47%
0.95	0.316	15.86%	61.19%	**25.92%**	13.13%	62.55%
0.90	0.447	28.35%	91.86%	30.87%	26.12%	91.63%

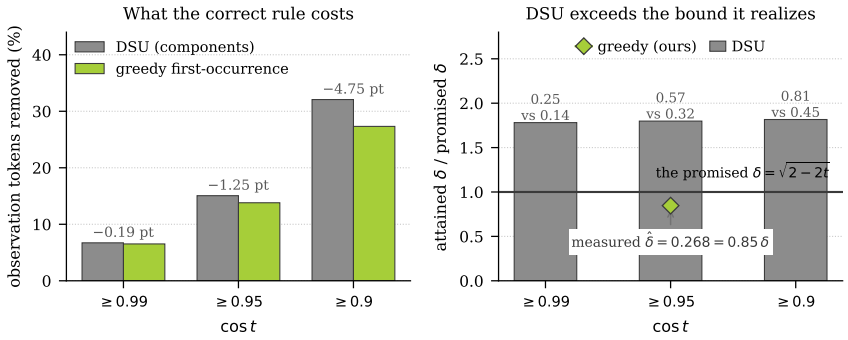

Figure 5: Representative selection and the substitution bound. **Left:** observation-token removal for connected components (DSU) and greedy first-occurrence. **Right:** maximum substitution distance relative to $\delta = \sqrt{2 - 2t}$. Greedy remains within the bound. The diamond marks $\hat{\delta} = 0.268$ at $\cos \geq 0.95$. Results use 600 trajectories and 9.07M observation tokens.

Attained removal. At the default threshold, the near-duplicate layer removes 15.86% of chunks against a ceiling of 61.19%, attaining 25.92% of that upper bound. This is a lower bound on the fraction of optimal removal achieved, since the ceiling may be loose. The corresponding fractions are 59.48% at $t = 0.99$ and 30.87% at $t = 0.90$. They should be compared between methods at a fixed threshold: increasing δ also loosens the second-pass bound. The auxiliary threshold $t' = 4t - 3$ is used only to compute the ceiling and is not a deployment setting.

C.4 Representative selection

The merge relation is not transitive: a can be close to b and b to c without a being close to c. Keeping one record per connected component of the threshold graph therefore does not ensure a δ-substitution. Greedy first-occurrence compares each arrival directly with retained records and accepts only substitutions within δ.

Greedy first-occurrence drops x only when an already retained y lies within δ. This directly satisfies the premise of Theorem 1. Figure 5 compares the removal and maximum substitution distance of this rule with connected-component grouping.

C.5 The measured bound

The pipeline reports $\hat{\delta} = \max_{x:\pi(x)\neq x} \|x - \pi(x)\|_2 \leq \delta$, with value zero when no substitutions occur. It can be computed in $O(|S|D)$ or maintained incrementally as records arrive. Theorem 1 then holds with $\hat{\delta}$ in place of δ. For a particular query, the deficit is also bounded by $\max_i |q \cdot (x_i - \pi(x_i))|$. On the 600 trajectories in Section 3, the realized maximum is 0.268 at a configured 0.316. This bounds embedding substitution and score loss. Evidence retention is evaluated separately in Section E.3.

D DETERMINISM

This section expands the reproducibility claim made for the online operator in §2.3.

We distinguish identical output bytes from identical deletion sets and test both across repeated runs, encoder batch sizes, and hardware configurations.

The pipeline by construction. For fixed embeddings and arithmetic, quantization, band hashing, and distance verification are repeatable, and the lowest-index rule removes dependence on traversal order. Repeated cuSZp runs at relative errors 10^{-3}, 10^{-2}, and 3×10^{-2} give one SHA-256 digest and one output size per setting, including under concurrent GPU load.

The measured bound. Measured cosines vary by 2.4×10^{-7}, which $\delta = \sqrt{2 - 2\cos}$ amplifies near one into a 4.9×10^{-4} spread in $\hat{\delta}$. Online and batch execution nevertheless produce byte-identical deletion sets on 600/600 trajectories.

The encoder. We chunk 15,169 records from 600 trajectories into 27,986 pieces, encode them at six batch sizes ($\{8, 16, 32, 64, 128, 256\}$), run the greedy rule on each embedding matrix, and compare SHA-256 digests. Relative to batch 64, coordinates vary by up to 1.1×10^{-6} and pairwise cosines by 1.8×10^{-7}, yet every threshold yields one digest across all batch sizes: 2,357 chunks deleted at $\cos \geq 0.99$, 4,439 at $\cos \geq 0.95$, Jaccard 1.000000, symmetric difference 0 on NVIDIA GB10.

Decision margins. Over 27,985 decisions the margin $|\max_j \cos(x_i, x_j) - t|$ is at minimum 6.7×10^{-6} at $t = 0.99$ and 8.3×10^{-6} at $t = 0.95$, 37.7× and 46.6× times the measured jitter; the first percentile is 0.0013, the median 0.077, and zero of decisions lie within ten times the jitter. Margins this wide explain the unchanged deletion sets.

SemDeDup seed sensitivity. Holding the 27,986 chunks and their order fixed and varying the k-means seed over five runs with 559 clusters, removal at $\cos \geq 0.99$ is 28.0% with spread 0.14%, but pairwise deletion-set Jaccard is only 0.686 at 0.99 and 0.731 at 0.95: all seeds share 43.7% and 50.9% of the deletion union, and 5,798 chunks change status. Similar removal rates therefore need not imply identical memories.

Cross-hardware. Both machines use the same checksum-verified parquet shard, tokenizer, fp32 CLS pooling, and greedy rule, differing in PyTorch version, CUDA version, and GPU architecture. The same probe run on an NVIDIA A100-SXM4-40GB under a different PyTorch build (2.14.0 against 2.13.0+cu132) produces the same two deletion sets byte for byte: 7,678 chunks dropped at $\cos \geq 0.99$ and 11,727 at $\cos \geq 0.95$, digests `b46a58ac82d22bfe` and `8e788921d75c7e76` on both machines.

LLM memory managers. Submitting a Mem0-style prompt that lists records and requests deletion indices to Qwen2.5-7B-Instruct under vLLM with prefix caching disabled: On twelve LHTB trajectories of 30 records each, ten repeats at temperature 0 with a fixed seed, the DELETE set was identical every time: pairwise Jaccard 1.000, 100% of trajectories returning a single distinct set. Repeating the measurement with the same prompt scheduled against 16 concurrent decoys, so that batch composition varied between repeats, changed nothing. Of twelve trajectories eight yield empty deletion sets, and the four nonempty ones have sizes $\{8, 10, 15, 29\}$. Reproducibility here rests on greedy decoding, a fixed seed, and a pinned model version rather than on embeddings and a merge threshold.

E EXTENDED EVALUATION

This section expands the main results of §3.2. Specifically, it reports the per-floor breakdowns the body summarizes into single numbers, and adds supplementary experiments that repeat the same measurement on a second corpus, a second encoder, and a second baseline configuration.

