VESTIGEKV: QUERY-INDEPENDENT SPARSE ATTENTION WITH CERTIFIED RECALL FOR NOPE-MLA CACHES

Anonymous authors
Paper under double-blind review

ABSTRACT

A long-lived KV cache must be compressed *before* the queries that will read it exist. Methods ranking by observed attention are built for the opposite order and are not compared against here: with no query, the statistic they rank by is undefined rather than weak. VestigeKV decides before any query exists and makes the decision non-destructive. Reading 11% of each row, a statistic the cache already carries, it partitions the cache into an attended tier and a GPU-resident archive no row ever leaves, reachable each step by a certified, query-adaptive trigger. The certificate is what carries the result: swapping the first tier's statistic for a stride, a norm, a recency window or a random draw leaves every cell of the retrieval table unchanged, so the recovery is the archive's and not the selector's, and we report that rather than claim the selector. Nothing is trained and nothing is stored below 16 bits. On Kimi Linear 48B the two tiers recover every needle trial at $32\times$ and $128\times$ (1.00, 1.00) and 32/32 on the served path at 128k; delete tier 2 and the algorithm is incomplete. Both tiers stay on the GPU and the cache *grows* $1.14\times$, so the win is bandwidth, not memory: the per-step scan reads $\sim$26% of the bytes dense attention would, and on a two-GPU `sglang` deployment the crossover sits at $\sim$24k context, reaching $1.45\times$ at 512k. Two conditions bound the claim, and both are measured. It holds under NoPE, where the score is position-free so a ranking taken at prefill is permanent: with rotation the only variable it inflates the winner set a fixed cone can select by 1.1–2.1$\times$ (Prop. 1), and exact merging is provably impossible there. And it holds on *natural* text, where the selector's overlap with an oracle sits at $3.3\times$ the random floor against $1.6\times$ on a synthetic haystack; the overflow fallback owns that minority regime. All thresholds were frozen before their data; 20 archived verdicts and 8 closed routes accompany the paper.

1 INTRODUCTION: CAPABILITY, COST, AND REQUIRED CHANGES

NoPE-MLA is no longer one checkpoint's quirk: Kimi K3 and GLM-5.3-Flash both ship it, so what a position-free cache makes possible is a question about a growing class of production models rather than about a single release. This paper cashes one such dividend, on stock Kimi Linear weights: the released experimental model of that line, a hybrid in which only 7 of 27 layers are full-attention MLA. VestigeKV changes those and nothing else, so every end-to-end ratio below is diluted by the untouched majority of the step (Appendix H.3); for K3's Gated-MLA the machinery extends by proof (Section 3d), the signal content untested (Limitations).

Structurally, VestigeKV is a sparse attention; what distinguishes it is the *origin of the pattern*. Fixed geometry (Zaheer et al., 2020; Xiao et al., 2024b) ignores content; trained patterns (Yuan et al., 2025; Lu et al., 2025) cost pre-training; score-based selection (Zhang et al., 2023; Li et al., 2024; Tang et al., 2024) is training-free but query-dependent. The remaining quadrant (*training-free and query-independent*) is what compression-before-the-query requires (Section 2). KVzip (Kim et al., 2025) reaches it by estimating importance from context-reconstruction passes; VestigeKV arrives from the opposite direction, reading the pattern off a channel the pretrained NoPE cache already carries at zero forward-pass cost on natural text, a route that exists only under NoPE (Section 3a, Appendix B). The rest of the paper is the one implementation in three passes: Section 3 is the

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.

mathematics the guarantees rest on, Section 4 the policy and the per-step flow, and Section 5 the kernels and the serving path a reimplementation would have to reproduce.

2 THE PROBLEM: COMPRESSION BEFORE THE QUERIES EXIST

In a long-lived cache, the query that will need a token arrives after the compression decision. A statistic of *observed* attention is then not weak but undefined (nothing has been observed), so the obstacle is a property of the order, not a deficiency of any method built for the other one. We make no performance claim against those methods; the question this order raises is what signal is available *before* a query, and the answer is that a NoPE-MLA cache already carries one.

3 THEORY: THE MATHEMATICS OF A POSITION-FREE CACHE

The cache row and the two maps that read it:

$$\tilde{C}_u = [\hat{c}_u;\, r_u], \qquad s_h(t,u) = \lambda \big\langle Q_t^h, \tilde{C}_u \big\rangle, \qquad v_u^h = W_{UV}^h \hat{c}_u, \tag{1}$$

with $Q_t^h = [W_{UK}^{h\top} q_t^{h,c};\; q_t^{h,r}]$ the absorbed query and λ the attention scale. Everything follows from one structural fact: under NoPE, *both* maps depend on u only through the row itself.

Lemma 1 (Exchangeability)**.** *NoPE-MLA attention output is a function of the multiset $\{\{\tilde{C}_u\}\}$: $o_t^h = \sum_u e^{s_h(t,u)} v_u^h \big/ \sum_u e^{s_h(t,u)}$, and both sums are symmetric in u by Eq. 1.* □

Three consequences follow, one per pillar of the paper. **(a) Query-independent salience exists (the selector).** The score is one bilinear form over the *stored* rows, $s_h(t,u) = \lambda \langle Q_t^h, \tilde{C}_u \rangle$, with the absorbed query Q_t^h confined to a fixed subspace of rank $\le d_{\text{head}} + d_r$ ($W_{UK}^{h\top}$ spans at most d_{head}; the decoupled branch adds d_r): every read direction lies in one fixed low-rank subspace, so the tokens that can ever win are the extreme points of the cache-row cloud against one fixed cone. Under RoPE the form is instead the family $A_h(\Delta) = W_q^\top R_\Delta W_k$ and the cone sweeps with position. What that sweep costs a selector chosen *before* its queries is not a matter of degree; it has a closed form.

Proposition 1 (Rank stability)**.** *Fix a head and a query $q = (q^c, q^r)$, and score row u at decode step t by $s_t(u) = \langle q^c, \hat{c}_u \rangle + \langle q^r, R_{u-t} r_u \rangle$, where $R_m = \bigoplus_{j \le p} \mathrm{Rot}(\theta_j m)$ and $R \equiv I$ under NoPE. If the θ_j are rationally independent, then for every pair of rows u, v*

$$\sup_t \big[s_t(u) - s_t(v) \big] = \underbrace{\big\langle q^c, \hat{c}_u - \hat{c}_v \big\rangle}_{\textit{position-free gap}} + \beta_{uv}, \qquad \beta_{uv} = \sum_{j \le p} \big\| q^{(j)} \big\| \, \big\| \mathrm{Rot}(\theta_j u)\, r_u^{(j)} - \mathrm{Rot}(\theta_j v)\, r_v^{(j)} \big\|. \tag{2}$$

Under NoPE $\beta \equiv 0$: the ranking of any two rows is fixed once and for all, the top-k set is one set, and its union over any horizon is exactly k. Under RoPE the ranking of u above v survives every step iff *the position-free gap covers β_{uv}, and the union exceeds k as soon as a single pair fails to.*

Equation 2 is measurable, so we measure it as a controlled ablation rather than reading it off a cross-architecture comparison: one checkpoint, one prefill, the same rows and the same absorbed queries, scored twice with the decoupled branch as stored and then rotated synthetically, so R is the only variable. Over 32 steps the union of per-head winner sets inflates in every one of the 7 MLA layers (by 3.2–8.5× for the top row and 1.1–2.1× for the top 64, the wider set moving less exactly as Eq. 2 predicts), and the rank-64 residual a query-independent sketch must carry grows 1.2–1.4×. Two limits: these rows were never shaped by rotation-aware training, so this isolates the geometry of R, not a RoPE-trained cache; and Eq. 2 is a supremum over an equidistributed phase the lowest frequencies have not reached at 4096 tokens, so the torus limit reads 1.1–14.0% against the 1.0–11.6% measured. A cross-model comparison would answer this with architecture, width and training data confounded into R, so we report none.

(b) Merging is impossible; selection is forced (the method). A representative cache that reproduces the full cache's softmax output on an open set of realizable queries needs at least one row per equivalence class of rows agreeing on that subspace (Proposition 2, Appendix A). Under RoPE equal content at distinct positions is never equal as rows, $\|R_u k - R_v k\|^2 = \sum_j 4 \sin^2 \frac{\theta_j (u-v)}{2} \|k^{(j)}\|^2 > 0$, so $B = T$: *exact* compression is zero, and every practical method must select. Under NoPE+MLA

the merge class is wider (positive rescalings of c_u at equal r_u), but a pre-registered measurement finds it empty on real corpora, so selection is forced there too. The bound is on *exact* reproduction; lossy merging with output compensation is placed by deployment, not refuted (Section 7).

(c) Per-token certificates (the recall tier). For any surrogate row with $\|\tilde{C}_u - C_{g(u)}\| \le \varepsilon_u$ and any future query, $|\Delta s| \le \lambda \|Q\| \varepsilon_u$ — query-*uniform* only because the form is fixed, and the bound Section 4.2 builds the archive index from.

(d) Gate commutation (extension to Gated-MLA). Write a gated layer as $o_t = G(x_t, a_t(S))$, $a_t(S) = \sum_{u \in S} p_u(S) v_u$, with $p(S)$ the softmax of the same fixed NoPE form restricted to S.

Lemma 2 (Gate commutation)**.** *Let $G(x, \cdot)$ be L-Lipschitz for every x. Then (i) the selected set, Lemma 1, the certificate of (c) and ranking stationarity are independent of G; (ii) $\|o_t(S) - o_t(S')\| \le L \, \|a_t(S) - a_t(S')\|$; (iii) a query-side positive gate and a query-independent per-row gate both leave the selected set unchanged.*

Proof. (i) S, p and every quantity the guarantees range over are determined by $\{s_h(t,u)\}_u$, which is computed before G is applied, so G appears in none of their statements. (ii) is Lipschitz continuity of $G(x_t, \cdot)$ applied to the two aggregates. (iii) For a query-side gate the layer is $G = \gamma(x_t) \odot a_t$ with $\gamma > 0$ elementwise; the scores are unchanged, so the order of $\{s_h(t,u)\}_u$, hence the selected set, is unchanged. For a per-row gate $v_u \mapsto g_u v_u$ with g_u independent of t, the layer is the ungated one on the values $g_u v_u$, and the scores again do not move. □ □

For an elementwise sigmoid gate $G(x, a) = \sigma(Wx) \odot a$ we have $\|G(x,a) - G(x,a')\| \le \|\sigma(Wx)\|_\infty \|a - a'\| \le \|a - a'\|$, so $L \le 1$ and (ii) *tightens* every perturbation bound rather than loosening it. The one form that does not transfer is a query-row-*coupled* score gate $s_h(t,u) \mapsto g(q_t, k_u)\, s_h(t,u)$: it makes the order of the scores query-dependent in a way the stored rows do not determine, which is the same obstruction as rotation and the boundary of the theory. Kimi K3's reported NoPE Gated-MLA is of the output-gated form: the mainstream instance of this architecture, which is why the extension is stated here rather than deferred, so the *machinery* extends by this argument; whether K3's training loads the branch with salience as Kimi Linear's does (Section 3.1) is an empirical property this study does not measure.

3.1 What tier 1 reads

The branch r_u exists because rotary position cannot pass MLA's low-rank bottleneck (R_{t-u} does not commute with the up-projection), so DeepSeek-V2 routes position around it in a per-token, MQA-shared channel. Under NoPE the released code caches the branch but never rotates it (`skip_rope=True` in `sglang`), so it is 64 dimensions of per-row state with no positional job left, readable at no forward-pass cost. That is what tier 1 scores; Section 4 reports that what the score has to do is spread the kept set, which this channel is not uniquely able to do.

The read budget. Decode attention is bandwidth-bound, so the KV-path cost is bytes per step. The three-term budget (attended rows, the index scan, the admitted set) lands at $\sim$294 against 1152 B/token, $\sim$26%, with the admitted fraction under 1% on document decode; Appendix H derives it (Eq. 7) and prices a full-precision index at $2.1\times$.

