LARK: LOW-RANK ARCHIVED RETENTION OF KV CACHE VIA LINEAR EXPECTED-ATTENTION SCORING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long chain-of-thought (CoT) reasoning generates tens of thousands of tokens, making KV cache the primary memory and bandwidth bottleneck. Token eviction irreversibly discards history; its accuracy hinges on ranking keys by attention from queries that do not yet exist, which content predicts only in their frame, reached by an inverse rotation per candidate. We propose **LARK**, a KV cache compression system built to *archive rather than discard*. LARK comprises: (1) **LEAP**, a linear scoring operator strictly equivalent to pre-RoPE expected-query phase scoring yet executing as a single projection on cached post-RoPE keys, with cost independent of offset and query-sample counts; (2) **LACE**, a low-rank archival pipeline whose evicted tokens are compressed into factors that leave the model cache yet keep participating in attention, re-scored at segment level by the same form; (3) a systems co-design enabling CUDA-graph-compatible variable-length continuous batching. Across four models and four benchmarks at a 32k generation limit, LARK retains accuracy far above eviction-based baselines as the budget tightens. At B=2048 it reaches near-lossless accuracy with 5.0% of the dense device KV memory, with 2.0–3.3× iso-batch throughput.

1 INTRODUCTION

Test-time reasoning improves the accuracy of large language models (DeepSeek-AI et al., 2025), but its memory cost grows with every token. Challenging problems can require a *long chain-of-thought* spanning 10^4–10^5 tokens. During decoding, the KV cache occupies more GPU memory, leaving room for fewer concurrent requests. Reading this history requires more memory traffic per step, slowing generation. Efficient long-CoT serving must reduce this cost while preserving the accuracy that longer reasoning enables.

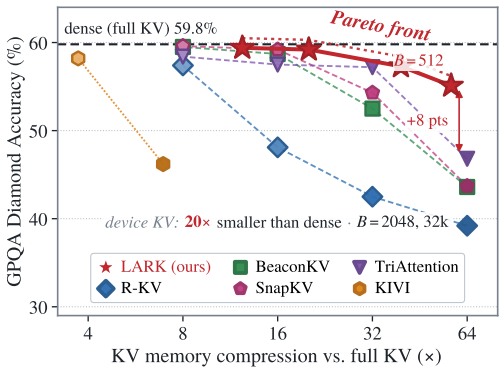


Figure 1: GPQA Diamond accuracy vs. device KV footprint at 32k (Qwen3-8B): LARK is 0.6 pt from dense on 5.0% of it at B=2048, 8 pts above every baseline at B=512.

KV cache compression spans eviction (Li et al., 2024; Kim et al., 2026), merging (Zhang et al., 2024a; Wang et al., 2024a), quantization (Liu et al., 2024c; Hooper et al., 2025), and low-rank approximation (Sun et al., 2025; Chang et al., 2024), applied during prefill (Xiao et al., 2024c; Cai et al., 2025; Jo et al., 2026) or decoding (Cai et al., 2026; Zarch et al., 2026). Retaining a subset of KV entries (e.g., via eviction) preserves their detail but limits historical coverage. Compact representations (e.g., merged states, quantized states, or low-rank approximations) can retain broader history at the cost of fidelity.

These trade-offs motivate combining individual retention with a compressed representation of the remaining history. We ask: *can a small retained KV cache support reasoning over a much larger history?* Supporting this broader historical access efficiently poses three challenges.

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.

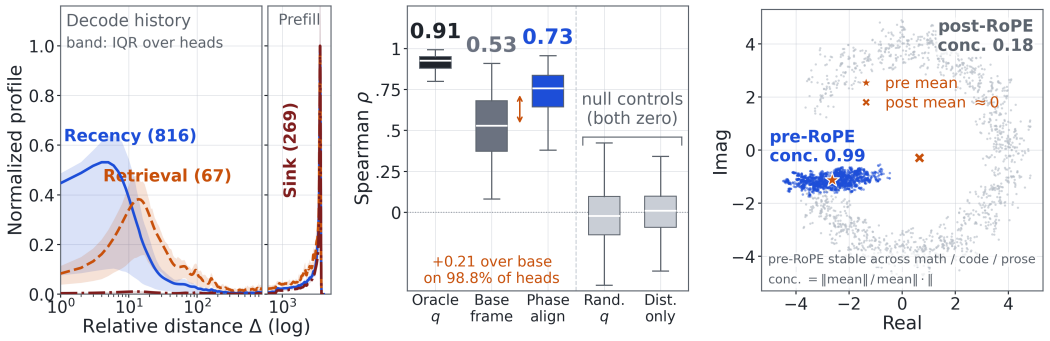


Figure 2: **Left-(a):** Three attention-head patterns based on realized future attention. **Mid-(b):** Spearman correlations between five predictors and realized future attention. **Right-(c):** Pre-RoPE query concentration vs. post-RoPE positional dispersion. (Appendix G)

Challenge 1 (C_1)**: Predicting future KV relevance.** A premise that receives little attention now may become essential later (An et al., 2024). Queries before positional encoding (pre-RoPE) cluster around a stable mean, a signal for predicting future query content (Mao et al., 2026) (§2.1). Yet relevance also depends on where those queries occur. Evaluating every key at many future positions would make online selection costly.

Challenge 2 (C_2)**: Preserving evicted history.** Even a fixed oracle ranking under a tight budget leaves much of the later attention to the history uncovered (§2.2). Low-rank archival extends coverage beyond this pool via shared structure, but must stay reselectable as queries change. Reconstructing full KV on every reuse costs computation and transient memory, so selection and attention must happen in compressed form.

Challenge 3 (C_3)**: Translating compression into serving gains.** Maintaining a fixed KV budget requires repeated compression. If decoding waits for each compression event, its overhead can erase the benefit of a smaller cache. The system must overlap this work with generation while preserving serving optimizations (e.g., PagedAttention and CUDA graph execution).

To address these challenges, we present **LARK** (Low-rank Archived Retention of KV Cache by Linear Expected-Attention Projection), a serving system in which *evicted tokens can still contribute to future attention*. Two components share one linear form: **LEAP**, a scorer that folds future-position effects into a single projection, and **LACE**, a low-rank archive that the same projection scores directly from its stored factors. **Serving optimizations** fuse attention over retained and archived entries, overlap archival with decoding, and keep CUDA graph execution under variable-length continuous batching. Figure 1 previews the trade-off; our contributions are:

(1) *Unified Linear Relevance Scoring*: We derive LEAP, an exact reformulation of linear expected-query scoring that uses one projection per cached key. The same operator scores archived groups directly from their compressed representations.

(2) *Low-Rank Archival with Direct Attention*: We design LACE to archive each compression event's evicted KV as a low-rank *segment*. GPU-resident segments attend directly, while host-resident ones stay eligible for reselection as reasoning evolves.

(3) *System Integration and Evaluation*: Across four models and four benchmarks at a 32k limit, LARK stays far above eviction baselines as budgets tighten; on Qwen3-8B at B=2,048 it reaches **90.4%** MATH-500 accuracy versus 90.6% for full-KV, using only **5.0%** of its device KV memory. At matched batch sizes, our Mini-SGLang implementation delivers **2.0–3.3**× full-KV decode throughput across budgets of 1,024 and 2,048 on H20 and RTX 5090 GPUs.

2 Motivation

Notation. An n_L-layer model has H heads of dimension d. Positions are $p, t \in \mathbb{N}$; t_0 is the compression time, so keys at $p \le t_0$ are eviction candidates and queries at $t_0+\tau$ ($\tau \ge 1$) are future queries. RoPE applies a block-diagonal rotation $R(\theta) = \bigoplus_j \mathrm{Rot}(\theta\omega_j)$, mapping base-frame

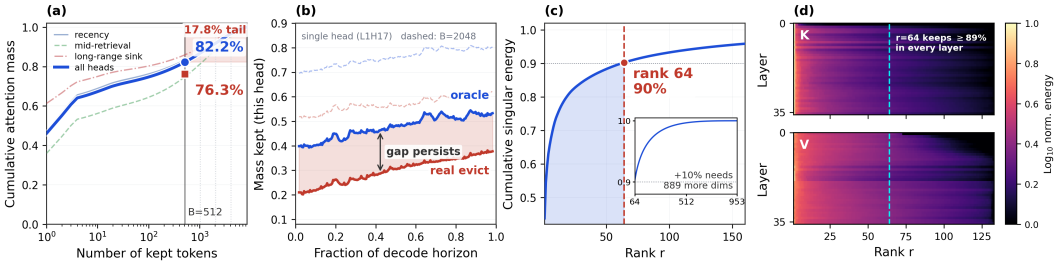


Figure 3: (a) Oracle mass over all heads vs. real eviction at B=512; (b) the same gap, one head over decode; (c) cumulative singular energy (rank-64 = 90%); (d) per-layer K/V spectra.

(pre-RoPE) $q_t, k_p \in \mathbb{R}^d$ to cached post-RoPE $\tilde{q}_t = R(t)q_t$, $\tilde{k}_p = R(p)k_p$; we use $R(a)R(b) = R(a{+}b)$ and $R(a)^\top = R(-a)$. The logit is $\ell_t(p) := \tilde{q}_t^\top \tilde{k}_p/\sqrt{d}$ and the attention mass $a_t(p) := \exp(\ell_t(p))/Z_t$. Future importance is estimated from a per-head base-frame expected query $\hat{\mu}_h$ over a geometric offset grid $\{\tau_j\}_{j=1}^J$ spanning the future horizon.

The core compression decision is whether a historical token will still receive future attention. Insights 1–2 (§2.1) motivate the scorer (C_1) from two mechanistic observations (derivation in Appendix B; measurement protocol in Appendix G); Insight 3 (§2.2) drives the archival tier (C_2).

2.1 Observations in Predicting Future Importance

Insight 1: Importance ranking requires content–position interaction; the scorer must align expected queries to the future query frame. Aggregating realized future attention by distance across heads reveals three modes (Figure 2): recency (816 heads), mid-retrieval (67), long-range sink (269). A five-predictor decomposition (Figure 2b) shows oracle content similarity reaches Spearman 0.91, but a random-query control is zero: the difficulty is the unknowable query side. Distance alone cannot rank keys within a stage; it enters only as the comparison frame. The same phase term in the base frame, key rotation uncancelled, reaches 0.53. Realigned to future positions it recovers 0.73, beating the base frame on 98.8% of heads—the gain is frame alignment, not positional resolution (Appendix F.5). Backward-looking attention is posterior: evidence for soon-to-be-retrieved keys has not yet appeared, so past importance is not future importance.

Insight 2: Only the pre-RoPE frame has a stable query center; it is reachable from post-RoPE keys at no extra cost. Pre-RoPE queries concentrate at a stable non-zero center (Figure 2c), whereas post-RoPE queries disperse into position-dependent arcs that confine observation-based statistics to a narrow recent window. This explains why pre-RoPE statistics provide a usable scoring signal (Mao et al., 2026)—but the scorer S_h^{pre} (below) requires per-candidate inverse RoPE (cost $\Theta(mdJ)$) or an extra pre-RoPE copy. We observe that the RoPE composition rules move the rotation from the key side to the statistics side, so its phase term is strictly equivalent to a post-RoPE one:

$$S_h^{\text{pre}}(p) \;:=\; \tfrac{1}{J\sqrt{d}} \textstyle\sum_j \hat{\mu}_h^\top R(p - t_0 - \tau_j)\, k_p \;=\; \tfrac{1}{J\sqrt{d}} \textstyle\sum_j \big(R(t_0 + \tau_j)\, \hat{\mu}_h\big)^\top \tilde{k}_{p,h} \;=:\; S_h^{\text{post}}(p) \quad (1)$$

i.e., a single linear projection on cached post-RoPE keys (proved in Appendix B). The scorer is instantiated as a contribution in §3.1.

2.2 Low-Rank Structure of Historical KV

Insight 3: Hard eviction loses a persistently attended tail; historical KV is low-rank, admitting archival in factors. Scoring determines *who* stays but cannot relieve capacity pressure: hard discard's cost is independent of scoring quality—an oracle ranking keys by their realized future attention, fixed once at the compression point as real eviction is, pays it too, so the gap is irreducible by better scoring. On an AIME trace (11.9k generated tokens; 8,479-token prefix, 1,152 heads), that oracle's top-512 keeps only 82.2% of the history block's future attention mass, and frozen eviction drops this to 76.3% (Figure 3a; panel b, one head over decode): hard eviction irreversibly loses a persistently attended tail—like peripheral vision, low in resolution yet indispensable.

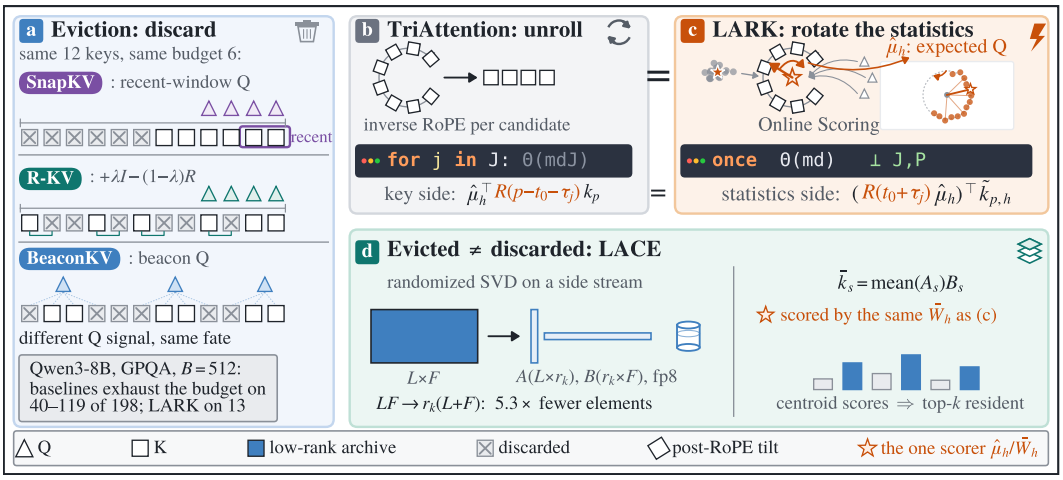


Figure 4: **(a)** Eviction discards history irreversibly. **(b)** TriAttention unrolls the grid per candidate ($\Theta(mdJ)$); **(c)** LEAP rotates the statistic $\hat{\mu}_h$ instead, giving the identical score in one $\Theta(md)$ projection. **(d)** LACE factorizes each evicted segment, scored by the same $\bar{W}_h$.

History should instead stay compact *and* usable in attention. Low-rank factorization does both. On a shorter trace (layer 17, 953-token history) the first 64 singular components hold 90% energy (Figure 3c; panel d, 190-token probe), and per-segment spectra decay faster than the global one, making rank 64 a conservative precision–memory operating point.

Off-diagonal profile correlations of 0.29–0.63 leave no shared subspace, so $\hat{\mu}_h$ stays per-head, whereas the offset grid depends only on the RoPE frequencies and is shared. V adopts rank 32 vs. K's 64 as attention-weighted averaging tolerates more reconstruction error (ablated in §4.4).

3 Method

We introduce **LARK**, which organizes the KV cache into three components (Figure 4): the LEAP scorer driving eviction and segment residency (§3.1), the LACE archival tier of evicted buffers (§3.2), and the system co-design (§3.3). Algorithm 1 (Appendix F) gives the overall workflow.

3.1 LEAP: Query-Aware Importance Scoring

Scoring must estimate each candidate key's future attention. We estimate the expected query online: at each compression event t_0, a probe set $\mathcal{P}$ mixing recent offsets (the current stage) with distant ones (the long-term mean) is inverse-rotated to the base frame and averaged per head,

$$\hat{\mu}_h \;=\; \frac{1}{|\mathcal{P}|}\sum_{t\in\mathcal{P}} q_{t,h}^{\text{base}}\,. \tag{2}$$

The probe ring fills before the first compression event, so $\hat{\mu}_h$ is read from the live trace. The ideal importance of a cached key $\tilde{k}_p$ is the $\pi_h(\tau)$-weighted average mass it receives at future offsets τ, $S_h^*(p) = \sum_\tau \pi_h(\tau)\, a_{t_0+\tau}(p)$. We average rather than accumulate, since a cumulative score couples to the remaining generation length. As S_h^* contains unknown future queries, we estimate the expected logit $\hat{S}_h(p) = \sum_\tau \pi_h(\tau)\,\tilde{\mu}_{h,\tau}^\top \tilde{k}_p/\sqrt{d}$ with $\tilde{\mu}_{h,\tau} := \mathbb{E}[\tilde{q}_{t_0+\tau}]$. Logit and mass are rank-equivalent for a realized query (Z_t is p-independent), so per-query selection is preserved.

Three residual gaps remain: a Jensen term biasing against keys with high logit variance, the cross-step aggregation, and the drift of the base-frame expected query, which $\hat{\mu}_h$ estimates in mean rather than assumes away. Appendix F.5 characterizes all three and bounds the drift.