Table 10: The ε error-bounded test over 600 SWE-smith trajectories, 15,169 observation records, 9.07M tokens, $\cos \geq 0.95$.

quantity	value	note
gross removal	23.19%	L1 followed by L0
net of stubs	22.50%	stubs cost 0.69% of input
chunks encoded	27,986	one encoding per chunk
per-turn latency	18.1 ms mean / 115 ms p99	a turn itself costs 306 ms–3.3 s
measured bound	$\hat{\delta} = 0.268 \leq 0.316$	attained, not configured
online $\equiv$ batch	byte-identical residuals	verified deletion-set equality, 600/600 pass

Table 11: Substitution distances and retrieval quality. Distance summaries are weighted by deleted tokens; $\hat{\delta}$ is the maximum within a run. *Keep* is the retained-chunk fraction. R@10 compares with the full index's top-10 after applying the substitution map.

	substitution distance d_i, token-weighted					retrieval	
$\cos t$	bound δ	median	p99	max	mean$/\delta$	keep	R@10
0.99	0.1414	0.0376	0.1401	0.1412	0.350	0.916	0.9978
0.98	0.2000	0.0820	0.1979	0.1999	0.409	0.896	0.9947
0.95	0.3162	0.2103	0.3148	0.3162	0.559	0.837	0.9846
0.9	0.4472	0.3598	0.4457	0.4472	0.706	0.696	0.9518

Table 12: Layer composition and order, $\cos \geq 0.95$, same denominator. The independence estimate is compared with measured joint removal.

configuration	removal	vs. default
L0 alone (exact hashing)	18.04%	-4.93 pt
L1 alone (near-duplicate)	13.81%	-9.16 pt
A: L0 $\rightarrow$ L1	20.52%	-2.45 pt
B: L1 $\rightarrow$ L0 (default)	22.97%	—
independence estimate (rejected)	29.35%	$+6.38$ pt

Table 13: Per-stage cost of one full compaction pass at trajectory scale.

stage	seconds	% of total
bge-base encode	13.945508	99.989%
Q_ε quantize	0.000138	0.001%
band digests	0.000078	0.001%
exact ℓ_2 verify	0.001201	0.009%
cuSZp byte layer	0.000069	0.000%
total	13.946994	

E.1 The online operator, measured (A1)

Table 10 reports the greedy rule with reference-stub costs included, and online and batch residuals are byte-identical on 600/600 trajectories. Running L0 before L1 costs 2.45 percentage points of joint removal (Table 12), and encoding accounts for nearly all compaction time (Table 13), so reusing existing embeddings is worth far more than optimizing the remaining stages.

E.2 Retrieval quality and the measured bound

Recording $d_i = \|x_i - x_{r(i)}\|_2$ for every substitution on 600 trajectories (27,986 chunks, 9.07M observation tokens), all satisfy the configured bound and the largest approach it at each threshold: at $\cos \geq 0.99$ the mean distance is 0.350 of the bound and 63.2% of merged tokens fall below half of it.

Table 14: Gold-patch safety, 598 trajectories, 6,276 witnessed evidence lines. "Kept" pools over lines (micro retention); "intact" is the fraction of trajectories losing none (macro). The no-compaction and L0 rows are harness checks and must read 100.000%; composing L1 with L0 must score what L1 scores alone.

method	removal	evidence kept	trajectories intact	lines lost
no compaction	0.00%	100.000%	100.0%	0
L0 alone (exact hashing)	18.02%	100.000%	100.0%	0
ours, $\cos \geq 0.99$	18.86%	100.000%	100.0%	0
ours, $\cos \geq 0.96$	21.49%	100.000%	100.0%	0
ours, $\cos \geq 0.95$	22.47%	99.984%	99.8%	1
ours, $\cos \geq 0.90$	31.99%	98.709%	93.0%	81
observation masking, $k = 50$	4.42%	99.140%	98.3%	54
observation masking, $k = 20$	34.06%	93.021%	81.9%	438
observation masking, $k = 10$	67.38%	80.752%	52.2%	1,208
observation masking, $k = 5$	77.94%	73.136%	35.6%	1,686
observation masking, $k = 3$	81.44%	71.606%	32.8%	1,782

Table 15: Ratio ledger, 600 trajectories, 9.07M observation tokens, $\cos \geq 0.95$. Context gain is $1/(1 - r)$; "share of turn" is per-event cost against a turn costing 1.003 s. "Net" charges us for reference stubs.

method	removal	context gain	cost / event	share of turn
L0 alone (xxHash3)	18.04%	1.220×	3.08 μs	3.1×10^{-6}
L1 alone (near-duplicate)	13.81%	1.160×	8.88 ms	0.0089
L1 → L0, gross	23.19%	1.302×	8.88 ms	0.0089
L1 → L0, net	22.50%	1.290×	8.88 ms	0.0089
ours − hashing	+4.47 pt	1.058×	2,883×	—

Retrieval cost. Indexing each trajectory's observation chunks and querying with its own action texts (13,133 queries over 29,322 chunks), the default $\cos \geq 0.95$ keeps 83.7% of chunks at recall@10 0.9846 (recall@1 0.9606, recall@20 0.9928) and $\cos \geq 0.99$ gives 0.9978. At $\cos \geq 0.90$, keeping 69.6%, recall@10 drops to 0.9518 — the threshold at which the resolve-rate ceiling also falls (Section E.6).

E.3 Evidence survival (A4)

Ground truth and criterion. SWE-smith applies a bug patch to a working repository, so the patch's + side is the defect the agent explores and the gold fix is that patch reversed. The patch comes from the task instance, not the misaligned `patch` column of the trajectory row: joining on `instance_id` matches 600 of 600, and the + side appears in 99.2% of instances against 4.2% for the − side. Evidence is the added plus context lines of at least 12 characters, so that `else:` does not count. We score the 6,276 such lines appearing verbatim in some observation over 598 trajectories — compaction cannot delete what the agent never saw — and a line survives if it is findable anywhere in the residual.

At $\cos \geq 0.95$ LAM removes 22.47% of observation tokens and retains 99.984% of measured evidence, losing 1 line in 1 of 598 trajectories, while keeping the last 50 observations removes 5.1× fewer tokens and loses 54× more evidence. At matched removal, masking with $k = 20$ removes 34.06% and retains 93.021% against 31.99% and 98.709% for LAM at $\cos \geq 0.90$. The 454 originally resolved trajectories behave the same (18.61% removal, 99.979% retention).

Stub exposure and specificity. A line survives only if its *content* is readable, never because a stub points at it. At $\cos \geq 0.95$ the near-duplicate layer suppressed 2,005 witnessed lines from the record they arrived in, of which 2,004 remain readable elsewhere and 1 does not. The test is specific: searched in an unrelated trajectory's memory at a fixed offset, only 0.7% of the 6,276 are found. This shows the evidence is still in memory, not that the agent would still produce the patch.

Table 16: Maximum evidence loss allowed by each floor across 35 settings from four families. "Tasks losing evidence" is $1 - \mathrm{macro_all_intact}$. "RSC decrease" is the largest percentage-point decrease from 75.83% among the two families with measured RSC.

floor τ	admissible	evidence loss	binding point	RSC drop	binding point
1.000	7	0.0%	(all tie)	−0.00 pp	(all tie)
0.999	10	0.7%	SemDeDup $\cos 0.9999$	−0.17 pp	ours t=0.95
0.990	16	5.5%	SemDeDup $\cos 0.995$	−1.83 pp	ours t=0.92
0.950	22	12.0%	ours t=0.88	−8.50 pp	ours t=0.88

Adding L1 above L0 buys 4.47 percentage points of net removal at 2,883× times L0's per-event cost (Table 15); what that is worth end to end is modeled in Appendix M.3.

E.4 What the evidence axis includes

Exact substring matching cannot score token-dropping compressors, so we relax it: a witnessed line survives if its alphanumeric tokens occur in order within a window at most 4× times their count. Under the same unrelated-trajectory control, 100.0% of lines match their own trajectory and 1.21% another, against 0.7% for substring matching; that control rate rises from 0.85% without dilation to 1.72% at eightfold, so we use fourfold.

This covers deletion and token dropping, including LLMLingua-2 (Pan et al., 2024), which LightMem uses at intake (Fang et al., 2026); we evaluate LightMem's configured `bert-base-multilingual-cased-meetingbank` checkpoint, not the XLM-R-large one modeled in Appendix M, and not its full consolidation system. Neither criterion reaches the paraphrases or latent representations of MemAgent (Yu et al., 2026), Gisting, ICAE, or xRAG, which need a semantic evaluation of their own.

E.5 The balance: one metric for evidence and tokens (A5)

Removal alone does not measure retained utility — masking with $k = 5$ removes 77.94% of tokens and loses about a quarter of the measured evidence — so rather than weight removal against loss we report maximum removal subject to a retention floor τ:

$$\mathrm{ECR@}\tau \;=\; \max\big\{\, \rho(m) \;:\; m \in \mathcal{M},\; \mathrm{micro}(m) \geq \tau \,\big\}, \tag{8}$$

over a method's own operating points $\mathcal{M}$ (our t, masking's k).

