RIX: REUSABLE ATTENTION INDEXING

Anonymous authors
Paper under double-blind review

ABSTRACT

Indexer-based sparse attention replaces full $O(L^2)$ attention with a learned index plus a top-k pass, but as contexts grow longer the indexer's own score computation becomes the bottleneck. Existing accelerations cluster keys into blocks or share indices across layers. Reusing indices across query positions is orthogonal to both, yet unsafe to exploit directly. We observe that adjacent queries' index sets coincide strongly, but rare query pairs drift sharply, so query-side reuse requires a safety assurance. Our proposed method, REUSABLE INDEXING (RIX), provides one with an $O(Hd)$ closed-form reusability metric, derived from attention score estimation under a uniform key model, and applies reusable indexing coherently by indexer design, calibration training and inference-time optimal scheduling. In our evaluation, Rix consistently outperforms fixed reuse scheduling at matched reuse rates.

1 INTRODUCTION

State-of-the-art agentic intelligence makes fundamental demands on long-context inference. Long-horizon tasks, e.g. repository-level software development being the canonical example (Jimenez et al., 2024; Yang et al., 2024), require models to sustain coherent reasoning over hundreds of thousands of tokens of accumulated context, spanning source files, tool outputs, test logs, and the agent's own action history. As these workloads push context windows from 128k toward $\sim$1M tokens, inference efficiency becomes the primary constraint on deploying such systems.

Full attention scales quadratically with context length (Vaswani et al., 2017), and sparse attention has emerged as a practical way to process contexts up to 128k tokens and beyond. Indexer-based designs (DeepSeek-AI et al., 2025) attach a lightweight learned indexer that scores every key for each query and restrict the main attention to the top-k keys, reducing the attended-to tokens at a small fraction of the cost of full attention.

As contexts grow even longer, however, indexing itself becomes the bottleneck. At 200k tokens, the indexer accounts for roughly 81% of end-to-end prefill latency and 41% of decode latency (Bai et al., 2026). Two scaling remedies have appeared. Along the key dimension, tokens are grouped into blocks and selected block-wise rather than individually (Lu et al., 2025; Lai et al., 2026), amortizing index evaluation over grouped keys; along the layer dimension, indices computed at one layer are reused by multiple subsequent layers (Bai et al., 2026; Sun et al., 2026; Gao et al., 2026).

However, the third axis, reusing indices across query positions, has to our knowledge received less interest. Our empirical study finds this axis to be simultaneously promising and hazardous. On the one hand, reusability is abundant. Most queries need very few keys to reach a high coverage threshold, and adjacent queries' high-mass key sets overlap heavily, so most queries could simply inherit a neighbor's index. On the other hand, the tail is sharp. A small fraction of corner queries demand a dramatically larger key budget than the moderate majority. For the worst 1% of corner queries there is no budget-efficient shared index that retains 70% of their attention mass beyond 64k context length. The dilemma sharpens because hardware friendliness demands the same top-k budget for every query to avoid ragged indices. Thus, adaptively scheduling index computation becomes a practical solution, serving fresh indices at the few corner queries and one shared index across the moderate majority. Previous work focuses on improvements within fixed grouping (Liu et al., 2026) or partitioning (Song et al., 2026). How to handle outliers by adaptive reuse scheduling remains an open question.

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.

The dilemma compels an adaptive reuse strategy that computes fresh, high-quality indices for outlier queries while saving FLOPS by reusing indices across ordinary ones. We propose REUSABLE INDEXING (RIX), an adaptive query-level reusable indexing method pivoted by a gating metric, realizing verifiable reuse decisions. Our contributions are threefold.

(1) A pairwise, key-agnostic reusability metric. Compute efficiency forces the metric to be key-agnostic, and we derive a closed form from attention score estimation under a uniform spherical key model, yielding a head-weighted sum of per-head von Mises–Fisher KL terms plus a head-mixture correction that keeps the predictor faithful at long anchor ages.

(2) A systematic indexer design aligned with the metric. The closed form mandates exp activation and an unparameterized K-norm; training distills the indexer with a per-query KL objective plus a Huber-style calibration loss for metric faithfulness across reuse rates, at negligible cost to the indexer fit in our measurements.

(3) An inference-time optimal reuse scheduler. Optimal Reuse Scheduler (ORS) places minimal fresh anchors, ensuring that every non-anchor position provably satisfies the metric threshold, with parallel and serialized realizations for prefill and decode.

We evaluate Rix on RULER (Hsieh et al., 2024) and LongBench (Bai et al., 2024) with the Qwen3 family (Yang et al., 2025) at 8 / 14 / 32B and KimiLinear-48B (Team et al., 2025). At matched reuse, the learned schedule beats fixed-period grouping in 24 of 25 controlled evaluations on RULER. Our method composes with PIVOT-style pooled anchors (Liu et al., 2026) and further improves over fixed pooling. We observe that at 75% reuse, our ORS kernel accelerates indexer forward by 2.6–3.8$\times$ from 128k to 1M context.

2 MOTIVATION: ABUNDANT REUSABILITY, SHARP CORNERS

Rix is motivated by imbalanced index reusability across queries. We measure the minimal number of keys a single query requires and the maximal number of consecutive queries that can share a single key set, to understand index reusability. To be specific, we run the per-layer attention-mass-based procedure of Algorithm 1 on Qwen3-8B and samples from our training corpus, which outputs Keys and Queries under a given Cov and Siz, the mass coverage and the key set size.

Algorithm 1 Greedy coverage-set construction for Keys and Queries.

Require: coverage target Cov, index budget Siz

attn $\leftarrow 1/H \sum_h \text{softmax}(Q_h K_h^\top / \sqrt{d})$ ▷ head-averaged attention mass
$\text{cov}(S, j) \leftarrow \sum_{k \in S} \text{attn}_{\text{offset}=j}[\ldots, k]$ ▷ mass of key set S on the query at offset j
$S_0 \leftarrow \text{TopP}(\text{attn}, p = \text{Cov}, dim = -1)$ ▷ first query's minimal coverage set
for $i = 1, 2, \ldots$ **do** ▷ extend the shared index greedily
  $S_i \leftarrow S_{i-1} \cup \text{TopP}\big(\text{attn}_{\text{offset}=i}[S_{i-1}^C],\ p = \text{Cov} - \text{cov}(S_{i-1}, i),\ dim = -1\big)$
  if $\|S_i\| > \text{Siz}$ **then break**
  end if ▷ S^C is the complement of S
end for
return $\text{Keys}(\text{Cov}), \text{Queries}(\text{Cov}, \text{Siz}) \leftarrow \|S_0\|, \max\{\, i : \|S_i\| \le \text{Siz} \,\} + 1$ ▷ 0 if no i qualifies

The results show that reuse is commonly cheap, but occasionally catastrophic. Table 1 reports the worst 50th, 90th and 99th percentiles of each quantity, with reuse lengths censored at 16. p90 of Keys(70%) is no larger than 479 under 128k context length and p90 of Queries$(70\%, 2048) \ge 12$ holds within 64k context window, showing that most queries are cheap to serve and easy to reuse. However, p99 of Keys(70%) is roughly $4\times$ and $40\times$ larger than its p90 and p50, showing that a sharp tail of queries is hard to serve. p99 of Queries ≤ 1 holds at 128k context length, meaning that mass-preserving reuse is infeasible on no less than 1% of queries.

A straightforward idea is to schedule an adaptive key budget for different queries to handle this imbalance. But this causes ragged indices, which are hardware-unfriendly. In the Rix design, we instead conduct adaptive reuse scheduling under a universal index top-k. With a key-agnostic reusability metric, Rix effectively identifies whether one query's index set can serve another, and saves compute on easy-to-serve queries by index reuse while still serving hard queries with fresh indices.