Evaluating this expected logit under the base-frame expected query $\hat{\mu}_h$ yields the pre-RoPE scorer

$$S_h^{\text{pre}}(p) \;=\; \frac{1}{J\sqrt{d}}\sum_{j=1}^{J} \hat{\mu}_h^\top\, R(p-t_0-\tau_j)\; k_p\,, \tag{3}$$

whose naive implementation requires per-candidate inverse RoPE (cost $\Theta(mdJ)$ over m candidates) or an extra pre-RoPE copy. We avoid both costs with an exact rearrangement: applying the RoPE composition rules moves the rotation from the key side to the statistics side,

$$S_h^{\text{pre}}(p) \;=\; \frac{1}{J}\sum_{j=1}^{J} \bar{W}_{h,j}^{\top}\,\tilde{k}_{p,h}\,, \qquad \bar{W}_{h,j} \;:=\; \frac{1}{\sqrt{d}}\,R(t_0+\tau_j)\,\hat{\mu}_h\,, \tag{4}$$

i.e., a single linear projection on cached post-RoPE keys, avoiding both. The derivation is in Appendix B. Scoring, archival, and attention thus share one post-RoPE representation.

The deployed proxy instantiates Eq. 4 on the geometric grid $\tau_j = 2^{j-1}$ (J=17), spanning the horizon logarithmically. As the aggregation is a linear mean, the J terms fold on the statistics side into one weight $\bar{W}_h = \frac{1}{\sqrt{d}}R(t_0)D\hat{\mu}_h$, where D depends only on the RoPE frequencies and grid, precomputed offline per model. Scoring is thus one projection, independent of J and probe count.

Summary. The ranking target is attention from queries yet to exist. Taking its expectation under an online expected query yields a score linear in the key, which LEAP rewrites into one equivalent post-RoPE projection. **One projection on cached keys estimates attention from queries that do not yet exist.**

3.2 LACE: Low-Rank Archival and Centroid Tiering

LARK's archival tier is **LACE** (low-rank archive with centroid exactness). LARK caps the pool at a budget B and archives in segments of L tokens: on reaching $B{+}L$, the L lowest-scoring tokens leave for the eviction buffer, archived as a unit. Let $K_s \in \mathbb{R}^{L\times F}$ collect the segment's post-RoPE keys, flattened over features ($F{=}H_{\text{kv}}d$) so all KV heads share one factorization:

$$K_s \;\approx\; A_k B_k\,, \qquad A_k \in \mathbb{R}^{L\times r_k}, \quad B_k \in \mathbb{R}^{r_k\times F}, \tag{5}$$

and likewise (A_v, B_v) for values, at the spectral operating point of §2.2. Factorizing per segment matters on two counts: per-segment spectra decay faster than the global one (§2.2) and a completed segment's factors are immutable, making archival a one-shot cost. Storage per segment falls from LF to $r(L{+}F)$ entries, and the archive grows per compression event rather than per token.

Archived segments never rejoin attention by reconstruction. The linear form of §3.1 scores each segment directly from its centroid $\bar{k}_s = \text{mean}(A_s)B_s$:

$$S_h^{\text{seg}}(s) \;=\; \frac{1}{J}\sum_{j} \bar{W}_{h,j}^{\top}\bar{k}_s\,, \tag{6}$$

the exact per-key score mean at one d-dimensional projection—no reconstruction. Linearity is what makes the archive re-scorable at all: a scorer needing the individual archived keys must first materialize A_sB_s, at 341× the FLOPs and a transient buffer 14× the factors it reads (Appendix F.4). LEAP is thus a precondition for LACE. The centroid score also drives a two-tier residency rule (Figure 6): the κ highest-scored segments keep their factors GPU-resident and join every decode step through the factorized branch (§3.3). The rest keep only their centroids resident, which cost little and suffice to re-score the archive, while their factors stay on the host, off the critical path. Each event thus updates the resident set, bounding attention as history grows.

Both tiers are stored in fp8 (e4m3) with per-column scales. Quantization composes multiplicatively with factorization, which removes cross-token redundancy while fp8 narrows each coefficient. Dequantization is fused into the attention kernel.

Summary. Why budget + Low-rank archive? A rank-64 archive keeps 90% of a key's energy (Figure 3c), which suffices to blur similar keys. Identity-critical tokens therefore stay verbatim and the concentrated residual survives in a few singular directions. **The budget preserves token-level fidelity; the archive preserves attention-level coverage.**

3.3 Algorithm–System Co-Design

The archival pipeline must hide its own factorization cost, let resident factors participate cheaply in attention, and integrate with PagedAttention, prefix caching, and CUDA graph (Figure 5).

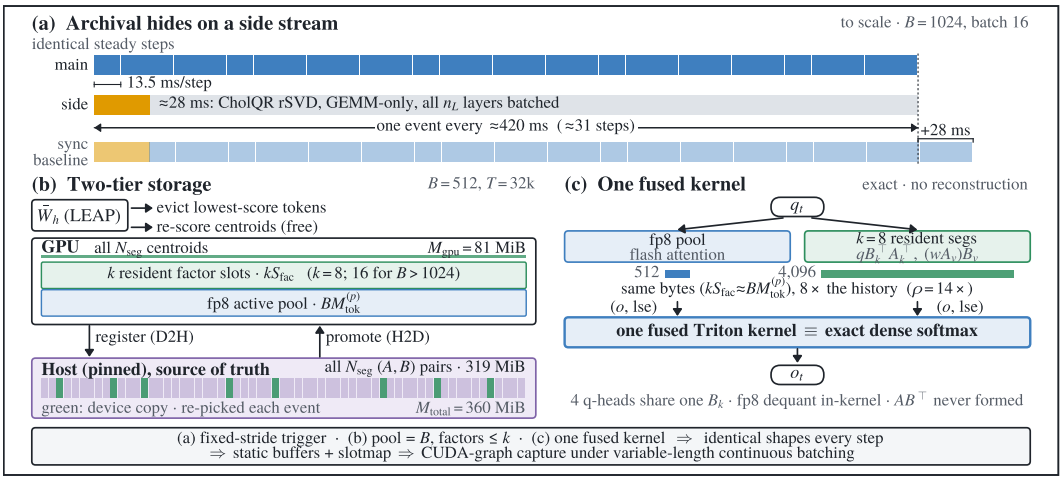


Figure 5: Systems view of LARK. (a) Archival on a side stream, drawn to scale (28/420 ms per event at batch 16). (b) Two-tier storage (M_{gpu}=81 MiB at B=512). (c) One fused kernel; factor branch covers 8× pool history for the same bytes (ρ=14×). Footer: CUDA-graph capture chain.

Factorized attention. Resident segments join attention without ever materializing A_kB_k: logits factor as $(qB_k^\top)A_k^\top$ and weighted sums as $(wA_v)B_v$, and both merge with the pool's flash attention via online softmax into one exact softmax. Given the factors this equals exact attention on the reconstructed KV, so the fusion itself adds no error. A kv-head-packed batched kernel amortizes the shared row factor across heads, while the residency cap keeps cost independent of archive size. The residual $K_s - A_kB_k$ is discarded without correction: archival is one-shot and the cost is bounded by the spectral operating point of §2.2. Residency corrects selection misses instead: every event re-scores all segments, so a dropped segment can re-enter.

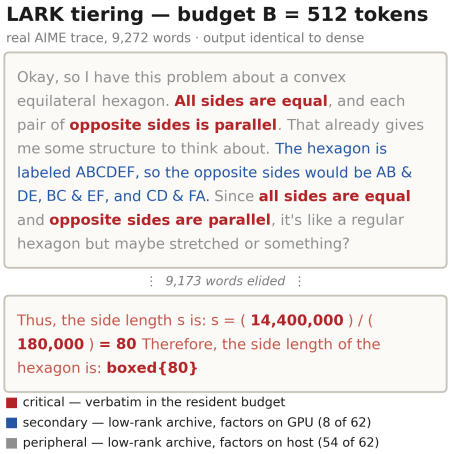


Figure 6: Illustrative tiering of a real trace (verbatim; Appendix E): the archive relocates rather than discards content.

Asynchronous factorization. Eq. 5 is realized by randomized SVD with CholQR power iteration, which is GEMM-only and avoids the fp32-bound cuSOLVER QR path, under a fixed seed making archival bit-reproducible. The resulting $(Q, Q^\top K)$ pair spans the same subspace as standard rSVD factors. All layers of one compression event form one batched decomposition, executed on a background stream while the main thread only decides and assembles segments, cutting archival cost an order of magnitude below serialization.

Serving-system integration. The method embeds in PagedAttention engines: compression swaps page indices and needs no kernel changes. Eviction truly frees memory rather than masking it—the commit step returns the segment's pages to the allocator and marks their page-table entries with sentinels, so a page-count check verifies real eviction at run time. Prompt tokens stay pinned for prefix-cache reuse. Fixed-stride triggering, static buffers, and a **slotmap indirection layer** (slot lifetime decoupled from batch position) enable CUDA graph capture under variable-length continuous batching, with a single capture reused across requests even as batch composition changes.

Summary. The factor branch reads the archive at a cost independent of its size, side-stream rSVD hides archiving behind decoding, and the slotmap keeps steady-state decoding CUDA-graph capturable. **Compressed memory, uncompressed throughput.**

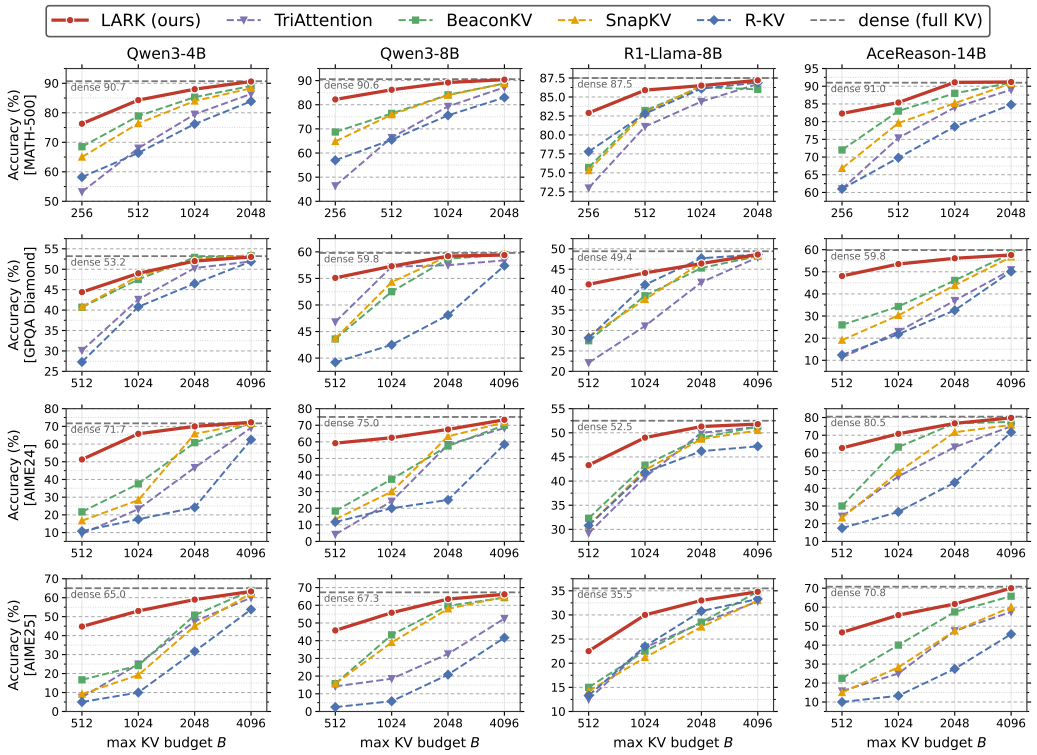


Figure 7: Accuracy vs. KV budget B (T=0.6, top-p 0.95, Avg@4): LARK against all baselines over four models (columns) and four benchmarks (rows); dashed lines mark dense full KV.

4 Experiments

4.1 Experimental Setup

Models and Datasets. We evaluate on four models: Qwen3-4B and Qwen3-8B (Qwen Team, 2025), DeepSeek-R1-Distill-Llama-8B (DeepSeek-AI et al., 2025), and AceReason-Nemotron-14B (Chen et al., 2025), over four benchmarks: MATH-500 (Lightman et al., 2023), AIME24 (Zhang & Math-AI, 2024), AIME25 (Zhang & Team, 2025), and GPQA Diamond (Rein et al., 2023).

Baselines. We compare against **dense** (full KV in bf16) as the accuracy reference, three eviction-based compressors evaluated at matched budgets, namely **SnapKV** (Li et al., 2024), **R-KV** (Cai et al., 2026), and **BeaconKV** (Kim et al., 2026); **TriAttention** (Mao et al., 2026), a trigonometric-compression method likewise built on RoPE structure. Every method is run under the same engine, generation limit, and grader, so differences reflect the compression policy rather than the harness.

Hyperparameters. Unless stated otherwise LARK uses a 32-slot probe ring, the grid $\tau_j{=}2^{j-1}$ ($J{=}17$), segment length $L{=}512$, ranks $r_k{=}64/r_v{=}32$ stored in fp8 (e4m3) for both tiers, a residency cap of $\kappa{=}8$ resident segments for $B{\leq}1024$ and $\kappa{=}16$ for $B{>}1024$, which keeps LARK's device footprint at the baselines' level across budgets (Appendix C), and no residual correction; $B{\in}\{256, 512, 1024, 2048\}$ on MATH-500 and $\{512, 1024, 2048, 4096\}$ on AIME24/AIME25/GPQA-Diamond (Appendix G).

Implementation. We implement LARK on Mini-SGLang (PagedAttention, radix prefix caching) with Triton kernels. Archival uses a dedicated side stream with static, fixed-address buffers so steady-state decoding stays CUDA-graph capturable (§3.3).

Evaluation Protocol. Experiments run on NVIDIA H20, with RTX 5090 added for efficiency. Accuracy is Avg@4 at temperature 0.6/top-p 0.95 for all tests (aligned with Kim et al., 2026; Mao et al., 2026). The generation limit is 32,768 tokens. A response reaching it without an answer counts as *budget-exhausted*. Memory is reported in binary units throughout (1 MiB$=2^{20}$ B), device capacities included.

4.2 Main Results

Figure 7 reports the accuracy. Two readings follow.

The gap to eviction opens as the budget tightens. At loose budgets all methods sit near dense. The ordering changes sharply at B=512. On Qwen3-8B, LARK retains 86.2% on MATH-500 against 76.4/75.9/65.5% for BeaconKV/SnapKV/R-KV, and 59.2% on AIME24 against 4.2–18.3%. On R1-Distill-Llama-8B it holds 41.3% on GPQA while baselines fall to the 25% random level. Raising GPQA to B=2048 gains LARK 4.1 pp versus 8.9–15.5 pp for baselines—roughly 4× the budget to match LARK's tightest setting. On MATH-500 (shortest chains) all four fall into a narrow 83.0–88.8% band.

Table 1: Decode efficiency; Peak in GiB (cap: H20 95.1, 5090 31.4).

Method	B	Acc	H20 Tok/s	H20 Peak	RTX 5090 Tok/s	RTX 5090 Peak
Iso-batch (batch 16-H20/2-5090)						
Full KV	–	90.6	366	88.0	110	24.6
R-KV	1024	75.6	1169	18.7	215	15.8
R-KV	2048	83.0	1079	21.4	186	16.2
LARK	1024	89.2	1191	18.4	225	15.7
LARK	2048	90.4	1032	21.0	219	16.1
Max-batch (batch in subscript)						
R-KV	1024	75.6	2499_{432}	93.3	961_{80}	31.2
R-KV	2048	83.0	1857_{240}	93.0	809_{40}	31.1
LARK	1024	89.2	2554_{548}	93.9	1056_{98}	30.7
LARK	2048	90.4	2182_{312}	94.1	856_{50}	31.0

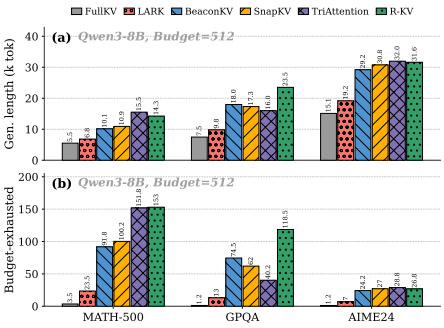


Figure 8: Qwen3-8B at B=512: (a) mean generated length; (b) exhaustion counts, MATH-500/GPQA/AIME24 out of 500/198/30.