Deriving the floor. Pooled retention hides trajectory-level loss: SemDeDup at $\cos 0.995$ retains 99.092% of lines, yet 5.5% of trajectories lose at least one. Across 35 settings from four families, raising the floor from 0.990 to 0.999 cuts the largest admissible fraction of affected trajectories from 5.5% to 0.7% and the largest RSC decrease from 1.83 to 0.17 percentage points (Table 16); against an uncompacted resolve rate of 75.83% on 455 of 600 trajectories (binomial standard error 1.75 points), those are about one standard error and one-tenth of one. We therefore use $\tau = 0.999$, chosen after examining the data. The transition lies between 99.889%, where the least intact setting (LAM at $t = 0.93$) retains all evidence in 98.997% of trajectories, and 99.917%, the loosest tested floor meeting the 99% requirement.

At $\tau = 0.999$ LAM removes the most of any evaluated setting (Table 17): SemDeDup reaches 19.60% and no masking setting qualifies. At perfect retention LAM removes 21.49%, while SemDeDup's tightest tested threshold, $\cos 0.9999$, retains 99.917% and does not qualify. At lower floors SemDeDup removes more: 22.84% against 22.47% at 0.99 and 33.74% against 31.99% at 0.95, although it clusters across trajectories here (Section E.8 restricts it to one). Which family is preferable depends on the required retention; only LAM carries a substitution bound and admits testing before prefill.

Table 17: Removal and evidence in one place, same corpus and denominator. ECR@τ is the most a family can remove while retaining $\geq \tau$ of the gold-patch evidence lines, maximized over its own operating points; "best" drops the constraint. Shading marks the column winner. SemDeDup is swept to $\cos 0.9999$, tighter than our own family.

	best	ECR@τ			
method	(unconstrained)	$\tau = 1.000$	0.999	0.990	0.950
L0 alone	18.02%	18.02%	18.02%	18.02%	18.02%
near-duplicate layer alone	26.09%	11.42%	13.08%	13.08%	26.09%
observation masking	81.44%	—	—	4.42%	4.42%
SemDeDup, 5 seeds	47.19%	—	19.60%	**22.84%**	**33.74%**
ours: near → line	31.99%	**21.49%**	**22.47%**	22.47%	31.99%

Table 18: Net reduction $\eta(\kappa)$ from equation 9, assuming κ repair turns per lost evidence line. Break-even values show when repair costs offset additional removal.

	ρ	$\eta(1)$	$\eta(3)$	$\eta(8)$	break-even κ vs. ours
ours, $\cos \geq 0.95$	0.2247	0.2246	0.2245	0.2241	—
ours, $\cos \geq 0.90$	0.3199	0.3141	0.3027	0.2741	16.9
masking, k=20	0.3406	0.3097	0.2478	0.0932	**3.8**
masking, k=10	0.6738	0.5884	0.4178	−0.0087	**5.3**
masking, k=5	0.7794	0.6603	0.4222	−0.1732	**4.7**

Table 19: Budget-feasibility rate BFR@B: the fraction of 600 trajectories whose observation stream fits under B tokens. Necessary, not sufficient.

	B=8K	B=16K	B=32K	B=64K
uncompacted	22.0%	67.2%	93.2%	99.3%
ours, $\cos \geq 0.99$	23.0%	71.5%	94.2%	99.8%
ours, $\cos \geq 0.95$	25.2%	75.0%	96.7%	100.0%
ours, $\cos \geq 0.90$	33.0%	83.7%	99.3%	100.0%

Turning loss into cost. Charging lost evidence κ repair turns at $\Delta = 636.8$ measured tokens each (§F), the *net* reduction is

$$\eta(\kappa) \;=\; \rho \;-\; \frac{\text{repairs} \cdot \kappa \cdot \Delta}{T}, \qquad T = 9{,}017{,}033 \text{ observation tokens}, \tag{9}$$

and we sweep κ rather than choose it.

Masking keeps a net-token advantage while repair is cheap, breaking even against LAM at around four repair turns per lost line depending on k and turning negative near $\kappa = 6.5$, or $\kappa = 15$ charging one repair per affected trajectory (Table 18). LAM at $\cos \geq 0.95$ loses one measured line, so the sweep barely moves its net reduction.

Cost in tokens and dollars. Removing 22.47% of 9,017,033 observation tokens is 3.39M context tokens per 1,000 trajectories, or $3.39p$ at p per million input tokens. That under-counts: without prompt caching a context token is re-read on every later turn, so at 26.28 turns per trajectory the saving is amplified by $\approx n/2 = 13.1\times$. We report the multiplier rather than a dollar figure, which would need a cache policy and a price list. Johnson & Lee (2026) make the same point from the other side: aggressive compression *increased* total cost by 1.8% despite large input reduction, because output tokens expanded and are priced higher. We do not measure output length.

Budget feasibility. BFR@B is the fraction of recorded observation streams fitting within B tokens after compaction — storage feasibility, not task success. The default setting raises it from 67.2% to 75.0% at $B = 16$K and from 93.2% to 96.7% at 32K; at 64K nearly all streams already fit.

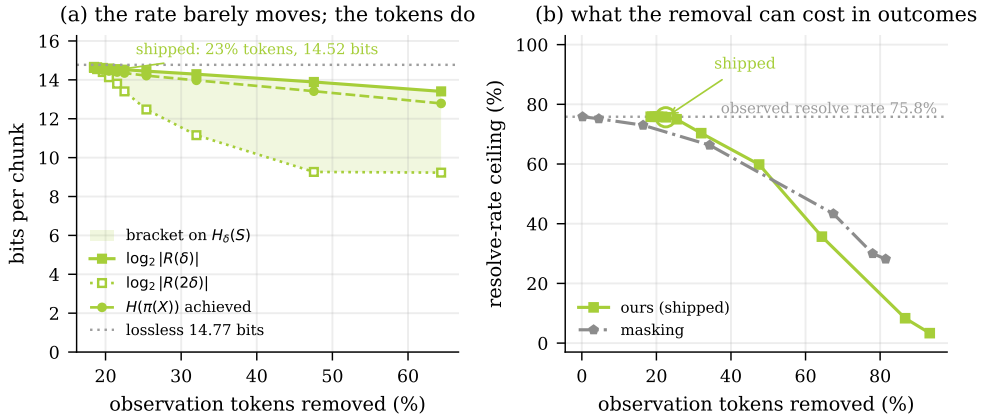

Figure 6: **(a)** Metric-entropy bounds from Corollary 3 and observation-token removal. The upper and lower edges use greedy passes at radii δ and 2δ. Index rate depends logarithmically on retained-set size, while context cost depends on token count. **(b)** Conditional resolve-rate ceiling (RSC, equation 10) versus token removal for LAM and observation masking on 600 trajectories, of which 455 were originally resolved. RSC remains near baseline through 22.50% removal and decreases at more aggressive settings.

E.6 Rate, tokens, and the outcome ceiling

We measure both halves of Corollary 2 on 600 trajectories over a dense sweep of δ. These were recorded *without* compaction, so a compacted rerun is unobservable, but each carries a `resolved` outcome (455 of 600, rate 75.83%); crossing that with per-trajectory evidence survival gives the *resolve-rate ceiling*

$$\mathrm{RSC}(\delta) \triangleq \frac{|\{\text{ resolved trajectories retaining } \textit{every} \text{ witnessed evidence line under } \delta \}|}{600}, \tag{10}$$

which bounds live task success only if every witnessed line is necessary, lost evidence cannot be reacquired, and compaction solves no previously unresolved task; otherwise it is an evidence-based proxy. The harness checks that the sweep is non-increasing and that $\mathrm{RSC}(0)$ matches the recorded resolve rate.

Against a lossless index of 14.77 bits per chunk, at $\delta = 0.3162$ the bound gives $[13.41, 14.52]$ bits and the achieved entropy is 14.35, a 1.69% reduction, while the same run removes 22.50% of observation tokens (Figure 6a) and RSC falls 0.17 percentage points, from 75.83% to 75.67%. At similar removal masking with $k = 20$ removes 34.28% at RSC 66.33% against 32.02% and 70.33% for LAM at $\cos \geq 0.90$; $k = 50$ removes 4.56% at 75.17%, and LAM at $\cos \geq 0.80$ removes 64.38% at 35.67% (Table 26). Hou & Yang (2026) report a similarly sharp task-success decrease under aggressive control-context compression, on measurements not directly comparable with ours.