Table 1: Worst 50th / 90th / 99th percentiles of Keys and Queries. Measured on Qwen3-8B, sequences sampled from our training corpus, all layers pooled. †: with YaRN $2\times$. ‡: with YaRN $4\times$.

context	Keys(70%)	Keys(90%)	Queries(70%, 1024)	Queries(70%, 2048)
16k	19 / 159 / 619	362 / 1550 / 3676	16 / 16 / 10	16 / 16 / 16
32k	26 / 235 / 859	546 / 2618 / 5748	16 / 16 / 2	16 / 16 / 16
64k†	30 / 353 / 1445	879 / 4202 / 9070	16 / 12 / 0	16 / 16 / 4
128k‡	51 / 479 / 2275	1687 / 6254 / 14749	16 / 4 / 0	16 / 16 / 0

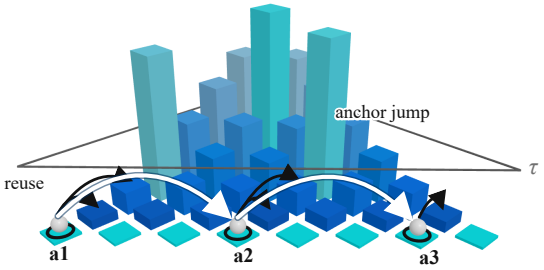


Figure 1: The jump game on the divergence grid. Bars show the calibrated reusability metric $\widehat{\mathrm{KL}}_{\mathrm{corr}}$ between query pairs of a causal 8×8 position grid. Adjacent queries diverge little and share one index, while rare corner pairs spike through the admission threshold τ and demand a fresh index. The optimal reuse scheduler walks the diagonal, reusing each anchor's index along its calm rows (dark arcs) and jumping to a fresh anchor exactly where the gate fires (white arcs), instead of grouping queries uniformly.

3 Method

Figure 1 outlines the method on a causal 8×8 position grid, a jump game in the divergence valley. We establish a closed-form divergence metric for the reuse error of a query pair (§3.1), design and train the indexer so the metric stays faithful without compromising indexing quality (§3.2), and realize optimal anchor scheduling as a jump game firing minimal anchors where the metric demands (§3.3).

3.1 The reusability metric

At inference time we ask, for a multi-head query state j, whether the index computed for an earlier query state i can serve j in place of a fresh computation. The metric is evaluated once per candidate pair per layer, so it cannot read cached keys or materialize score rows without costing as much as the index computation it gates. Thus, it must be key-agnostic, a function of the two query states alone. We make it closed form by assuming the indexer keys are uniformly distributed on the sphere: each head's served distribution is then a von Mises–Fisher (vMF) law whose KL divergence decomposes into tabulated one-dimensional kernel evaluations, and the full metric costs $O(Hd)$ flops plus $O(H)$ kernel lookups per pair per layer, negligible next to the $O(L\,H\,d)$ index computation.

Score functions and the closed form. We take nonnegative mixtures of exponentiated terms over H heads,

$$s(k;\,w,q)\;=\;\sum_{h=1}^{H} w_h\,\exp\!\big(q_h^{\top}k/\sqrt{d}\big),\qquad w_h\geq 0, \tag{1}$$

where q_h is the h-th indexer query head and w_h its weight. The distribution served by a query state is the normalized score map $s/\int s$, and the reuse error of serving query j from the index of anchor i is the divergence between the two served distributions. The indexer normalizes every key to RMS norm $\sqrt{d}$ and folds the $1/\sqrt{d}$ attention scale into the key, so $\tilde{k}=k/\sqrt{d}$ is unit-norm by construction,

and we model $\tilde{k} \sim \mathrm{Unif}(\mathbb{S}^{d-1})$. Each head's normalized score map is then a vMF law $p_h(x \mid q_h) \propto \exp(q_h^\top x)$ with concentration $\kappa = \|q_h\|$ carried on the query side, whose normalizer is the one-dimensional kernel expectation $f(\kappa; d) = \mathbb{E}_{x \sim \mathrm{Unif}(\mathbb{S}^{d-1})}[\exp(\kappa\, e^\top x)]$, a closed form in modified Bessel functions. Writing $\log Z(\kappa) = \log f(\kappa; d)$ and $A(\kappa) = \partial_\kappa \log Z(\kappa)$ for the vMF mean length, the divergence of the served row of query j against the anchor row of query i decomposes per head into the closed form

$$\widehat{\mathrm{KL}}_h(i,j) \;=\; A(\kappa_t)\Big(\kappa_t - \frac{q_{j,h}^\top q_{i,h}}{\kappa_t}\Big) + \log Z(\kappa_a) - \log Z(\kappa_t), \tag{2}$$

with $\kappa_t = \|q_{j,h}\|$ and $\kappa_a = \|q_{i,h}\|$ the served and anchor concentrations. For deployment, both $\log Z$ and A are evaluated from a single tabulated analytic surrogate of f (Figure 4, Appendix D).

The mixture correction. Aggregating equation 2 by head weights omits the mixture coupling. Drift that reallocates mass across heads moves the served mixture $p(x) \propto \sum_h w_h \exp(q_h^\top x)$ at first order while every per-head divergence stays small. The exact mixture divergence has no closed form, but the joint divergence over the latent pair of head and key does, and it upper-bounds the mixture divergence by the chain rule. With the head shares $\pi_h(q) = w_h\, Z(\|q_h\|) / \sum_{h'} w_{h'}\, Z(\|q_{h'}\|)$, the mass each head contributes to the total served energy, the joint divergence decomposes into the divergence of the shares plus the share-weighted per-head divergences, giving the reusability metric

$$\widehat{\mathrm{KL}}_{\mathrm{aug}}(i,j) \;=\; \mathrm{KL}\big(\pi(q_j) \,\|\, \pi(q_i)\big) \;+\; \sum_h \pi_h(q_j)\, \widehat{\mathrm{KL}}_h(i,j). \tag{3}$$

The first term accounts for head-weight and per-head norm drift, and it is exactly the term the head-weighted sum cannot see. The correction is decisive at large anchor ages, where the plain head-weighted sum decorrelates from the exact mixture divergence while equation 3 keeps a flat per-age admission AUC.

Calibration and deployment. Real key clouds cluster where the uniform-sphere model assumes the worst case, so equation 3 overestimates the realized reuse divergence by a per-layer scale. The scale is absorbed by a learned calibration trained jointly with the indexer, and the ranking fidelity of the isotropic surrogate under clustered keys is an empirical property, which we later validate directly against oracle attention rows,

$$\widehat{\mathrm{KL}}_{\mathrm{corr}}(i,j) \;=\; e^{-\tau_\pi}\, \mathrm{KL}\big(\pi(q_j) \,\|\, \pi(q_i)\big) \;+\; \sum_h \pi_h(q_j)\, \widehat{\mathrm{KL}}_h(i,j)\, e^{-\tau_h}, \tag{4}$$

with a per-head correction τ_h on the per-head terms and a scalar correction τ_π on the share term. Because $\pi_h(q_j)$ varies pair by pair, a per-head correction reshapes the admission *ranking* rather than merely rescaling each layer. Calibrated during training, the metric operates in true divergence units, and every reuse decision in the system applies one rule with one universal threshold: admit the pair iff $\beta \ln \widehat{\mathrm{KL}}_{\mathrm{corr}}(i,j) + c < \ln \tau$. A common offset of the scales τ_h, τ_π rescales $\widehat{\mathrm{KL}}_{\mathrm{corr}}$ uniformly and is absorbed by the intercept c, so the affine gate is the canonical deployment form.

3.2 Indexer design and training

Each attention layer carries a learned DSA-style (DeepSeek-AI et al., 2025) multi-query indexer: H indexer query heads over a single shared key head, having the same head dimension d as the backbone models. The indexer state $(q_{1:H}, w_{1:H})$ defines the score function of the reusability metric, and the main attention is restricted to the top-k indices under s, optionally unioned with a trailing local window. Our metric mandates the choice of exp activation and unparameterized key normalization, so that each head's served distribution is a vMF law whose KL divergence is closed form under the spherical key model. The query-side normalization keeps a learned per-dimension scale that the metric reads as the concentration $\kappa = \|q_h\|$, and the head weights are softplus-rectified, which keeps the mixture nonnegative. The indexer design is formulated as follows, with Norm_θ the learned per-dimension query scale and Norm the unparameterized RMS key norm to norm $\sqrt{d}$.

$$
\begin{aligned}
Q, K &= \mathrm{PE}\big(\mathrm{Norm}_\theta(XW^Q),\ \mathrm{Norm}(XW^K)\big), \mathrm{PE} \in \{\mathrm{RoPE}, \mathrm{NoPE}\} \\
w &= \mathrm{softplus}(XW^w)/\sqrt{H}, \\
s_j(k) &= \textstyle\sum_{h=1}^{H} w_{j,h}\ \exp\big(Q_{j,h}^\top K_k/\sqrt{d}\,\big), \\
\mathcal{I}_j &= \text{top-k}(s_j)\ \cup\ \{\, j{-}W, \ldots, j \,\},
\end{aligned}
\tag{5}
$$

The indexer is trained with the backbone frozen.