Budget exhaustion as the failure mechanism. Eviction fails by non-termination, not degradation: on R1-Distill-Llama-8B at GPQA B=512, LARK exhausts the budget on 6.5/198 problems (Avg@4; decode 12.2k), vs. 107–130 (54–66%, 22–25k) for baselines (all cells: Appendix D.1), a 16–20× reduction in exhausted responses at the same budget. Deleting still-attended history prevents convergence, as the model restates reasoning until the limit, whereas archival keeps it recallable. Figure 8 shows the coupling on Qwen3-8B: panel (b) orders the methods by exhaustion and panel (a) reproduces that order in decode length, dataset by dataset. On AIME24 every baseline exhausts 24–29 of 30 problems and runs to within 11% of the 32k limit, so the median chain never converges, while LARK exhausts 7. On GPQA LARK's 13 stays well below the baselines' 40–119. A truncated response is graded wrong regardless of the reasoning before it, so non-termination rather than degraded reasoning turns a tight budget into lost accuracy, as §2.2 argued.

4.3 Inference Efficiency

Memory footprint. Appendix C's ledger predicts the device footprint in closed form. Figure 9 checks it against KV allocation measured to 8k and extended to 32k at saturation, against dense bf16, 2/4-bit quantization, and 1024-token eviction. Dense grows linearly to 4.52 GiB and quantization inherits it (1.21/0.65 GiB), whereas eviction flattens at 180 MiB and LARK at 126 MiB at B=1024, i.e. 2.7% of dense and 70% of eviction's own footprint, 8% above the ledger's 116.9 MiB (prefill and allocator residue). Only centroids grow once the cap binds.

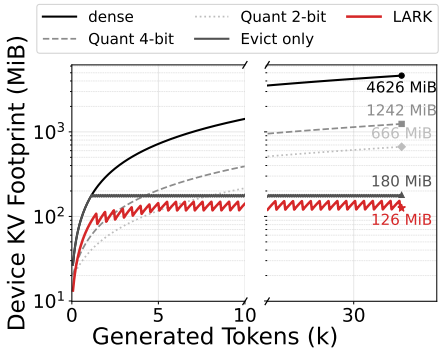


Figure 9: Device KV Mem footprint.

End-to-end throughput. Table 1 reports decode efficiency on Mini-SGLang (§3.3) at 32k under two regimes (Ramachandran et al., 2026; Kim et al., 2026): a common batch, and the largest batch each footprint sustains. At a matched batch LARK turns the saved memory into 3.3× dense throughput on H20 and leads the eviction baseline in three of four cells (Appendix D.4). At each method's own limit the smaller footprint buys concurrency, which more than offsets the factor branch's per-step cost: LARK leads in all four cells, by 2.2% to 17.5%.

Component overhead breakdown. Table 2 accounts for archival over a 32k run at B=1024, batch 16, cumulative over 1,024 events (Appendix D.4). The side stream costs ≈28 ms per event against a ≈420 ms interval, entering the critical path only if the interval drops below it.

Table 2: LARK component overhead.

Stage	Total (ms)	ms/step	Loc
End-to-end decode	–	13.4	–
Scoring	1445	0.041	GPU
Selection + retention	916	0.026	host
Batched rSVD (K+V)	13571	0.39	side
Package	7966	0.23	side
Mem Swap	9237	0.27	side+host

4.4 Ablation Study

All ablations run at batch 16, B=1024. Other parameters match the main experiments. Acc is on MATH-500. Blue marks deployed settings, two of them because deployment is budget-dependent.

Ablation on each component. Each row of Table 3 adds one component to the softmax-mass eviction baseline—LEAP scoring (§3.1), the low-rank archive (§3.2), then top-κ centroid tiering.

Table 3: Cumulative ablation; Lat. = per-step latency.

Configuration	Acc	Lat. (ms)
Softmax scoring	74.0	10.77
+ LEAP	81.5	10.79
+ low-rank archive (all)	89.6	20.45
+ top-κ tiering (LARK)	89.2	13.43

The Softmax-scoring row is the anchor-less streaming null. Published baselines surpass it by augmenting the same observational signal with prefill, beacon, or redundancy anchors. LEAP improves ranking at no memory cost (74.0→81.5, above TriAttention's 79.3) and makes scoring flat in P. The low-rank archive lifts accuracy 8.1 points to 89.6 with all segments resident. Top-κ tiering recovers most of it at a 0.4-point cost and keeps the step affordable: full residency nearly doubles it (10.8→20.5 ms); tiering returns it to 13.4 ms. The stack nets 15.2 points at 25% higher step time.

Ablation on rank. Table 4 sweeps ranks around 64/32. The key axis saturates at 64: halving to 32/32 costs 5.0 points, while doubling to 128/64 adds only 0.3 points at twice the bytes and an 8% throughput cost. The value axis is slightly flatter but non-monotone: 64/16 loses 2.9 points and 64/64 sits 0.4 below 64/32 rather than above. That the key side matters somewhat more is consistent with keys setting the logits and hence the ranking, while values enter an averaging that partially cancels the error, as in asymmetric KV-cache quantization (Hooper et al., 2025; Tao et al., 2025).

Table 4: Ablation on rank.

r_k/r_v	S_{seg} (MiB)	Tok/s	Acc
32/32	3.45	1276.1	84.2
64/16	4.29	1221.3	86.3
64/32	5.14	1191.0	89.2
64/64	6.83	1148.9	88.8
128/64	10.21	1093.0	89.5

Ablation on top-κ. The cap κ bounds how many of the N_{seg}=62 archived segments keep resident factors, while all centroids stay resident for re-scoring, so only resident segments join attention. The footprint is affine in κ (Table 5): 76.4 MiB fixed plus S_{fac}=5.07 MiB per segment. Accuracy rises steeply until κ=8 (82.8→89.2, at 30% of full-residency footprint) and is flat beyond: $\kappa \geq 16$ stays within 88.9–89.6, so full residency buys 0.4 points for 3.3× the footprint (117.0→390.9 MiB). We deploy κ=8 for $B \leq 1024$ and κ=16 above, which holds the footprint at the baselines' level across budgets.

Table 5: Residency-cap ablation; memory from Eqs. 12, 14.

κ	$\kappa\, S_{\text{fac}}$ (MiB)	Peak (MiB)	Acc
2	10.1	86.5	82.8
4	20.3	96.7	84.4
8	40.6	117.0	89.2
16	81.1	157.5	88.9
32	162.2	238.6	89.3
all (62)	314.4	390.9	89.6

5 Conclusion

LARK compresses the KV cache of long-CoT reasoning by archiving evicted history rather than discarding it: **LEAP** ranks keys with a single projection, and **LACE** keeps evicted tokens as low-rank factors that rejoin exact attention, all on a side stream that preserves CUDA-graph capture. On Qwen3-8B at 32k, LARK matches dense accuracy (MATH-500 90.4 vs. 90.6) on 5.0% of the dense device KV memory at 2.0–3.3× iso-batch throughput. We hope LARK helps bring long-horizon reasoning within reach of memory-constrained serving: letting a single commodity GPU sustain tens of thousands of chain-of-thought tokens for agentic, multi-step tasks without trading away accuracy or throughput.

AI USE STATEMENT

In this work, we used generative AI tools for code assistance during implementation and for polishing text. All AI-assisted work has been reviewed. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Complete hyperparameter configurations are provided in Appendix G. All methods are served by Mini-SGLang, the compact reference implementation of SGLang released by its developers under the MIT licence, so the host engine underlying every reported number is public; Appendix G states what we add to it. Implementation details including CUDA graph integration and variable-length batching are described in §3.3. Evaluation protocols and grading scripts are specified in §4.1.

REFERENCES

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. Training-free long-context scaling of large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 1493–1510. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/an24b.html.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling, 2025. URL https://arxiv.org/abs/2406.02069.

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. R-kv: Redundancy-aware kv cache compression for reasoning models, 2026. URL https://arxiv.org/abs/2505.24133.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: Compressing kv-cache with low-rank projection, 2024. URL https://arxiv.org/abs/2407.21118.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng

Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948. Published version: *Nature* 645:633–638, 2025.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference, 2025. URL https://arxiv.org/abs/2407.11550.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. Seerattention: Learning intrinsic sparse attention in your llms, 2025. URL https://arxiv.org/abs/2410.13276.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms, 2024. URL https://arxiv.org/abs/2310.01801.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization, 2025. URL https://arxiv.org/abs/2401.18079.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention, 2024. URL https://arxiv.org/abs/2407.02490.

Dongwon Jo, Jiwon Song, Yulhwa Kim, and Jae-Joon Kim. Fastkv: Decoupling of context reduction and kv cache compression for prefill-decoding acceleration, 2026. URL https://arxiv.org/abs/2502.01068.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An efficient error reduction framework for KV cache compression in LLM inference. In Mehdi Rezagholizadeh, Peyman Passban, Soheila Samiee, Vahid Partovi Nia, Yu Cheng, Yue Deng, Qun Liu, and Boxing Chen (eds.), *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pp. 305–321. PMLR, 14 Dec 2024. URL https://proceedings.mlr.press/v262/kang24a.html.

Janghyeon Kim, Minsoo Kim, Kyuhong Shim, and Jungwook Choi. BeaconKV: Key-value cache compression guided by beacon queries for efficient large reasoning model inference. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=bgAbffQCrL.

Junhyuck Kim, Jongho Park, Jaewoong Cho, and Dimitris Papailiopoulos. Lexico: Extreme kv cache compression via sparse coding over universal dictionaries, 2024. URL https://arxiv.org/abs/2412.08890.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 155–172, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/lee.

Tenghui Li, Guoxu Zhou, Xuyang Zhao, Yuning Qiu, and Qibin Zhao. Efficient low rank attention for long-context inference in large language models, 2025. URL https://arxiv.org/abs/2510.23649.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation, 2024. URL https://arxiv.org/abs/2404.14469.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models, 2024a. URL https://arxiv.org/abs/2405.14366.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. Chunkkv: Semantic-preserving kv cache compression for efficient long-context llm inference, 2025. URL https://arxiv.org/abs/2502.00299.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. Cachegen: Kv cache compression and streaming for fast large language model serving, 2024b. URL https://arxiv.org/abs/2310.07240.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time, 2023. URL https://arxiv.org/abs/2305.17118.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 21–27 Jul 2024c. URL https://proceedings.mlr.press/v235/liu24bz.html.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms, 2025. URL https://arxiv.org/abs/2502.13189.

Weian Mao, Xi Lin, Wei Huang, Yuxin Xie, Tianfu Fu, Bohan Zhuang, Song Han, and Yukang Chen. Triattention: Efficient long reasoning with trigonometric KV compression. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=0tgzJK50Jz.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 37396–37412. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/nawrot24a.html.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state rnns, 2024. URL https://arxiv.org/abs/2401.06104.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Akshat Ramachandran, Marina Neseem, Charbel Sakr, Rangharajan Venkatesan, Brucek Khailany, and Tushar Krishna. ThinKV: Thought-adaptive KV cache compression for efficient reasoning models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=M3CeHnZKNC.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57355–57373. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/sun25b.html.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient kv cache compression through retrieval heads, 2024a. URL https://arxiv.org/abs/2407.15891.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/tang24l.html.

Qian Tao, Wenyuan Yu, and Jingren Zhou. AsymKV: Enabling 1-bit quantization of KV cache with layer-wise asymmetric quantization configurations. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 2316–2328, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.158/.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks, 2024a. URL https://arxiv.org/abs/2407.08454.

Zihao Wang, Bin Cui, and Shaoduo Gan. Squeezeattention: 2d management of kv-cache in llm inference via layer-wise optimal budget, 2024b. URL https://arxiv.org/abs/2404.04793.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory, 2024a. URL https://arxiv.org/abs/2402.04617.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads, 2024b. URL https://arxiv.org/abs/2410.10819.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024c. URL https://arxiv.org/abs/2309.17453.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Amir Zandieh, Majid Daliri, and Insu Han. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead, 2024. URL https://arxiv.org/abs/2406.03482.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate, 2025. URL https://arxiv.org/abs/2504.19874.

Hossein Entezari Zarch, Lei Gao, Chaoyi Jiang, and Murali Annavaram. Delta: Dynamic layer-aware token attention for efficient long-context reasoning, 2026. URL https://arxiv.org/abs/2510.09883.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (aime) 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. CaM: Cache merging for memory-efficient LLMs inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58840–58850. PMLR, 21–27 Jul 2024a. URL https://proceedings.mlr.press/v235/zhang24n.html.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023. URL https://arxiv.org/abs/2306.14048.

Zhenyu Zhang, Shiwei Liu, Runjin Chen, Bhavya Kailkhura, Beidi Chen, and Zhangyang Wang. Q-hitter: A better token oracle for efficient llm inference via sparse-quantized kv cache. In P. Gibbons, G. Pekhimenko, and C. De Sa (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 381–394, 2024b. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/bbb7506579431a85861a05fff048d3e1-Paper-Conference.pdf.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VqkAKQibpq.

CONTENTS OF THE APPENDIX

A Related Work: A Survey of KV-Cache Compression . . . 17
A.1 Positioning . . . 18
B The LEAP Scoring Operator: Derivation . . . 18
C Memory Ledger of the Archival Pipeline . . . 19
C.1 Closed-Form Component Costs . . . 19
C.2 Instantiated Footprints . . . 20
D Supplementary Evaluations . . . 21
D.1 Termination Behavior Across the Full Sweep . . . 21
D.2 Comparison with KV Quantization . . . 23
D.3 Long-Prefill Compression on LongBench . . . 24
D.4 Decode Efficiency and the fp8 Pool . . . 25
E Qualitative Trace: Importance Tiering on a Real AIME Problem . . . 25
F Algorithmic Mechanism: Scoring Form and Segment-Level Tiering . . . 28
F.1 Pseudocode of the LARK Pipeline . . . 28
F.2 Why Linear Unnormalized Scoring Rather Than Softmax-Mass Scoring . . . 29
F.3 Relation to Trigonometric Pre-RoPE Scoring . . . 31
F.4 Reusing the LEAP Score for Segment-Level Tiering . . . 32
F.5 Mechanistic Reading of the Scoring Formula . . . 33
G Deployment Configuration and Reproducibility . . . 35
H Limitations and Future Work . . . 36

NOTATION

Table 6 collects the symbols used throughout the paper, grouped by type; calligraphic letters denote sets and operators.

Table 6: Symbols and description in this paper.