E.7 Composite reduction: deduplication × byte codec

Deduplication shrinks the number of vectors and the codec shrinks each survivor, so the two multiply on the index axis but *not* on the token axis: the codec compresses embeddings, not the text the agent carries forward, so none of these ratios is a context-length reduction. Crossing five-plus merge thresholds with the integrated byte codecs (`fp16`, per-vector `int8`, sign-binary, cuSZp at several relative error bounds) gives 84 combinations on one 5,882-vector index, scored against the uncompressed, undeduplicated ground truth (recall@10 0.6536).

Composite storage reduction ranges from 4.63× to 42.88× and recall@10 from 0.6543 to 0.6204, but at the most compressed setting retrieved-set Jaccard is 0.560: recall alone conceals a substantial change in which items come back. Because the substitution bound does not cover byte-codec error we select $\cos \geq 0.95$ with the conservative codec — 1.188× × 3.90× = 4.63× at recall 0.6543 and Jaccard 0.990 — over the higher-reduction 9.11× at recall 0.6491 and Jaccard 0.903.

Table 20: Composite storage reduction and recall@10, including the selected setting. Reduction multiplies deduplication and codec ratios on the embedding index. The full-precision, full-size index has recall 0.6536; small differences around it should not be interpreted as quality gains.

$\cos t$	byte codec	dedup	codec	composite	recall@10
0.9	binary (sign)	1.396×	30.72×	42.88×	0.6204
0.9	cuSZp/fixed, $\varepsilon_b = 0.03$	1.396×	9.46×	13.20×	0.6354
0.9	cuSZp/fixed, $\varepsilon_b = 0.01$	1.396×	6.53×	9.11×	0.6491
0.9	cuSZp/fixed, $\varepsilon_b = 0.003$	1.396×	4.94×	6.90×	0.6549
0.9	int8 (per-vector)	1.396×	3.96×	5.53×	0.6556
operating point we ship					
0.95	cuSZp/fixed, $\varepsilon_b = 0.001$	1.188×	3.90×	4.63×	0.6543

Table 21: Near-duplicate rate on LHTB, token-weighted, within trajectory, against the SWE-bench observation rate from Table 15. Same encoder, window and rule.

	LHTB		SWE-bench	
$\cos t$	all context	observations	observations	ratio
L0 (byte-exact)	2.08%	3.35%	—	—
0.999	6.28%	8.07%	—	—
0.99	14.44%	16.78%	6.52%	2.6×
0.98	18.71%	20.54%	8.39%	2.4×
0.95	33.07%	33.39%	13.81%	2.4×
0.9	58.72%	56.72%	27.32%	2.1×

E.8 SemDeDup matched on reach

The main SemDeDup baseline clusters the full corpus and so may match across trajectories, while LAM searches only the current one. Rerunning it per trajectory, with k scaled to hold chunks per cluster fixed and the combined deletion sets scored by the same criterion, removal falls to 13.69% at $\cos t = 0.95$ and 6.45% at 0.99, against 33.74% and 24.55% corpus-wide — a factor of 2.46× at 0.95.

At matched scope SemDeDup removes 13.69% retaining 99.939% of evidence, against 13.81% and 99.984% for LAM's L1 alone, rising to 22.47% once composed with L0; SemDeDup could be composed with an exact layer too. Its within-trajectory removal varies by 0.03 pts across five seeds, less than in the global experiment, though equal rates do not imply equal deletion sets.

E.9 A second corpus: long-horizon terminal agents

LHTB (Long-Horizon Terminal-Bench) (Li et al., 2026) contributes a different scaffold, task distribution, and model set, with longer trajectories than SWE-smith: 1,459 trajectories from 28 models over 131,357 steps (median 41, 90th percentile 230, maximum 1,074) and 12.1B cumulative prompt tokens. The same encoder, 512-token windows, and arrival-order greedy rule run over 400 trajectories: 138,652 windows, 64,510 of them observations, 22.0M observation tokens. Each step carries an agent message, a tool call, and an observation, so Table 21 separates observation-only from whole-context removal.

At $\cos t = 0.95$ LAM removes 33.39% of observation tokens against 13.81% on SWE-smith, a factor of 2.4×, and at 0.99 16.78% against 6.52%, a factor of 2.6×; whole-context removal is close to observation-only at 0.95 (33.07% and 33.39%). Exact record duplication accounts for 3.35%, at a different granularity from the line-level L0 hashing reported on SWE-smith. The experiment measures redundancy, not task success under compaction.

Trajectories come from the public leaderboard dump `IntelligenceLab/LHTB-leaderboard` on the Hugging Face Hub, pinned at commit `c999a3b`. The dump accumulates community submissions on top of the baselines the LHTB paper itself reports, which is why it spans 28 models rather than the 17 evaluated there; we read every `agent/trajectory.json`

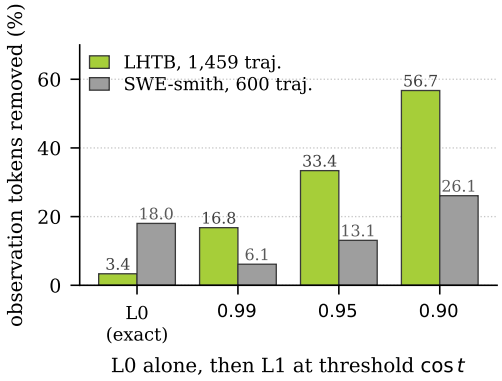

Figure 7: The removal axis on a second scaffold. LHTB provides no gold patches, so only removal is plotted. L0 is record-level there and line-level on SWE-smith.

under `submissions/long-horizon-terminal-bench/1.0/` and take the harness and model from the submission directory name.

E.10 Band width and quantizer step: the full grid

We sweep band width $r \in \{2,4,8,16,32\}$ and quantizer step $\varepsilon = \mathrm{rel} \cdot \sigma$ for $\mathrm{rel} \in 0.01 - -0.3$ on 5,011 observation records from 200 trajectories (283,980 within-trajectory pairs). Recall against the exhaustive $\cos \geq t$ pair set has to be full to reproduce exhaustive greedy removal; lower recall still preserves the substitution bound.

At the default $\cos \geq 0.95$, full recall requires $r = 4$ and $\mathrm{rel} = 0.2$, examining 31.6% of pairs for a 3.2× reduction, while accepting recall 0.979 at $r = 8$, $\mathrm{rel} = 0.3$ gives 26×. Since a band match requires agreement in all its coordinates, every threshold needs its own setting: the best $\cos \geq 0.99$ configuration is $r = 8$, $\mathrm{rel} = 0.2$ at 1.46% of pairs (68×, precision 0.565). Appendix K gives the grid.

E.11 Brute-force scan crossover

Exact search costs scale linearly in retained embeddings, and within-trajectory indexing confines it to one agent's history: replaying 300 trajectories gives median 36 survivors and maximum 112, occupying 344 KB, with search taking 44 μs of the 16.4 ms call (0.46% of elapsed time).

A synthetic index locates the crossover with encoding cost. Exact search is a matrix–vector product: at a million survivors it reaches 229 GB/s, 97% of measured triad bandwidth (Appendix G), costing 13.4 μs per thousand survivors, and matches encoding cost only at 1.2 M survivors — 10,000× the largest observed index and 3.8 GB of embedding storage. Bounding index growth therefore keeps these workloads clear of approximate search.

F Performance model

This section expands the end-to-end latency result of §3.3 and the model behind it in §2.6. Specifically, Appendix M charges each compaction method for inference, compression, and token removal; here we model end-to-end latency and evaluate scheduling and CPU placement.

F.1 The φ decomposition (P1)

Split one agent turn into the part scaling with context length C — KV reads during decode, prefill of arriving tokens against the existing context — and the part that does not, dominated by streaming the weights once per decode step. With φ the first part's share of turn latency, context reduction cannot beat the Amdahl ceiling $1/(1-\varphi)$.

Table 22: φ, the context-scaling share of one turn, and the ceiling it puts on *any* compactor. Llama-3.1-8B on GB10, $\Delta = 637$ arriving tokens and $G_a = 39$ decoded tokens per turn (measured). Weight reads amortize across a decode batch; KV reads do not.