We adopt the indexer training objective of DSA (DeepSeek-AI et al., 2025), a per-query forward KL from the sums of main attention scores to the normalized index distribution,

$$
\mathcal{L}_{\text{mono}} \ =\ \mathrm{KL}\big(s_{\text{ref}} \,\big\|\, \hat{s}\big),\ s_{\text{ref}} = \frac{1}{H^*}\sum_h \mathrm{softmax}\big(q_h^* K^{*\top}/\sqrt{d}\big) \tag{6}
$$

with $\hat{s}$ the normalized indexer score row of the same query. To support variable reuse rates, we measure the realized error as the divergence of the served row against the deployed mixture on the deployed support S,

$$
\mathrm{KL}_{\text{full}} \ =\ -\log p_j^{\text{ref}}(S)\ +\ \mathrm{KL}\big(p_j^{\text{ref}}|_S \,\|\, \tilde{p}|_S\big), \tag{7}
$$

where p_j^{ref} is the backbone's realized attention row of the served query j and $\tilde{p}$ is the deployed mixture renormalized on S, the served row itself on the local window and the anchor row on the remainder of S. We train this target by an additional Huber-style calibration term,

$$
\mathcal{L}_{\text{calib}} \ =\ \mathrm{Huber}_1\Big(\beta \ln \widehat{\mathrm{KL}}_{\text{corr}}\ +\ c\ -\ \ln\ \mathrm{sg}[\mathrm{KL}_{\text{full}}]\Big), \tag{8}
$$

with per-layer calibration parameters τ_h, τ_π, the log-space slope β, and the intercept c. The total training objective is $\mathcal{L}_{\text{mono}} + \lambda_c \mathcal{L}_{\text{calib}}$ with $\lambda_c = 1$. Details can be found in Appendix D.2.

Initialization and schedule. A head-copy initialization from the backbone's attention projections makes a limited token budget sufficient, and a two-stage schedule, dense warmup at up to 32k followed by sparse context expansion to 128k, scales the recipe across the model family and context lengths.

3.3 Optimal reuse scheduling

We organize reuse in the block view by segmenting sequences into consecutive blocks. To respect causality in language modeling, the first position of a block is chosen as its anchor, whose top-k indices serve the block's reuse positions, and each position additionally keeps its own intra-block causal tail through a trailing local window. If the indexer selects k keys per position and a block holds r reuse positions, the whole block shares the $k - r - 1$ global keys selected at the anchor outside the block, and each position adds its own causal keys inside the block, at most $r + 1$ of them. Admission is decided by the calibrated gate through the affine predictor of equation 8: with gate matrix $G_{i,\Delta} = [\beta \ln \widehat{\mathrm{KL}}_{\text{corr}}(i, i{+}\Delta) + c < \ln \tau]$, a block anchored at i covering r reuse positions is legal iff $G_{i,1} \wedge \cdots \wedge G_{i,r} = 1$. Blocks are additionally capped by a constant $\Delta_{\max} \le W$ for compute efficiency. The legal next anchor of i lies in $(i, N_i]$ with $N_i = i + \min\big(\{\Delta \in [1, \Delta_{\max}] : G_{i,\Delta} = 0\} \cup \{\Delta_{\max}{+}1\}\big)$, formulating the jump game below.

The anchor jump game. Choosing anchors to minimize fresh index computations is the jump game. Letting $f(a)$ be the farthest right end reachable with a anchors,

$$
f(a) = \max_{f(a-2) < i \le f(a-1)+1} N_i - 1, \tag{9}
$$

which unrolls into an $O(n)$ sequential greedy: the a-th anchor is the arg max of N_i over the reachable interval $(f(a{-}2),\ f(a{-}1){+}1]$, and the walk stops at the first anchor whose block reaches the

sequence end. We prove that the greedy is optimal and develop a parallel version for prefill (Appendix A). Recently, Liu et al. (2026) proved that mean-pooled query states in a block serve as higher-quality anchors by sacrificing causality. We further develop a pooled version of the jump game to compose ORS with their proposed method, PIVOT (Appendix A.6).

4 Experiments

4.1 Experimental setup

Backbones and indexer. Main results use the frozen Qwen3 family (Yang et al., 2025) at {8,14,32}B, a GQA-based series with a 40k native context window, and KimiLinear-48B-A3B-Instruct (Team et al., 2025), an MLA-based hybrid attention backbone with a 1M context window. The indexer is our MQA design, trained by per-query KL distillation plus the calibration term. Data and training details are provided in Appendix D. Two controls share the same settings for ablation: a ReLU-activated variant and a variant trained with the mean-grouping objective.

Evaluation We evaluate on RULER (Hsieh et al., 2024), which probes controlled long-context retrieval and reasoning, and LongBench (Bai et al., 2024), which covers diverse practical long-context tasks. The compared schedules combine an anchor construction with a placement rule. FirstAnchor reuses the index of a block's first row across the whole block; PIVOT-reuse (Liu et al., 2026) computes the block's shared index from block-mean query states. We do not report PIVOT-refine results because it is secondary indexing, which seamlessly applies to all controls and is orthogonal to our settings. On each anchor we evaluate fixed-period grouping and two ORS modes: $\text{ORS}_{\text{relative}}$, which bisects the per-sequence threshold to pin a target reuse rate for controlled comparisons, and $\text{ORS}_{\tau=t}$, the deployment mode under one fixed universal threshold. Together with the dense backbone (dense) and full indexing (idx), these give the reported variants; tables sort variants by realized reuse, reported in the first column. PIVOT-reuse anchors read the block's future rows, so at decode such grouping must speculate block membership and recompute on underfilled groups. All PIVOT-reuse variants are therefore evaluated as prefill controls, while every FirstAnchor variant is fully causal. Unless noted otherwise, evaluation runs on vLLM with a trailing window of $W = 8$, an anchor age cap of 8, and indexer top-$k = 2048$. We use greedy decoding to ensure deterministic results.

4.2 Main results

Acceleration Figure 2 tracks the per-layer indexer forward cost of the 32B model from 32k to 1M context with the production stack on a single H200. The speedup by 75% reuse over full indexing widens from $1.6\times$ at 32k to $3.8\times$ at 1M, approaching the reuse-implied bound, and ORS adds little over fixed-period grouping at matched reuse. The indexer forward is independent of the trailing window and weakly dependent on top-k. Attention-layer, end-to-end costs and more details are deferred to Appendix E.

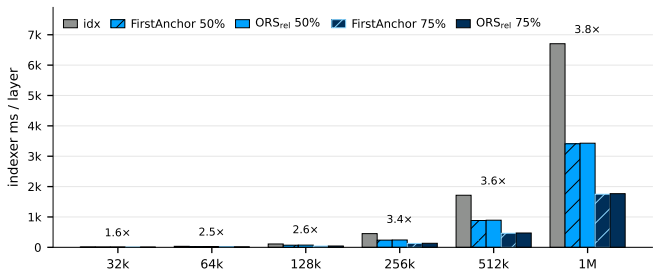


Figure 2: Indexer forward cost per layer of Qwen3-32B across context lengths (on single H200, bf16, top-k 1024, age cap 8, fixed reuse thresholds; ORS_{rel} in the legend denotes $\text{ORS}_{\text{relative}}$+FirstAnchor). Annotations give the full-indexing to 75%-reuse speedup at each length.

RULER Table 2 reports the comparison, with variants sorted by reuse rate. Full indexing tracks dense attention within 0.8–3.0 points at trained context lengths; under YaRN $4\times$ extrapolation the gap widens to 3.6–8.7 points, which Appendix D.6 attributes to backbone head disagreement that YaRN amplifies. At matched reuse, ORS placement beats fixed grouping in 24 of the 25 (anchor, reuse, model) combinations, the single reversal being the 87.5%-reuse pooled arm on Qwen3-14B (-1.4 points). The universal-threshold mode ($\text{ORS}_{\tau=t}$) matches full indexing within 0.4 points at $\tau = 0.2$ while reusing 9–42% of index computations, with the realized rate floating per backbone. Gains vary with anchor construction, architecture, and task hardness.