4 The algorithm: two tiers and the decode flow

The method is one object with two tiers; neither half is the method alone. Section 4.1 partitions every cache row into an attended tier and a GPU-resident archive; Section 4.2 is the certified, query-adaptive trigger that reads the archive back. **Delete tier 2 and what is left is not the algorithm**: the partition alone is lossy eviction (Table 4). Delete tier 1 instead and the trigger has nothing to read (the delta from classic MLA, drawn per layer: Figure 4, Appendix H).

That pair of ablations does not establish what tier 1 has to *be*, and the answer is less than we assumed. Tier 2 scans every archived row whatever tier 1 kept, so the selector moves the byte budget and not the reachable set. Replacing σ with a uniform stride, the branch norm, a recency window or a random draw, both tiers on, same 24 trials and same protocol as Table 4: 0 of the 8 cells move, and

all 5 selectors read 1.00 at $32\times$ and 1.00 at $128\times$. At these ratios the recovery is the archive's and the certificate's, not the selector's, and a claim that the branch is what makes the method work is not supported by this table.

What the selector does change is the cost of reaching that recovery, and the telemetry separates it where the needle cannot. Under the sound bound, *every* archived row beats the kept tier's maximum for a recency window (1.000), the branch norm (1.000) and a random draw (1.000): the admission test excludes nothing, and the calibration lands at 0.988 recall where it was asked for 0.90, which is over-fetching. Under σ the same rate is 0.996 and the calibration lands at 0.929. A uniform stride reaches the same two figures (0.997, 0.927), and the branch norm does not, so what tier 1 has to do is spread the kept set along the sequence rather than carry salience in it.

4.1 Tier 1: the eviction policy

For a closed prefix block of branch vectors $r_{0:T}$, $\mathcal{F}$ the sequence-axis rFFT, bandwidth κ:

$$\sigma_u = \left\| r_u - \left(\mathcal{F}^{-1} \mathbb{1}_{[0,\kappa)} \mathcal{F} r\right)_u \right\| \quad \text{per fixed 4096-token block;} \tag{3}$$

A row whose branch departs from its block's low-pass trend is locally distinctive along the sequence axis. We adopt the high-pass residual as a heuristic *motivated by* Section 3(a)'s extreme-point view, not as an equivalence: extremity in the row cloud is permutation-invariant while σ is order-dependent and reads the branch alone, so what licenses the metric is the measurement record, not the geometry. The transform runs over fixed-length blocks, never the whole prefix: κ counts frequency *bins*, so a whole-prefix transform would let the cutoff period drift with context ($\geq$256 tokens at T=4096, $\geq$32k at 512k), while fixed windows keep σ scale-invariant and incremental — each block transformed once at close, prefill and decode sharing one implementation. We keep the σ-top-m rows, $m = \rho T$, plus 4 sinks and the unclosed tail (the up-to-one-block remainder where decoded tokens live); every other row moves bit-exactly to the archive of Section 4.2. *No row is ever deleted in deployment*: the invariant is a partition, a cache-manager policy and not a model change, with nothing stored below 16 bits and so no quantization error budget, and the selector reading one contiguous 64-dim slice. It stays live during decode — every 4096 generated tokens the newest block closes, its σ joins the record, and one global top-m over all closed rows rebalances the tier through cached projections.

4.2 Tier 2: the recall tier

The archive exists to be read every step, not to be a graveyard (contrast evict-and-summarize (Chen et al., 2024)). The trigger is *query-adaptive, not quota*: each step's fired set is the archived rows that, after certified inflation, could still beat the kept tier's best for *this* query. It is empty when the query is well served and the minimal sufficient set for the recall target τ when its target was evicted, where a fixed budget both under- and over-fetches by construction. Archived row u keeps an index entry $(r_u,\ V_r^\top \hat{c}_u,\ \eta_u)$: the *exact* branch summand of every future logit (all 64 dims of it, a width the architecture fixes), then a sketch of the content summand at the index's own rank r, and the sketch residual norm $\eta_u = \|(I - V_r V_r^\top)\hat{c}_u\|$. V_r is fitted on the request's own *decode* queries (prefill queries reach the serving backend un-absorbed and are unusable), so the index builds in two stages (Figure 1): a provisional index serving from the first decode step, then a calibrated build on a side stream once 8–64 decode queries are collected (Section 5). That stage over-fetches and falls back to dense, so the warm-up spends throughput and never recall. Per decode step, over archived rows:

$$\text{score}_h(t,u) = \underbrace{\lambda\, q_t^{h,r\top} r_u}_{\text{exact summand}} + \underbrace{\lambda\, (V_r^\top q_{t,h}^{c\prime})^\top (V_r^\top \hat{c}_u)}_{\text{sketch}} + z\ \underbrace{\lambda\, \|(I - V_r V_r^\top) q_{t,h}^{c\prime}\|\, \eta_u / \sqrt{d_c - r}}_{\text{certificate scale}}, \tag{4}$$

with $q_{t,h}^{c\prime} = W_{UK}^{h\top} q_t^{h,c}$ head h's absorbed content query, d_c=512 the content-latent width, and the division by $\sqrt{d_c - r}$ putting the residual on a per-dimension scale before calibration. V_r is one PCA basis per (layer, request), *shared across heads*: per-head bases would multiply the scan traffic by H, and Section 6.2 notes this sharing as a plausible structural contributor to the sketch's resolution limit; fetch every row whose certified score beats its own head's kept maximum, and the fetched set is the union over heads. No quota ranks the fired set: the fetch is bounded by a designed 4096-row buffer (static, as CUDA graphs require), and an overflow attends densely.

The deployed backend evaluates Eq. 4 one step late. A decode step records its own query inside the CUDA graph, and the scan that reads it runs at the next step, so step t admits rows against q_{t-1} while Eq. 4 and the calibration below are written for q_t. The guarantee therefore holds for the query the scan saw, and transfers to the current one only insofar as consecutive decode queries admit the same rows. We do not model that gap; what we can say is that every number in this paper was measured with the lag in place, so it is priced into them rather than absent from them, and that removing it costs a second graph launch per step, which is the trade the implementation took. z has a closed form: over the calibration pool (every decode (query, head) pair whose best archived row exists, pooled across the layer's heads), the required inflation is $z_t = (s_t - \mathrm{idx}_t)/\mathrm{cert}_t$ (s_t the target's *exact* score, idx_t and cert_t its indexed and certificate summands). Of the two statistics available here this is the more conservative: it bounds the index's own error, while calibrating against the head's kept maximum is tight and fetches less. z is then the conformal quantile $z_{(\lceil (n+1)\tau_s \rceil)}$ of these (Shafer & Vovk, 2008), which by exchangeability alone guarantees $P(\text{target recovered}) \geq \tau_s$ on a fresh query, with no distributional assumption and no external calibration corpus (no row is ever deleted, so labels stay computable whenever queries exist). Head pooling is what meets the minimum sample size derived below within an 8–64-query window; until the pool reaches it, the tier serves the provisional clamp $z{=}Z_{\max}$ (8.0), which over-fetches and never under-recalls. The scan level $\tau_s = \tau/(1-\beta)$ and the gate's false-close budget $\beta = (1-\tau)/2$ both derive from the single recall target τ by an even split of the miss budget, and the minimum calibration sample size $\lceil \tau_s/(1-\tau_s) \rceil$ is the smallest n for which that order statistic exists: the tier has exactly one quality knob. The guarantee rests on exchangeability between the calibration window and future hard queries, and the answer step is not a fresh draw from that window (it matters precisely because it needs an archived row), so coverage is marginal over queries and weakest on the ones that decide the answer. The design does not assume otherwise: the un-calibrated default over-fetches, so a violated premise costs throughput, not recall, and the adaptive rebuild and self-disabling entropy gate (Section 6.3) are the runtime mitigations.

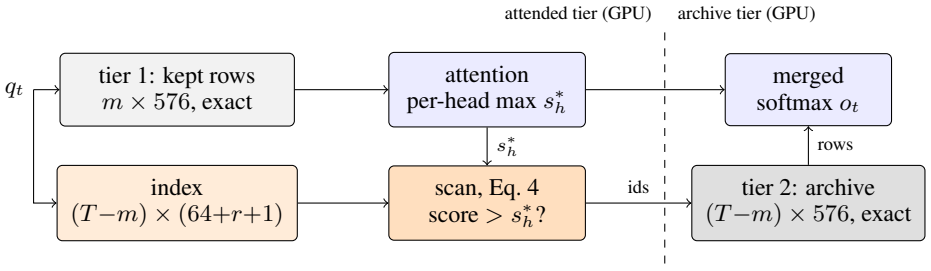


Figure 1: Per decode step: tier-1 attention yields the per-head maxima s_h^*; the index scan admits the archived rows that could compete on any head; fetched rows join that query's softmax as exact copies (Lemma 1). Both tiers are GPU-resident: the archive is a cheaper *read discipline* over the same memory, not a demotion to slower storage.

Two lemmas make the construction precise.

Lemma 3 (Index certificate)**.** *For every future query and archived row,* $s(t,u) = \lambda q_t^{r\top} r_u + \lambda (V_r^\top q_t^{c\prime})^\top (V_r^\top \hat{c}_u) + \delta_{t,u}$ *with* $|\delta_{t,u}| \leq \lambda \|(I - V_r V_r^\top) q_t^{c\prime}\| \, \eta_u$. *The first two terms are computable from the index alone; the bound is Cauchy–Schwarz on the sketch residual and holds for* all *queries because the form is fixed: under RoPE it would have to hold over the rotation orbit.* □

Lemma 4 (Bounded leakage)**.** *Let* F *be the rows admitted to a step's softmax (kept* $\cup$ *fetched) and* $s^* = \max_{u \in F} s(t,u)$. *If every excluded row satisfies* $s(t,u) \leq s^* - \gamma$, *the total excluded softmax weight is at most* $(T - |F|)\, e^{-\gamma}$ *relative to the maximal included weight, and the output error is bounded by that mass times* $2 \max_u \|v_u\|$ *(Appendix A).*

Lemma 3 is what the scan computes; Lemma 4 is why fetching only score-competitive rows suffices: the trigger exists to make the premise of Lemma 4 hold with high measured probability (0.929 in distribution). At the deployed $\delta{=}0$ the margin is zero and the bound vacuous: the lemma is the design target the trigger enforces, not the source of the guarantee, and a positive margin is what the δ sweep of Appendix G priced. A sound test is available and useless — the Cauchy–Schwarz bound fires on 99.6% of steps, which is no compression at all — so the adopted z is calibrated rather than sound: the distribution-free conformal quantile of the empirically required inflations.

5 KERNELS AND THE SERVING IMPLEMENTATION

The mathematics of Section 3 buys a specific engineering property, and this section makes it concrete (context lengths are written with $1\text{k} = 1024$ tokens): we port VestigeKV into standard `sglang` as an attention backend (`-attention-backend vestigekv_mla`) and serve Kimi Linear 48B on two GPUs, with the eviction running under CUDA graphs at zero per-step host cost. One attention kernel is written and it is a *fork*: decode stage 1 is upstream's grouped MLA kernel with three lines changed: the two that read a lane's segment bounds and the one that reads a row id. The tiling, the head-block grouping, the split schedule, the logit cap, the page-size arithmetic and the LSE output are upstream's, and stage 2 runs unmodified, so the arithmetic is not ours; only where a row comes from is. The rows are the two arrays the backend already maintains (the kept table and the fetch buffer, or the page table when a lane is fenced), so nothing copies them anywhere first. The kernels below are the selection pipeline around that read. The backend enforces its own precondition rather than trusting the operator: selection is refused unless the model is a NoPE-MLA cache (`mla_use_nope`), because a positional encoding on the decoupled branch destroys the eviction signal (Section 3a, Appendix B); it also refuses `page_size`$\neq 1$ and speculative decoding, which the kept-index tables do not yet address. A config that fails any check raises rather than degrades silently.