	$C = 3$K	$C = 8$K	$C = 32$K	$C = 128$K
batch 1	2.5% (1.03×)	6.4% (1.07×)	21.6% (1.28×)	52.4% (2.10×)
batch 32	23.5% (1.31×)	45.0% (1.82×)	76.6% (4.28×)	92.9% (14.1×)

Table 23: Modeled *mean latency per turn* with the same compressor, corpus, and deletion set. Only the schedule differs. Delete-in-place prefills the arriving tokens and then removes duplicates; the ε error-bounded rule removes before prefill. Llama-3.1-8B, one growth-regime session of 1,453 turns at $B_{ctx} = 32$K; the turn shape it replays ($\Delta = 637$ arriving tokens, 94% of it observation) is measured on 600 SWE-smith trajectories. These are modeled, not wall-clock, times.

policy	batch 1 (s/turn)	batch 32 (s/turn)	% of total
delete in place (prefill, then remove)	816.6	814.0	98.6%
ε error-bounded (remove, then prefill)	11.4	8.9	0.1%
policy gain	**71.4**×	**91.6**×	

At batch 1 and a 32K context $\varphi = 21.6\%$, capping context-reduction speedup at 1.28×; at batch 32 the same context gives $\varphi = 76.6\%$ and 4.28× (Table 22). Batch size, context length, and model architecture therefore move the speedup on their own, before any compactor acts.

Accounting behind Table 3. The end-to-end table in §3.3 runs the loop above for 1,453 turns in the growth regime at $B_{ctx} = 32$K, charging each method its own measured removal and the least expensive schedule its rule permits. Specifically, a method that is not prefix-stable can still defer compaction to the budget wall instead of deleting eagerly, and that cheaper option is what it is charged; only LAM and L0 admit. Furthermore the ordering there does not follow removal, because what each method pays is set by how often its rule forces a re-prefill rather than by how much it removes. Compaction latency itself is a separate and much smaller axis (Appendix M). Without compaction the run exhausts the KV cache at turn 1,341 of 1,453.

End-to-end accounting. Without overlap, total latency includes compaction and inference on the retained context:

$$t_{e2e} = \sum_{\text{turns}} [\text{prefill} + \text{decode}] + \sum_{\text{events}} [\text{compact} + \text{re-prefill}] .$$

Over 600 measured trajectories a turn adds $\Delta = 637$ tokens, 94% of it tool observation, while the agent *decodes* only $G_a = 39$: most arriving tokens are eligible for compaction, and what their removal is worth depends on φ.

F.2 Where the latency is: prefix-stability (S1)

We hold the compressor, corpus, and deletion set fixed and vary only when records are tested (Table 23). Deleting a record from an existing context invalidates the KV entries after it, and rebuilding a 360K-token cache costs ≈538 s in the model against ≈12 s for the turn's inference. Deletion in place therefore spends 98.6% of modeled runtime on compaction and re-prefill, 816.6 s of the average turn; testing before prefill preserves the cache and cuts that to 0.1% and 11.4 s, for 71.4× end to end at batch 1 and 91.6× at batch 32.

The re-prefill term, measured. Since 538 comes from the cost model, we also measure prefill directly for Qwen2.5-7B-Instruct under vLLM on four NVIDIA A100-SXM4-40GB with prefix caching enabled, issuing a salted prompt and then repeating it unchanged at each context length up to 30,720 tokens. The server reports 99.7% of the repeat as cached, and at the longest context cold and cached prefill take 1.12 s and 0.09 s, a 12.1× difference — independent confirmation that preserving the cached prefix cuts prefill cost. The cost is also nonlinear in context length: the effective rate

peaks at 36.9 K tok/s near 4K tokens and falls to 27.3 K tok/s at 30,720, and $t = c + C/r + kC^2$ fits better than a line, predicting 64 s at 360K tokens against 13 s for the linear fit (4.8× apart). Both extrapolations lie outside the measured range. They also sit 8.4× below the 538 s the cost model charges, because the two are different stacks: the model is configured for Llama-3.1-8B on one NVIDIA GB10 at 237 GB/s of achieved bandwidth, while this measurement runs Qwen2.5-7B-Instruct on four NVIDIA A100-SXM4-40GB at 1,365 GB/s each. Aggregate bandwidth alone separates them by 23×, so the model's figure is below what hardware scaling would predict rather than above it. Neither stack is measured at 360K tokens, and the headline rests on the model's.

How much a deletion invalidates. Deleting in place is not a competing system; it is the same operator on a different schedule, and the cost it pays turns on how much of the KV cache a deletion voids. That is measurable rather than assumed. A deletion invalidates every entry at or after the earliest deleted position, and a duplicate by definition repeats something earlier, so the first one arrives early: across 300 trajectories the median first duplicate sits 4.8% of the way in, and the invalidated share of context is 0.94 on average and 0.98 at the median, with 76% of trajectories above 0.95. Measured on exact hashing, the least aggressive layer, so these are lower bounds. Charging the whole context is therefore close to what the schedule actually costs; recomputing at the measured mean instead gives 63.5×–81.5× rather than 71.4×–91.6×.

The three factors together. In the evaluated regime the removal L1 adds over L0 is worth 1.07×, kernel optimization almost nothing because encoding dominates compaction time, and avoiding re-prefill 71.4×–91.6×. That last gain goes to any prefix-preserving rule, L0 included; summarization rewrites existing context and must rebuild the affected cache entries instead.

Index footprint. The index holds one fp32 embedding per retained chunk and one 64-bit hash per distinct line. Across 600 trajectories at $\cos \geq 0.95$ it is 0.128 MB on average, 0.34 MB at the 99th percentile and 0.44 MB at maximum, or 8.5 KB per 1K observation tokens, of which embeddings are 94%. A log–log fit of bytes against tokens gives an exponent of 0.71, sublinear over the measured range. The index is 0.015% of the corresponding 867 MB KV cache for Qwen2.5-7B-Instruct.

F.3 Where the compactor runs

The test can run on the CPU while the GPU serves the model, over embeddings that are already available. We measure whether that overlap costs serving throughput or latency.

Setup. Two nodes of the NVIDIA A100-SXM4-40GB system: node 1 serves Qwen2.5-7B-Instruct under vLLM across four NVIDIA A100-SXM4-40GB at tensor parallelism 4 and runs the compactor on its 64 CPU cores, node 2 runs the load generator. Prefix caching is disabled so that reuse of the shared prompt prefix cannot hide the interference. The five conditions are the server alone (A), the compactor alone (B), both overlapped under saturating load (C) and at a fixed arrival rate of 5 req/s (D), and (D) with the compactor confined by `numactl` to the two of four NUMA domains furthest from the GPUs (E). Compactor rates cover the whole benchmark window, so idle time counts against them.

Throughput and latency under overlap. At the fixed arrival rate, output throughput is 1.0000× of baseline with 8 compactor threads and 0.9999× with 16; median TTFT moves from 157.0 ms to 166.5 ms and median inter-token latency by 1.0027×. The compactor processes 4,647 context tokens per second against 5,001 alone, 0.929 of its standalone throughput. Under saturating load, serving throughput is 0.9990×, TTFT 1.0004×, and the compactor 0.907 of standalone. At 32 threads TTFT rises to 192.5 ms, inter-token latency to 1.1585×, and compactor throughput falls to 0.620 — CPU oversubscription, with compaction sharing cores with server tokenization and scheduling.

NUMA placement. Confining compaction to the two NUMA domains furthest from the GPUs gives server throughput 0.9966× and compactor throughput 0.540 of standalone: unrestricted placement wins, since pinning costs the compactor memory bandwidth.