Table 2: RULER suite means, winners within each anchor group in bold. Column one lists realized reuse per model, in column order where it differs across models. Rows marked † run $W{=}16$. Context lengths marked ‡ use YaRN $4\times$. Pooled-anchor variants are prefill-only and use block-JSD admission.

		Qwen3-8B		Qwen3-14B	Qwen3-32B	KimiLinear
reuse	variant	32k	128k‡	128k‡	128k‡	128k
0%	dense	**95.97**	**70.84**	**82.88**	**83.28**	**93.45**
0%	idx	92.94	67.28	74.15	75.62	92.65
0%	idx†	92.86	67.24	73.83	75.46	93.07
38/28/33/42/9%	$\text{ORS}_{\tau=0.2}$+FirstAnchor	92.95	67.23	74.10	75.61	92.99
50%	FirstAnchor	89.83	60.06	66.35	68.68	91.56
50%	$\text{ORS}_{\text{relative}}$+FirstAnchor	**93.03**	**68.07**	**70.17**	**73.05**	**92.95**
50%	PIVOT-reuse†	91.52	64.92	70.52	72.24	92.16
50%	$\text{ORS}_{\text{relative}}$+PIVOT-reuse†	**92.98**	**67.07**	**71.41**	**74.38**	**92.96**
73/63/69/72/30%	$\text{ORS}_{\tau=0.4}$+FirstAnchor	88.22	68.43	66.08	68.29	92.63
75%	FirstAnchor	81.04	55.26	60.51	63.19	88.68
75%	$\text{ORS}_{\text{relative}}$+FirstAnchor	**85.97**	**64.00**	**62.25**	**65.42**	**92.18**
75%	PIVOT-reuse†	86.87	60.21	62.30	65.43	91.43
75%	$\text{ORS}_{\text{relative}}$+PIVOT-reuse†	**91.40**	**65.50**	**64.41**	**68.22**	**92.72**
83/76/82/81/58%	$\text{ORS}_{\tau=0.2}$+PIVOT-reuse†	88.24	66.21	59.44	67.56	92.80
84/80/82/82/53%	$\text{ORS}_{\tau=0.6}$+FirstAnchor	83.45	64.84	57.33	60.58	92.59
87.5%	PIVOT-reuse†	81.23	56.62	**55.40**	60.51	90.26
87.5%	$\text{ORS}_{\text{relative}}$+PIVOT-reuse†	**85.35**	**61.10**	54.02	**64.39**	**91.41**
93/91/92/91/83%	$\text{ORS}_{\tau=0.4}$+PIVOT-reuse†	80.63	60.76	48.30	57.28	92.29
94/93/93/93/87%	$\text{ORS}_{\tau=0.6}$+PIVOT-reuse†	77.43	57.40	48.01	55.96	90.83

LongBench While RULER stresses retrieval, LongBench (Bai et al., 2024) evaluates general long-context reasoning (Table 3), which we find friendly to indexing and reuse. The idx variant lands within half a macro point of dense attention on all four backbones. Reuse under ORS stays cheap at up to 75% reuse, and the only visible degradation comes from FirstAnchor at 75% reuse, which trails full indexing by 0.6 to 1.5 macro points.

Table 3: LongBench v1 macro means at 32k, variants sorted by realized reuse, winners at each reuse level in bold.

reuse	variant	Qwen3-8B	Qwen3-14B	Qwen3-32B	KimiLinear-48B
0%	dense	**46.93**	**47.60**	40.06	47.68
0%	idx	46.75	47.08	**40.13**	**47.86**
50%	FirstAnchor	46.31	**46.83**	39.49	47.38
50%	$\text{ORS}_{\text{relative}}$+FirstAnchor	**46.78**	46.82	**40.13**	**47.83**
75%	FirstAnchor	45.88	46.27	39.55	46.41
75%	$\text{ORS}_{\text{relative}}$+FirstAnchor	**46.56**	**46.46**	**39.84**	**47.62**

4.3 Ablations

QK activation Table 4 compares the exp indexer against a ReLU-activated variant trained on the same data and token budget. The ReLU training is metastable in the deep layers and closes at an order of magnitude worse rank KL, even when trained with more carefully searched learning rates (Appendix D.2); downstream, the ReLU indexer trails its exp counterpart by 6.7 and 1.8 points at full indexing (8B and 32B), and the deficit persists under reuse at both sizes. Our KL metric reads each head's served distribution as a vMF law, which presupposes the exp form, so the ReLU indexer runs only the non-gated variants and both indexer are trained with $\mathcal{L}_{\text{calib}}$ removed.

Table 4: Activation ablation on suite means of RULER at 32k.

model	activation	idx	FirstAnchor (50% reuse)	FirstAnchor (75% reuse)
8B	exp	**92.94**	**89.83**	**81.04**
8B	ReLU	86.21	81.02	75.75
32B	exp	**97.54**	**96.32**	**91.99**
32B	ReLU	95.72	91.99	85.24

Training objective Our proposed objective distills each query's full attention row, whereas the grouped alternative (group2mean) replaces s_{ref} in $\mathcal{L}_{\text{mono}}$ with $1/2 s_{\text{ref}}[:-1] + 1/2 s_{\text{ref}}[1:]$. Table 5 carries the gap downstream. Full indexing drops 3.8 suite points under the grouped objective. The suite-mean deficit persists at both controlled reuse rates, at 3.1–4.4 points at 50% reuse and 1.9–2.0 at 75%. Noticeably, the controlled ORS variants still beat their fixed first-anchor counterparts, even on a checkpoint specifically trained for the grouped objective.

Table 5: Training-objective ablation at 8B, RULER-32k, each cell group2mean / ours under an identical protocol. mk2/mk3: multi-key needle retrieval with two and three needles; vt: variable tracking.

reuse	variant	suite	mk2	mk3	vt
0%	dense	95.97 / 95.97	100 / 100	99 / 99	84.6 / 84.6
0%	idx	89.18 / **92.94**	77 / 88	60 / 82	83.4 / 83.4
50%	FirstAnchor	86.69 / **89.83**	64 / 72	51 / 70	83.2 / 86.4
50%	$\text{ORS}_{\text{relative}}$+FirstAnchor	88.68 / **93.03**	73 / 87	56 / 75	85.8 / 89.0
75%	FirstAnchor	79.16 / **81.04**	39 / 43	21 / 27	79.8 / 77.2
75%	$\text{ORS}_{\text{relative}}$+FirstAnchor	83.96 / **85.97**	52 / 64	36 / 43	83.0 / 78.2

4.4 KL metric fidelity

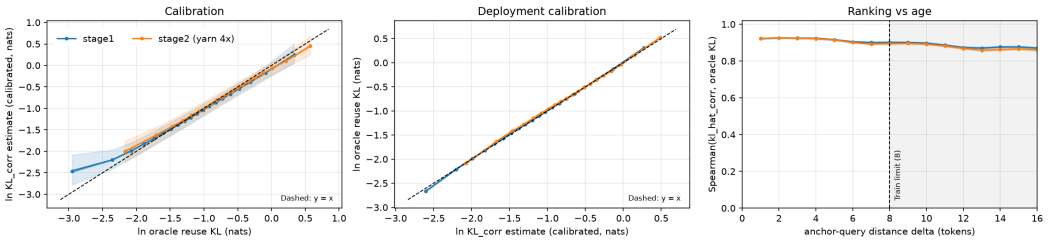


Figure 3: Calibrated $\widehat{\text{KL}}_{\text{corr}}$ against oracle reuse KL, measured on Qwen3-8B, stage-1 and stage-2 indexers. Left: predictor against the oracle. Middle: the deployment calibration $\mathrm{E}[\text{oracle} \mid \text{estimate}]$. Right: per-distance ranking quality.