Decode step: six kernels inside the model graph. Per layer the attended set is kept $\cup$ tail $\cup$ fired(q_t), both tiers always on. The whole recall step runs as *six fused Triton kernels baked into the decode CUDA graph* (Table 1), with the forked stage 1 launched once per MLA layer: one `cudaGraphLaunch` per step, no second graph, no VestigeKV-triggered recapture (worst-case-capacity buffers refresh in place). Stage 1 attends only tail $\cup$ fired; the kept partition's partial output and normalizer come from the Triton prologue, so no row is scored twice. Partitions merge exactly through their log-sum-exps: $\ell = \text{logaddexp}(\ell_A, \ell_B)$ and $o = \varsigma(\ell_A - \ell_B)o_A + \varsigma(\ell_B - \ell_A)o_B$ with ς logistic, associative, so kept, tail and fired combine in any order and an empty fetched set is the identity. Calibration fits on the request's first decode queries with a provisional index serving from step one; the $O(S)$ calibrated build runs on a side stream. The recall tier has no production off-switch, and the ablation that deletes tier 2 is why (Table 4): recall is a component of the method, not a tunable on top of it.

Table 1: The six kernels of the decode graph, per layer. P pairs (lane, layer), H=32, D=64 branch dims, R=64 sketch rank, arena = the layer's archive rows. Shapes are what the kernel indexes, not what it copies.

kernel	reads	writes	grid
prologue scores	q $[L, R_1, H, 576]$; kept rows $[P, N_k, 576]$ bf16, or pool ids $[P, N_k]$ int32 + base $[P]$ int64	partial (o, ℓ)	$(P, \lceil N_k/B \rceil)$
prologue merge	the partials; N_k, arena lengths $[P]$ int64	(o, ℓ), s_h^*, $q^r/q^{sk}/\|q^\perp\|$	(P)
archive scan	$q^r[P, D, H]$, $q^{sk}[P, R, H]$, $\|q^\perp\|[P, H]$ fp32; sidecars [arena, D] bf16, sketch [arena, R] fp16	fire mask [arena] int8, per-bucket counts	$(\lceil \text{arena}/64 \rceil, P)$
compaction scan	per-bucket counts	exclusive offsets, total	(1)
compaction write	fire mask, offsets, row ids [arena] int32	fired ids, length, overflow	$(\lceil \text{arena}/64 \rceil, P)$
CSR pack	kept $[L, R_1, \text{cap}]$, kept/fetch lengths, slots [bs] int64	stage-1 `indptr`, `indices`	(bs)

Speedup accounting: the gains and their regimes. Decode step time is $t(S) = C_0 + T_{\text{net}} + A(S)$, and attention $A(S)$ is the only term any selection scheme can shrink. On this deployment (one node, two RTX PRO 6000 96 GB GPUs, TP=2, SHM collectives) the measured fixed part is $C_0 + T_{\text{net}} \approx 3.9$ ms (MoE with 3B active parameters, 20 linear-attention layers, sampling, and the per-step TP collectives; the two are not separately profiled on this testbed), and $A_{\text{dense}} \approx 0.41$ ms at S=64k by measured-slope calibration: attention is 10% of the step there. VestigeKV replaces $A_{\text{dense}} = kS$

with $k\rho S$ (kept attention) $+ k_s(1-\rho)S$ (the archive scan at 258 B/row, ≈260 with alignment, on the 16-bit index) $+ R$, where the fused pipeline's fixed adder measures R = 68 µs, batch-independent (the 4k-context arms measure even). The S-slope ratio is $\rho + k_s/k \approx 0.26$: the scan must touch every archived row every step (what the recall guarantee costs), so the attention slope drops to a quarter, not to zero — and the end-to-end curve of Figure 2 understates even that, because this testbed's fixed non-attention part (3.9 ms of a 7.1 ms step at 512k) suppresses the ratio.

Regimes without gains. The calibration window (8–64 decode steps) fits the certificate on live queries, a per-request cost bounded by the window and independent of S; it is under 0.3% of a 4k-token decode and is *not* why bs=1 sits near dense at 64k+4k. That near-parity is the steady-state attention share (10% of the step at 64k), where compression's saving is roughly offset by the recall scan, so the advantage appears where attention's share grows, with batch and with context. The gain is speed: both tiers are GPU-resident and nothing is deleted. Unlike classic sparse attention, the attention volume is not a fixed budget: the tier-1 set is *proportional* (ρT plus the tail) and the fired set is query-dependent, zero to a measured worst case of 3,983 rows. The 4096-row fetch buffer of Section 4.2 sits just above that, and an overflowed lane is a runtime scalar that switches stage 1's row source to the request's own page table (`req_to_token[slot,:seq]`), so one recorded kernel serves both cases: no recapture, no second kernel, no copy made in advance. Capping the fetch is ArkVale's design; placing the cap *behind* the admission rule rather than inside it is what lets an overflow be exact, so an overflow is a detection and not a miss — the certificate has asked for more rows than the graph's width, and the step is served densely. Like the recall tier, it has no production off-switch. The slope ratio sits *below* the $3.9\times$ byte account of Section 3.1, and has to: the account prices the KV-path reads alone, while the measured slope also carries the scan's arithmetic and the kernel's other traffic. The account is a ceiling on what the read term can buy, not a prediction the ratio reproduces.

NoPE dividend: the kept set is frozen before any query exists, so the sparse pattern is a compile-time constant of the decode graph: selection costs zero on the token path. Under RoPE: a RoPE selector must consult the live query, forcing an out-of-graph rebuild per step: the pattern cannot be captured.

6 Experiments and recommended configuration

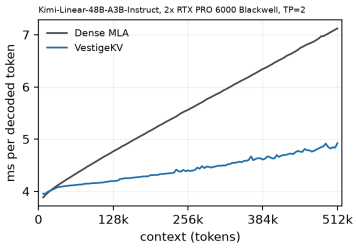
Protocol. Every serving number follows the *continuous-generation* protocol, because that is the deployment form: bs=1 latency is one request prefilled at 4k and decoded continuously to 512k[1] (decode-time closes and the live archive run throughout); throughput is 64k-prefill/4k-decode swept over batch, loaded by `sglang`'s standard `bench_serving` with the server-side decode log as per-token authority. Both arms share one node under an identical deployment configuration, against the wrapped base itself as baseline, and disable the radix cache, so no answer can come from a reused prefix rather than from attention (Appendix F). Quality tables report the deployed form, always both tiers; ablations that delete tier 2 are Appendix D.

6.1 Cost accounting: the cache memory–time product

Weights are a resident constant identical across arms, so the decision-relevant account is the variable part: the cache itself. Both coefficients below are measured on the deployment (Appendix H).

Memory: $1+\alpha$, invariant in length and batch. The stock pool row costs c_{kv}=1152 B per token per MLA layer. VestigeKV adds a 4 B index entry per kept row and a 16-bit rank-r sketch ($2r$+6 B) per archived row, plus fixed per-slot staging: measured α_{steady}=0.14, α_{peak}=0.17 (peak decides whether a maximal request fits; steady decides capacity). The scan's ≈260 B/row is read *traffic*, not memory — its sidecar is the row's own branch, read in place. Archive tables are sized by the KV pool itself, so α is flat in context length and batch (Appendix H). In capacity-bound serving the $+14\%$ footprint lowers peak concurrency $\sim 12\%$ until the ≈128k crossover pays it back. Below the activation threshold nothing is archived and $\alpha \approx 0.004$.

[1] Below the crossover the method pays a fixed ≈37 µs/step graph-binding cost (≈1%); production keeps an activation threshold that is byte-identical dense below it, disabled throughout this paper (Appendix H).



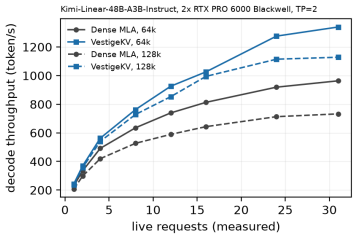


Figure 2: **Left:** bs=1 continuous decode, one script and one sitting, radix cache off. A point at N is the mean over the 4096 tokens ending at N, from the server's own log, so a block close and an index rebuild sit inside the number rather than being summarized out of it. Crossover ∼24k; VestigeKV's slope is nearly flat, 1.9 ns per step per cached token against dense 6.3; 1.26× at 256k, 1.45× at 512k. Their ratio 3.3× is the speedup on attention alone; end-to-end is lower because 20 of 27 layers are linear attention and untouched (Appendix H.3). **Right:** decode throughput against the concurrency the server actually ran, 64k and 128k prefill, radix cache off. Each point is the tokens a window of steady decode generated over its wall clock, so prefill is outside it. The advantage grows with batch, because attention does not amortize while shared weights do (1.25 at twelve live requests, 1.39 at twenty-four), and again with context (1.45 at twelve on 128k prompts); both series reach 31. The 128k series is measured at 0.87 on both arms, because at the default memory fraction the VestigeKV arm exhausts this GPU above sixteen 128k requests and the dense arm does not, and a comparison needs the same knob on each (Appendix F).

A request holds its cache for its whole lifetime, so the memory–time product $M_{kv} \cdot T$ is the account that closes: the arms' ratio crosses 1 at A=1.14 and reaches 0.79 at 512k (Appendix H).

The needle test (primary metric). A magic digit string no language prior can guess is spliced at a random depth into filler, the document is compressed whole at prefill *before any query exists*, and a trial is intact when the answer-token NLL stays within 1 nat of dense (Appendix F). **Delete tier 2** and needles are lost at every ratio past 8× where the complete algorithm recovers every trial (Table 4; 24/24 [0.86,1.00] at L=8192), and what is left of the selector does not even degrade monotonically with length (Appendix D): the archive is what is worth keeping, not the selector worth tuning.

6.2 Corroboration on the deployment path

Table 2 gates retrieval (the 128k serving needle, RULER), reasoning (GSM8K) and open-ended generation (MAUVE) with recall live every step. VestigeKV passes every frozen bar. RULER is not one of them, and there it trails by -0.022 (50 per cell). The arms answer the same items, so the reading is paired and the gap is resolved rather than absorbed: sign-flip $p = <10^{-5}$ over 3250 items. That is a real loss and we do not report it as noise. It is carried by the two tasks whose *haystack is itself made of key–value pairs* (-0.056, -0.128 with UUID keys); multiplicity alone costs nothing, +0.000 on the three that plant or ask four. What fails is resolution: a rank-r sketch fitted to the archive's own content, and shared across a layer's heads, cannot separate rows drawn from the bulk it was fitted to. A second pass at 128k and 256k finds the same direction, -0.046 over 26 cells, paired $p = 0.008$; per-cell readings, the band, and the copy-fidelity split are Appendix G.

RULER is synthetic by construction, which is the regime this method does not claim: its haystack is assembled from key–value pairs rather than written, so the structure tier 1 selects on is not there to be selected. A high fallback rate there is the design boundary working, not a surprise, and an answer 14 tokens long leaves a per-request adaptation nothing to amortize over besides, ours included, so the same prefill decoded for 4096 steps moves the rate from 0.337 to 0.012. What the fallback bounds is the *throughput* cost of such input: the certificate reports the step it cannot serve sparsely at the recall target, and that step is served exactly. It does not bound the loss above (there the certificate is satisfied and the sketch still cannot resolve the row, so nothing fires), which is why both are reported. Real-document evidence sits inside RULER too, `qa_squad` and `qa_hotpot` at -0.056 and -0.040, with one cell escaping (`qa_hotpot` at 64k, 0.62→0.54), which we read rather than the mean as the honest risk summary. Continuing published novels for 4096 tokens gives decode something to amortize over: 1.072/1.148/1.270× dense at 64k/128k/256k of real prose, server-side over the decode.

Table 2: Serving-path quality, dense vs. VestigeKV, both checkpoints. Bars frozen before the data (GSM8K within 2σ of dense, MAUVE $\geq$ dense -0.10, needle $\geq$ dense); every row passes. GSM8K is questions answered correctly out of 1209; the arms differ by 10 against a two-sigma band of 19.3. Protocols: Tables 5, 6.

capability	benchmark	Base		Instruct	
		dense	vestigekv	dense	vestigekv
retrieval	serving needle 128k, En / Zh	32/32 / 8/8 recovered, every arm and checkpoint			
	RULER mean, 65 cells	0.922	0.860	0.942	0.920
reasoning	GSM8K-Platinum 64-shot	1027	1031	1107	1097
generation	MAUVE, 4k prefill	0.9997	0.9960	0.9345	0.9805
	MAUVE, 64k prefill	0.9954	0.9987	0.9905	0.9861

6.3 Recommended configuration

The deployment exposes one quality parameter, the recall target $\tau = 0.90$: the gate budget, the conformal order statistic, $\mathrm{min_hard}$, the calibration window and the $Z_{\max}$ clamp all follow from it in closed form. The structural constants ($\rho = 1/32$, bandwidth $\kappa = 16$ bins, sketch rank $r = 64$, four sinks, the unclosed tail, the 4096-token close granularity) are not tuned per deployment; every default traces to a pre-registered measurement (Appendix H).