Symbols	Description
Model and configuration scalars	
$n_L,\ H,\ H_{\mathrm{kv}}$	Number of transformer layers; of query heads; of KV heads under grouped-query attention.
$d,\ F{=}H_{\mathrm{kv}}d$	Per-head dimension; flattened KV feature width of one layer.
$p,\ t,\ t_0$	Cached-key position; decode step (query position); time of a compression event.
$\tau,\ \tau_j{=}2^{j-1},\ J$	Future offset ($\tau \geq 1$); geometric offset grid; grid size ($J{=}17$).
$P,\ m$	Probe count $\lvert\mathcal{P}\rvert$; number of eviction candidates.
$B,\ L,\ W$	Active budget in tokens; segment length, i.e. the eviction-buffer length ($L{=}512$); attention window over the accumulating buffer ($W{=}128$).
$r_k,\ r_v,\ \kappa$	Archival ranks of the key and value factors (64 and 32); residency cap, i.e. the number of GPU-resident segments (top-κ).
$T,\ N_{\mathrm{seg}}(T)$	Generated tokens of a request; number of archived segments (Eq. 9).
$b_{\mathrm{kv}}, b_p, b_f, b_s, b_c$	Byte widths of an uncompressed KV element, a pool element, a factor element, a per-rank scale, and a centroid element.
$\ell_t(p),\ a_t(p),\ Z_t$	Attention logit, attention mass, and softmax normalizer at step t.
$\rho,\ c_f,\ \psi_f^0$	Compression factor of an archived segment (Eq. 13); per-frequency amplitude weight and phase shift induced by D.
Importance scores	
$S_h^*(p),\ \hat{S}_h(p)$	Ideal importance, i.e. the π_h-weighted future attention mass of key p at head h, and its expected-logit estimate.
$S_h^{\mathrm{pre}}(p)$	Pre-RoPE phase scorer (Eq. 3) and its equivalent post-RoPE form (Eq. 4).
$S(p),\ S_h^{\mathrm{seg}}(s)$	Deployed token-level score driving budget selection; centroid score of archived segment s (Eq. 6).
$\pi_h(\tau)$	Normalized weight over future offsets at head h.
Memory ledger (Appendix C)	
$M_{\mathrm{tok}}, M_{\mathrm{tok}}^{(p)}, M_{\mathrm{dense}}, M_{\mathrm{pool}}$	Bytes of one KV token at the bf16 reference width and in the fp8 pool; uncompressed cache after T tokens; footprint of the active pool.
$S_{\mathrm{fac}}, S_{\mathrm{cen}}, S_{\mathrm{seg}}$	Bytes of one segment's factors, of its centroids, and their sum (Eq. 12).
$M_{\mathrm{gpu}}, M_{\mathrm{host}}, M_{\mathrm{total}}$	Device, host, and conservatively summed footprints (Eq. 14).
Vectors	
q_t, k_p; $\tilde{q}_t, \tilde{k}_p$	Base-frame (pre-RoPE) and cached post-RoPE query and key; $\tilde{k}_{p,h}$ is the head-h slice.
$q_{t,h}^{\mathrm{base}},\ \bar{\mu}_h$	Probe query inverse-rotated back to the base frame, and the online base-frame expected query of head h (Eq. 2).
$\tilde{\mu}_{h,\tau}$	Expected post-RoPE query at future offset τ.
$\bar{W}_{h,j},\ \bar{W}_h,\ \bar{W}$	LEAP scoring direction per offset, after grid folding, and stacked over heads.
$\bar{k}_s$	Centroid of archived segment s, $\bar{k}_s = \mathrm{mean}(A_s)B_s$.
$\ell_s,\ w_s,\ o$	Factorized logits of a resident segment, its softmax weights, and the attention output of a decode step.
Matrices	
$R(\theta),\ \mathrm{Rot}(\cdot),\ \omega_j$	Block-diagonal RoPE rotation, its $2{\times}2$ block, and the block frequencies; $R(a)R(b) = R(a{+}b)$.
$D = \frac{1}{J}\sum_j R(\tau_j)$	Grid-folding constant, precomputed offline per model.
$K_s,\ V_s \in \mathbb{R}^{L\times F}$	Post-RoPE keys and values of one segment, flattened over heads.
A_k, B_k (A_v, B_v); Q	Rank-r_k (r_v) factors of K_s (V_s), $K_s \approx A_k B_k$ (Eq. 5), with subscript s indexing the segment; Q is the orthonormal basis from the CholQR power iteration.
Sets and operators	
$\mathcal{P},\ \mathcal{B},\ \mathcal{T},\ \mathcal{E}$	Probe ring of recent and distant query offsets (32 slots); fp8 active budget pool ($\lvert\mathcal{B}\rvert \leq B$); accumulating buffer of tokens appended since the last event ($\lvert\mathcal{T}\rvert < L$); length-$L$ eviction buffer.
$\mathcal{A},\ \mathcal{R}$	Archive of low-rank segments and its GPU-resident subset ($\lvert\mathcal{R}\rvert \leq \kappa$).
$\mathcal{Q}(\cdot), \mathrm{rSVD}(\cdot, r),$ $\mathrm{CholQR}(\cdot), \mathrm{mean}(\cdot)$	fp8 (e4m3) quantizer with per-column scales; randomized SVD at rank r; its GEMM-only orthogonalization; the row mean.
$\mathrm{FlashAttn}(\cdot),$ $\mathrm{OnlineMerge}(\cdot), \mathbb{E}[\cdot], \Theta(\cdot)$	Flash attention over the active pool; online-softmax merge of the pool and factor branches; expectation; asymptotic cost.

A Related Work: A Survey of KV-Cache Compression

This appendix extends the related-work discussion of the main text. It groups prior work into five families (the four compressive routes of §1, plus the sparse-read line, which reduces what is read rather than what is stored) and, within each family, into the technical routes that family has taken.

Token selection and eviction. Four routes divide this family. (i) *Position priors*: retain attention sinks plus a sliding window, at fixed cost and with no scoring at all (Xiao et al., 2024c). (ii) *Observed-mass scoring*: rank candidates by the attention they have already received, either in one pass over an observation window at the end of the prompt (Li et al., 2024) or continuously during decoding (Zhang et al., 2023; Liu et al., 2023; Oren et al., 2024; Zhang et al., 2024b); the variants differ in the statistic (cumulative mass, persistence, entropy, query-conditioned mass) rather than in its provenance. (iii) *Budget allocation*: spend a global budget unevenly across structural axes, per head (Ge et al., 2024; Feng et al., 2025), per layer (Cai et al., 2025), or over the joint layer–token space (Wang et al., 2024b), with Ada-KV deriving the allocation from a bound on the eviction-induced attention-output error. (iv) *Coarser granularity*: evict spans rather than tokens, at chunk (Liu et al., 2025) or thought (Ramachandran et al., 2026) level. Whatever the route, the signal is *past observed* attention and the discard is irreversible, so a token that only becomes important later cannot be recovered (§2.2); the scores additionally inherit softmax-normalization dilution and recency/sink bias. LARK replaces the signal with an expected-logit score and the discard with a recoverable archive.

Quantization and serialized transfer. Two routes, both shrinking bytes rather than the token set. (i) *Low-precision storage*, differing in where the quantization grid is fitted: per channel for keys and per token group for values (Liu et al., 2024c), calibrated non-uniform grids (Hooper et al., 2025), sign quantization after an inner-product-preserving Johnson–Lindenstrauss transform (Zandieh et al., 2024), residual quantization with a sparse correction (Kang et al., 2024), and online vector quantization at near-optimal distortion (Zandieh et al., 2025). (ii) *Layout and transport*: bitstream serialization for cross-request and cross-node reuse (Liu et al., 2024b), and paged memory management (Kwon et al., 2023)—not compression, but the engineering basis of our slotmap layer (§3.3). Precision is orthogonal to which tokens are kept, so this family stacks with LARK instead of competing with it; our own ledger books both tiers as fp8 (Appendix C).

Sparse attention and retrieval. Three routes, all reducing what is *read* rather than what is stored. (i) *Offline pattern fitting*: assign each head a sparse mask and execute it with matching kernels (Jiang et al., 2024), or make the sparsity a trainable part of the architecture (Yuan et al., 2025). (ii) *Per-query block selection*: index history in blocks and fetch only the promising ones, chosen by cheap upper bounds on the attention score (Tang et al., 2024b), by representative tokens over host-resident blocks (Xiao et al., 2024a), by a learned router (Lu et al., 2025), or by learned gates (Gao et al., 2025). (iii) *Head specialization*: give retrieval heads the full cache and streaming heads a constant-size one (Xiao et al., 2024b). Because these methods delete nothing, they are complementary to LARK: they sparsify the read path per query while leaving the stored representation intact, whereas LARK reshapes the storage hierarchy itself.

Merging and aggregation. Two routes, both trading exactness for a smaller cache. (i) *Along the token axis*: fold to-be-evicted entries into surviving ones, weighted by attention prominence (Zhang et al., 2024a) or under a dual-cache policy (Nawrot et al., 2024). (ii) *Along the depth axis*: merge the KV states of adjacent middle-to-deep layers whose representations are near-duplicates (Liu et al., 2024a). The dividing line from LARK is structural: merged directions cannot be separated again, whereas the rank set aside by archival stays reachable through factorized attention.

Low-rank projection and offloading. Two routes, both reshaping the stored representation rather than discarding it. (i) *Projected storage*: shrink the KV feature dimension with a learned or fitted basis, by weight-side projection (Chang et al., 2024), a universal sparse-coding dictionary (Kim et al., 2024), or joint query–key factors (Li et al., 2025). (ii) *Low-rank-guided offloading*: keep a compact proxy on the device to decide what to fetch from a host-resident full cache (Sun et al., 2025; Lee et al., 2024; Tang et al., 2024a). This family shares LARK's vocabulary but not its mechanism, and three differences separate them: these decompositions are static, offline and global, whereas

LARK's are online, per segment and committed incrementally; their factors serve as prefetch hints or reconstruction approximations, whereas LARK's enter attention directly with no reconstruction; and none couples decomposition to the retention decision, whereas LARK drives both from one linear score. Their budgets are also not commensurable with ours: a budget here bounds only the device-resident proxy, while the full cache is retained off-device and every retrieved token is reconstructed before use, so at a matched B both the footprint and the per-step arithmetic stay far above an eviction baseline's. We therefore restrict the iso-budget comparison to eviction methods, whose budget fixes memory and compute at once, and treat this family as complementary.

A.1 Positioning

Prior work answers *what to keep* with observation-based heuristics and *how to save* with quantization, merging, sparse reads or static decomposition; none provides all three of a future-oriented signal, recoverable discards, and compressed factors that participate directly in attention. LARK occupies that intersection, and the per-family dividing lines above, together with Appendix F.3 against the TriAttention line, separate it from each family.

Scope of the experimental comparison. Our protocol fixes the weights, the engine and the grader and sweeps one controlled variable, the device-resident KV budget B; every recent method that accepts that variable is included. SnapKV, R-KV (Cai et al., 2026), Ada-KV, H_2O and the two ICML 2026 entries BeaconKV and TriAttention together span window aggregation, redundancy-aware selection, per-head allocation, accumulated-mass scoring, beacon queries and trigonometric compression, and KIVI joins the same axis in bytes rather than tokens (Appendix D.2). The sparse-read line has no such knob: it keeps the whole history resident and spends its budget on how much of it to read, so it composes with LARK rather than substituting for it.

B The LEAP Scoring Operator: Derivation

§3.1 states the post-RoPE form of the scorer (Eq. 4) and the single-weight collapse $\bar{W}_h = \frac{1}{\sqrt{d}} R(t_0) D \hat{\mu}_h$ without proof; both are derived here, together with the exactness of the segment score (Eq. 6).

Exact position cancellation. Fix a head and omit h. Splitting the rotation by $R(a)R(b) = R(a+b)$ gives $R(p - t_0 - \tau_j) = R\big(-(t_0+\tau_j)\big)R(p)$; inserting this into the pre-RoPE scorer (Eq. 3) and using $\tilde{k}_p = R(p)k_p$ and $R(-a)^\top = R(a)$,

$$\hat{\mu}^\top R(p - t_0 - \tau_j)\, k_p \;=\; \hat{\mu}^\top R\big(-(t_0+\tau_j)\big)\tilde{k}_p \;=\; \big(R(t_0+\tau_j)\hat{\mu}\big)^\top \tilde{k}_p\,. \tag{7}$$

Averaging over j yields Eq. 4 with $\bar{W}_j = \frac{1}{\sqrt{d}} R(t_0+\tau_j)\hat{\mu}$. Intuitively, the $R(p)$ already contained in $\tilde{k}_p$ and the $R(-p)$ implicit in $R(p - t_0 - \tau_j)$ compose to the identity, so p enters only through the cached post-RoPE key: neither inverse RoPE nor a pre-RoPE copy is required, and each candidate costs one inner product per offset.

Folding the offset grid: explicit D. Eq. 4 is linear in the weights, so any fixed aggregation over offsets moves to the statistics side. The deployed scorer uses the grid mean (a weighted π_h is identical, with $D_\pi = \sum_j \pi_j R(\tau_j)$ below). From $R(t_0+\tau_j) = R(t_0)R(\tau_j)$,

$$\bar{W} \;=\; \frac{1}{J}\sum_j \bar{W}_j \;=\; \frac{1}{\sqrt{d}}\, R(t_0)\Big(\frac{1}{J}\sum_j R(\tau_j)\Big)\hat{\mu} \;=:\; \frac{1}{\sqrt{d}}\, R(t_0)\, D\, \hat{\mu}\,, \tag{8}$$

where $D = \frac{1}{J}\sum_j R(\tau_j)$ is block-diagonal with 2×2 blocks $\frac{1}{J}\sum_j \mathrm{Rot}(\tau_j \omega_f)$ and depends only on the RoPE frequencies and the grid; it is computed once per model. Scoring m candidates is therefore the single projection $[\tilde{k}_1, \ldots, \tilde{k}_m]^\top \bar{W}$ at cost $\Theta(md)$, independent of both J and the probe count that formed $\hat{\mu}$.

Segment scores are exact, not approximate. The same linearity makes the archived-segment score (Eq. 6) exactly the mean of the per-key scores inside the segment: with factors (A_k, B_k) and $\tilde{k}_p \approx (A_k)_p B_k$, $\frac{1}{|s|}\sum_{p\in s} \bar{W}^\top \tilde{k}_p = \bar{W}^\top \Big(\frac{1}{|s|}\sum_{p\in s}(A_k)_p\Big) B_k = \bar{W}^\top \bar{k}_s$, using $\bar{k}_s = \mathrm{mean}(A_k)B_k$; with the unfolded weights this is $\frac{1}{J}\sum_j \bar{W}_j^\top \bar{k}_s$. One d-dimensional projection thus scores a whole segment without reconstructing it, and no separate scorer needs to be calibrated. All folds are per head, so the equivalence holds head by head.

C Memory Ledger of the Archival Pipeline

This appendix derives the memory footprint of each component of LARK in closed form and then instantiates it for two of the evaluated models. All quantities are exact byte counts of the deployed data structures; no compression ratio is asserted beyond what the formulas give.

C.1 Closed-Form Component Costs

Notation. Let n_L be the number of layers, H_{kv} the number of KV heads, d the head dimension, and $F = H_{\mathrm{kv}} d$ the flattened feature width per layer. Let b_{kv}, b_p, b_f, b_s, b_c denote the byte widths of an uncompressed KV element, the active-pool element, the archived factor element, the per-rank scale, and the segment centroid, respectively; in the deployed configuration $b_{\mathrm{kv}}{=}b_s{=}b_c{=}2$ (bf16) and $b_p{=}b_f{=}1$ (fp8-e4m3). The active budget is B tokens, the segment length L, the archival ranks r_k and r_v, and the GPU residency cap κ segments. For a request that generates T tokens, the number of archived segments is

$$N_{\mathrm{seg}}(T) \;=\; \max\big(0,\; \big\lfloor \tfrac{T-B}{L} \big\rfloor\big)\,. \tag{9}$$

Uncompressed reference. One token of KV, stored for all layers and both K and V, costs

$$M_{\mathrm{tok}} \;=\; 2\, n_L H_{\mathrm{kv}} d\, b_{\mathrm{kv}}, \qquad M_{\mathrm{dense}}(T) \;=\; T\, M_{\mathrm{tok}}. \tag{10}$$

Active budget pool. The pool stores B tokens quantized to fp8 (e4m3) with per-column scales (§3.2), hence

$$M_{\mathrm{tok}}^{(p)} \;=\; 2\, n_L H_{\mathrm{kv}} d\, b_p, \qquad M_{\mathrm{pool}} \;=\; B\, M_{\mathrm{tok}}^{(p)}. \tag{11}$$

The budget-only column of Table 7 charges the same B tokens at the bf16 reference width M_{tok} of Eq. 10, whereas every LARK total below charges them at $M_{\mathrm{tok}}^{(p)} = M_{\mathrm{tok}}\, b_p / b_{\mathrm{kv}}$, half as much.

Eviction buffer. Compression fires whenever the retained set reaches $B{+}L$, at which point the L lowest-scoring tokens leave, so the model cache oscillates in $[B, B{+}L]$ and contributes a steady-state mean of $L/2$ tokens above the budget. Residency and attention are two distinct windows here, and only the first is a memory quantity: within a period the newest $W{=}128$ tokens of the accumulating buffer are the ones that participate in attention (Appendix G), while the rest of the buffer stays resident in the fp8 pool, is charged in full below, and enters the candidate set of the next compression event exactly like the attended part. The $[B, B{+}L]$ oscillation and the $L/2$ steady-state term are therefore independent of W, which bounds the per-step attention length rather than the footprint. Two conventions for charging this oscillation are used consistently below. Under the *at-cap* convention the pool term of Eq. 11 is evaluated at the cap B, and the eviction baseline is likewise charged BM_{tok}; under the *steady-state* convention both sides are charged at $B{+}L/2$. Table 7 reports the at-cap convention, which is the peak-inclusive one; a measured allocation trace such as Figure 9 realizes the steady-state one. The distinction matters for cross-method ratios rather than for LARK alone, because the extra $L/2$ tokens are charged at bf16 on the baseline but at fp8 in LARK's pool.

One archived segment. A segment stores the four factors of Eq. 5 in fp8 together with one scale per rank per layer, and one centroid per layer for segment-level scoring. The two parts have different residency requirements, so we keep them separate:

$$S_{\mathrm{fac}} = n_L\big[(r_k{+}r_v)(L{+}F)\, b_f + (r_k{+}r_v)\, b_s\big], \qquad S_{\mathrm{cen}} = n_L F\, b_c, \qquad S_{\mathrm{seg}} = S_{\mathrm{fac}} + S_{\mathrm{cen}}. \tag{12}$$

Amortizing a full segment over the L tokens it replaces gives an effective per-token archival cost of S_{seg}/L, and hence a compression factor relative to the uncompressed reference of

$$\rho \;=\; \frac{M_{\text{tok}}}{S_{\text{seg}}/L} \;=\; \frac{2H_{\text{kv}}\,d\,b_{\text{kv}}\,L}{(r_k{+}r_v)(L{+}F)\,b_f + (r_k{+}r_v)\,b_s + F\,b_c}, \tag{13}$$

which is independent of n_L and of T. With $L{\ll}F$ the factor term is dominated by the shared bases B_k, B_v rather than by the per-token codes A_k, A_v, so ρ saturates as L grows; conversely ρ scales linearly in L while $L{<}F$.