We validate the calibrated predictor $\widehat{\text{KL}}_{\text{corr}}$ of equation 4 against the oracle reuse KL on the distribution the gate is calibrated on. The measurement is a forward-only replay of the trainer over the training corpus (Figure 3, Qwen3-8B). The predictor ranks pairs by their true reuse damage at Spearman 0.92 at distance 1, decaying only to 0.90 at the training age cap of 8 and 0.87 at distance 16,

indicating that ranking quality survives well past the ages the gate was trained on. The deployment curve E[oracle | estimate] lies on the identity across the observed range, and a threshold on the estimate therefore reads directly as a bound on realized damage. The residual error is conservative. Below the identity crossing at KL ≈ 0.4 nats, squarely in the band where deployment thresholds live, the KL of safe pairs is over-estimated, which means mispredicted pairs err toward recomputation. The stage-2 indexer, trained at 128k under YaRN $4\times$ extrapolation, coincides with the stage-1 indexer on all three panels, indicating that YaRN rescaling leaves the gate's calibration and ranking intact.

5 Related Work

Sparse attention. Heuristic designs fix the key subset in advance: block-sparse, strided, and random patterns (Zaheer et al., 2020), sliding windows with global tokens (Beltagy et al., 2020), and attention sinks for streaming (Xiao et al., 2024). A second line instead shrinks the cache, evicting keys under a budget by importance estimation (Zhang et al., 2023; Li et al., 2024; Cai et al., 2025; Xiao et al., 2025); these methods decide which keys survive rather than which computation is shared. A third line keeps the cache intact and approximates the score matrix through cheap QK indexing, selecting keys per query at inference time by query-aware page summaries (Tang et al., 2024), two-pass proxy scores (Ribar et al., 2024), per-head dynamic patterns (Jiang et al., 2024), or learned gates (Gao et al., 2024). Among these, natively trained indexers take this furthest. NSA (Yuan et al., 2025) combines block compression, token selection, and sliding windows in one trainable design, MoBA (Lu et al., 2025) routes queries to KV blocks, and DeepSeek-V3.2's DSA (DeepSeek-AI et al., 2025) attaches the lightweight lightning indexer whose MQA design Rix adopts. All of these methods still score every query position; Rix exploits the orthogonal query axis and can in principle be stacked on any of them.

Sharing attention across queries. That similar queries can share attention work has classic roots in content-based routing, from LSH bucketing (Kitaev et al., 2020) and spherical k-means routing (Roy et al., 2021) to cluster-centroid broadcast (Vyas et al., 2020). Closer to Rix, a recent line reuses selections between consecutive or similar queries in already-sparse attention. Wu et al. (2025) cache the previous query's selected tokens under a cosine-similarity threshold, ReTopK (Yao et al., 2026) recalls supports of similar past queries with an exact-search fallback, PIVOT (Liu et al., 2026) aggregates a fixed-size group of consecutive queries into one proxy scan that serves the whole group, and Wang et al. (2026b) speculate on the previous step's block selection at decode time. These works establish the query axis as a real efficiency axis, but are restricted to fixed-period re-indexing. Rix identifies corner cases where reuse becomes unsafe and schedules adaptive query-level reuse, preserving index quality while still exploiting the cheap majority.

Sharing indices across layers. A complementary axis amortizes indices vertically. IndexCache (Bai et al., 2026) lets most layers inherit the indices of the nearest full layer, deployed at production scale in GLM-5.2; CLSA (Sun et al., 2026) shares one routing index across the layers of KV-sharing architectures; HySparse (Gao et al., 2026) interleaves full and sparse layers that inherit token selections; Kascade (Deshmukh et al., 2025) chooses anchor layers by dynamic programming over a cross-layer similarity matrix; query-axis reuse is orthogonal and composable with all of them.

6 Conclusion

We study the third axis of index acceleration, reusing indices across queries, and find it abundant but hazardous: cheap for most query pairs, catastrophic for a sharp tail. Our proposed method, Rix, focuses on reuse safety by emitting adaptive reuse schedules. A closed-form reusability metric derived under a spherical key model predicts the reuse divergence from anchor queries to non-anchor queries, the indexer and distillation recipe keep the metric faithful across different reuse rates, and the optimal reuse scheduler places minimal anchors where the gate fires. On LongBench and RULER, at 32k and 128k, Rix outperforms fixed grouping at matched reuse in all but one case in our evaluation, and shows comparable reuse–quality efficiency under fixed global gating thresholds. Rix also provides flexible quality–cost tradeoffs by supporting adjustable thresholds.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods, specifically the construction of the GPU kernel stack (Triton/TileLang kernels) described in Appendix B. We do not use generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for or assist with proofs, propose or refine hypotheses, or design research methodology or experiments: in particular, the theory (the reusability metrics, the mixture-duality analysis, the parallel-ORS construction and its proof), the algorithm designs (indexer architecture, ORS, training objectives), and all experimental design decisions (choice of model family and scales, training corpora, and evaluation protocols) were made without generative AI. The remaining tasks with required disclosure (synthetic data generation, translation, dataset cleaning and reformatting, qualitative data analysis, and results interpretation) are not applicable to this work. Additionally, we used generative AI tools for drafting and editing parts of this manuscript. We have reviewed all AI-assisted work: AI-assisted kernel code was verified against pure-PyTorch reference implementations and is covered by the test suite described in Appendix B, and all AI-assisted text was reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The reusability metrics, ORS schedules, and training objectives are fully specified in §3, with a complete correctness proof of the parallel ORS in Appendix A. Every kernel ships with a pure-PyTorch reference that doubles as a test oracle (Appendix B). Training configurations, corpora, and evaluation protocols are given in Appendix D; anonymized source code will be submitted as supplementary material.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, Vienna, Austria, July 2025. Association for Computational Linguistics.

Yushi Bai, Qian Dong, Ting Jiang, Xin Lv, Zhengxiao Du, Aohan Zeng, Jie Tang, and Juanzi Li. Indexcache: Accelerating sparse attention via cross-layer index reuse. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=74886M5htU.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

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.

Zhi Chen, Qiguang Chen, Libo Qin, Qipeng Guo, Haijun Lv, Yicheng Zou, Hang Yan, Kai Chen, and Dahua Lin. What are the essential factors in crafting effective long context multi-hop instruction datasets? insights and best practices. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27129–27151, Vienna, Austria, July 2025. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, et al. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Dhruv Deshmukh, Saurabh Goyal, Nipun Kwatra, and Ramachandran Ramjee. Kascade: A practical sparse attention method for long-context LLM inference. *arXiv preprint arXiv:2512.16391*, 2025.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Learning intrinsic sparse attention in your LLMs. *arXiv preprint arXiv:2410.13276*, 2024.

Yizhao Gao, Jianyu Wei, Qihao Zhang, Yu Cheng, Shimao Chen, Zhengju Tang, Zihan Jiang, Yifan Song, Hailin Zhang, Liang Zhao, Bo Yang, Gang Wang, Shijie Cao, and Fuli Luo. HySparse: A hybrid sparse attention architecture with oracle token selection and KV cache sharing. *arXiv preprint arXiv:2602.03560*, 2026.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=kIoBbc76Sy`.

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, pp. 52481–52515. Curran Associates, Inc., 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkgNKkHtvB`.

Xunhao Lai, Weiqi Xu, Yufeng Yang, Qiaorui Chen, Yang Xu, Lunbin Zeng, Xiaolong Li, Haohai Sun, Haichao Zhu, Vito Zhang, Jinkai Hu, Jiayao Li, Rui Gao, Zekun Li, Songquan Zhu, Jingkai Zhou, and Pengyu Zhao. MiniMax sparse attention. *arXiv preprint arXiv:2606.13392*, 2026.

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 *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. doi: 10.52202/079017-0722.

Hong Liu, Yuan Cheng, Lin Niu, Yi Su, Yufei Xue, Anmin Liu, Guanghua Yu, and Jianchen Zhu. PIVOT: Efficient query-group indexing for token-level sparse attention. *arXiv preprint arXiv:2607.24593*, 2026.

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, et al. MoBA: Mixture of block attention for long-context LLMs. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 17790–17815. Curran Associates, Inc., 2025.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 31932–31951, 2024.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. SparQ attention: Bandwidth-efficient 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. 42558–42583. PMLR, 21–27 Jul 2024.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Chuxu Song, Zhencan Peng, Jiuqi Wei, and Chuanhui Yang. CSAttention: Centroid-scoring attention for accelerating LLM inference. *arXiv preprint arXiv:2604.08584*, 2026.