Every remaining quantity a reimplementation needs, at $\tau = 0.90$ on this checkpoint. Geometry: $H = 32$ heads, $d_{\mathrm{head}} = 128$, $d_r = 64$, $d_c = 512$, so a row is 576 values and $\lambda = 192^{-1/2}$. Derived from τ: gate budget $\beta = (1-\tau)/2 = 0.05$; per-scan target $\tau_s = \tau/(1-\beta)$; $\mathrm{min_hard} = \lceil \tau_s/(1-\tau_s) \rceil = 18$, the fewest hard samples for which the conformal quantile is defined; $Z_{\max} = 8$. Calibration collects 8 decode steps and doubles the window to at most 64 until $\mathrm{min_hard}$ hard samples exist; a prefix that never reaches it serves clamped at $Z_{\max}$, which over-fetches and cannot under-recall. The entropy gate is one threshold: the β-quantile of the kept tier's softmax entropy over the hard calibration queries; below it the scan still runs and the fetch it would have triggered is skipped; if that threshold would leave the gate open on more than 0.60 of calibration queries it is discarding nothing and is disabled outright. A miscalibrated gate costs throughput, never recall. Row identifiers are `int32`; the activation threshold, which serves a request densely below a configurable length, defaults to zero and is zero in every measurement here.

7 Related work

The separating axis is *when* the compression decision is taken relative to the queries that will read it, which places methods rather than ranking them (Table 8, Appendix E). A long-lived cache admits three placements for the decision: KVzip (Kim et al., 2025) spends prefill compute that grows with context to decide what to discard; ArkVale (Chen et al., 2024) spends nothing and accepts a digest bound valid only over the rotation orbit (Appendix B); and exact merging (Tian et al., 2026) is bounded by Proposition 2. Query-aware page selection (Tang et al., 2024) shares our first column instead: it deletes nothing, and its decision cannot be taken before the query arrives. VestigeKV settles only how much to read back, the one decision an archive lets it revise every step. The one head-to-head a fair comparison admits is therefore against KVzip, which also decides before the queries arrive: Table 4, at matched budget against the random floor.

8 Limitations and conclusion

The salience channel is measured on one model family; the dynamic that loads it is not understood (three pre-registered explanations refuted, Appendix G), so until reproduced on another NoPE-MLA model it is a property of this checkpoint, not the architecture. It is also a property of *natural* text, and measurably so. Against an oracle built from each request's own absorbed decode queries at the same budget, tier 1's selection overlaps by 9.9% on real prose and 4.7% on RULER's haystack, against a 3% random floor: 3.3× random where the text was written, 1.6× where it was assembled from key–value pairs (35 snapshots per corpus, dumps not retained; supplement). The overflow fallback covers that minority regime, at a cost this paper reports rather than assumes.

THE USE OF LARGE LANGUAGE MODELS

In this work, we used generative AI tools to assist in the writing and checking of proofs and to brainstorm arguments; to implement the experiment code and orchestrate the benchmark campaigns, including the automatic regeneration of every numeric macro in this paper from archived run records; and to provide feedback on experiment design. We also used generative AI tools to edit the manuscript for grammar and readability, to fill in supporting and appendix text, to draft the outline, to check academic conventions, to search and screen the literature, to produce the figures programmatically, and to check and compress the submission formatting. We have not used generative AI tools to originate the research: the theoretical framework, the mathematical claims, and their derivations are the authors' own, no synthetic data was generated, and no results were interpreted by AI. The main-body framing, including the method and procedure descriptions, was written by the authors. We have reviewed all AI-assisted work: every proof was checked manually, every reported number either regenerates from an archived run record or carries its source in a comment (Reproducibility statement), and the gates of Appendix F were driven with known-bad inputs before being trusted. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The supplementary archive contains: 20 pre-registration files with decision rules dated before their data and verdicts appended after, including retracted predictions; the run records behind 89% of the numbers printed here, with the scripts that regenerate those numbers from them; the full harness with its gates; and the queue file that ran every serving experiment, which is also the record of the exact commands, seeds and ordering, together with the per-job status, wall time and log path the runner wrote beside it. The remaining 11% are hand-maintained values, each carrying its source in a comment, and *those source files are not in the archive*: they predate it, and the trees that produced them did not survive. We report that rather than let a reader discover it: the regenerable set is what can be checked, and it is the set every headline number in Section 6 is drawn from. The serving stack is one upstream `sglang` tree (release tag `v0.5.20`) plus the additive backend commits, whose full diff accompanies the archive as an anonymized patch file. The measurements were taken on an earlier upstream revision and the tree was rebased onto this tag afterwards; the rebase is reported rather than assumed harmless, and it is checkable in the archive. The eighteen intervening upstream commits touch no file under `layers/attention/` and none of the five shared files this backend modifies. Their one numerical change is a clamp fix in the FP8 MoE path, which this bf16 checkpoint does not take, and the subsystems they do change (hierarchical cache, DP attention) are disabled in every run here. The RULER tables and the serving pair were measured on one tree, within nine hours of each other, and that tree already carries the two corrections found after the first serving freeze: a per-request sketch basis, and int64 row addressing on pools beyond 2^{31} elements. Neither changes the arithmetic of a measured step, and the earlier records taken before them were retired rather than reported (Appendix F). Raw activation dumps are excluded for size and are regenerated by the included extraction script from the public checkpoints.

REFERENCES

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. In *Second Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=ayi7qezU87`.

Renze Chen, Zhuofeng Wang, Beiquan Cao, Tong Wu, Size Zheng, Xiuhong Li, Xuechao Wei, Shengen Yan, Meng Li, and Yun Liang. ArkVale: Efficient generative LLM inference with recallable key-value eviction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=4oAt5L4lYe`.

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. In *Advances*

in Neural Information Processing Systems, volume 37, 2024. doi: 10.52202/079017-1663. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5dfbe6f5671e82c76841ba687a8a9ecb-Abstract-Conference.html.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=JFygzwx8SJ.

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. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/28ab418242603e0f7323e54185d19bde-Abstract-Conference.html.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, Tianqi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. MatryoshkaKV: Adaptive KV compression via trainable orthogonal projection. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 86669–86690, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d7b351608d824a4680344a02b180a947-Abstract-Conference.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, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=RlqYCpTu1P.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1043. URL https://aclanthology.org/2024.emnlp-main.1043/.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(12):371–421, 2008. URL https://www.jmlr.org/papers/v9/shafer08a.html.

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, 2025. URL https://proceedings.mlr.press/v267/sun25b.html.

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, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Yuxuan Tian, Zihan Wang, Yebo Peng, Aomufei Yuan, Zhiming Wang, Bairen Yi, Xin Liu, Yong Cui, and Tong Yang. KeepKV: Achieving periodic lossless KV cache compression for efficient LLM inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33259–33267, 2026. doi: 10.1609/aaai.v40i39.40611. URL https://ojs.aaai.org/index.php/AAAI/article/view/40611.

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. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. doi: 10.52202/079017-3801. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5e5fd18f863cbe6d8ae392a93fd271c9-Abstract-Conference.html.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cFu7ze7xUm.

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/.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract-Conference.html.

A PROOFS

Proof of Lemma 4. Each excluded weight is at most $e^{-\gamma}$ times the maximal included one, so the excluded mass is at most $(T - |F|)e^{-\gamma}$ relative to it; summing over excluded rows bounds the output difference by that mass times $\max_u \|v_u - \tilde{o}_t\| \le 2 \max_u \|v_u\|$. □ □

Proposition 2 (Minimum exact cache)**.** *Let V be the subspace of realizable absorbed queries ($\dim V \le d_{\text{head}}+d_r$ per head) and call two cache rows equivalent when they agree on V and carry the same value vector. Provided the value map separates inequivalent rows, a representative cache $\{(C_g, b_g)\}_{g=1}^{B}$ with logit offsets whose softmax output equals the full cache's on a relatively open set of V satisfies $B \ge \#\{\text{equivalence classes}\}$.*

Proof of Proposition 1. Factor $\mathrm{Rot}(\theta_j(u-t)) = \mathrm{Rot}(-\theta_j t)\,\mathrm{Rot}(\theta_j u)$ and put $w_j = \mathrm{Rot}(\theta_j u) r_u^{(j)} - \mathrm{Rot}(\theta_j v) r_v^{(j)}$, a vector independent of t. Then $s_t(u) - s_t(v)$ is the position-free gap plus $\sum_j \langle q^{(j)}, \mathrm{Rot}(-\theta_j t) w_j \rangle = \sum_j \|q^{(j)}\| \|w_j\| \cos(\theta_j t + \psi_j)$. By Weyl's theorem the phase vector $(\theta_j t)_j$ is equidistributed on $\mathbb{T}^p$, so some t brings every cosine within ϵ of 1 simultaneously

and the supremum is $\sum_j \|q^{(j)}\| \|w_j\|$. Setting $R = I$ gives $w_j = 0$ whenever $r_u = r_v$ and, in general, a t-free difference. □ □

Proof of Proposition 2. Write the full cache's softmax weights $p_u(Q) \propto e^{\langle Q, \tilde{C}_u \rangle}$ and the representative's $p'_g(Q) \propto e^{b_g + \langle Q, C_g \rangle}$. Restrict to a relatively open sub-chamber $\Omega \subseteq V$ on which all affine scores differ and both lists are strictly ordered. Because the value map separates inequivalent rows, equality of the output vectors on Ω forces equality of the ordered softmax lists, hence of the ordered affine score functions. Two affine functions $\langle \cdot, a \rangle + \alpha$ and $\langle \cdot, a' \rangle + \alpha'$ that agree on an open set satisfy $a = a'$ and $\alpha = \alpha'$; applied from the largest score downward, every equivalence class's score $\langle \cdot, \tilde{C}_u \rangle$ appears among the representatives, with e^{b_g} reproducing the class multiplicity m_u (the m_u identical rows of a class contribute one representative of weight m_u). Hence $B \geq \#\{\text{classes}\}$. Equivalently, clearing denominators turns output equality into an identity of finite sums of real exponentials, which are linearly independent on open sets for distinct exponents (restrict to a generic line and invoke Vandermonde). Under RoPE the displayed distance bound makes all T rows pairwise distinct on $V = \mathbb{R}^d$. Under NoPE+MLA, dilation $c_v = a\, c_u$ ($a > 0$) with $r_v = r_u$ normalizes to one stored row, exact up to the RMSNorm stabilizer, i.e. when $\|c\|^2 \gg d\epsilon$, so the widening of the merge class lives at the pre-cache activation level and the stored cache sees vector equality. The value side is consistent because v is a function of the row; in standard attention, with V cached independently, no such consistency holds and the numerator identity is not even well-posed. □

A.1 Ranking stationarity: the commutation of eviction with time

A consequence of Lemma 1 worth isolating, because it licenses a schedule the experiments use and RoPE cannot.

Corollary 1 (Ranking stationarity)**.** *Under NoPE, $s(q_t, u) = \lambda \langle Q_t, \tilde{C}_u \rangle$ (Eq. 1) contains neither t nor the age of row u. Hence any row-intrinsic ranking (such as σ), once computed, is valid at every later step, and for a fixed budget m a global top-m maintained incrementally (a heap, later rows displacing earlier ones) equals the one-shot global top-m computed at the end. Where the budget grows with T (the deployed $m = \rho T$), the equivalence is re-established at each block close by a fresh global top-m over all closed rows rather than maintained incrementally. No decision ever needs re-scoring; the rebalance re-ranks, it does not re-measure.*

The measured footprint is the streaming result of Section 6: blockwise compression equals one-shot global compression trial by trial at both ratios, and so does blockwise compression with global rebalance (constant m, later rows displacing earlier ones) at matched final budget. Under RoPE the corollary fails at an intrinsic rate: the cached score $s(q_t, u) = \lambda\, q_t^\top R_{t-u} W_K \tilde{C}_u = \sum_i a_i(u) \cos(\omega_i(t-u) + \phi_i(u))$ oscillates per frequency band, so the pairwise order of any two rows flips as t grows and a frozen ranking goes stale; the only remedy is re-scoring every archived row against the current position at every rebalance, so no eviction decision is ever final. NoPE deletes this staleness failure mode entirely; the remaining failure of a constant-m schedule is genuine budget contention (a later high-σ row displacing a needle), a memory-for-quality trade, not signal failure. The schedule itself is prior art (H2O (Zhang et al., 2023) lineage); the dividend attaches to the signal.