Consequences for the cost model. At the selected setting the CPU processes context at roughly 4.7× the GPU's output-token rate, so this workload overlaps with serving at little throughput cost,

Table 24: CPU compaction overlapped with GPU serving on two NVIDIA A100-SXM4-40GB nodes. At 32 threads the compactor contends with the server's own CPU work and both sides lose.

compactor	GPU server			CPU compactor	
threads	out tok/s	TTFT p50	TPOT p50	ctx tok/s	vs. solo
fixed arrival rate, 5 req/s — server unsaturated					
none	990.8	157.0 ms	11.81 ms	—	—
8	990.8	164.7 ms	11.84 ms	2,937	0.966
16	990.7	166.5 ms	11.84 ms	4,647	0.929
32	989.0	192.5 ms	13.68 ms	3,809	0.620
saturating load — server queue-bound					
none	1267.9	100.3 s	77.18 ms	—	—
8	1266.2	100.4 s	77.26 ms	2,908	0.946
16	1266.6	100.3 s	77.25 ms	4,536	0.907
32	1258.5	101.4 s	77.34 ms	3,747	0.610
16, pinned	1263.6	100.7 s	77.32 ms	2,703	0.540

and overlap reduces what compaction contributes to elapsed time below what the sequential model charges. The result assumes spare CPU capacity and excludes GPU encoding when embeddings must be generated.

G Measured hardware constants

This section supplies the device constants the performance model of §2.6 consumes.

The paper uses two GPU systems throughout, an NVIDIA GB10 and an NVIDIA A100-SXM4-40GB; Table 5 compares them. The constants below are measured on a single GB10 (Grace Blackwell, 48 SM). Eight of the nine rows are kernel microbenchmarks at sizes large enough to reach steady state; only β comes from real data.

Table 25: Hardware anchors and the benchmark used to obtain each.

quantity	measured	benchmark
streaming bandwidth	237.4 GB/s	triad $c = a + b$, 3×512 MB fp32; $\mu_m = 0.869$ of the 273 GB/s spec
fp16 tensor matmul	90.3 TFLOP/s	8192^3 GEMM
machine balance	380 FLOP/byte	derived F/B; everything below is bandwidth-bound
bge-base encode	71,708 tok/s	512 *synthetic* docs $\times$ 512 tok = 262K tok, batch 64
Q_ε quantize	65.4 GB/s	2^{26} fp32 $\rightarrow$ int16, 6 B/element moved; unfused PyTorch
xxHash3 (CPU)	38.2 GB/s	64 MB buffer, single thread
exact ℓ_2 verify	13.0 M pairs/s	65,536 candidate pairs at $D = 768$
cuSZp	53.3 GB/s	2^{24} fp32, error bound 1% of $\max \lvert x \rvert$; ratio 5.18$\times$
bytes per token β	3.68 B/tok	SWE-smith `tool` split, 200 trajectories, observation records

H Compaction latency on a second device

I Deferred proofs

Corollary 2 (distribution-free rate–distortion bound)

Proof. Let (R, π) be a δ-substitution of S and let $X \sim P$ be supported on S. Because π is a deterministic function of X we have $H(\pi(X) \mid X) = 0$, so

$$I(X; \pi(X)) \;=\; H(\pi(X)) - H(\pi(X) \mid X) \;=\; H(\pi(X)) \;\leq\; \log_2 |R|,$$

the last step because $\pi(X)$ takes at most $|R|$ values. By Definition 1, $\|x - \pi(x)\|_2 \leq \delta$ for *every* $x \in S$, hence $\|X - \pi(X)\|_2 \leq \delta$ almost surely and, taking expectations, $\mathbb{E}\|X - \pi(X)\|_2 \leq \delta$. The

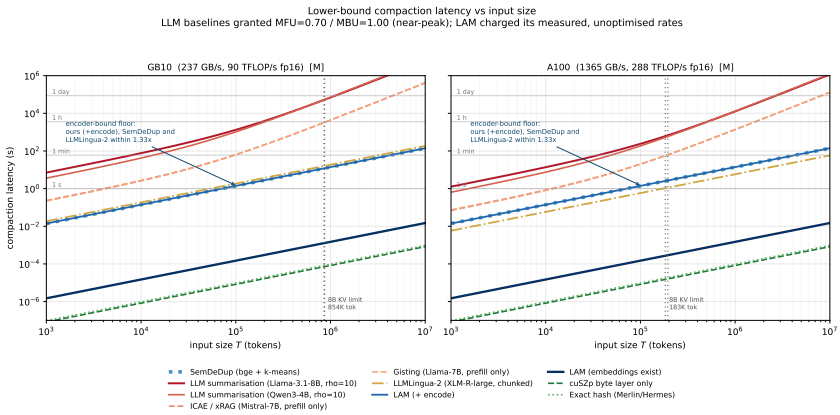

Figure 8: Compaction latency on a second device (NVIDIA A100-SXM4-40GB), the same comparison as Figure 9(a). The ordering and the crossovers are unchanged, so the gap is structural rather than an artifact of one bandwidth-starved machine. Dotted verticals mark where the KV cache no longer fits at batch 1.

joint law of $(X, \pi(X))$ is therefore feasible for the rate–distortion problem at distortion δ, and since $R_P(\delta)$ is the infimum of $I(X;\hat{X})$ over feasible couplings, $R_P(\delta) \le I(X;\pi(X))$. The almost-sure variant restricts the feasible set to couplings satisfying the constraint pointwise, which $(X, \pi(X))$ does. □

The bound holds for every P supported on S, and claims tightness for none. The cardinality bound $\log_2 |R|$ equals the achieved entropy $H(\pi(X))$ when retained clusters carry equal probability mass; their difference in Table 26 is the empirical imbalance.

COROLLARY 3 (MEASURED BRACKET)

Proof. We take the four terms of equation 4 left to right.

$\log_2 |R_{\text{greedy}}(2\delta)| \le \sup_P R_P^{\text{as}}(\delta)$. Write $P^\star = R_{\text{greedy}}(2\delta)$; by the packing half of Proposition 2 at radius 2δ, its elements are pairwise separated by more than 2δ. Let P be uniform on $P^\star$ and let $\hat{X}$ be any reconstruction with $\|X - \hat{X}\|_2 \le \delta$ almost surely. If distinct $p, p' \in P^\star$ were assigned a common $\hat{x}$, the triangle inequality would give $\|p - p'\|_2 \le 2\delta$, contradicting the separation. So $\hat{X}$ determines X, giving $I(X;\hat{X}) = H(X) = \log_2 |P^\star|$.

$\sup_P R_P^{\text{as}}(\delta) \le H_\delta(S)$. Let C be a minimal internal δ-covering, $|C| = N(S,\delta)$, and for any P take $\hat{X}$ to be a nearest element of C to X, which satisfies the constraint by definition. Then $R_P^{\text{as}}(\delta) \le I(X;\hat{X}) \le H(\hat{X}) \le \log_2 |C|$.

$H_\delta(S) \le \log_2 |R_{\text{greedy}}(\delta)|$. By the covering half of Proposition 2, $R_{\text{greedy}}(\delta)$ is an internal δ-covering of S, so its cardinality is at least $N(S,\delta)$. □

The middle two quantities need not be equal: the lower bound uses a 2δ-packing and the upper bound a δ-covering, with $M(S,2\delta) \le N(S,\delta) \le M(S,\delta)$. Two greedy passes then give computable bounds.

J THE DENSE δ SWEEP

Table 26 is the full sweep behind Figure 6. The outer bit columns are the bracket of Corollary 3, greedy first-occurrence at radius 2δ and at δ. The middle column is not part of it: $H(\pi(X))$ is the rate the δ run *achieves* on the empirical source, which equation 3 bounds only above by $\log_2 |R(\delta)|$ — every row satisfies that, and at $t = 0.995$ it falls just under the 2δ column, which is permitted because the bracket constrains the supremum over sources rather than one empirical source. A dash

in the 2δ column marks a threshold whose partner $t' = 4t - 3$ falls below 0.2, where that run merges nearly everything and the ceiling is vacuous. Masking is not a δ-substitution, so its rows carry no bit columns and appear only for the matched-removal comparison. RSC is the conditional ceiling in equation 10, not measured task success.