Yutao Sun, Yanqi Zhang, Li Dong, Jianyong Wang, and Furu Wei. You only index once: Cross-layer sparse attention with shared routing. *arXiv preprint arXiv:2606.06467*, 2026.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, pp. 10–19, New York, NY, USA, 2019. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21665–21674. Curran Associates, Inc., 2020.

Lei Wang, Yu Cheng, Yining Shi, Zhiwen Mo, Zhengju Tang, Wenhao Xie, Tong Wu, Lingxiao Ma, Yuqing Xia, Jilong Xue, Fan Yang, and Zhi Yang. Tilelang: Bridge programmability and performance in modern neural kernels. In *International Conference on Learning Representations*, volume 2026, pp. 73099–73134, 2026a.

Qi Wang and Lizhang Chen. Mix-context post-training dataset for 128k context extension, 2026. URL `https://huggingface.co/datasets/ghostcc3`.

Tianyu Wang, Gourav Rattihalli, Aditya Dhakal, Junbo Li, Zhiwei Ren, Dejan Milojicic, and Longfei Shangguan. Predict, reuse, and repair: Accelerating dynamic sparse attention for long-context LLM decoding. *arXiv preprint arXiv:2606.30389*, 2026b.

Wei Wu, Zhuoshi Pan, Kun Fu, Chao Wang, Liyi Chen, Yunchu Bai, Tianfu Wang, Zheng Wang, and Hui Xiong. TokenSelect: Efficient long-context inference and length extrapolation for LLMs via dynamic token-level KV cache selection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21264–21281, Suzhou, China, November 2025. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

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 *International Conference on Learning Representations*, volume 2025, pp. 37228–37253, 2025.

Peng Xu, Wei Ping, Xianchao Wu, Chejian Xu, Zihan Liu, Mohammad Shoeybi, and Bryan Catanzaro. ChatQA 2: Bridging the gap to proprietary LLMs in long context and RAG capabilities. In *International Conference on Learning Representations*, volume 2025, pp. 67244–67258, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024.

Wenshuai Yao, Wenyong Zhou, Hanyong Shao, Yizhe Chen, Zhiyuan Ning, Yuannuo Feng, Ru Huang, and Kechao Tang. Recall before you rank: Similarity-guided top-k reuse for efficient long-context attention. *arXiv preprint arXiv:2607.27692*, 2026.

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 *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.

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 *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020.

Jiajie Zhang, Yushi Bai, Xin Lv, Wanjun Gu, Danqing Liu, Minhao Zou, Shulin Cao, Lei Hou, Yuxiao Dong, Ling Feng, and Juanzi Li. LongCite: Enabling LLMs to generate fine-grained citations in long-context QA. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5098–5122, Vienna, Austria, July 2025. Association for Computational Linguistics.

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 *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023.

Appendices

A Parallel ORS: correctness proof

This appendix proves that the serial ORS greedy uses the minimum number of anchors (§A.1) and that the anchor-count binary-lifting scheduler outlined in §3.3 returns **exactly** its anchor set (§A.2–A.3).

A.1 Optimality of the serial greedy

Positions are $0, \dots, S-1$. For each position j, the gate scan (§3.3, capped at the age cap) yields $F[j] = \min(j + N_j,\ S)$, the smallest position *not* coverable by a legal block anchored at j; a block anchored at j legally covers $[j, F[j]-1]$, and $F[j] \geq S$ means the block reaches the end of the sequence — such anchors are **terminal**. The serial ORS greedy (jump game) is the orbit

$$a_0 = 0, \qquad a_{t+1} = W(a_t), \qquad W(j) = \operatorname*{arg\,max}_{j' \in (j,\ \min(F[j],\ S-1)]} F[j'], \tag{10}$$

with first-achieving tie-breaking, stopping at the first terminal anchor. To make W a total map, terminal rows are defined to be absorbing: $W(j) = j$ when $F[j] \geq S$. Writing $\tau = \min\{t \geq 0 : F[a_t] \geq S\}$, the serial schedule is $\{a_0, \dots, a_\tau\}$ with $A = \tau + 1$ anchors. The window in W is non-empty for non-terminal j (it contains $j+1 \leq \min(F[j], S-1)$), so W is well-defined and deterministic. (The walk map W is unrelated to the trailing window W of the main text; no window appears in this appendix.)

A legal schedule is a strictly increasing anchor sequence $0 = b_0 < b_1 < \cdots < b_{m-1}$ with $b_{t+1} \leq F[b_t]$ and $F[b_{m-1}] \geq S$: block t covers $[b_t, F[b_t]-1]$, so the next block must open no later than $F[b_t]$ for coverage to stay contiguous, and the last block must reach the sequence end. The schedule costs m fresh index computations, and the jump game of §3.3 asks for the minimum m. Write the recurrence of §3.3 in the notation of this appendix, the block right end $F[i]-1$ in place of N_i-1 (the cap at S only clamps values the walk has already stopped at), with $f(0) = f(-1) = -1$:

$$f(a) \ = \ \max\big\{\, F[i] - 1 \ :\ f(a-2) < i \leq f(a-1) + 1 \,\big\}. \tag{11}$$

Proposition (the frontier recurrence is exact). *$f(a)$ is the largest position coverable by a legal schedule of a anchors: the right-hand side is attained by a legal a-anchor schedule, and no legal a-anchor schedule reaches beyond it.*

By induction on a. Base $a = 1$: the band is $\{0\}$, position 0 is the only legal first anchor, and $f(1) = F[0] - 1$ is its block end. Step $a \geq 2$. Attainment: by the hypothesis some legal $(a-1)$-anchor schedule covers $[0, f(a-1)]$, so every i in the band opens a legal a-th block, the arg max included, whose block then covers through $f(a)$; note also $f(a) \geq f(a-1) + 1$, since the band contains $f(a-1) + 1$ and a block covers its own anchor. Unreachability: let b be the last anchor of any legal a-anchor schedule. Contiguity bounds b by one past the reach of the schedule's first $a - 1$ anchors, so $b \leq f(a-1) + 1$ by the hypothesis. If $b > f(a-2)$, then b lies in the band and $F[b] - 1 \leq f(a)$ by the maximum. If $b \leq f(a-2)$, then b is itself a legal $(a-1)$-th anchor — an $(a-2)$-anchor schedule covers $[0, f(a-2)] \supseteq [0, b-1]$ — and the hypothesis at $a-1$ gives $F[b] - 1 \leq f(a-1) < f(a)$. Either way the schedule ends no further than $f(a)$. ■

Corollary (optimality of the greedy). *The serial ORS greedy solves the jump game with $A = \tau + 1$ anchors, the minimum over legal schedules.*

The walk stops at the first a with $f(a) \geq S - 1$; since f is strictly increasing, every schedule of $m \leq a - 1$ anchors ends at or before $f(m) \leq f(a-1) < S - 1$ and is incomplete. The orbit attains the frontier at every step, $F[a_t] - 1 = f(t+1)$: its window contains every band candidate to the right of a_t, band candidates at or below a_t were passed over by an earlier window and satisfy $F[i] \leq F[a_t] = f(t+1) + 1$, and window candidates at or below $f(t-1)$ are legal t-th anchors, bounded by $f(t) < f(t+1)$ by the proposition. ■

A.2 Setup

The parallel scheduler reproduces the orbit of §A.1 by binary lifting on the anchor count. The lifting tables are built by

$$\text{Land}(j,1) = W(j), \quad \text{End}(j,1) = F[W(j)], \tag{12}$$

$$\text{Land}(j,2n) = \text{Land}(\text{Land}(j,n),n), \quad \text{End}(j,2n) = \text{End}(\text{Land}(j,n),n). \tag{13}$$

Theorem. *(i)* For every position j and every $k \geq 0$, $\text{Land}(j,2^k) = W^{2^k}(j)$ and $\text{End}(j,2^k) = F\big[W^{2^k}(j)\big]$. *(ii)* The count phase returns $A = \tau + 1$. *(iii)* The enumeration phase maps each ordinal $d \in [0, A)$ to a_d. Hence the lifted schedule coincides with the serial ORS schedule position-for-position.