B The digest obstruction under RoPE

Table 3: Each ArkVale-style limitation, its mechanism, and the effect of removing rotation. The two worst-layer recall rows this table carried are withdrawn: their record and the script that produced it are in no surviving tree, so they cannot be checked and we do not print them. What remains is the radius argument, which is analytic (Eq. 5), and the winner-set union, which has its own record.

limitation	mechanism	RoPE	NoPE
digest looseness	orbit-inflated radius (Eq. 5)	1.78–1.86× spread	(removed)
per-step re-ranking	winner set sweeps with position	10.98–18.96% union	6.01–14.12% union

The prior recall design (Chen et al., 2024) is solving a different problem, and we run no head-to-head against it for that reason. It evicts pages from the GPU cache and keeps a backup it can recall from, so what it buys is GPU capacity and what a recall costs is a transfer. VestigeKV keeps both tiers resident, so its archive costs capacity and a recall costs no transfer at all. The two are not alternatives for one deployment: a reader who is short of GPU memory wants the former, and a reader who has the memory and wants the bytes per step down wants the latter. What does carry over is the digest, because a design that recalls at all needs a cheap score for a row it is not reading, and that is where the positional geometry decides the outcome.

Their score for an evicted page K is a bounding-sphere digest, $I(q, K) = \max_{k \in K} q \cdot k \le q \cdot c + r\|q\|$, sound only when r encloses every key, and adopts a non-enclosing radius because the sound one overestimates, giving up the guarantee. Under RoPE that dilemma is structural. A page is P consecutive positions of cached keys $R_u k_u$; even for *constant* content k (j indexes the $d/2$ two-dimensional rotation blocks),

$$\|R_u k - R_v k\|^2 = \sum\nolimits_j 4 \sin^2 \tfrac{\theta_j (u-v)}{2} \, \|k^{(j)}\|^2 \implies r^2 \gtrsim \sum\nolimits_{\theta_j \ge 2/P} \|k^{(j)}\|^2, \tag{5}$$

so the radius is of order the fast-frequency key norm regardless of content coherence: the digest cannot be tight. The center fares no better: per frequency pair the page mean carries the Dirichlet factor

$$\left| \tfrac{1}{P} \sum\nolimits_{u=0}^{P-1} e^{i \theta_j u} \right| = \frac{|\sin(P\theta_j/2)|}{P\,|\sin(\theta_j/2)|} \ll 1 \quad \text{for fast } \theta_j, \tag{6}$$

so $q \cdot c$ is blind to exactly the dimensions inflating r. NoPE removes both at the root (rows are content-only) and replaces the estimate with structure the digest never had: an exact summand (Lemma 3), a per-token certificate with a measured accuracy knob, and by Section 3(a) a fixed winner set, so rankings may be prefetched where the RoPE family $A_h(\Delta)$ forces re-ranking every step. Table 3 holds the measured comparison.

C Experiment tables (harness)

Table 4: The one head-to-head a fair comparison admits: KVzip also decides before the queries arrive, so the two share a quadrant of Table 8. Needle intact rate, 24 trials per cell at L=8192, matched budget, harness protocol of Appendix C, Instruct checkpoint. The first row is the floor: a uniform random draw at the same budget. Both ranges are printed because they are different operating points, and the comparison is only fair when that is said: 2–4× is the range KVzip's release targets and where it is designed to work, while 32–128× is an order of magnitude outside it. Cells there report what the method does when extrapolated past its stated range, not a failure of it, and we draw no conclusion about KVzip from them. What the row does separate is where each design spends: KVzip buys its ordering with a scoring pass, a per-request cost that grows with S, and then evicts; VestigeKV spends 8–64 decode queries, flat in S, and keeps a certified path back to every archived row. Neither is free, and the second is the one this paper argues for only in the long-context regime where a per-request cost has decode steps to amortize over. **Tier 2 is deleted here in the third row, so the selector's own contribution is visible; with recall on, every cell of that row reads** 1.00**. Ablation only: what is left is an incomplete algorithm, not a configuration this paper deploys.**

	2×	4×	8×	32×	128×
random, budget-matched	0.54	0.17	0.04	0.00	0.00
KVzip (Kim et al., 2025)	1.00	0.54	0.00	0.00	0.00
VestigeKV, **tier 2 deleted**	1.00	1.00	1.00	0.92	0.83
VestigeKV, both tiers	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

Provenance of the gap, and the effect of a recall margin. With stats on, the calibrated regime is tight: over 126950 decode steps of a 4k→128k stream the fetched-row count per scan is p50/p90/p99 = 0/0/4096 rows with a dense-fallback rate of 0.012 on this rank: the rate is *per rank* and the two disagree by an order of magnitude on the same job (0.0027 and 0.0548 at 256k), so a single-rank figure understates the work the fence does; the 64k RULER prompts, whose answers average 14

Table 5: Deployed-path quality, final fused code; bars frozen before data, PASS on both checkpoints. MAUVE: gpt2-large, per-context sampling seeds shared across arms, divergence-derived; the bar is one-sided, so a gap in either direction inside it is divergence, not signal. Needle: one fact spliced at random depth into 128k filler, greedy, fixed seed identical across arms. GSM8K-Platinum is questions answered correctly out of 1209. Harness LM cost transfers via bit-identical row sets: median ΔCE +0.0124 nats.

experiment	Base dense	Base vestigekv	Instruct dense	Instruct vestigekv	bar
gsm8k-platinum 64-shot	1027	1031	1107	1097	within 2σ of dense
MAUVE 4k (16 ctx)	0.9997	0.9960	0.9345	0.9805	$\geq$ dense -0.10
MAUVE 64k (64 ctx)	0.9954	0.9987	0.9905	0.9861	$\geq$ dense -0.10
serving needle 128k ($\times$32)	32/32	32/32	32/32	32/32	$\geq$ dense
+ Chinese corpus	8/8	8/8	8/8	8/8	$\geq$ dense

Table 6: RULER on Kimi Linear Instruct, dense/vestigekv per cell (50 samples per cell; one sample moves a cell by 0.02). lm-eval 0.4.13 `ruler` group, its generators and lm-eval seeded 0, raw `/v1/completions` prompts, greedy, 128 tokens; server: CUDA graph on, radix cache off, `-random-seed 0`, TP=2 on one node; every `-vestigekv-*` flag at its default. Dense 139 min, VestigeKV 142 min for 3250 prompts. Row and column means are computed from unrounded accuracies; displayed cells are rounded to two decimals.

task	4k	8k	16k	32k	64k	mean
niah_single_1	1.00/1.00	1.00/1.00	1.00/1.00	0.96/0.96	0.96/0.98	0.984/0.988
niah_single_2	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.000/1.000
niah_single_3	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.000/1.000
niah_multikey_1	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.000/1.000
niah_multikey_2	1.00/1.00	1.00/0.98	1.00/0.92	1.00/0.90	1.00/0.92	1.000/0.944
niah_multikey_3	1.00/1.00	1.00/0.92	1.00/0.84	1.00/0.84	1.00/0.76	1.000/0.872
niah_multiquery	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.000/1.000
niah_multivalue	1.00/1.00	1.00/1.00	1.00/1.00	0.99/0.99	1.00/1.00	0.998/0.999
vt	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	0.999/0.999
cwe	0.99/1.00	0.97/0.95	0.98/0.98	0.98/0.98	0.97/0.97	0.977/0.977
fwe	0.79/0.79	0.97/0.97	0.99/0.98	0.98/0.95	0.71/0.69	0.885/0.875
qa_squad	0.79/0.78	0.78/0.73	0.74/0.67	0.74/0.71	0.76/0.64	0.760/0.704
qa_hotpot	0.70/0.66	0.68/0.66	0.62/0.60	0.60/0.56	0.62/0.54	0.644/0.604
mean	0.944/0.940	0.953/0.940	0.948/0.922	0.942/0.915	0.924/0.884	0.942/0.920

decode steps, are served mostly by the provisional index (identity basis, $z{=}Z_{\max}$) and fall back on 0.337 of scans (p50 784 rows). We first attributed the residual RULER gap to the criterion rather than the certificate, on the ground that a row is fired only when some head could score it above its best kept row, so near-tied distractors would never be "needed". Instrumenting the *answer* steps refutes that. Over 3248 (step, layer, lane) records of `niah_multikey_3` at 64k, the attended set omits the highest-scoring row of the whole context on 20% of records (22.9% of those served by a *calibrated* index) and the omitted row ranks 24th at the median under the certified score (1177th at p90) while falling 7.06 scaled logits short of the best kept row. Seven logits is about $1100\times$ in softmax weight and 33 times the certificate's own scale, so it is a bias in the order the sketch induces, not a threshold the certificate could be inflated past. The single-needle control (`niah_single_1`, which matches dense at 0.988) omits the top row on 6.9% of records. The gap is a resolution failure of the rank-64 content sketch on queries that must match several distinct keys, and it is the open problem this work does not close. A pre-registered sweep of a recall margin δ below the kept maximum (a dropped row then holds at most $e^{-\delta}$ of the maximum's weight; sweep on the affected tasks at 16k–64k) recovers them (multi-key-3 0.667$\rightarrow$1.000, two-hop QA 0.567$\rightarrow$0.667, targeted mean 0.722$\rightarrow$0.878 at $\delta{=}2$) but every candidate failed the frozen short-answer cost gate (fetch p50 at the capacity, fallback 0.43$\rightarrow$0.71), because the margin compounds with the provisional index. The default stays $\delta{=}0$; the archive carries the verdict and the follow-up pre-registration (calibration during prefill from absorbed prompt queries, so decode never starts on the provisional index).

D BASELINE BEHAVIOR ON THIS MODEL

The deployed form always runs both tiers. **Tier 2 is deleted here throughout this appendix, because the transplanted policies have no recall tier of their own, so deleting ours is what makes the component counts match and the rows comparable. Ablation only: what is left is an incomplete algorithm, not a configuration this paper deploys.** Nothing here is runnable in production and none of it is this paper's quality; the tables support one sentence of the main text: with tier 2 deleted, the partition is lossy eviction.

Table 7: **Tier 2 deleted throughout (see above).** Needle intact rate when compression precedes the query. The denominators are not uniform: each transplant row is a single run (24 trials at L=8192, 12 at L=32768) while the VestigeKV rows pool two seeds (36 trials at L=8192 under 32× and 128×, 24 at L=32768, the 8× cell from a separate width sweep). VestigeKV is not a row here: the same arm, at the same length and ratios, is Table 4, measured on one surviving record at the deployed bandwidth and seed. Printing it twice from two seed pools put two numbers in this paper for one measurement, which is worse than printing it once. **The non-VestigeKV rows are our own transplants of those policies into an HF-forward harness, not the authors' implementations**, and no quality or efficiency claim against them is made here or anywhere in this paper: they are run outside the order they were designed for, so the rows report what that order removes. Each remains strong where the query is present at compression. The recent-window variant is the control that separates a missing input from a small budget. Every number the paper's claims rest on comes from a standard harness instead: RULER through `lm-eval`, GSM8K-Platinum through `sglang`'s own few-shot runner, MAUVE through the reference implementation, and every serving number through `sglang`'s `bench_serving`.

	8×	32×	128×
H2O (Zhang et al., 2023)	0.00	0.00	0.00
SnapKV (Li et al., 2024)	0.33	0.00	0.00
H2O + recent (half budget)	0.00	0.00	0.00
StreamingLLM (Xiao et al., 2024b)	0.00	0.00	0.00
H2O / SnapKV, L=32768	0.08 / 0.17	0.00 / 0.00	0.00 / 0.00

E PLACEMENT BY DEPLOYMENT

The axis of Section 7 in full, with the warm-up each family pays and what that warm-up settles.