Table 26: Rate, removal and the outcome ceiling across the sweep. Bits per chunk; $\log_2 |S| = 14.77$ lossless.

		bits per chunk					
t	δ	$\log_2 \lvert R(2\delta)\rvert$	$H(\pi(X))$	$\log_2 \lvert R(\delta)\rvert$	tokens rm.	traj. intact	RSC
0.995	0.1000	14.61	14.59	14.67	18.47%	100.0%	75.83%
0.990	0.1414	14.55	14.56	14.65	18.89%	100.0%	75.83%
0.980	0.2000	14.40	14.50	14.61	19.64%	100.0%	75.83%
0.970	0.2449	14.14	14.45	14.58	20.45%	100.0%	75.83%
0.960	0.2828	13.80	14.40	14.55	21.53%	100.0%	75.83%
0.950	0.3162	13.41	14.35	14.52	22.50%	99.8%	75.67%
0.930	0.3742	12.48	14.22	14.45	25.40%	99.0%	75.00%
0.900	0.4472	11.15	13.98	14.29	32.02%	93.0%	70.33%
0.850	0.5477	9.26	13.42	13.89	47.55%	76.9%	59.83%
0.800	0.6325	9.23	12.79	13.41	64.38%	43.1%	35.67%
0.700	0.7746	—	11.39	12.20	86.71%	9.5%	8.33%
0.600	0.8944	—	10.21	11.15	93.31%	3.3%	3.33%
mask k=100	—	—	—	—	0.20%	99.8%	75.83%
mask k=50	—	—	—	—	4.56%	98.3%	75.17%
mask k=30	—	—	—	—	16.43%	92.5%	73.00%
mask k=20	—	—	—	—	34.28%	81.9%	66.33%
mask k=10	—	—	—	—	67.47%	52.2%	43.33%
mask k=5	—	—	—	—	78.00%	35.6%	30.00%
mask k=3	—	—	—	—	81.49%	32.8%	28.17%

K Additional measurements

K.1 Rank invariance, in full

We evaluate ranking margins on the LoCoMo retrieval store — 5,882 dialogue-turn records and 1,536 questions with gold evidence, excluding the unanswerable split, under `bge-base-en-v1.5` at $D = 768$ — which isolates the margin requirement from SWE-smith evidence retention. For unit queries a coordinatewise tolerance gives $|\Delta \cos| \le 2\varepsilon\sqrt{D}$, while distance verification gives $|\Delta \cos| = |q \cdot (a - b)| \le \|a - b\|_2 \le \sqrt{2 - 2t}$ at merge threshold t, linking the substitution bound to the merge threshold without a $\sqrt{D}$ factor.

Invariance rates. At $\varepsilon = 0.001$ the coordinatewise bound proves invariance for 15.62% of queries at $k = 1$ and 0.59% at $k = 3$; the ℓ_2 bound proves 3.71% at $k = 1$ in its tightest tested setting and 0.46% at $\cos \ge 0.99$. At the default $\cos \ge 0.95$ no tested k is guaranteed invariant: median adjacent-rank margins are 0.0185, 0.0056, and 0.0010 at $k = 1, 3, 20$ against bounds 0.1414 and 0.3162. Margins this tight are why Section C states the guarantee on scores rather than ranks.

K.2 Band-width study

Table 29 is the grid behind Section E.10, over 200 trajectories (5,011 records, 283,980 within-trajectory pairs), with σ =0.0361 the RMS coordinate magnitude of the encoder's output. Recall is over the true pairs at the stated cosine, so 1.000 means the banded index proposed every pair exhaustive comparison would merge.

The grid trades pair recall against candidate count. At $r = 2$, full recall at $\cos \ge 0.99$ costs 90.9% candidates with $\varepsilon/\sigma = 0.08$; the cheapest full-recall setting is $r = 8$, $\varepsilon/\sigma = 0.2$, examining 1.46% of pairs for a 68× reduction; at $r = 32$ recall never exceeds 0.770 at $\cos \ge 0.99$ or 0.314 at

Table 27: Share of queries whose top-k is provably invariant, by bound and by k. 1,536 queries against 5,882 LoCoMo records. The default operating point is $t = 0.95$.

			queries with bounded-invariant top-k				
relation	parameter	bound on $\lvert\Delta \cos\rvert$	k=1	k=3	k=5	k=10	k=20
ℓ_∞ relaxed, $2\varepsilon\sqrt{D}$	$\varepsilon = 0.001$	0.0554	15.62%	0.59%	0.13%	0.00%	0.00%
	$\varepsilon = 0.003$	0.1663	0.26%	0.00%	0.00%	0.00%	0.00%
	$\varepsilon = 0.01$	0.5543	0.00%	0.00%	0.00%	0.00%	0.00%
	$\varepsilon = 0.03$	1.6628	0.00%	0.00%	0.00%	0.00%	0.00%
ℓ_2 tight, $\sqrt{2-2t}$	$t = 0.995$	0.1000	3.71%	0.00%	0.00%	0.00%	0.00%
	$t = 0.99$	0.1414	0.46%	0.00%	0.00%	0.00%	0.00%
	$t = 0.98$	0.2000	0.07%	0.00%	0.00%	0.00%	0.00%
	$t = 0.95$	0.3162	0.00%	0.00%	0.00%	0.00%	0.00%
	$t = 0.90$	0.4472	0.00%	0.00%	0.00%	0.00%	0.00%
	$t = 0.85$	0.5477	0.00%	0.00%	0.00%	0.00%	0.00%

Table 28: Rank-k/rank-$(k{+}1)$ cosine margin over 1,536 queries. Compare against the bound column of Table 27.

k	median	p90	p99	max
1	0.0185	0.0702	0.1296	0.2358
3	0.0056	0.0201	0.0483	0.0886
5	0.0036	0.0133	0.0299	0.0698
10	0.0018	0.0067	0.0148	0.0247
20	0.0010	0.0034	0.0075	0.0147

Table 29: Band-width sweep over the exhaustive within-trajectory pair set. Candidate rate is the fraction of pairs sent to distance verification; its reciprocal gives the reduction in comparisons. We select settings with full recall and low candidate rate.

	ε/σ							
r	0.01	0.02	0.03	0.05	0.08	0.12	0.2	0.3
recall of true $\cos \geq 0.99$ *pairs*								
2	0.974	1.000	1.000	1.000	1.000	1.000	1.000	1.000
4	0.581	0.774	0.885	0.990	1.000	1.000	1.000	1.000
8	0.400	0.506	0.590	0.743	0.879	0.986	1.000	1.000
16	0.332	0.398	0.467	0.537	0.668	0.779	0.934	0.992
32	0.323	0.333	0.358	0.414	0.486	0.550	0.685	0.769
candidate pairs, % of all within-trajectory pairs								
2	5.40	17.36	33.19	64.21	90.95	99.18	100.00	100.00
4	0.48	0.66	0.79	1.19	2.61	7.54	31.65	73.06
8	0.33	0.42	0.49	0.61	0.73	0.90	1.46	3.84
16	0.27	0.33	0.39	0.44	0.55	0.64	0.79	0.98
32	0.27	0.28	0.30	0.34	0.40	0.45	0.57	0.64
recall of true $\cos \geq 0.95$ *pairs*								
2	0.574	0.844	0.955	1.000	1.000	1.000	1.000	1.000
4	0.237	0.322	0.378	0.488	0.672	0.931	1.000	1.000
8	0.163	0.206	0.240	0.303	0.361	0.441	0.651	0.979
16	0.135	0.162	0.191	0.219	0.272	0.318	0.391	0.481
32	0.132	0.136	0.146	0.169	0.198	0.224	0.279	0.314

$\cos \geq 0.95$. Settings are specific to the threshold: the best 0.99 configuration reaches only 0.651 recall at 0.95.

L DESIGN CHOICES AND THE SCOPE

These choices connect the implementation to its guarantees and cost model.

1. **Direct ℓ_2 verification** (Section 2). For unit-normalized embeddings cosine maps directly to substitution distance. Quantization proposes candidates; the distance check decides the merge.
2. **A query-independent substitution bound** (Appendix K.1). The theorem bounds maximum retrieval-score loss with no ranking-margin assumption; ranking quality and evidence retention are measured separately.
3. **Retained representatives** (Section C.4). Every deleted record lies within δ of its representative. Greedy first-occurrence enforces that without assuming the merge relation is transitive.
4. **Per-arrival accounting** (Section M.3). Removal is charged once per arriving token, the context grows by the retained arrivals, and a hard size limit needs eviction on top of deduplication.
5. **Explicit reproducibility conditions** (Section D). LAM is deterministic for fixed embeddings, arrival order, and arithmetic; cross-batch and cross-hardware stability are measured separately. The LLM deletion baseline reproduces only under its tested serving configuration.