Device and host totals. Host memory holds every segment as the authoritative copy. On the device, the factors of at most κ segments are resident, but *all* centroids must be available because every compression event re-scores the entire archive to choose the top-κ (§3.2). Hence

$$M_{\text{gpu}}(B,T) = B\,M_{\text{tok}}^{(p)} + \min\big(N_{\text{seg}},\kappa\big)S_{\text{fac}} + N_{\text{seg}}S_{\text{cen}}, \qquad M_{\text{host}}(T) = N_{\text{seg}}\,S_{\text{seg}}. \tag{14}$$

The factor term is bounded, constant in T once $N_{\text{seg}}{\geq}k$, while the centroid term grows in T at S_{cen}/L bytes per token, two orders of magnitude below M_{tok} and therefore negligible in practice but not formally constant. LARK thus converts an unbounded device footprint into a nearly bounded device footprint plus a slowly growing host footprint. Single-number claims in the main text are device-side, i.e. M_{gpu} at the generation limit; alongside it we state the conservative sum $M_{\text{total}} = BM_{\text{tok}}^{(p)} + N_{\text{seg}}S_{\text{seg}}$, which charges every segment once at full cost, and Table 8 compares the two axes against the baselines.

Depth invariance of the relative footprint. Every segment term carries the same factor n_L as M_{tok} does, so the ratios

$$\frac{S_{\text{fac}}}{M_{\text{tok}}} = \frac{(r_k{+}r_v)(L{+}F)b_f + (r_k{+}r_v)b_s}{2H_{\text{kv}}\,d\,b_{\text{kv}}}, \qquad \frac{S_{\text{cen}}}{M_{\text{tok}}} = \frac{F\,b_c}{2H_{\text{kv}}\,d\,b_{\text{kv}}} \tag{15}$$

are free of n_L. Consequently $M_{\text{gpu}}/M_{\text{dense}}$ and $M_{\text{total}}/M_{\text{dense}}$ depend only on B/T, κ, L, the ranks, F, and the fixed byte widths—not on model depth. Two models that share (H_{kv}, d) therefore have identical *relative* footprints even though their absolute footprints differ; Table 7 therefore lists absolute footprints for Qwen3-8B only, with one set of fractions shared by both models.

C.2 Instantiated Footprints

Table 7 evaluates Eqs. 9–15 at the generation limit $T{=}32{,}768$ for the deployed configuration $L{=}512$, $r_k{=}64$, $r_v{=}32$ under the deployed cap schedule (§4.1). Both tabulated models have $H_{\text{kv}}{=}8$ and $d{=}128$, hence $F{=}1024$, and differ only in depth ($n_L{=}36$ for Qwen3-8B, 32 for R1-Distill-Llama-8B), giving $M_{\text{tok}}{=}0.141/0.125$ MiB, $S_{\text{fac}}{=}5.07/4.51$ MiB and $S_{\text{cen}}{=}0.07/0.06$ MiB respectively; the fp8 pool is charged at $M_{\text{tok}}^{(p)}{=}M_{\text{tok}}/2$ (0.070/0.063 MiB), and Eq. 13 gives $\rho{=}14.0$ for both, i.e. an archived token costs $1/14$ of an uncompressed one.

Table 7: Memory ledger (MiB) at the generation limit $T{=}32{,}768$ tokens, evaluated for Qwen3-8B ($M_{\text{dense}}(T){=}4{,}608$ MiB). LARK's pool stores the B retained tokens in fp8 and is charged at $M_{\text{tok}}^{(p)}{=}M_{\text{tok}}/2$ inside M_{gpu} and M_{total} (Eq. 11); the residency cap follows the deployed schedule $\kappa{=}8$ for $B{\leq}1024$ and $\kappa{=}16$ for $B{>}1024$. The fractions of $M_{\text{dense}}(T)$ are depth-invariant (Eq. 15) and therefore apply equally to R1-Distill-Llama-8B ($M_{\text{dense}}(T){=}4{,}096$ MiB).

		on device: M_{gpu}		**device + host:** M_{total}	
budget B	**archived** N_{seg}	Qwen3-8B (MiB)	fraction of M_{dense}	Qwen3-8B (MiB)	fraction of M_{dense}
512	63	81.0	1.8%	359.8	7.8%
1024	62	116.9	2.5%	390.6	8.5%
2048	60	229.3	5.0%	452.4	9.8%
4096	56	373.0	8.1%	575.8	12.5%

Composition of the footprint. The archive is a fixed cost (288 to 324 MiB on host and device combined across the sweep), while the fp8 pool grows from 36 MiB at B=512 to 288 MiB at B=4096, so the archive's share of the footprint falls as the budget grows. Two thirds of a segment sits in the shared bases B_k, B_v rather than the per-token codes A_k, A_v, since the basis term rF of Eq. 12 exceeds the code term rL whenever $F{>}L$, as at F=1024, L=512. Sharing bases across segments or lowering the basis-side rank would shrink the archive further; the results here do not require it.

A matched budget is not a matched byte count. An eviction baseline at budget B holds BM_{tok} in bf16 and nothing else; LARK holds the same B tokens in fp8 at half that, and spends the difference on resident factors and all centroids. Eq. 14 then puts the device-side ratio to the baseline at $1/2 + [\min(N_{\text{seg}}, \kappa)S_{\text{fac}} + N_{\text{seg}}S_{\text{cen}}]/(BM_{\text{tok}})$, i.e. 1.13×, 0.81×, 0.80×, 0.65× over the four budgets at the 32k cap, which pins the pool to its trough while billing the archive at its peak, and 0.92×, 0.75×, 0.76×, 0.64× under the steady-state convention ($B{+}L/2$ in both pool terms, since archival fires every L tokens and the pool oscillates in $[B, B{+}L]$), below parity throughout because those $L/2$ tokens cost the baseline twice what they cost the fp8 pool. Both series decrease with B, and decrease further at the measured decode lengths, where N_{seg} sits below the cap: the measurement of §4.3 lands at 0.70× at B=1024, below both series. The sweep of §4.2 is therefore iso-*budget* with LARK carrying a recallable archive at no extra device memory from B=1024 on, while at B=512 the two conventions bracket parity and the fp8 saving very nearly pays for the archive outright.

The iso-*byte* axis. The device axis is the operative one: HBM, not host DRAM, bounds batch size and hence throughput (§4.3), and the pinned host copy transfers off the critical path (Table 2). Table 8 prices both axes on GPQA Diamond against the strongest baseline at LARK's byte count, interpolated log-linearly in bytes between the swept budgets. On the device axis LARK leads by 6.5 points at the tightest budget, narrowing as the fixed archive amortizes. Charging the host copy at HBM prices costs 4.2 points at B=512 when every segment is billed at the 32k cap; at the measured average lengths (18 archived segments rather than 63), LARK leads at three of the four budgets and trails by 0.4 at the fourth.

Table 8: Accuracy gap (points) of LARK over the strongest eviction baseline *at equal bytes* on GPQA Diamond (Qwen3-8B, sampling, Avg@4, the protocol of §4.1). Baselines are charged BM_{tok} (bf16, device, 72/144/288/576 MiB); LARK is charged M_{gpu} or M_{total} (Eq. 14), each evaluated either at the 32k cap or at the measured average length, with LARK's four footprints listed in the row label. Baseline accuracy at LARK's byte count is interpolated log-linearly in bytes between the swept budgets.

axis charged to LARK (footprint at B=512/1024/2048/4096)	**gap at budget B**			
	512	1024	2048	4096
M_{gpu}, at cap (81/117/229/373 MiB)	+6.5	+3.2	+1.6	+0.1
M_{gpu}, measured (78/113/206/324 MiB)	+7.1	+3.7	+1.8	+0.1
M_{total}, at cap (360/391/452/576 MiB)	−4.2	−2.1	−0.3	−0.2
M_{total}, measured (129/144/206/324 MiB)	−0.4	+0.1	+1.8	+0.1

D Supplementary Evaluations

D.1 Termination Behavior Across the Full Sweep

Figure 8 showed the coupling between budget exhaustion and generated length at a single (model, budget) point. Figures 10 and 11 give the same two diagnostics over the complete sweep: four models (rows) × four budget steps (columns) × four benchmarks (bar groups within a panel), with dense repeated in every panel as a budget-free reference. Because MATH-500 is swept one octave lower, a column holds the same budget step of each benchmark rather than one shared value, and each tick label carries the budget that group actually used.

Across the sweep the two diagnostics move together: in every panel the ordering of methods by budget-exhausted count reproduces their ordering by generated length, and both fall monotonically

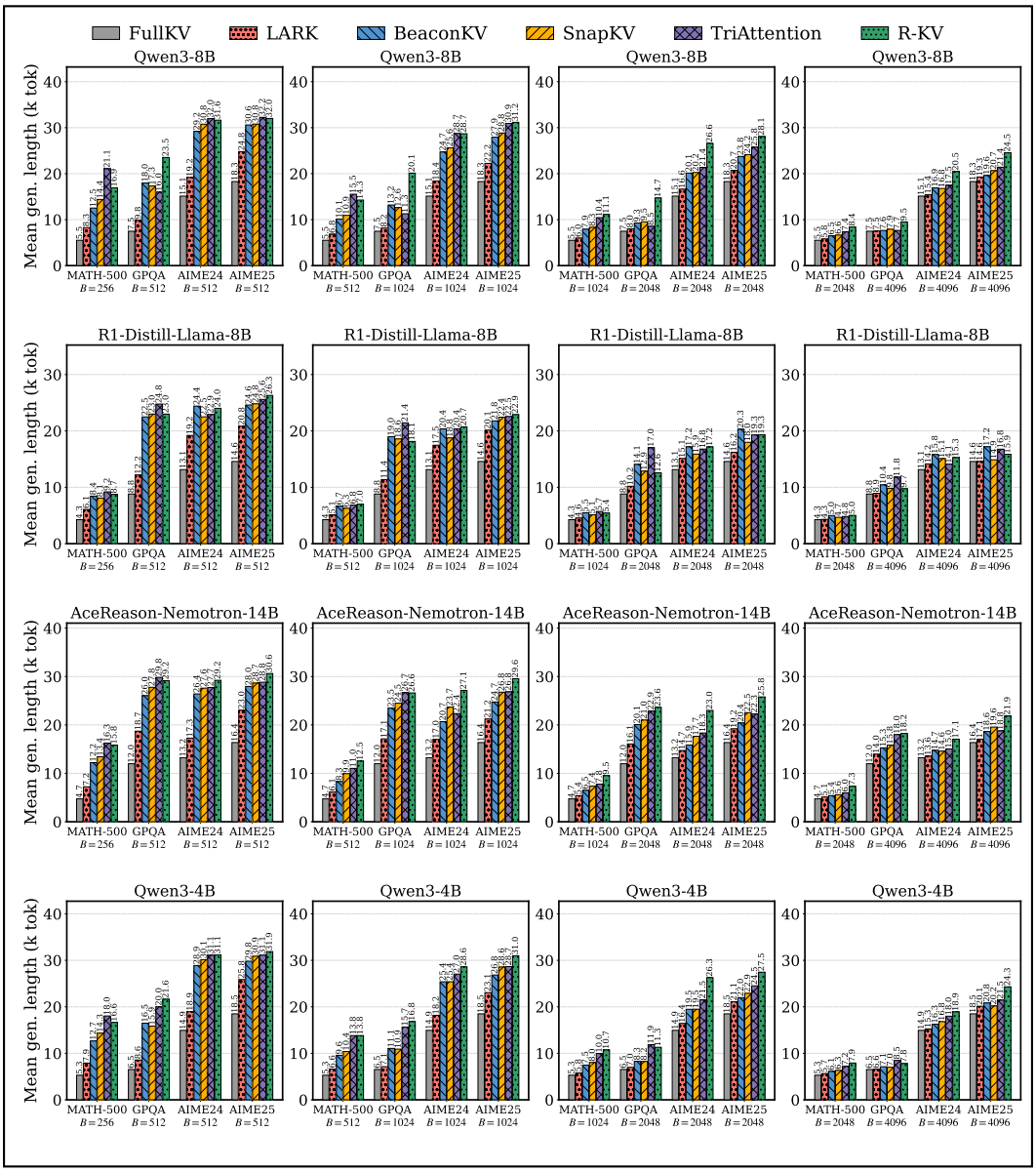


Figure 10: Mean generated length across models and KV budgets; dense is budget-free and repeated for reference.

in B. The trend is uniform, with LARK lowest on both axes in 59 of the 64 (model, benchmark, budget) cells, and the effect is largest where the budget is tightest: at a matched budget the best baseline still generates up to $1.85\times$ longer than LARK, and matching LARK's tightest setting costs an eviction baseline 4–$8\times$ the budget. The mechanism is the irreversibility argument of §2.2. Eviction shrinks the addressable context to the budget itself, so a token that leaves the window is gone for good and the model re-derives what it can no longer read; LACE's top-κ low-rank archival instead keeps the evicted history addressable at a fraction of its bytes, so the index over the context stays far longer than B and recently generated reasoning does not fall out of reach as the chain grows. Chains therefore converge rather than restate themselves, which is what turns the shorter generated lengths of Figure 10 into the lower budget-exhausted counts of Figure 11.

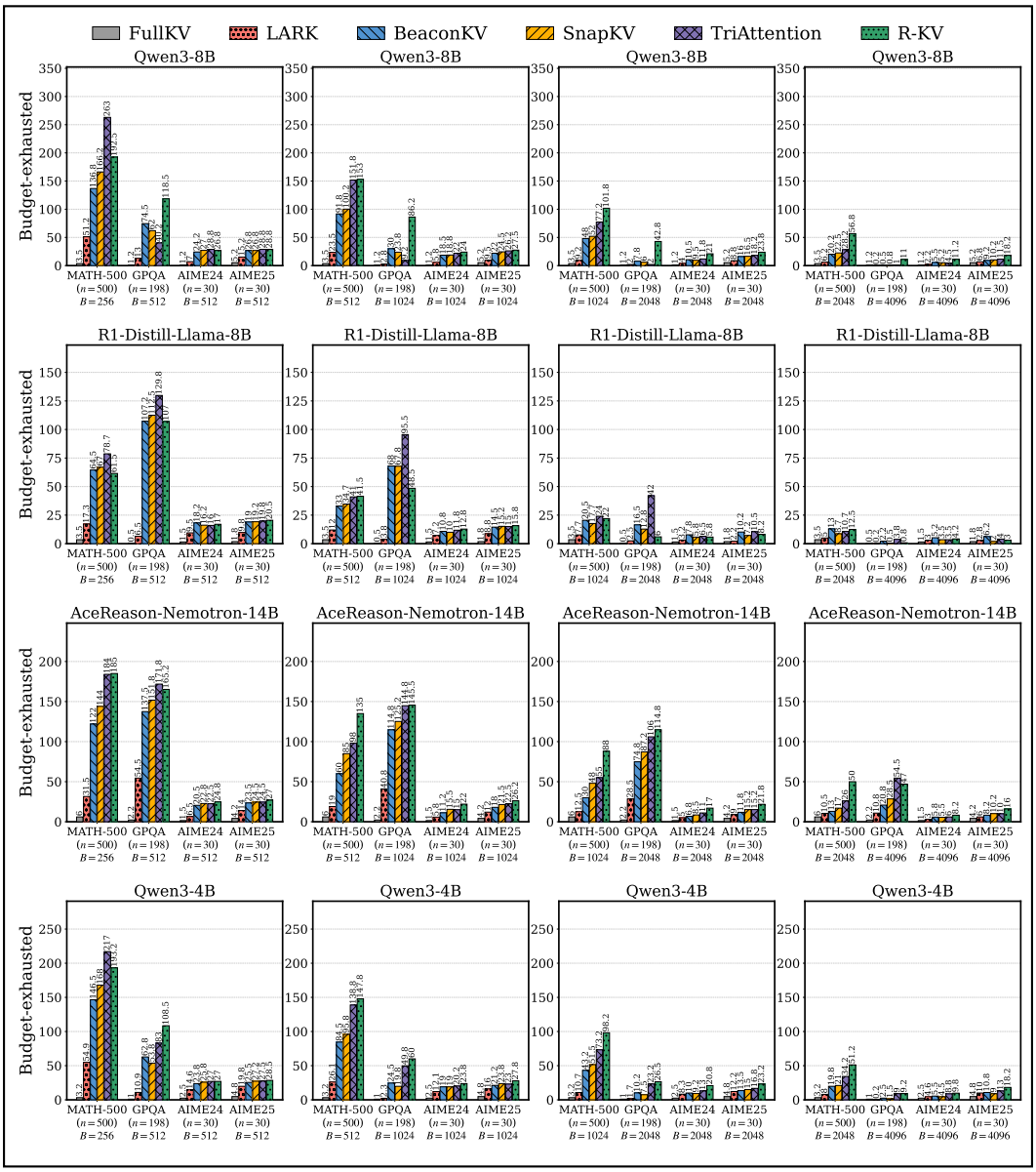


Figure 11: Responses whose decode length reached the 32k cap, out of n problems per benchmark; dense is budget-free and repeated for reference.