A.3 Proof

Lemma 1 (doubling invariant). *Part (i) of the theorem holds.*

By induction on k. Base $k = 0$: definitions of $\text{Land}(j,1)$ and $\text{End}(j,1)$. Step: suppose the claim holds at $n = 2^{k-1}$. Since W is total, its iterates compose, $W^n \circ W^n = W^{2n}$, and

$$\text{Land}(j,2n) = \text{Land}(W^n(j),n) = W^n(W^n(j)) = W^{2n}(j), \tag{14}$$

$$\text{End}(j,2n) = \text{End}(W^n(j),n) = F\big[W^n(W^n(j))\big] = F\big[W^{2n}(j)\big]. \qquad \blacksquare \tag{15}$$

Lemma 2 (termination is monotone). *If $F[a_t] \geq S$ then $F[a_{t'}] \geq S$ for all $t' \geq t$: the predicate $P(t) = [F[a_t] < S]$ is downward-closed, $\{t : P(t)\} = \{0, \ldots, \tau - 1\}$.*

Terminal rows are absorbing: $F[a_t] \geq S$ implies $a_{t+1} = W(a_t) = a_t$, hence $a_{t'} = a_t$ for all $t' \geq t$. ■

Lemma 3 (count phase). *The high-to-low bit walk — cur $\leftarrow 0$, $n \leftarrow 0$; for k descending, if $\text{End}(\text{cur}, 2^k) < S$ then cur $\leftarrow \text{Land}(\text{cur}, 2^k)$, $n \mathrel{+}= 2^k$ — ends with cur $= a_n$ and $n = \max\{t : P(t)\}$ whenever $\tau \geq 1$, and $n = 0$ when $\tau = 0$. The returned value $A = n + 1 + [F[\text{cur}] < S]$ equals $\tau + 1$.*

By Lemma 1 the bit test reads $\text{End}(\text{cur}, 2^k) = F\big[W^{2^k}(a_n)\big] = F[a_{n+2^k}]$, so the test is exactly $P(n+2^k)$: a jump is taken iff the anchor 2^k steps ahead is non-terminal. We claim the walk maintains the invariant $P(t)$ for all $t \leq n$. It holds at $n = 0$ when $\tau \geq 1$; a taken jump lands at $n + 2^k$ with $P(n+2^k)$ true, and monotonicity (Lemma 2) gives P on the whole interval $[0, n+2^k]$ — the invariant is preserved. At the end, n is the largest value the walk can reach with $P(n)$ true. It remains to show the greedy descent reaches $n^* = \max\{t : P(t)\}$. Suppose for contradiction the walk finishes at $n < n^*$. Let k^* be the largest bit of $n^* - n$ not taken below its position; when the walk inspected bit k^* at some $n' \leq n$, the test $P(n' + 2^{k^*})$ was evaluated with $n' + 2^{k^*} \leq n + (n^* - n) = n^*$ (all bits of $n^* - n$ above k^*) were taken by descent order), so by monotonicity $P(n' + 2^{k^*})$ was true and the jump would have been taken — contradiction. Hence $n = n^* = \tau - 1$ when $\tau \geq 1$, and $F[\text{cur}] < S$ by definition of n^*, giving $A = (\tau - 1) + 1 + 1 = \tau + 1$. When $\tau = 0$ (the very first anchor is terminal), every test reads $F[W^{2^k}(0)] = F[0] \geq S$ (absorbing), no jump is taken, and $A = 0 + 1 + [F[0] < S] = 1 = \tau + 1$. ■

Lemma 4 (enumeration phase). *Ordinal $d \in [0, A)$ maps to a_d, independently for all d.*

Write $d = \sum_k d_k 2^k$. The kernel applies $\text{Land}(\cdot, 2^k)$ for each set bit $d_k = 1$, starting from position 0. Each such map is W^{2^k} (Lemma 1), and iterates of one function add under composition, $W^a \circ W^b = W^{a+b}$. Thus, the bits apply in any order and the result is $W^{\sum_k d_k 2^k}(0) = W^d(0) = a_d$. The tables are read-only after the doubling phase, so all ordinals are computed independently and in parallel; each reuse position's source anchor ordinal comes from the same decomposition, replacing the searchsorted against the anchor list. ■

Bit-exactness. The serial walk and the lifting read the same map W: the materialization pass uses the serial scan's window $(j, \min(F[j], S-1)]$ and its first-achieving tie-break, and every table entry derives from W by function composition over integer indices where no floating-point arithmetic is involved anywhere in the scheduler. The agreement is therefore exact; the serial Triton walk is retained as the test oracle, and the fused lifting is validated against it. ■

A.4 Complexity

W materialization is one windowed-argmax pass, $O(S\Delta_{\max})$ work — negligible under the deployment age cap $\Delta_{\max}$=8 (and, conversely, the lifting only pays off *because* of the cap: uncapped, this pass alone costs as much as the gate scan it replaces). Doubling costs $O(S \log A)$ gathers over $\lceil \log_2 A \rceil \approx 15$ rounds at 128k; the count is a single $O(\log S)$-depth bit walk in one program; enumeration is fully parallel over ordinals. The serial walk it replaces is $O(A)$ latency-bound iterations, $A \approx S/\hat{m} \approx 35.6$k at $S = 131072$, $\hat{m} \approx 3.7$ — 33.0 ms vs 2.0 ms end-to-end for the *unfused* torch prototype (W 0.44 ms + tables 0.59 ms + count 0.31 ms + enumerate 0.66 ms), 16.5× before kernel fusion. All tables are int32 (deployed contexts $\leq 10^7 \ll 2^{31}$).

A.5 Iso-reuse control

Comparing schedules fairly requires holding realized reuse fixed across variants, since reuse level alone moves quality. For evaluation we therefore use a bisection variant of ORS that re-finds the threshold τ per layer per prefill to land on a target reuse rate. The bisected variants realize 50% and 75% reuse within ± 0.003 of target on every task, and they are the controls of the frontier comparison of §4.2.

A.6 Pooled anchors: the block JSD, blocked prefix sums, and per-beginning bisection

Pooled anchors and the block JSD. The first-row anchor is one composition of a block's queries; the block mean is another. Pooling the block's query states and head weights, $(\bar{q}_h, \bar{w}_h) = \frac{1}{M}\sum_{b\in B}(q_{b,h},\, w_{b,h})$, composes an anchor whose per-head served law is exactly the geometric centroid of the block's per-head laws, the normalized product of experts $\big(\prod_{b\in B} p_h(\cdot \mid q_{b,h})\big)^{1/M} \propto p_h(\cdot \mid \bar{q}_h)$, since vMF densities multiply by adding their natural parameters. The expected reuse damage of serving block B from one shared distribution is then the block's Jensen–Shannon divergence,

$$\mathrm{JSD}(B) \;=\; \frac{1}{M}\sum_{b\in B} \mathrm{KL}\big(p_b \,\|\, m\big), \qquad m = \frac{1}{M}\sum_{b\in B} p_b, \tag{16}$$

the mutual information between a uniformly drawn row of the block and the key. Because the arithmetic mixture m is the block's forward-KL barycenter, $\mathrm{JSD}(B)$ lower-bounds the mean reuse cost of *any* anchor, first-row, pooled, or otherwise, so it certifies block reusability before an anchor is chosen. The deployed estimate substitutes the pooled state into the augmented decomposition: with $K_{\text{pool}}(b) = \widehat{\mathrm{KL}}_{\text{aug}}\big(b \to (\bar{q}, \bar{w})\big)$ the pairwise form of equation 3 evaluated against the pooled anchor, the block mean $\frac{1}{M}\sum_b K_{\text{pool}}(b)$ upper-bounds equation 16 by the same mixture bound and stays closed form in the same tabulated kernels, and admission is the same affine rule, $\beta \ln\big(\frac{1}{M}\sum_b K_{\text{pool}}(b)\big) + c < \ln\tau$. The jump game is unchanged, with the gate now pricing blocks rather than pairs, and the scheduler evaluates the statistic at $O(\log M)$ candidate spans per position from blocked prefix sums, derived below. Averaging over the block's rows also smooths the statistic where the pairwise gate is spiky, and we find the pooled gate robust to calibration error at scales where the first-row gate is not (§4.2). Pooled anchors read the block's future rows and are therefore a prefill construction; decode keeps first-row anchors.