M COST ACCOUNTING AND COMPACTION LATENCY

This section supplies the per-method cost accounting the performance model of §2.6 applies.

M.1 THREE COST CLASSES

A wall-clock comparison has to charge each method for what it does, which separates them into three classes.

Class 1: decode-bound LLM (summarization). With input T and ratio ρ, the method decodes $G = T/\rho$ tokens:

$$t = \underbrace{\frac{2PT + 2LdT^2}{\mu_c F}}_{\text{prefill}} + \underbrace{\frac{GPb_w + 2LH_{kv}d_h b_{kv}(GT + G^2/2)}{\mu_m B}}_{\text{decode}} + G\tau. \tag{11}$$

The term $GPb_w/(\mu_m B)$ is linear in T and *irreducible by parallelism*, since decode is sequential; on our hardware an 8B model in fp16 costs 67 ms per token from this term alone.

Class 2: prefill-only LLM (Gisting, ICAE, xRAG, LLMLingua-2). One forward pass emits the representation, so there is no decode term. A causal backbone gives $t = (2PT + 2LdT^2)/(\mu_c F)$, quadratic in T; a *chunked encoder* windows attention for $t = (2PT + 4LdTW)/(\mu_c F)$, linear. LLMLingua-2 is in the second group, so it is the cheaper LLM baseline.

Class 3: no LLM (all of LAM's layers, and byte codecs generally):

$$t = \frac{T}{R_{enc}} + \frac{6ND}{B_q} + \frac{2ND}{R_{hash}} + \frac{n_{cand}}{R_{\ell_2}} + \frac{\beta T(1 + 1/\mathrm{CR})}{R_{cuSZp}}. \tag{12}$$

Every stage is below the machine balance (380 FLOP/byte, measured), so all are bandwidth-bound.

Fairness rule. LLM baselines get near-peak efficiency ($\mu_c = 0.70$, $\mu_m = 1.00$) while our pipeline is charged its measured, unoptimized rates, so the comparison favors the baselines. These are modeling assumptions, not runtime bounds; Appendix G lists the hardware measurements.

M.2 COMPACTION LATENCY AND ITS LIMITS AS AN ARGUMENT

Figure 9(a) compares the three classes and Figure 8 repeats it on a second device; vertical lines mark the modeled KV capacity at batch 1 (854K tokens for Llama-3.1-8B on GB10), beyond which the

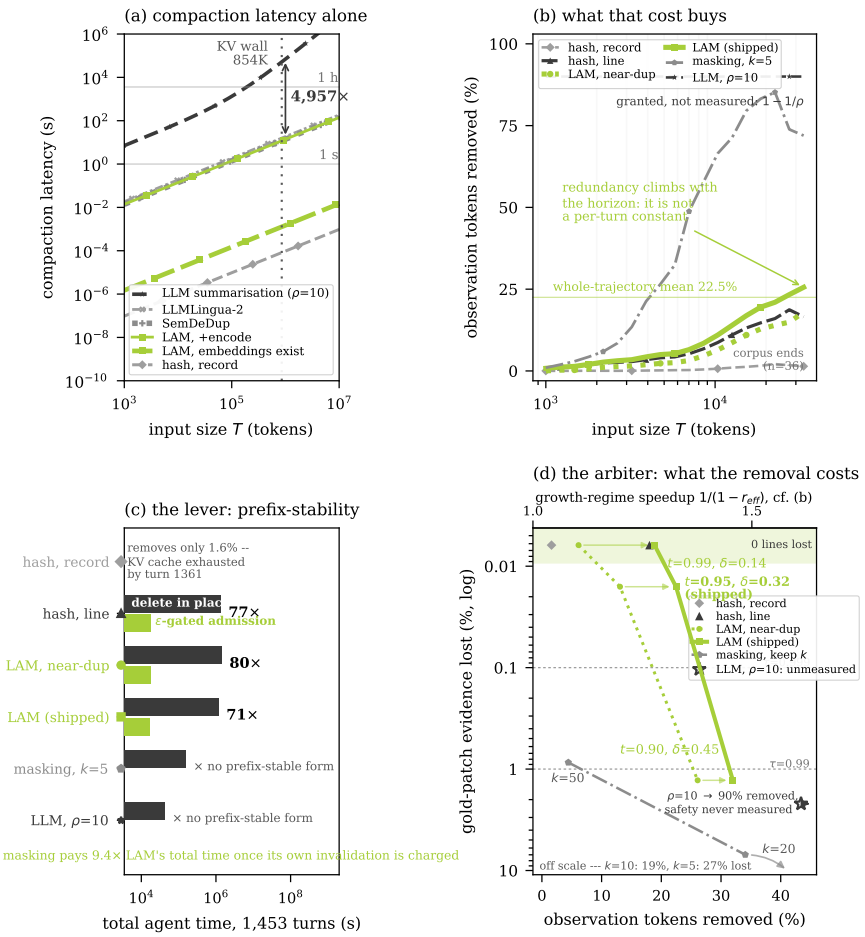

Figure 9: Compaction cost, removal, scheduling, and evidence retention. **(a)** Modeled compaction latency on T tokens, excluding subsequent agent inference. **(b)** Observation-token removal versus input size over 600 trajectories. Removal rises from 0.80% at $T = 1K$ to 25.58% at $T = 33K$, with whole-trajectory mean 22.50%. Curves stop when too few trajectories reach the input size; summarization removal is assumed at $1 - 1/\rho$. **(c)** Scheduling at fixed compressor, corpus, and removal. Methods that rewrite or delete existing context pay re-prefill costs; observation masking costs 9.4× times LAM's modeled total time. For rules supporting both schedules, testing first gives 71.4×–91.6× speedup. **(d)** Gold-patch evidence loss on an inverted logarithmic axis. Zero-loss methods appear in the shaded band. The plot is clipped at 10% loss. The default setting loses 1 of 6,276 lines (0.016%), compared with masking losses of 6.98%, 19.2%, and 26.9% at $k = 20, 10, 5$. The τ floors and 20% band mark retention requirements. Summarization evidence retention is unmeasured.

unmodified full-context configuration does not fit. At $T = 10^6$ the modeled compaction speedups are 4,957× over summarization, 1.33× over LLMLingua-2, and 1.00× over SemDeDup. The LLMLingua-2 cost model uses XLM-R-large; the evidence experiment in Appendix E.4 uses the smaller `bert-base-multilingual` variant configured by LightMem.

Encoding is 99.989% of LAM's cost at $T = 10^6$ and SemDeDup uses the same encoder, so the two cost about the same at trajectory scale. Where a memory store already holds the embeddings, LAM costs 1.49 ms against 0.0963 ms for L0 alone, a factor of 15.4×. We report both cases because reuse depends on the deployment.

Table 30: Snapshot and per-arrival accounting for the same measured removal rate.

	snapshot assumption	per-arrival (correct)
update rule	$C \leftarrow C(1-r)$ every event	$C \leftarrow C + \Delta(1-r)$ per arrival
steady state	$C^* = \Delta / r_{\mathit{eff}}$	none; only eviction bounds C
mean context, 32K budget	2.8K	28K
speedup of L1 over L0 alone	1.22×	1.07× growth, 1.00× hard budget

M.3 Per-arrival removal and context growth

The measured removal rate r is the fraction of arriving tokens duplicating earlier content. A token can be removed once, so context grows as $C \leftarrow C + \Delta(1-r)$ per arrival; applying r to the full context on every event would count the same duplicates again. Testing first imposes no steady-state size on its own, so a hard budget also needs eviction (Table 30).

The two accountings diverge where it matters. In the growth regime LAM supports 1,727 turns before the KV capacity limit against 1,615 for L0 alone, a modeled 1.07× speedup. Under a hard budget every method eventually reaches B_{ctx}, so additional removal gives 1.00× latency benefit; it buys delayed eviction of nonredundant content instead, a quality effect the model does not capture.