D.2 Comparison with KV Quantization

Table 9 reports a head-to-head on the full MATH-500 set (500 problems, 32k generation limit) between LARK at three budgets and KIVI KV-quantization at its two supported precisions. Beyond accuracy, it reports three diagnostics that expose *how* each method loses: the average generated length, the number of responses that reach the 32k limit (a truncated chain is graded wrong regardless of the reasoning before it), and the peak device KV footprint at the generation limit (KIVI read off the measured curves of Figure 9; LARK from the ledger Eq. 14 under the at-cap convention, with N_{seg}=63/62/60 and κ=8/8/16 at T=32k, i.e. the M_{gpu} column of Table 7; KIVI's footprint grows unboundedly with generated length, LARK's saturates). Ledger and measured numbers are not interchangeable within one ratio: at B=1024 the measured plateau of Figure 9 is 126 MiB against the ledger's 116.9 MiB (+8%), and 1024-token eviction measures 180 MiB against a nominal BM_{tok}=144 MiB (+25%); the excess is prefill KV and allocator residue, which Eq. 14 does not model.

Table 9: LARK vs. KIVI on the full MATH-500 (Qwen3-8B, 32k limit): MATH-500 Acc, average generated length, exhausted count (responses hitting the 32,768-token cap), and peak device KV footprint at the limit (MiB, as in Figure 9). LARK rows are the Avg@4 means of the main sweep (Figure 7), hence fractional counts; KIVI is a single pass under the same limit and grader.

Method	Acc (%)	Avg decode	Peak KV (MiB)	Exhausted (/500)
LARK, B=512	86.2	6,830	81.0	23.5
LARK, B=1024	89.2	5,981	116.9	9.2
LARK, B=2048	90.4	5,795	229.3	6.2
KIVI, 2-bit	80.8	8,611	666	60
KIVI, 4-bit	89.8	5,473	1242	9

Two contrasts stand out. First, accuracy alone understates the difference in regime: KIVI-4 comes within 0.8 of dense (89.8 vs. 90.6) by retaining every token at reduced precision, but its footprint still grows with generated length (1.21 GiB at 32k, 10.6× LARK's B=1024 footprint and growing linearly beyond it), whereas LARK's footprint is bounded by construction. LARK reaches the same accuracy without that growth, coming within 0.6 of KIVI-4 at B=1024 (89.2, 10.6× less KV) and passing it at B=2048 (90.4 against 89.8, still 5.4× less), so the bound on memory is not paid for in accuracy. Second, the failure modes differ: at B=512 LARK reaches the limit on 23.5 of 500 responses against KIVI-2's 60 at 8.2× the footprint—quantization noise, like eviction, breaks termination on the hardest chains before it degrades the rest. KIVI-2 (80.8) lands 5.4 points below even LARK's tightest budget, so blurring the whole history is worse than keeping a bounded slice of it exactly and archiving the remainder.

D.3 Long-Prefill Compression on LongBench

The main evaluations so far are decode-heavy (long reasoning chains over short prompts). We additionally evaluate the complementary *long-prefill* regime, with long inputs and short answers, on five LongBench subsets (Qasper, HotpotQA, QMSum, TriviaQA, LCC; 200 samples each), where the prompt itself dominates the KV cache. LARK compresses once at the end of prefill: the attention-sink prefix is pinned, the top-scoring fraction of prompt tokens is kept exactly (post-RoPE linear score with an observation-window probe, W=32), and the evicted tokens are archived into low-rank segments (512 tokens per segment, K/V ranks 64/32, fp8 factors) that remain attended throughout decoding; the decode region is never compressed. The keep fraction (10%/25%/50% of prompt tokens) is the single knob. We follow the kvpress evaluation protocol: official per-dataset templates, the chat template applied to the context only, head truncation at 15,500 tokens, generation capped at the dataset answer length plus 20, and the corresponding metrics (F1 for QA, ROUGE-L for QMSum, code similarity for LCC). H_2O (Zhang et al., 2023) and Ada-KV (Feng et al., 2025) are reimplemented in the same serving engine and evaluated under the identical protocol and budget convention (fraction of prompt KV physically retained; a 512-token question region is pinned for all methods, matching kvpress).

Table 10: Long-prefill compression on LongBench (Qwen3-8B, kvpress-aligned protocol, n=200 per subset). The keep ratio refers to the fraction of prompt tokens retained exactly; scores are ×100. Bold: best among compressed methods per column within each keep block.

Keep	Method	Qasper	HotpotQA	QMSum	TriviaQA	LCC	Avg
100%	Dense	43.79	63.34	24.27	90.91	68.60	58.18
10%	H2O	25.90	53.48	21.15	83.78	11.24	39.11
10%	AdaKV	25.92	53.79	20.87	84.62	11.14	39.27
10%	LARK	**40.78**	**61.91**	**23.56**	**89.27**	**69.39**	**56.98**
25%	H2O	35.73	59.67	**23.39**	89.85	44.02	50.53
25%	AdaKV	35.32	58.91	22.96	90.24	44.08	50.30
25%	LARK	**42.73**	**62.10**	23.34	**90.36**	**70.60**	**57.83**
50%	H2O	41.00	60.74	**23.96**	**90.49**	67.54	56.75
50%	AdaKV	41.27	61.27	23.92	**90.49**	67.17	56.82
50%	LARK	**45.12**	**62.19**	23.57	90.02	**70.17**	**58.21**

Table 10 reports the results. LARK's accuracy stays close to dense at every compression ratio: averaged over the five subsets it reaches 56.98/57.83/58.21 against dense's 58.18 at 10%/25%/50%

keep, and even at the most aggressive 10% setting it stays within roughly three points of dense on every subset. The eviction baselines degrade substantially once compression becomes aggressive, and the gap is largest exactly where information cannot be re-derived: on LCC, code completion depends on precise local context, so H2O/AdaKV collapse to ∼11 at 10% keep and only approach dense at 50%, whereas LARK's low-rank archive preserves the evicted content and remains close to dense across all three ratios. Consequently LARK's margin over both baselines widens as compression grows (+17.8 average points at 10%, +7.4 at 25%, +1.4 at 50%), and H2O and AdaKV agree within 0.5 points on every cell, confirming that the comparison is not an implementation artifact. Note that the keep ratio counts exactly retained tokens; the archived segments add a bounded overhead of roughly 1/14 of the evicted volume (fp8 low-rank factors), so LARK's actual KV footprint is ≈16%/30%/54% of dense at the three settings—slightly above the nominal keep ratio, but with no information discarded.

D.4 Decode Efficiency and the fp8 Pool

Provenance of Table 2. The breakdown is one instrumented run at B=1024, batch 16, true 32k depth: 34,819 decode steps and $16\times(32768/512)$=1,024 compression events, so Total is cumulative and per-event costs are Total/1024. The side-stream chain is 13.3+7.8+6.5=27.6 ms per event (rSVD, package, tier write), quoted as ≈28 ms in §4.3 and Figure 5a; the stages that do sit on the critical path, namely scoring, selection, and slotmap commit, total 4.8 ms per event, i.e. 0.14 ms/step or 1% of the step. With 512-token segments and 16 concurrent requests an event arrives every 512/16=32 steps (≈420 ms), a 7% duty cycle for the side stream; at the max-batch H20 point of Table 1 (2554 tok/s at batch 548, 215 ms/step) the interval is 200 ms and the duty cycle 14%, and that throughput is measured end-to-end with archival enabled.

Against R-KV at a matched budget (Table 1) LARK measures +1.9%/−4.4% on H20 at B=1024/2048 and +4.7%/+17.7% on the RTX 5090; at each method's own memory limit, +2.2%/+17.5% and +9.9%/+5.8%. Seven of the eight cells favour LARK, including all four at each method's own memory limit; the single regression costs 4.4%, and it is on the far higher-bandwidth H20.

Because LARK's pool is fp8 where the baselines' is bf16 (Appendix C), those margins could be read as an artifact of moving half the bytes. Table 11 rules that out: a bf16 pool is 1.5% *faster*: fp8's return is memory (−20.5% peak). The ledger corroborates this: a rate tracking KV bytes would lead by 23–26% in every cell (Eq. 14 gives ratios of 0.81/0.80), whereas the measured margins above range from −4.4% to +17.7%. The ablation is H20-only; on the 5090, whose narrower bus leaves more room for a precision effect, part of the +17.7% may still come from pool width.

Table 11: Active-pool precision ablation on H20: LARK at B=1024, **batch** 64, true 32k depth, with the active pool in bf16 instead of the deployed fp8 (e4m3); the archived factors stay fp8 in both rows, so the rows differ only in pool width. Peak is peak device memory in GiB, as in Table 1. Note the batch: this is not that table's batch-16 operating point, so the rates are not comparable to it.

Active pool	Tok/s	ms/step	Peak (GiB)
bf16	**2086.7**	**30.67**	34.27
fp8 (e4m3, deployed)	2054.5	31.15	**27.24**
effect of fp8	−1.5%	+1.6%	−20.5%

This ablation is measured on H20 at one batch and one budget, which settles the attribution for the two H20 cells, including the −4.4% regression. We did not repeat it on the RTX 5090, whose narrower memory bus leaves more room for a precision effect; the +17.7% cell is therefore the one where part of the margin may still be attributable to pool width rather than to archival, and §4.3 does not claim otherwise.

E Qualitative Trace: Importance Tiering on a Real AIME Problem

Figure 6 excerpts two spans of the trace examined here; this appendix gives the full picture. We re-run a single problem: item 26 of our AIME 2024 set (equilateral hexagon; answer 80), on which LARK at B=512 is correct on 4/4 seeds while R-KV is correct on 0/4. The problem is run under

three configurations that share an identical sampling protocol (temperature 0.6, top-p 0.95, seed 1): dense full KV (Figure 12), R-KV eviction at B=512 (Figure 13), and LARK at B=512 (Figure 14). Each figure shows verbatim excerpts with elisions made explicit; tier labels are annotated for exposition, not read out of the compressor.

Dense — full KV cache

9,272 words · correct (answers 80)

Okay, so I have this problem about a convex equilateral hexagon. All sides are equal, and each pair of opposite sides is parallel. That already gives me some structure to think about. The hexagon is labeled ABCDEF, so the opposite sides would be AB & DE, BC & EF, and CD & FA. Since all sides are equal and opposite sides are parallel, it's like a regular hexagon but maybe stretched or something?

⋮ *6,346 words elided* ⋮

So, equation (3) becomes: -19s/240 = (181 s / 240) - (5 s² / (4 d)) - (5 s f / (4 d)) Multiply both sides by 240 to eliminate denominators: -19s = 181 s - 300*(5 s² / (4 d)) - 300*(5 s f / (4 d)) Simplify: -19s = 181s - (1500 s² / (4 d)) - (1500 s f / (4 d)) Convert 1500/4 to 375: = 181s - (375 s² / d) - (375 s f / d) Bring all terms to left: -19s - 181s + 375 s² / d + 375 s f / d = 0 Simplify: -200s + 375 s² / d + 375 s f / d = 0 Factor out s: s [-200 + 375 s / d + 375 f / d] = 0 Since s ≠ 0, -200 + 375 s / d + 375 f / d = 0

⋮ *2,537 words elided* ⋮

We use the formula for the side length s of such a hexagon inscribed in a triangle with sides a, b, and c: s = (a · b · c) / (a · b + b · c + c · a) Substituting the given side lengths of the triangle: a = 200, b = 240, c = 300 Calculate the numerator: a · b · c = 200 · 240 · 300 = 14,400,000 Calculate the denominator: a · b + b · c + c · a = 200 · 240 + 240 · 300 + 300 · 200 = 48,000 + 72,000 + 60,000 = 180,000 Thus, the side length s is: s = (14,400,000) / (180,000) = 80 Therefore, the side length of the hexagon is: boxed{80}

AIME 2024 item 26 (equilateral hexagon, answer 80) · AceReason-Nemotron-14B · temp 0.6, top-p 0.95, seed 1
Excerpts are verbatim from the saved traces (LaTeX delimiters stripped); elided portions marked ⋮ .

Figure 12: Dense reference trace (9,272 words, correct): opening, one excerpt of the running algebra, and the closing derivation.

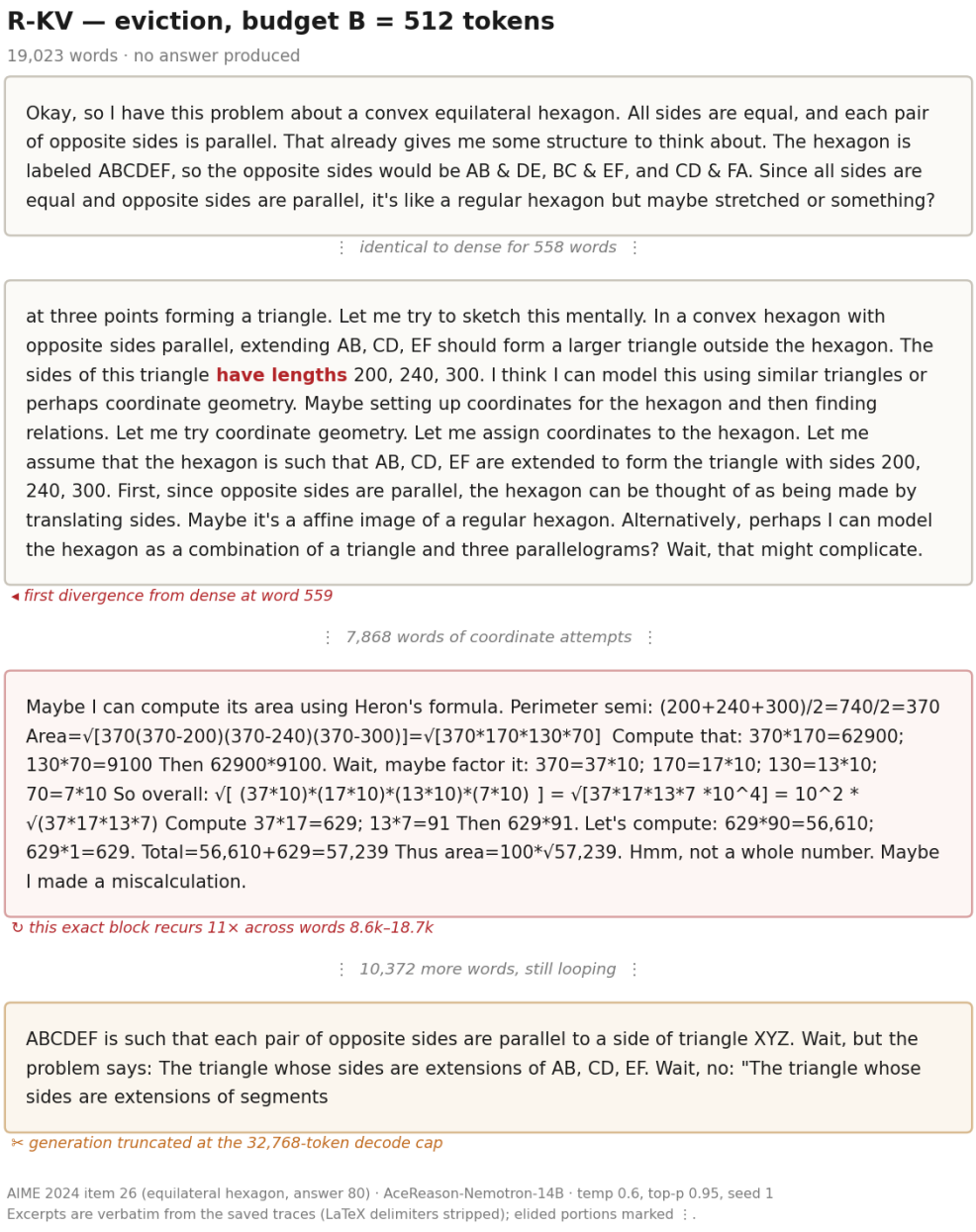


Figure 13: R-KV trace (19,023 words, incorrect): after a 558-word prefix identical to dense, the chain diverges and degenerates (the highlighted Heron block recurs verbatim $11\times$), and generation ends mid-sentence at the 32,768-token cap with no boxed answer.

The three runs share their first 558 words. Beyond that point, R-KV loses the constraint that selects the correct similar-triangle configuration and cycles through coordinate re-starts and repeated Heron evaluations until the decode cap, never producing an answer; LARK's archive instead keeps the evicted content reachable, and on this item it recovers the same answer as dense at one sixth of the nominal KV footprint. This is a single-item illustration of where content resides, not a claim of token-level equality; the aggregate behavior is the budget sweep of §4.2.



Figure 14: Tiering view of the same trace under LARK at B=512. The text is the reference excerpt of Figure 12; the tier annotation is illustrative—the givens and the closing derivation reside verbatim in the budget, the reduced constraint that the later algebra keeps citing stands in for the κ=8 segments holding GPU-resident factors, and the bulk of the intermediate algebra is archived with its factors on the host, re-scored at every event and recallable.