Three adjacent placements sit off this axis and are worth naming. MInference (Jiang et al., 2024) fixes a sparse pattern at prefill and pays full attention to discover it: it accelerates prefill, not the long-lived decode cache. Offload systems (Xiao et al., 2024a; Sun et al., 2025) keep recall by moving rows to host memory; VestigeKV's archive stays on the GPU, trading their capacity ceiling for zero PCIe traffic per step. Irminsul and Kamera[2] already exploit NoPE's position independence for cross-request prefix reuse; the increment here is that the same property makes a pre-query eviction signal exist, and provably so.

F MEASUREMENT INTEGRITY: FULL PROTOCOL

What the harness shares with the deployed system. The needle and ablation tables (Appendices C and D) come from an HF-forward harness (`harness/` in the code release) that records exactly which rows selection and recall keep and fetch. Two links tie it to the deployed system. The fused kernels' fired-and-fetched row sets are bit-identical to the harness's on shared inputs, and a per-step row-set assertion held over a full benchmark.

Where the harness differs, and which way that points. Those links hold the row sets fixed; they do not make the harness the deployed algorithm. It runs its own forward pass rather than replaying

[2]Both are arXiv preprints with no peer-reviewed version at the time of writing, so they are named in prose and carry no bibliography entry, the convention this paper already uses for Kimi K3.

Table 8: Placement by deployment, not by score. The first column is the only one a deployment chooses; the rest follow. A policy that deletes must be right at compression time; one that can recall need only be right about what to read back. Query-aware page selection deletes nothing either, and pays for it in the first column: its decision cannot be taken until the query exists.

family	decision taken	warm-up: cost	deletes?	recall?
observed attention (Zhang et al., 2023; Li et al., 2024; Oren et al., 2024; Xiao et al., 2024b)	queries present	hold everything meanwhile	yes	no
structural splits (Xiao et al., 2025; Cai et al., 2025)	offline profile	offline, once	yes	no
trained pattern (Yuan et al., 2025; Lu et al., 2025)	pre-training	training	model changed	—
trained projection (Lin et al., 2025)	training	training	model changed	—
surrogate scoring (Kim et al., 2025)	before queries	$\approx 2\times$ prefill, grows with S	yes	no
page digests (Chen et al., 2024)	before queries	none, by not adapting	yes	bounded
query-aware page selection (Tang et al., 2024)	queries present	none	**no**	n/a
exact merging (Tian et al., 2026)	before queries	none	merges	—
VestigeKV	before queries	8–64 decode queries, flat in S	**no**	yes

engine traces, and it differs in four places. Each one is stated with its *direction*, because all four point the same way, and it is not the flattering one.

(i) Fetch quota. The harness admits, per query and head, only the top-16 firing rows by certified score; the engine admits every firing row. Its admitted set is a subset of the engine's. *(ii) Sinks.* The engine always keeps four sink rows in the attended tier; the harness keeps none, so the engine keeps strictly more. *(iii) Inflation.* The harness takes the smallest z on a grid that reaches recall τ on its calibration rows; the engine takes the distribution-free conformal quantile at $\tau_s = \tau/(1-\beta)$ with the finite-sample correction of Section 6.3, which is larger, so the engine fires and fetches more. *(iv) Sketch basis.* The harness fits V on prefill queries; the engine fits it on the decode calibration window, which is the distribution the evaluated query is actually drawn from.

In all four the engine admits a superset of what the harness admits, or fits a better-matched basis, so for retrieval the harness is a *lower bound* on the engine and not a flattering proxy: a needle the harness recovers is one the engine recovers. The exception is not about level but about statistic: a z chosen in-sample makes the harness's own calibration recall optimistic as a number, which is why the recall this paper reports (0.929) is read off the engine and the harness's is not quoted. Every headline result — Table 2, the 32/32 serving needle, the latency and throughput curves, and the spectral and oracle evidence of Section 6.2, which is computed from the engine's own calibration dumps — is measured on the served system with none of the four in play. The harness carries the ablations, where all arms share the four differences and they cancel in the contrast.

Why needle and not continuation loss. Continuation loss cannot arbitrate retrieval: the recent-only floor matches dense on CE and still pays $+15$ nats on the needle answer (Appendix C). Needle is therefore the primary metric, and CE enters only as corroboration.

Pairs are launched alike or discarded. The two arms of a latency pair run under one node and one configuration, and a pair whose launch knobs differ is discarded rather than reported. That happened to this curve's predecessor, whose replacement is the pair Figure 2 reduces. The live-

archive gate that admits generation-time eviction reports parity PASS, a median ΔCE of +0.0124 nats and needle 8/8.

The n=50 grid's generations, and what survives. Those generations were later overwritten by a run of the same shape on the other checkpoint. The per-item scores survive in the archive's reduction, and they are what the paired reading needs; the paired reading itself was added after the run and outside its pre-registration, and is reported as such.

Two objections this design anticipates. A reader may ask whether our KVzip transcription is simply broken. A broken one could not clear the random floor at $2\times$ (1.00 vs 0.54) or $4\times$ (0.54 vs 0.17), and it does. A reader may also ask why the head-to-head compares retrieval and not speed. Both serving arms are kernel-optimized, so an untuned transplant would measure implementations rather than methods; against the observed-attention family nothing is claimed at all, because what separates them is Table 8's axis and not a score.

What a cache policy owes. Not universal applicability, but noticing where it does not apply. The gap between the fallback rate on RULER and the rate on a long decode is the point: the certificate reports the step it cannot serve sparsely, and that step is served exactly.

Every check was driven with a known-bad input and shown to refuse it before being trusted; every threshold was frozen, dated, before its data; all reported rates are exact fractions over the pooled trial counts stated in each table. The serving port carried a per-decode-step attended-row-set assertion on every pipeline rank and every layer, each failure mode unit-tested to fire on the broken form; the final benchmark campaign ran a full proof round with the assertion enabled on both ranks and both arms across all batch sizes: zero violations. The assertion was a debug build and is not a switch on the released branch, whose instrumentation surface is the six environment keys `environ.py` documents, so the reader inherits the result rather than the means to re-run it. All wall-clock numbers come from `sglang`'s own benchmark module and its server-side decode log, with no benchmarking code of ours; each curve is collected within a single launch so fixed launch-state terms cancel in the Δ-vs-4k reading; every launch log prints the checkpoint's content-level fingerprint.

The server log is the per-token authority on a long stream, and the reason is worth stating because the alternative is worse than noisy. At one detokenize-and-emit event per generated token, the cost of that path grows with the length of the stream; once emission falls behind decode the server accumulates a send backlog and the client's arrival stamps measure its own drain rate. The faster arm builds the deeper backlog, so the metric reports it as the slower one. On the pair that diagnosed this the server finished generating at 2297 s while the client was still receiving at 5308 s. The client-side records of those runs are therefore deleted rather than archived, with a manifest naming each file and why; a reader who reduced them would compute the wrong sign. The benchmark client now measures the collapse in arrival rate across a stream and records `itl_trustworthy` with it, so a run that hits this regime says so in its own output instead of producing a number that looks ordinary. Each gate paired with the known-bad input it was proven to catch: Table 9.

Experiment procedure: one tree, one queue. One tree: the serving stack is release tag `v0.5.20` plus this work's commits, and every measurement reported here runs on it. Serving experiments are entries in a queue file, executed by a runner; nobody types a launch command, and no agent starts one. Before the runner is allowed near a GPU an auditor resolves each pending entry into the command and the output path it will actually produce, and refuses nine things: an entry pinned to any other tree, an output file that already exists, paired arms whose environment or arguments differ in anything but the arm, an arm with no launcher, a client the runner does not implement, a seeded client with no seed given or a seed other than 0, a launch that does not appear verbatim in the repository's registry, a captured graph wider than the batch it decodes, and a serving-performance entry that leaves the radix cache on. Every run on every line uses seed 0: a paper reporting seed 7 owes its reader an account of the six before it, and the only cheap way to owe nothing is to take the first one. A registered name is not a registered experiment either (the knobs can be edited afterwards and the name still matches), so what the auditor compares is the canonical launch, character for character. The offline harness experiments are the same discipline in script form, each declaring what it varies and what it holds fixed, and the declaration is required rather than encouraged: writing

it is where a loose variable gets noticed, which is before the GPU hours rather than after. Each refusal, here as elsewhere, was driven with a known-bad input before it was trusted.

The sparse arm does not repeat bit-for-bit; the dense arm does. Tier 2 is built twice for every (request, layer): a provisional index on the first decode step, and the calibrated one once enough real decode queries exist to fit it. The calibrated build runs on a background thread and a side stream, and the decode loop adopts it at whichever step finds it finished, so the step at which a request stops using the provisional index is set by thread and stream scheduling rather than by the prompt, the weights or the token stream. Greedy decoding removes the seed, so the dense arm's output is a function of the token stream alone and repeats exactly; the sparse arm carries this one further input, which no seed controls. The two indexes inflate the certificate differently, so they fire different rows and give different logits; almost always the argmax is unmoved and the runs agree, and occasionally it is not and they diverge from that token on. Repeating GSM8K-Platinum end to end moves the sparse arm by 7 questions of 1209 and the dense arm by 0. Both indexes are sound, so what is not unique here is the output and not the recall guarantee, and the quality bars are frozen in questions rather than in the direction of a difference, which is what lets a conclusion survive a wobble of this size. Making the transition deterministic means blocking decode on the build; that is the right fix and it is not on this tree, because it would move every measured quality number. `docs/nondeterminism.md` carries the full account.

The throughput count is a lower bound on the sparse arm. Each throughput point counts tokens into the KV pool over a window where the live request count is constant, and is admitted only if that count agrees with the server's own reported rate to within 2%. The two agree on the dense arm at every width we ran (worst case 0.6%). On the VestigeKV arm they diverge, in one direction, by an amount that grows with concurrency: under 1% up to 16 live, and past the tolerance above it. The cause is that the pool is what is counted and eviction removes rows from it, so a block close during the window makes the pool grow by less than the arm generated, monotonically, which is why it does not look like a round boundary. The count therefore understates the sparse arm and understates it more as concurrency rises, so every ratio reported here is conservative, and the series ends where the agreement check stops admitting points rather than where the hardware stops. We did not widen the tolerance to admit them: a check retuned until it accepts the data it was built to audit is not a check.

Corrections found after the serving freeze. Two defects surfaced while running RULER and are fixed on the archived follow-up tree. (i) The sketch basis of each layer's first calibrated build was cached for the process lifetime and reused by every later request; the certificate stays sound under any orthonormal basis but loosens, so the frozen serving numbers over-fetched and are conservative. (ii) The index tables moved to int32, and the decode kernel multiplied a row id by the row stride in that dtype: on the 5.1M-row Kimi Linear pool ($5.1\mathrm{M} \times 576 > 2^{31}$) rows past 3.7M overflowed to wrong addresses (37 of 260 garbage answers on a 13-task probe; 0 on the frozen tree). Row addressing keeps the base kernel's int64; both fixes carry regression tests, and the RULER tables were measured after them. Neither fix touches the arithmetic of a measured step: the addressing change widens a pointer only, and the per-request basis changes how tight the certificate is, not what it certifies; the per-step row-set assertion above passed on both trees. The left panel of Figure 2 was re-collected after the fixes: both arms in one sitting on the released tree, and the two server logs it reduces ship in the archive. Every value in it, the rightmost context included, is derived from those logs by one script, the rightmost context included: a point at N is the 4096 tokens ending at N, so the last one ends where the stream does and no endpoint has to be argued for. Choosing that endpoint by hand is how the previous figure came to report a context nobody could derive, a round-sounding 508k that appears in no record and sits 16k short of the end. Neither did its ratio survive. That $1.53\times$ was read across two arms from different jobs, differing in three launch knobs (mamba cache slots, captured graph width, radix cache) besides the attention backend. Against the controlled pair's $1.45\times$ the gap is mostly the graph width: an unused captured lane scans rows in proportion to the context, so it costs 0.4% below 64k and 5.6% at the end of the stream. The number above is the controlled one.