F Algorithmic Mechanism: Scoring Form and Segment-Level Tiering

F.1 Pseudocode of the LARK Pipeline

Algorithm 1 consolidates the scoring, archival, and tiering steps of §3 into a single executable summary; every quantity below is either an input or computed from preceding lines.

Algorithm 1 LARK: LEAP linear expected-attention scoring and LACE low-rank archival with centroid tiering.

Require: model with RoPE rotation $R(\cdot)$, head dimension d, KV heads h; cached post-RoPE keys/values $(\tilde{k}_p, \tilde{v}_p)$ and queries $\tilde{q}_t$; probe ring $\mathcal{P}$; offset grid $\{\tau_j\}_{j=1}^{J}$; budget B; segment length L; attention window W; ranks r_k, r_v; residency cap κ; fp8 quantizer $\mathcal{Q}$

Ensure: fp8 active pool $\mathcal{B}$ with $|\mathcal{B}| \le B$ and accumulating buffer $\mathcal{T}$ with $|\mathcal{T}| < L$; archive $\mathcal{A}$ of low-rank segments; resident set $\mathcal{R}$ with $|\mathcal{R}| \le \kappa$

$D \leftarrow \frac{1}{J}\sum_{j=1}^{J} R(\tau_j)$ {grid-folding constant; once per model}
$\mathcal{B} \leftarrow \mathcal{Q}$(first B tokens); $\mathcal{A} \leftarrow \emptyset$; $\mathcal{R} \leftarrow \emptyset$
for each compression event at time t_0 (every L generated tokens) **do**
  {Step 1: online expected query and folded scorer}
  for each head h **do**
    $\hat{\mu}_h \leftarrow \frac{1}{|\mathcal{P}|}\sum_{t\in\mathcal{P}} R(-t)\,\tilde{q}_{t,h}$ {Eq. 2, probes inverse-rotated to the base frame}
    $\bar{W}_h \leftarrow \frac{1}{\sqrt{d}}\,R(t_0)\,D\,\hat{\mu}_h$ {folded LEAP direction, Eq. 4}
  end for
  {Step 2: linear scoring and budget selection}
  for each candidate $p \in \mathcal{B} \cup \mathcal{T}$ **do**
    $S(p) \leftarrow \bar{W}^{\top}\tilde{k}_p$ {heads stacked; one linear projection per cached key, scored over the whole buffer including its unattended part}
  end for
  $\mathcal{B} \leftarrow \mathcal{Q}$(top-$B$ candidates by S); $\mathcal{E} \leftarrow$ the L lowest-score candidates; $\mathcal{T} \leftarrow \emptyset$
  {Step 3 (LACE): low-rank archival of the evicted buffer}
  for each layer **do**
    $K_s \leftarrow$ post-RoPE keys of $\mathcal{E}$ reshaped to $\mathbb{R}^{L\times F}$, V_s likewise $\{F = H_{\mathrm{kv}}d\}$
    $(A_k, B_k) \leftarrow \mathrm{rSVD}(K_s, r_k)$; $(A_v, B_v) \leftarrow \mathrm{rSVD}(V_s, r_v)$ {Eq. 5, batched CholQR on a side stream}
    $\bar{k}_s \leftarrow \mathrm{mean}(A_k)\,B_k$ {segment centroid}
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{(A_k, B_k, A_v, B_v, \bar{k}_s)\}$; free the pages of $\mathcal{E}$
  end for
  {Step 4 (LACE): centroid re-scoring and residency tiering}
  for each segment $s \in \mathcal{A}$ **do**
    $S^{\mathrm{seg}}(s) \leftarrow \bar{W}^{\top}\bar{k}_s$ {Eq. 6, exact mean of per-key scores}
  end for
  $\mathcal{R} \leftarrow$ top-κ segments by S^{seg}
end for
for each decode step with query q **do**
  append the step's key/value to $\mathcal{T}$ {resident and fp8-quantized; scored and archived regardless of W}
  $(\ell^{\mathrm{pool}}, o^{\mathrm{pool}}) \leftarrow \mathrm{FlashAttn}(q, \mathcal{B} \cup \mathrm{newest}_W(\mathcal{T}))$ {attention window: only the newest W of the buffer are attended}
  for each $s \in \mathcal{R}$ **do**
    $\ell_s \leftarrow (qB_{k,s}^{\top})A_{k,s}^{\top}$ {factorized logits, $AB^{\top}$ never materialized}
  end for
  $o \leftarrow \mathrm{OnlineMerge}\big(\ell^{\mathrm{pool}}, o^{\mathrm{pool}}, \{(\ell_s, A_{v,s}, B_{v,s})\}_{s\in\mathcal{R}}\big)$ {single exact softmax; values as $(w_s A_{v,s})B_{v,s}$}
end for
return $(\mathcal{B}, \mathcal{A}, \mathcal{R})$ and per-step outputs o

F.2 Why Linear Unnormalized Scoring Rather Than Softmax-Mass Scoring

§3.1 adopts a linear, unnormalized scoring form rather than the more intuitive alternative of using the softmax attention mass itself as the score. This subsection answers "why not softmax" with 384 real compression events: the softmax form exhibits three structural weaknesses at the segment level, namely mass concentration, cross-round incomparability, and ranking distortion in the decision band, each corresponding precisely to a design property of LEAP. (The numerical equivalence of the D-folding acceleration is derived in Appendix B.)

Protocol. During the MATH-100 run (first 100 problems of MATH-500 under the 32k protocol; §4.1), at each compression event we record two quantities: the true softmax attention mass each archived segment subsequently receives under the runtime probes (the observational ground truth), and the LEAP centroid score; the three panels of Figure 15 share this same 384-event sample.

(a) Mass concentration: resolution wasted on dominant candidates. The top-1 segment alone receives on average 22% of the mass and the top-3 take 49% (Gini 0.37); the two most recent segments take 34%—mass concentrates on a few sink/recency-dominated candidates. Consequently the few dominant candidates consume the dynamic range of the softmax score, and the ranking of the remaining majority of candidates is compressed into a thin tail in which they are hard to separate. The linear form normalizes nothing over the candidate set, so its scores are not squeezed by dominant candidates.

(b) Denominator contamination: incomparable across rounds. For the same segment, whose content has not changed, the median ratio of its softmax mass between consecutive compression rounds is only 0.92; 28% of segments drift by more than $\pm 20\%$, and the dilution correlates negatively with candidate-set growth (Spearman -0.26 over 5,965 same-segment pairs). The softmax score is therefore a "relative share" contaminated by the candidate-set size, unable to support the continuous residency decisions made across rounds. (Part of this drift is a task component from the evolution of the probe set, and the two cannot be fully separated; the crucial contrast is that LEAP outputs per-key absolute scores without a candidate-axis normalization denominator, so its cross-round variation arises solely from the evolution of the query statistics.)

(c) Ranking distortion in the low-score band: least reliable where decisions are made. The per-round Spearman correlation is 0.90 over the full candidate set but drops to 0.80 in the low-mass bottom half. The high overall correlation is therefore only an average, yet the residency/eviction boundary lies precisely in the low-score band—the interval where the softmax form's ranking signal is weakest. This is why the fidelity validation in Appendix F.4 focuses on hit rate in the decision band rather than reporting only the overall Spearman.

Summary. The three weaknesses attack the resolution, temporal comparability, and decision-band reliability of the softmax form, respectively; the linear unnormalized form avoids all three—the empirical rationale for the form choice in §3.1.

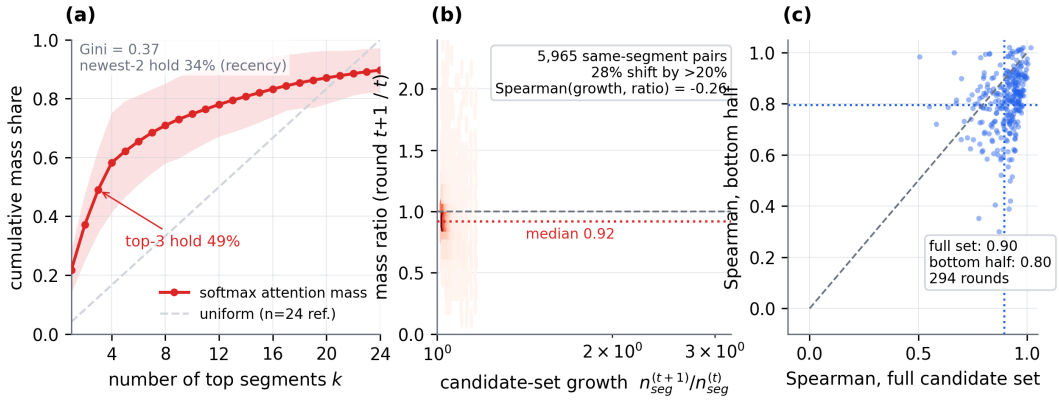


Figure 15: Three intrinsic weaknesses of softmax-form scoring over 384 real compression events (ground truth: future softmax attention mass per segment; numerics and interpretation in Appendix F.2). **(a)** Cumulative mass share of the top-κ segments (red: cross-event mean with IQR band; gray dashed: uniform reference): mass concentrates on a few dominant candidates, compressing resolution over the rest. **(b)** Same-segment mass ratio between consecutive rounds vs. candidate-set growth: systematic dilution below the no-change line (1.0) makes the scores incomparable across rounds. **(c)** Per-round Spearman, full set (x) vs. low-mass bottom half (y): the ranking signal degrades precisely in the decision band.

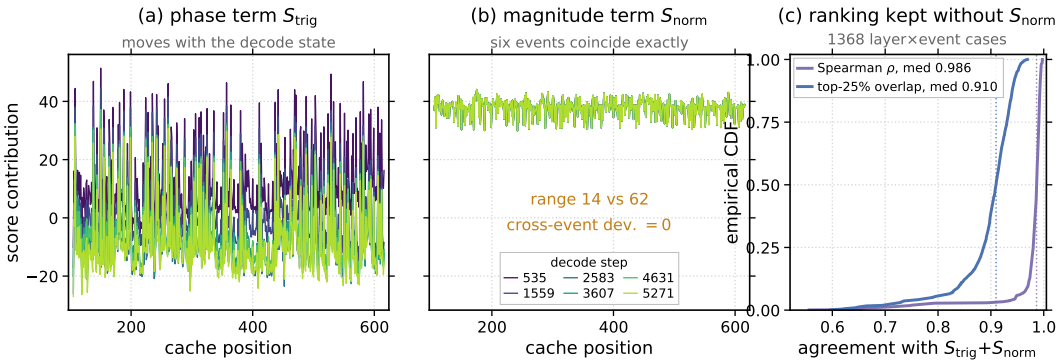


Figure 16: **Why the magnitude term is not carried into LEAP.** Score components of the trigonometric scorer on an AIME trace (Qwen3-8B, 36 layers × 38 compression events). **(a,b)** Layer 1, six events spanning decode steps 535–5271, on a shared y scale: the phase term S_{trig} moves with the decode state, whereas S_{norm} is a narrow band whose six curves coincide to the last bit—it carries no offset dependence, hence no evidence about which key a future query will attend to. Annotated ranges are for the first event shown. **(c)** What dropping it costs, over all 1368 (layer, event) cases: empirical CDFs of the Spearman correlation between the ranking of S_{trig} alone and that of the full score, and of the overlap of the resulting top-25% retained sets.

F.3 Relation to Trigonometric Pre-RoPE Scoring

The closest prior form is the trigonometric scorer of TriAttention (Mao et al., 2026), which we credit for showing that an *offline-calibrated, per-model* pre-RoPE prior can rank keys. LEAP is not that scorer made cheaper. Eq. 7 is an exact identity, but it relates two forms of *one* term, the expected-logit phase term of Eq. 3; the two deployed scorers agree on none of the three things that decide a retained set: which terms they carry, what evaluating the phase costs, and where the statistic comes from. The deltas below are not independent refinements of a shared scorer—Delta 1 is a precondition for the archival tier to exist at all, and the linearity it buys is what lets Delta 2's collapse apply. In complex channels f ($d/2$ per head), with $\bar{q}_f$ the calibrated mean query, $k_{p,f}$ the base-frame key, $\phi_{pf} = \arg\big(\bar{q}_f \overline{k_{p,f}}\big)$ and s_f a per-frequency weight, that scorer reads

$$S(p) \;=\; \underbrace{\sum_f s_f\,|\bar{q}_f|\,|k_{p,f}|\,\cos\big(\Delta_p\omega_f + \phi_{pf}\big)}_{S_{\text{trig}}(p)} \;+\; \underbrace{\sum_f s_f\big(\mathbb{E}|q_f| - |\bar{q}_f|\big)\,|k_{p,f}|}_{S_{\text{norm}}(p)}, \qquad \Delta_p := p - t_0 - \tau\,, \tag{16}$$

the first term being Eq. 3 in Cartesian form, the second weighting key magnitudes by a Jensen gap constant in head and frequency.

Delta 1: the magnitude term is dropped, and could not be kept. S_{norm} carries no Δ_p: it is a weighted ℓ_1 norm of the candidate's channel magnitudes, hence silent on the only question the scorer is asked, which future query will attend where. Figure 16(a,b) shows this directly—over 1332 event-to-event comparisons its largest deviation is exactly 0. Nor is dropping it optional: a magnitude is not linear in the key, and both mechanisms of Appendix B—the offset-grid collapse (Eq. 8) and the exact segment score (Eq. 6)—require that linearity. A scorer carrying S_{norm} stays $\Theta(mdJ)$ and must reconstruct $A_k B_k$ before it can rank a segment, at the 341× FLOP and 14× transient-memory cost of Appendix F.4. Keeping the term is therefore not a scorer substitution but the removal of the archival tier, hence the end-to-end comparison below.

Over the 1368 (layer, event) cases of Figure 16(c), S_{trig} alone reproduces the full score's ranking at Spearman median 0.986 (min 0.556) and its top-25% retained set at median overlap 0.910 (min 0.597), while the token-to-token spread of S_{norm} averages 20.4% of that of S_{trig}.

Delta 2: the phase term is folded into a constant. Where Eq. 16 evaluates a cosine per (candidate, offset, frequency) triple—the $\Theta(mdJ)$ path of Figure 4(b)—Eqs. 7–8 move all J rotations

onto the statistic, where they collapse into one model-level constant D, leaving a single $\Theta(md)$ projection on the post-RoPE key that scoring, archival and attention share, at a cost independent of J and of the probe count.

Delta 3: the statistic is a per-request state, not a per-model constant. The trigonometric $\bar{q}_f$ and $\mathbb{E}|q_f|$ come from an offline calibration pass and are held fixed across requests and events; LEAP forms $\hat{\mu}_h$ from a 32-slot probe ring on the trace being served (Eq. 2), refreshed at every event. Figure 16(a) argues for doing so: within one request the phase term drifts by roughly a full token-to-token spread between events 1 and 38 at the same cache positions, so the statistic is not a per-model constant at the scale eviction decisions are made on. An offline prior and a per-event estimate are therefore different signals, not two implementations of one.

End to end. On one model, dataset, budget and harness (Qwen3-8B, MATH-500, B=1024, batch 16), the observational softmax statistic reaches 74.0, the trigonometric method 79.3, and the LEAP row of Table 3 81.5 (archive switched off, hence scorer and fp8 pool alone); the accuracy above that comes from the archival tier the three deltas make possible. Since both rows run the same archive setting, the 79.3→81.5 gap isolates the scorer change itself: the deltas are worth 2.2 points before the tier they enable contributes anything.

F.4 Reusing the LEAP Score for Segment-Level Tiering

LARK's segment-level tiering (§3.2) does not introduce a second scoring machinery: it *reuses* the LEAP linear score already computed for key selection as the residency decision. At each compression event, segments are ranked by their centroid scores, the top κ reside on the GPU, and the remainder stay archived in low-rank form on the host; because the score is evaluated on the archived low-rank centroids themselves, the decision adds no attention computation.

This reuse is a feasibility condition. Linearity in the key makes the segment mean commute with the factorization, $\mathrm{mean}(A_sB_s)=\mathrm{mean}(A_s)B_s$, so one $r\times F$ product scores a segment from its stored code mean—$r(L+F)\approx 98$k FLOPs per segment and layer at the deployed L=512, F=1024, r_k=64. A scorer needing the individual archived keys, softmax-mass scoring included, must first materialize A_sB_s: $LrF\approx 34$M FLOPs (341×) plus a transient bf16 buffer of 2.0 MiB against the 0.14 MiB its fp8 factors occupy (14×), for every segment at every event. The pre-RoPE route adds instead an inverse rotation per candidate and a second, pre-RoPE copy of the cache (Appendix B). LEAP is therefore what makes an archive re-scorable at all, which is also why the ablation of §4.4 is cumulative in this order: an "archive + softmax scoring" row would be a different storage design, not a scoring-function swap.