Table 9: Gates and process.

gate	known-bad input it was proven to catch
identity operator (ΔCE $= 0$)	`eager`/`sdpa` kernel mismatch (1.4×10^{-1})
uncompressed-baseline gate	trials the base model cannot retrieve
hook fire counts	silent non-capture (verdicts about nothing)
leakage floor (cache destroyed $\Rightarrow$ 0.00)	uncompressed layers recovering the needle
pre-registration, 20 verdicts	three retracted mechanisms; a rejected learned selector

G LOCALISATION OF THE GAP, AND THE STUDY'S BOUNDARIES

Which band the n=50 grid is read against. Both arms answer the same 3250 items under one seed, so they are not independent samples and a band built as if they were charges the estimate with between-item variance that pairing removes. Paired: 136 items answered differently, 110 of them against VestigeKV, $2\sigma = 0.006$ against a gap of -0.022, exact sign-flip $p = <10^{-5}$. An unpaired band over the same cells is wide enough to admit the gap, which is why the earlier reading of this grid called it indistinguishable from dense; the pairing is what resolves it, and the per-item scores it needs survive even though the generations of this grid do not.

The same correction applies to the 128k and 256k pass and in the same direction: 8 disagreeing items, 8 against VestigeKV, $2\sigma = 0.035$, exact sign-flip $p = 0.008$, against an arm-mean band of 0.059 that treats them as independent.

The gap, in full. The gap is carried by the two whose *haystack is itself made of key–value pairs* (-0.056, -0.128 with UUID keys). Multiplicity alone costs nothing (`multikey_1`, `multiquery` and `multivalue` plant or ask four, all +0.000) because a rank-r sketch fitted to the archive's own content, and shared across a layer's heads, cannot separate rows drawn from the bulk it was fitted to (Table 6). Reading the generations splits that gap in half *on Instruct*: at n=50 only 6 of 100 items return a *different* needle, and as many return the right one mis-copied. On Base the split does not hold and the gap is selection almost throughout (70 against 11); both are Appendix G.

That grid stops at 64k. A second pass runs the same thirteen tasks at 128k and 256k (26 cells at n=5) with its reading fixed beforehand: 0.926→0.880, a gap of -0.046 inside an arm-mean band of 0.059. That band treats the arms as independent and they are not, both answering the same 130 items under one seed; a paired reading is sharper in the direction that costs this paper: 8 disagreeing items, 8 against VestigeKV, $2\sigma = 0.035$, exact sign-flip $p = 0.008$: a real effect that five prompts per cell still cannot size. It deepens on the same two key–value tasks (6 of the 26 cells move, 0 upward) and spreads to `fwe` and `qa_hotpot`, which the 64k grid does not implicate (per-cell readings, Appendix G).

RULER is adverse twice over: planted content, by the mechanism above, and 14-step answers that never amortize calibration, so the index never leaves its conservative clamp and 0.337 of its scans overflow into the exact path against 0.012 on a long decode (Appendix G). The second of those is a property of the evaluation rather than of any one method: an answer a dozen tokens long gives a per-request adaptation nothing to amortize over, ours included, and the same prefill decoded for 4096 steps instead moves the rate by two orders of magnitude. That is what the fallback is for. The certificate reports a step it cannot serve sparsely at the recall target, and the step is served exactly, paying throughput rather than answers. The fallback is a component, not a setting: the release branch exposes no switch for it, only a debug key. RULER is used here as an adversarial test of the cache policy, not a capability score: it asks whether an evicted row is the one the next query needs. Real-document evidence sits inside RULER too: `qa_squad` and `qa_hotpot` pose real questions over real paragraphs from their own corpora, answered extractively, -0.056 and -0.040, both inside 1.5σ at n=50 on the length-averaged mean. One cell escapes, `qa_hotpot` at 64k (0.62→0.54): two-hop prose chains plausibly stress the same rank-64 sketch resolution as the key–value haystacks, and we read that cell, not the mean, as the honest risk summary. Continuing published novels for 4096 tokens gives the decode something to amortize over: 1.072/1.148/1.270× dense at 64k/128k/256k of real prose, server-side over the decode; the curve is no artifact of synthetic input.

Table 10 is the evidence for the body's claim that the RULER gap sits only where the haystack is itself made of key–value pairs: the thirteen tasks sorted by the parameters RULER's own generator sets, against the measured gap. Multiplicity is visible as a non-cause: four planted needles, or four answers asked for, cost +0.000.

Table 10: The thirteen tasks by their generator's parameters, against the measured gap at n=50. `needle` as a haystack type means the filler *is* competing key–value pairs rather than prose. Six rows carry a negative delta; the two that lose by more than 2σ are exactly the two where that holds, and the other four are inside the band (Table 6).

task	haystack	key	value	Δ
`niah_single_1`	repeated filler	words	numbers	+0.004
`niah_single_2`	essays	words	numbers	+0.000
`niah_single_3`	essays	words	**UUIDs**	+0.000
`niah_multikey_1`	essays	words (×4)	numbers	+0.000
`niah_multikey_2`	**key–value pairs**	words	numbers	**-0.056**
`niah_multikey_3`	**key–value pairs**	**UUIDs**	**UUIDs**	**-0.128**
`niah_multiquery`	essays	words	numbers (4 asked)	+0.000
`niah_multivalue`	essays	words	numbers (×4)	+0.001
`vt`	variable-assignment chain			+0.000
`cwe`	word-frequency aggregate			-0.000
`fwe`	word-frequency aggregate			-0.011
`qa_squad`	*real* paragraphs + question			-0.056
`qa_hotpot`	*real* paragraphs, two hops			-0.040

The short-decode cost is the other half and is paid in throughput, not answers: a 14-step answer keeps the index on the $Z_{\max}$ clamp, which over-fetches and falls back to dense, so 0.337 of RULER's scans overflow where long decodes exhibit 0.012. The clamp in fact loses the top archived row *less* often than the fitted certificate (16.0% against 22.9%), so a longer-decoding RULER would have shown a marginally wider accuracy gap, not a narrower one.

The contribution of the fallback is therefore measured at both ends of that range rather than only where it fires. **The overflow fallback is deleted here in both arms. Ablation only: what is left is an incomplete algorithm, not a configuration this paper deploys.** On RULER it is deleted where the rate is highest, which is the arm a reviewer should distrust on its own: a third of the scans served exactly is also what an exact method wearing a sparse method's name would look like. Deleting it there costs the mean of 9 cells at 50 items each, 0.81 against 0.52, with 7 cells down and 2 up. A sign test does not resolve that at this width (p=0.180), and the honest reading is the single widest cell, `niah_multikey_3` at 64k, falling 0.82 to 0.18. Why the rate is that high is the short generation, and the fetch distribution shows the mechanism rather than asserting it. A 64k RULER answer averages 14 decode steps, too few to amortize a calibration, so the index stays provisional (identity basis, z=$Z_{\max}$), which is the certificate at its most conservative, asking for the most rows it ever asks for. Under it the median step already fetches 784 rows and the ninetieth percentile stands at 4096, the buffer's whole capacity of 4096: a step that reaches capacity *is* an overflow, so the rate is a property of that distribution's upper tail and not of a policy failing to find rows. This is a warm-up in the sense of Table 8 (a cost paid before the decision is informed) but not a cold start a benchmark could be run past: a 14-step answer ends before the stage does, so here the stage is the whole of the decode. Calibration collapses the same distribution: on the 128k stream the median and the ninetieth percentile both fetch 0, and the rate follows it down. The same deletion on a 128k stream is the control that separates the two readings. There the calibrated index asks for the exact path on 0.012 of scans against 0.337 (not nothing, but a thirtieth as often) and deleting the fallback moves the decode by 0.3%, from 5.444 to 5.426 ms per token. So what the deletion costs on RULER is what the provisional index costs before it calibrates, and what it costs a long deployment is bounded by a rate that small. Neither arm is a configuration: the release branch has no switch for the fallback, and an overflow is the certificate reporting that a step cannot be served sparsely at the recall target, not a row the policy failed to find. The full argument, the two-protocol distinction for LongBench v2, and the warm-up comparison against observed-attention eviction are in the archive.

Why the 128k series runs at a smaller memory fraction. At the default static fraction the VestigeKV arm exhausts this GPU above sixteen concurrent 128k requests and the dense arm does not, twice

and identically: 31 live, 0.82 of the token pool, and a failed allocation of 20 MiB against 15 MiB free. The pool is not what runs out (both arms size it to the same 5020421 tokens); the allocator is already running with expandable segments, so this is headroom rather than fragmentation. After startup the dense arm has 9.02 GB free and VestigeKV 8.22, the 0.80 GB its own tables cost, and the per-step working set at that width consumes the rest. The ceiling is one deployment knob's, not the method's: at 0.87 the same shape completes and serves all 32 requests at 31 live. The 128k series in the figure is therefore the whole sweep re-measured at 0.87 on both arms, so that each point compares the two arms under the same knob; the 64k series keeps the default, where neither arm is near the limit. Decode context parallelism, which would shard the sequence and the memory with it, is not supported by this backend and is not rejected at startup either. A reproducer should not enable it expecting this configuration to hold.

What the synthetic filler puts in the residual. A pre-review reader asked whether pairing the branch's dimensions into complex numbers and low-passing the complex spectrum would separate rows better, by analogy with RoPE. It would not, twice: the branch is un-roped, so there is no rotation to exploit, and a symmetric complex low-pass *is* the real one it would replace, $\mathrm{lp}(a{+}ib) = \mathrm{lp}(a){+}i\,\mathrm{lp}(b)$, a form also indifferent to which dimension is paired with which, which no transform using the pairing could be. The question underneath is empirical, though, and its answer is unflattering to the benchmark rather than to the transform. Taking the sidecar's spectrum along the token axis, RULER's haystack carries a standing component at bin 71–72 of a 4096-row block, a period near sixty tokens, at 724 times the median bin above the cutoff at layer 19, in every one of 5 requests. The component is present in every layer; its prominence is not, ranging over an order of magnitude across them, and it is the *bin*, not the peakiness, that separates the two corpora without overlap. Notching it leaves 33% of the rows tier 1 keeps at a 3% budget. Real prose has its largest above-cutoff component at bin 16, the first bin past the cutoff, where any smooth signal leaves a shoulder: 91 times the median over 5 requests, and 75% of the kept set survives. Part of what tier 1 selects on RULER is therefore selected by the filler's periodicity, a third way the benchmark is adverse, and not something the method carries into a deployment.

LongBench, and why it takes two protocols. Scoring LongBench v2's four choices from one token reads logits produced by `forward_extend`, which is the unmodified base kernel over the full pool; the compression changes only `forward_decode`. Both arms are therefore equal by construction, and they are: 0.333 against 0.333 on all 300 questions. We keep that as an exactness check on real documents, where an inequality would mean the premise is wrong, and we do not report it as a retrieval result. The result is the second protocol, which answers by generating and so puts real decode steps on the compressed path (Table 11). The continuation runs are unrelated to either: their text is full-length published novels, and the benchmark plays no part beyond having been a convenient way to obtain them (provenance in the archive).

Table 11: LongBench, both arms, Instruct checkpoint, one sitting per pair. **Top:** v2 answered by generating, so 300 questions each run a 120k prefill and up to 1024 decode steps on the compressed path. Accuracy is four-way, chance 0.25. The arms differ on 17+13 questions of 300, gaining 17 and losing 13, which is the paired view two close accuracies can hide; wall clock is 47.8 against 47.9 minutes (+0.2%), and a repeat of the compressed arm moves 12.59%, so a difference below that is not resolved here. The official length buckets short and medium are both present; the 120k subset contains no long-bucket item, so that cell is not reported. **Bottom:** v1 summarization, ROUGE-L F1 $\times 100$ under LongBench's own `eval.py` scoring, n=200 per subset and none truncated, generating up to 512 tokens per document. Both are generative long-document tasks rather than needles, and neither is a frozen gate: they were added after the pre-registration and are reported, not tested against a bar.