Offline hit-rate validation. We instrument full traces on MATH-100 (§4.1; 384 compression events), recording at each event both the centroid linear score and the true future softmax attention mass each segment subsequently receives. Figure 17 reports both views, averaged over events with bootstrap 95% confidence bands. Selecting by score recovers 0.908 of the true top-16 against the recency baseline's 0.840 (top-12: 0.893 vs. 0.812; top-8: 0.854 vs. 0.819), a lead held at every κ below the oracle ceiling of 0.918, and the rank–rank scatter concentrates along the diagonal at Spearman $\rho = 0.891$, Pearson $r = 0.90$. The cheap linear score therefore tracks the softmax-mass ranking closely, which licenses its reuse as the residency decision. Age alone is a strong prior at this granularity (ρ=0.793 on the same events), which does not contradict the vanishing position-only correlation of Figure 2b: that panel ranks *individual* keys at one boundary by the head's own distance profile binned over $\Delta\leq 512$, and a 512-token block averages away exactly the within-block content spikes that distance cannot predict.

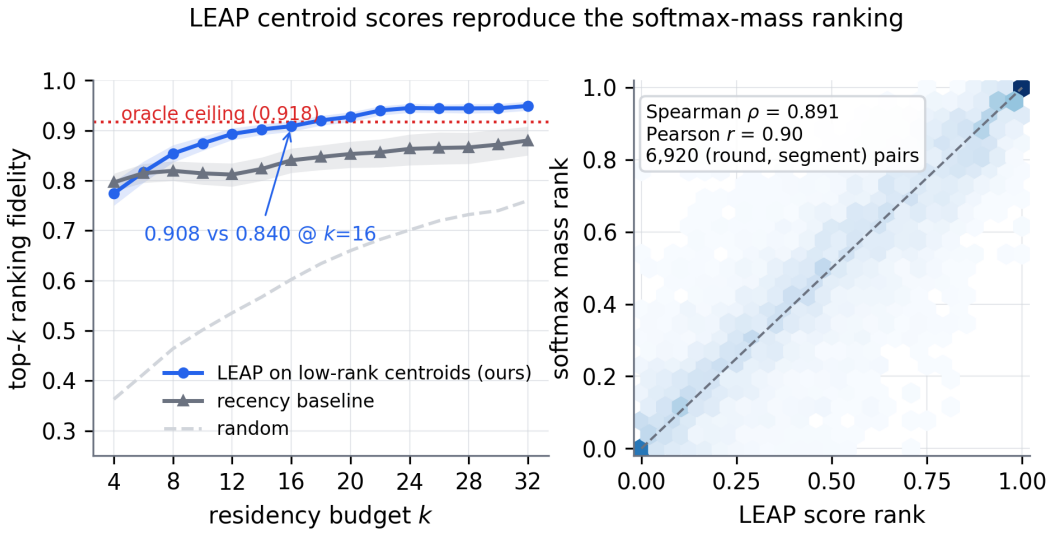


Figure 17: Segment-level ranking fidelity of the LEAP centroid score (protocol and confidence bands in Appendix F.4). **Left:** top-κ hit rate against the true future attention mass: LEAP (blue) vs. the recency (gray triangles) and random (light dashed) baselines; the red dotted line is the oracle ceiling (0.918). **Right:** rank–rank scatter of normalized score rank vs. truth rank over 6,920 (round, segment) pairs; darker color indicates higher density.

F.5 Mechanistic Reading of the Scoring Formula

The offset grid acts as a low-pass frequency mask. Appendix B shows $D = \frac{1}{J}\sum_j R(\tau_j)$ to be block-diagonal, applying per frequency block an amplitude c_f and a fixed phase ψ_f^0, independent of t_0. Under the deployed parameters (d=128, τ_j=2^{j-1}, J=17) the spectrum is a low-pass trend with a non-monotonic high-frequency comb (Figure 18). The retained blocks are not the position-discriminating ones: Table 12 concentrates the amplitude on blocks rotating by less than one radian over the 32k horizon, suppressing those that could resolve $|p-t_0|$. Consistently, the position autocorrelation $K(\Delta) = \sum_f c_f^2 \cos(\omega_f \Delta)/\sum_f c_f^2$ decays gently over the horizon (global minimum 0.403 at $\Delta\approx$27,177, endpoint $K(32768)$=0.508), versus 0.242 for J=1 (D=I). The deployed fold thus moves the operator toward content matching on the low-frequency subspace with gentle long-range decay.

Table 12: Spectral anatomy of the deployed D: the amplitude retained by the fold concentrates on position-invariant frequency blocks (all values recomputable from the grid definition).

Band	Blocks	$\sum_f c_f^2$ share	c_f range	max phase $\omega_f \cdot 32768$
$f \geq 49$	15	51.5%	0.922–1.000	<1 rad
$40 \leq f \leq 48$	9	20.4%	0.682–0.889	1–6 rad
$f \leq 39$	40	28.0%	0.057–0.816	>6 rad

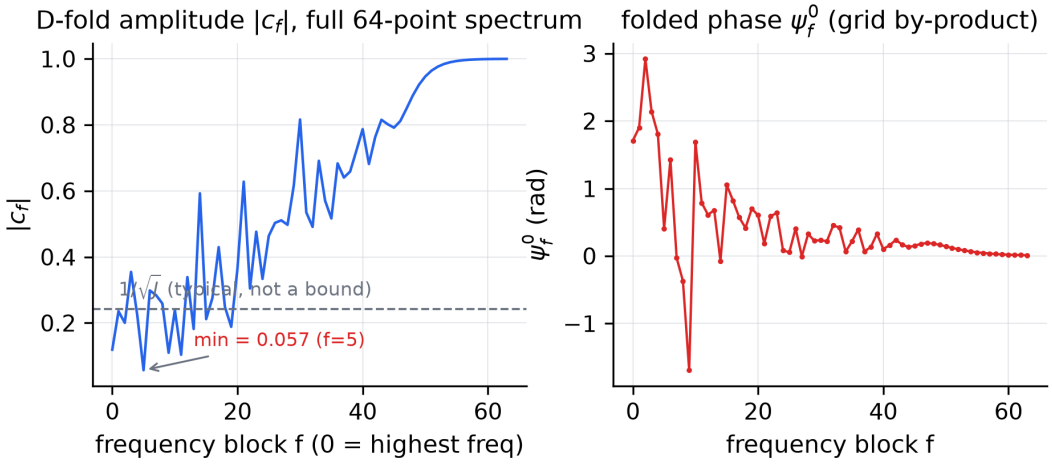


Figure 18: Per-frequency amplitude $|c_f|$ of the deployed D-folding (Appendix F.5): a low-pass trend with a non-monotonic high-frequency comb residual (minimum 0.057 at $f{=}5$; $1/\sqrt{J}$ is a typical amplitude, not a bound). **Right:** the folded phase ψ_f^0, a by-product of the grid.

The logit substitution: two approximation sources. Relative to the expected-mass ideal $\mathbb{E}[a_t(p)]$, the expected-logit scorer carries two gaps. (i) *Jensen gap*: the expectation enters before the convex map $\ell \mapsto e^{\ell}$, and $\mathbb{E}[a_t(p)] \approx e^{\mathbb{E}[\ell_t(p)]}(1 + \frac{1}{2}\mathrm{Var}[\ell_t(p)])/Z_t$, so $\hat{S}_h$ drops an $O(1)$ multiplicative variance bonus and under-ranks keys retrieved strongly only on some futures. For top-B selection this is tolerable: the exponential mean term spans orders of magnitude across candidates while the bonus does not; the persistently attended tail of §2.1 is the high-mean class the mean captures; and the disadvantaged sporadic class stays recallable from the archive, re-scored there against realized queries (§3.2). A correction would need per-head second moments $\mathbb{E}[\tilde{q}\tilde{q}^{\top}]$, a key-quadratic form that breaks the linear structure behind centroid reuse—the deliberate cost of the linear form. (ii) *Aggregation gap*: masses normalize per step, so π-averaged masses and π-averaged logits agree only up to the spread of Z_t across steps. The rank fidelity against realized mass reported below bounds both residuals.

The base-frame expected query: drift-freeness suffices. Folding the J rotations into one offline constant requires a base-frame statistic shared across offsets: a τ-dependent $\mu_{h,\tau}^{\mathrm{base}}$ would stop $\sum_j R(\tau_j)\mu_{h,\tau_j}^{\mathrm{base}}$ from factoring as $D\hat{\mu}_h$, and the cost would recover its dependence on J. The assumption needed is weaker than stationarity. Writing $\bar{\mu}_h = \sum_\tau \pi_h(\tau)\mu_{h,\tau}^{\mathrm{base}}$, the deployed $\hat{\mu}_h$ targets this π-weighted mean, and the residual $\sum_\tau \pi_h(\tau)(\mu_{h,\tau}^{\mathrm{base}} - \bar{\mu}_h)^{\top} R(t_0+\tau)\tilde{k}_p/\sqrt{d}$ vanishes to first order whenever the drift is uncorrelated with the grid phases, and is bounded by $\max_\tau \|\mu_{h,\tau}^{\mathrm{base}} - \bar{\mu}_h\|_2 \|\tilde{k}_p\|_2/\sqrt{d}$ in general. Two deployment choices keep it small: $\hat{\mu}_h$ is re-estimated at every compression event, so drift-freeness is required over one segment ($L{=}512$ tokens) rather than the full 32k horizon; and the probe ring mixes recent with distant offsets, estimating $\bar{\mu}_h$ rather than the instantaneous mean. The failure mode is a reasoning-stage switch inside the window, which moves $\bar{\mu}_h$ after the kept set is frozen—recoverable, again, from the archive. The uniform mean over the geometric grid is an implicit π_h uniform on a logarithmic axis; a weighted $D_\pi = \sum_j \pi_j R(\tau_j)$ folds identically, so the grid weighting is a design choice rather than a constraint.

Window-free fidelity, and the fidelity that decides eviction. Sweeping the observational softmax baseline over its window on the same trace and head cases gives Spearman $0.643/0.671/0.727/0.779/0.792/0.822$ at $W{=}16/32/64/128/256/512$: monotone in W, since a longer window scores more recently attended keys, and the longest (0.822) exceeds LEAP's window-free 0.733. But global correlation is the wrong target: under deployment the retained set is chosen among keys *older* than any window—even $W{=}512$ is a fraction of the 32k horizon—so their scores collapse and the rule degenerates towards "keep the most recent B". At segment granularity, measured directly, recency alone recovers $\rho{=}0.793$ against LEAP's 0.891 (Appendix F.4).

Table 3 isolates this at deployment scale: its *Softmax scoring* row is this statistic at $W{=}32$, matched to LEAP's 32-slot probe ring, and substituting LEAP is worth $+7.5$ at $B{=}1024$.

Cost is decoupled from the probe count. The softmax path costs $O(P{\times}m)$ per-sample scoring that resists batching; LEAP averages the P probes into $O(1)$ per-head statistics and one bmm independent of P (Appendix B). Growing P from 16 to 256 raises the softmax cost by $\approx$9.3$\times$ while LEAP stays flat, and on the multi-head sampling harness (1,152 sampled heads, $m{\approx}400$) the scaled implementation scores an event in $\approx$20 ms versus $\approx$332 ms; the deployed configuration of Table 2 scores one in 1.4 ms.

Boundary of the centroid reuse. The mean form underestimates segments whose quality is dominated by a few critical tokens (heavy tail); a max-type bound from per-rank min/max of the factors (Tang et al., 2024b) is a future control (mean and sum agree in ordering at equal segment lengths). Softmax scoring of centroids is possible, as page-level selection does (Tang et al., 2024b), at the cost of the linear form's single projection and its direct action on the factors.

G Deployment Configuration and Reproducibility

This appendix consolidates the deployed configuration referenced by §4.1 and the Reproducibility Statement.

- **Scorer:** LEAP with runtime query statistics—a 32-slot probe ring, the geometric offset grid $\tau_j = 2^{j-1}$ with $J{=}17$, and grid-mean folding (Appendix B).
- **Archival:** segment length $L{=}512$; ranks $r_k{=}64$ and $r_v{=}32$; fp8 (e4m3) storage for both the active pool and the archived factors; no residual correction; randomized SVD with CholQR power iteration under a fixed seed, making archival bit-reproducible.
- **Residency:** $\kappa{=}8$ GPU-resident segment slots for $B{\leq}1024$ and $\kappa{=}16$ for $B{>}1024$, re-scored at every compression event; active budget B swept over $\{256, 512, 1024, 2048\}$ on MATH-500 and $\{512, 1024, 2048, 4096\}$ on AIME24, AIME25, and GPQA.
- **Cadence and attention window:** compression fires on a fixed period matched to the segment length (one event every $L{=}512$ decode steps), so the keep-set re-scoring interval and the archival granularity coincide; within a period only the newest $W{=}128$ tokens of the accumulating buffer participate in attention, while the rest of the buffer stays resident in the fp8 pool, is charged in full, and enters the candidate set of the next event exactly like the attended part (Appendix C). Archival itself happens only at an event, which writes the evicted segment as low-rank factors and holds the non-resident ones in pinned host memory.
- **Serving engine:** We integrated our method into the *Mini-SGLang*[1], the SGLang developers' own "compact implementation of SGLang", MIT-licensed and hosted in the same `sgl-project` organisation as SGLang (Zheng et al., 2024); its radix prefix caching and CUDA-graph decoding are inherited from that lineage. We chose the compact tree purely for development agility; for the quantities we report the two engines are interchangeable. Our additions to that engine are the Triton kernels for the factorized-attention branch, the kv-head-packed batched kernel and fp8 dequantization, plus the background archival side stream (§3.3); they are applied identically for LARK and for the baselines reimplemented in the same engine (Appendix A).
- **Evaluation:** 8$\times$H20 GPUs for accuracy, with an RTX 5090 (31.4 GiB) added for the efficiency measurements of §4.3; temperature 0.6/top-p 0.95 sampling with Avg@4 accuracy for all four models; generation limit 32,768 tokens; identical budget protocol and grader for all methods (§4.1).
- **Baselines:** every baseline runs with the hyperparameters recommended by its public reference implementation, with no per-dataset or per-model tuning—across all reported runs the only quantity we vary is the budget B, to match the protocol above.
- **Motivating measurements (Figure 2a,b):** One full-KV Qwen3-8B trace (1,362 tokens, $t_0{=}953$). (a) 1,152 heads (36$\times$32), $\Delta{\leq}1400$, binned by the dyadic window (1–7/7–65/65+) of largest mean profile (816/67/269). (b) one Spearman ρ per head case,

[1] https://github.com/sgl-project/mini-sglang (accessed August 2026); Python package `minisgl` version 0.1.0, MIT licence, "Copyright (c) 2026 sgl-project".

160 cases (layers $\{4, 9, 17, 26, 35\}$, $\Delta \leq 512$); predictors: oracle q; random q; position only; $\hat{\mu}_h$ base-frame (key rotation uncancelled); $\hat{\mu}_h$ phase-aligned (Eq. 1). Case means $0.909/0.527/0.733$; paired wins $158/160{=}98.8\%$; mean difference $+0.21{=}0.207$.

H Limitations and Future Work

Beyond long-output reasoning: long-prefill scenarios. This work targets the KV pressure of long chain-of-thought decoding, but the same machinery is promising for long-prefill workloads: prompt-derived KV exhibits the same low-rank concentration (§2.2), and at prefill time the future queries are partially available, so the expected-query statistics are even better conditioned. Extending low-rank archival (and its interplay with prefill-side eviction) to bound both prefill memory and attention cost is left to future work.

Total-byte accounting and the host archive. Our footprint claim is device-side: at a matched budget LARK occupies no more HBM than an eviction baseline from $B{=}1024$ on, and brackets parity at the tightest budget ($1.13\times$ with every segment billed at the 32k cap, $0.92\times$ with the pool at its steady-state average; Appendix C.2), while retaining a recallable archive. If the pinned host copy is priced like HBM, the advantage at the tightest budget is parity or slightly negative (Table 8), because roughly two thirds of a segment's bytes sit in the shared bases B_k, B_v rather than in the per-token codes. Sharing bases across segments, assigning a lower rank to the basis side, or spilling cold segments to NVMe would attack that term directly. So would sub-byte quantization of our deliberately conservative fp8 (e4m3) factors (Liu et al., 2024c; Hooper et al., 2025; Zandieh et al., 2025): it narrows each retained coefficient while the projection removes redundancy across tokens, so the two compose rather than compete. We have not explored these here, and we do not claim total-byte superiority.

Disaggregated serving. The system is presented for a single node hosting both prefill and decode; prefill–decode disaggregation is not yet supported. Because the archive is authoritative on the host and compact, shipping factors and centroids, rather than the full KV, between disaggregated prefill and decode instances is an attractive direction, as is generalizing the slotmap indirection layer to other paging designs.