LongBench v2 (generative, four-way)	dense	VestigeKV
overall (300 questions)	0.300	0.313
single-document QA	0.314	0.355
multi-document QA	0.314	0.279
long-dialogue history	0.308	0.333
long in-context learning	0.229	0.286
code repository	0.267	0.200
structured data	0.250	0.250
easy / hard	0.296 / 0.302	0.287 / 0.328
short / medium	0.331 / 0.254	0.343 / 0.270
LongBench v1 (summarization, ROUGE-L)	**dense**	**VestigeKV**
`gov_report`	29.41	28.18
`qmsum`	23.99	23.47
`multi_news`	24.78	24.70
mean	26.06	25.45

Every scope boundary of this study, with its measured status:

limitation	status
one measured model	all measurements are on Kimi Linear 48B, the only released NoPE-MLA line. Rate tables are measured on the *Base* checkpoint; the serving quality, needle and throughput gates replicate on the post-trained *Instruct* checkpoint (Table 5). For Kimi K3's Gated-MLA the machinery extends by the gate-commutation argument; the branch's salience *content* there is an untested empirical claim. We predict it carries the same content; the test is one forward pass: branch-only scoring must match whole-latent scoring on needle retrieval, as it does here (Section 3.1)
one NoPE-MLA family	selector *ordering* replicates on a NoPE GQA hybrid (Granite-4.0-H), eviction *depth* does not: depth claims are MLA-conditional
mechanism of the depth	open; three pre-registered candidate explanations were refuted by our own runs, recorded in the pre-registration archive
row reachability	existence is constructive (the partition deletes nothing); per-step consultation is probabilistic (recall 0.9–1.0, 1.00 end to end; the fired set itself is deterministic by construction given the query). Pure deletion loses rows irrecoverably (+15 nats of NLL on the needle answer)

limitation	status
tasks, length	needle probes to 256k (serving: 128k, English and Chinese corpora), held-out CE at 8k, 64-shot GSM8K-Platinum (n=1209) and MAUVE (4k and 64k prefill) on the serving path; serving latency measured to 512k; RULER 4k–64k on the serving path (Table 6), longer RULER lengths in the archive as they complete. Multi-token real-document evidence is LongBench v1 summarization and v2 answered generatively (Table 11), both on the Instruct checkpoint only, plus the novel-continuation decode and RULER's prose-QA cells; L-Eval and InfiniteBench are not run

The gap's two failures, only one of them selection. `niah_multikey_3` and `niah_single_3` are the same 36-character UUID copy at the same lengths; only the haystack differs, prose against key–value pairs. Run at n=50 over 32k and 64k, 100 items each (`mk3-copyfidelity-n50` in the code release):

	`multikey_3`	`single_3`
dense	100/100	100/100
VestigeKV	88/100	100/100

The copy never fails without compression, and under compression it never fails in prose either. On the *Instruct* checkpoint the 6+6 failures in a key–value haystack split evenly.

The split is a property of that checkpoint, not of the method. The same classification over the Base pair, restricted to the same two lengths so the populations match, reads 19 correct of 100, with 70 returning a different needle and only 11 the right one mis-copied. Base is therefore not a worse instance of the same failure: it is a different one, selection almost throughout, where Instruct is half selection and half copy. The dense arm loses nothing on either checkpoint (100 and 100 of 100), so the difference is the compression's and not the task's. We report this because the body's reading of the gap was obtained on Instruct and does not transfer, and because the rate tables of this paper are measured on Base.

The Instruct split, in detail. 6 return a different needle: the selection failure of Section 6.2, a rank-r sketch unable to separate rows drawn from the bulk it was fitted to. The other 6 return the *right* needle with one or two characters wrong, and they are not a blend with a competitor. Each sits at edit distance 1–2 from its target and 21–24 from the nearest other UUID in its own haystack. The right row was reached and a token or two of a ∼20-step verbatim span came out wrong. The cost of a haystack of look-alike rows is therefore not only the ranking but the fidelity of a long copy, and the paper's account covers the first half only. We do not have a mechanism for the second: separating a step that failed to fetch its row from one that fetched it and decoded under the omitted mass needs per-step fetched-row telemetry, which the deployed backend does not record.

H COST ACCOUNTING: THE TWO DERIVATIONS

Both derivations price the same per-step read that Figure 3 shows: the branch slice, not the row.

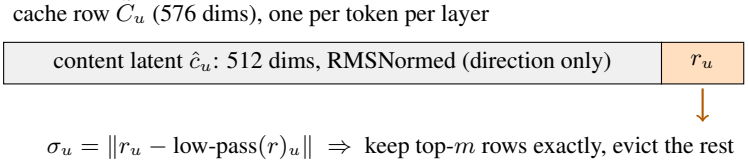


Figure 3: For eviction, VestigeKV reads only the branch (orange, 11% of the row); the decode-time scan additionally reads the rank-r sketch and residual (Section 4.2). Kept rows are exact; nothing else is stored.

H.1 THE READ BUDGET

Decode-time attention is bandwidth-bound: KV-path time tracks bytes read per step. The three-term budget follows from the construction above:

$$\text{reads}(\rho, r) = \underbrace{\rho \cdot 576}_{\text{attended}} + \underbrace{(1-\rho)(64+r+1)}_{\text{index scan}} + \underbrace{f \cdot 576}_{\text{admitted}} \text{ dims/token}, \tag{7}$$

where the $+1$ counts the per-row residual scalar η_u. In *bytes* (what bandwidth prices): rows are 1152 B, the 16-bit index scan reads its $(64{+}r{+}1)$ dims at 2 B (258 B/row payload, $\approx$260 with alignment; the first 128 B is the row's own 64-dim branch, read in place from the pool rather than duplicated), and the budget lands at $\sim$294 vs. 1152 B/token, $3.9\times$ (a full-precision index blunts this to $2.1\times$). At $\rho{=}1/32$, $r{=}64$ the admitted fraction f measures 0.4–0.8% on document decode (0.5–2.6% on needle contexts); the trigger term is under 1%. Eq. 7 is the asymptotic form: the unclosed tail adds up to one 4096-token block of full 576-dim rows, a per-token contribution of up to $4096/T$ that vanishes as T grows and that, with the fixed adder R, accounts for the measured crossover sitting right of this account's slope-only value. The ratio is asymptotic in context and batch (attention never amortizes, shared weights do), so the reported ratios are a conservative floor; the pipeline is six fused kernels inside the decode CUDA graph, plus the forked stage 1 once per MLA layer (Section 5).

H.2 THE CACHE MEMORY–TIME PRODUCT

A request occupies its cache for its whole lifetime, so the resource a deployment actually spends is $C = M_{kv} \cdot T$ (GB·s), and by Little's law it is the throughput denominator: memory-capped serving admits $C_{mem}/(M_{kv}T)$ requests/s. Writing C_{vk} and C_{dense} for the two arms' products, their ratio is

$$C_{\text{vk}}/C_{\text{dense}} \;=\; \big(1+\alpha\big) \,/\, A(S), \tag{8}$$

with $A(S)$ the end-to-end speedup of Figure 2. Below the threshold $A{=}1$, $\alpha \approx 0$: parity. Above it the product crosses 1 where $A(S) = 1.14$ ($\approx$128k on this testbed), reaches $1.14/1.45 \approx 0.79$ at 512k, and tends to $(1{+}\alpha)(\rho{+}k_s/k) \approx 0.30$ asymptotically. The memory–time crossover, not the speed crossover, is where the method starts paying for itself; both sit right of their attention-only values only because this testbed's step is MoE-dominated (Section 5). A production deployment can also bypass the scan outright when the context sits below the crossover or the expected decode is shorter than the calibration window; this paper disables that threshold everywhere (Section 6).

H.3 WHAT THE END-TO-END RATIOS DILUTE

Every speedup this paper reports is measured on a whole decode step, and VestigeKV changes one part of that step. Kimi Linear is a hybrid: 7 of its 27 layers are full-attention MLA and the other 20 are linear attention, which keeps no KV cache and so has nothing for an eviction policy to do. Those 20 layers, the MoE, the projections and sampling are all inside the denominator of $1.26\times$ and $1.45\times$, and none of them is touched.

The quantity that is not diluted is already measured. A curve's slope is the marginal cost of one more cached token, and attention is the only term whose cost grows with the cache; everything the method leaves alone is a constant per step, hence sits in the intercept. So $6.3/1.9$, or $3.3\times$, is the speedup on the part VestigeKV changes, read off the same serving runs as the end-to-end figures rather than from a separate harness. It also sharpens the account at the end of the previous subsection: the crossovers sit right of their attention-only values not only because the step is MoE-dominated, but because a majority of the layers have no KV cache at all.

We deliberately do not report a standalone kernel benchmark against dense attention, although the shapes would permit one. The decode fork cannot run without its tier state (a kept table, an archive, a fetch buffer and a calibrated index), and in a synthetic harness that state is chosen by the person writing the harness. The kept fraction ρ and the fire rate f of Equation 7 are exactly the two numbers such a benchmark would have to assume, and they are exactly the two that decide its result: it would report its own assumptions back. The slope is free of that, because ρ and f there are whatever the workload produced. A reader who wants the analytic counterpart has Equation 7, which prices the same comparison in bytes and needs no device at all.

The split also bounds how the result should be read elsewhere. On a homogeneous MLA model every layer is one VestigeKV changes, so the same per-layer gain reaches a larger share of the step, and the end-to-end ratio moves toward 3.3× rather than away from it. The numbers here are the pessimistic end of that range, not a representative point in it.

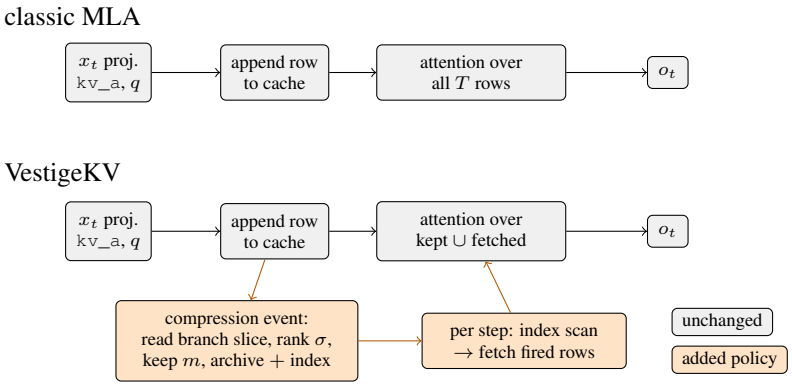


Figure 4: The delta from classic MLA, per layer. Every model operation (gray) is byte-identical: projections, kernels, arithmetic, weights. The orange path is cache-membership policy, and removing the whole of it (the kill-switch) recovers classic MLA exactly, byte-for-byte.

Lifecycle of a request. Every stage below is stated in Sections 4 and 5; collected here so a reimplementation need not read the code for the order.

1. **Block close** (prefill, and again every 4096 generated tokens). The closing block's branch vectors go through the sequence-axis rFFT at κ=16 bins; σ_u is the norm of the high-pass residual (Eq. 3). A block is transformed once and its σ never changes.
2. **Rebalance.** One global top-m over all closed rows, $m = \rho T$, plus 4 sinks and the unclosed tail, is the attended tier. Every other closed row moves to the archive. Nothing is deleted, so a rebalance may move a row back.
3. **Index write.** Each archived row gets $(r_u,\ V_r^\top \hat{c}_u,\ \eta_u)$. Until calibration completes, V_r is the provisional basis and the tier serves at the clamp z=$Z_{\max}$.
4. **Calibration** (first decode steps, side stream). Decode (query, head) pairs whose best archived row exists enter the pool; at $\lceil \tau_s/(1-\tau_s) \rceil$ pairs, z becomes the conformal quantile of the required inflations and V_r is refitted. The step never waits on this.
5. **Decode step**, one CUDA-graph launch: prologue online-softmax over kept ∪ sinks yields each head's s_h^* and a partial (o, ℓ); the entropy gate may skip the scan where that softmax is already peaked; the scan evaluates Eq. 4 over archived rows and fires those beating s_h^*; compaction turns the fired mask into a row list; a per-layer write records the row counts stage 2 needs; the forked stage 1 attends tail ∪ fired, reading those rows where they already live; the two partitions merge by log-sum-exp.
6. **Overflow.** If the fired set exceeds the 4096-row buffer, a runtime scalar switches that lane's row source to its own page table: the step is then dense and exact, in the same graph, with no recapture.