The deployed realization caps blocks at P rows, a power of two, the anchor row included, so the anchor age never exceeds $P-1$. With the block-end map $N(i)$ fixed by the gate there is no arg max left to take: anchors are the plain orbit $a_0 = 0$, $a_{t+1} = N(a_t)+1$, and the binary lifting of §A.2–A.3 carries over verbatim, bit-exact against the serial orbit. What remains is pricing $N(i)$ itself, since the naive gate loops the block's rows for every candidate span at $O(PHd)$ per candidate.

The block-sum closure. For a candidate block $[i, j]$ of $L = j - i + 1$ rows with pooled state $(\bar{q}, \bar{w})$, pooled shares $\bar{\pi}_h \propto \bar{w}_h\, Z(\|\bar{q}_h\|)$, and per-row concentrations $\kappa_{b,h} = \|q_{b,h}\|$, summing the per-row decomposition equation 4 of all block rows against the pooled state gives the ln-of-mean JSD estimate in closed form,

$$G(i,j) \;=\; e^{-\tau_\pi} \sum_h \Big[S_h^{p \ln p} - S_h^p \ln \bar{\pi}_h \Big] \;+\; \sum_h e^{-\tau_h} \Big[S_h^{pA\kappa} - \bar{q}_h^\top S_h^c + \log Z(\|\bar{q}_h\|)\, S_h^p - S_h^{p \log Z} \Big], \tag{17}$$

with admission iff $\beta \ln(G/L) + c < \ln \tau$. The seven per-head block sums — two vectors $S_h^q = \sum_b q_{b,h}$ and $S_h^c = \sum_b \pi_{b,h} \big(A(\kappa_{b,h})/\kappa_{b,h}\big)\, q_{b,h}$, and five scalars $S_h^w = \sum_b w_{b,h}$, $S_h^p = \sum_b \pi_{b,h}$, $S_h^{p \ln p} = \sum_b \pi_{b,h} \ln \pi_{b,h}$, $S_h^{p \log Z} = \sum_b \pi_{b,h} \log Z(\kappa_{b,h})$, $S_h^{pA\kappa} = \sum_b \pi_{b,h} A(\kappa_{b,h})\, \kappa_{b,h}$, with $\bar{q}_h = S_h^q / L$ and $\bar{w}_h = S_h^w / L$ — reduce the statistic to $O(Hd)$ per candidate. Only the ln-of-mean form closes this way, the division by L happening once inside the logarithm; the max-over-rows analogue of the first-row gate does not close over block sums. The sums come from within-block inclusive prefix sums over blocks of P rows: a span $[i, j]$ never crosses more than one block boundary, so with $S(t)$ the inclusive prefix sum of a quantity x within its block and x_i the raw row value,

$$\sum_{b=i}^{j} x_b \;=\; S(j) \;-\; S(i) \;+\; x_i \;+\; \big[\lfloor j/P \rfloor \neq \lfloor i/P \rfloor\big]\; S\big(i - (i \bmod P) + P - 1\big), \tag{18}$$

one $O(1)$ lookup per span after a single $O(SHd)$ prefix pass per layer.

Per-beginning bisection. Rather than scanning all P candidate ends, the scheduler lifts accept-only from the always-legal singleton $j = i$, a one-row block having zero JSD, with halving steps $P/2, P/4, \ldots, 1$: each round takes the candidate $\min(j + \text{step}, S-1)$ iff it is legal. The steps sum to $P - 1$, so the search covers exactly $[i, i + P - 1]$ in $\log_2 P$ gate evaluations per position. Two approximation notes. The block JSD is only approximately monotone in span, rising and plateauing with small wiggles, so the bisected block end can differ from the exhaustive largest legal end near the boundary, a slack the iso-reuse threshold bisection of §A.5 absorbs; and legality is exact at both extremes, the singleton always passing and the full span evaluated at the first step. Each round is one kernel launch with the running j carried in global memory: fusing all $\log_2 P$ rounds into a single launch keeps ∼800 KB of per-program prefix rows live across rounds and thrashes the 50 MB L2 at ∼200 concurrent programs, a measured 8.7 → 1.9 ms regression at 128k from the restructure.

Thresholds and measured cost. The pooled statistic's distribution differs from the pairwise gate's in that centroid-reference divergences concentrate at smaller values within redundant blocks, and the first-row gate's self-divergence zero disappears. Thus, fixed thresholds do not transfer from the first-row recipes: $\text{ORS}_{\text{relative}}$ re-bisects τ per sequence with a galloping bracket plus twelve bisection steps, and production fixes one threshold per backbone and length, as for the first-row variants. Measured per-layer gate cost on H200 at the 32B geometry (H=8, d=128, P=16): at 128k, 1.20 ms of per-row terms shared with the first-row gate, 0.46 ms of prefix sums, 1.95 ms of bisection, 0.91 ms of pooled-state gather, and 0.36 ms of schedule walk, 4.7 ms in total against 3.4 ms for the first-row gate; at 1M, 33.2 ms against 22.5 ms. A naive per-row evaluation of the same JSD statistic, without the closure, costs ∼6 ms at 128k.

B Kernel implementation

Our kernels are implemented in Triton (Tillet et al., 2019) and TileLang (Wang et al., 2026a). Each ships with its pure-torch reference, which doubles as the test oracle. Benchmarks use Qwen3-14B shapes (H=40, H_{kv}=8, H_{idx}=6, d=128) on H200 unless noted.

B.1 Training path

Indexer scores Fuses $\text{QK}^\top$ → positional causal/padding/window mask → max-subtract → exp → w-weighted head reduction, never materializing the head dimension. Rewritten from two-pass to

single-pass online-max (per-tile fixup sweep rescales earlier tiles by $\exp(m_{\text{tile}} - m_{\text{final}})$): 114 → 59 ms at S=131072. At S=16384: reference einsum path 9.66 GiB / 0.039 s → fused 2.16 GiB / 0.019 s.

Head-pooled oracle softmax The fp32 head-pooled softmax oracle was ~97% of the fused distill step; the kernel folds the GQA group into one row block (each K tile loaded once per KV group), two passes (online max/sum, then normalized probabilities, atomic-added $\times 1/H^*$), never materializing $[H^*, c, S]$ or $[S, S]$. A later rewrite (2-pass online max/sum + hi/lo pool dot + streaming reduce) reaches 191 ms/offset at 128k (115 TF/s, the exact-softmax floor under the 1e-4 rowsum test). End-to-end effect on the distill step (B=1, chunk 1024, A100-40G):

S	torch oracle peak	s/step	Triton oracle peak	s/step	speedup
16384	1.76 GiB	0.607	0.42 GiB	0.275	2.2×
32768	3.51 GiB	2.746	0.82 GiB	1.099	2.5×
65536	7.00 GiB	11.268	1.63 GiB	4.346	2.6×
131072	13.99 GiB	45.204	3.24 GiB	17.336	2.6×

Fused KL + coverage distillation Chunked fwd+bwd autograd Function removing the $[B, H, S, S]$ energies and the $[B, S, S]$ log-pred/oracle materializations from the loss path; the oracle is recomputed per chunk in backward to keep peak activation at $O(\text{chunk} \cdot S)$. The pre-fusion path OOMs at S=16384 already (≈35 GiB of intermediates); the fused path fits 131072 comfortably (table above). The stage-2 anchor loss variant evaluates KL+coverage at anchor rows only against streamed reuse targets (fused cross kernel, t-centric dq/dw + s-centric dk, bf16×2 hi/lo split dots for grad accuracy $\leq 5.7 \times 10^{-3}$ rel): 447 → 101 ms fwd+bwd per layer at S=131072, A=16384 anchors (4.4×). The coverage term gathers only each anchor row's top-2048 target keys (TopK over the key axis, a 2048-wide column subset per row instead of the full 128k) which is the OOM-prevention trick behind stage 2's "sparse" qualifier (§3.2).

Fidelity. Fused vs full-matrix reference on real Qwen3-8B tensors (layers 0/15/35, 30 docs): loss max rel. diff 1.8×10^{-4}, Triton-vs-torch oracle 4×10^{-5}; 30-step AdamW dynamics and recall@2048 trajectories match the reference indistinguishably.

B.2 Inference path

TileLang indexed attention Gathered sparse attention: per-anchor top-K indices $[B, A, K_{\text{sel}}]$ + row→anchor map `src` + trailing window (dense band), flash-style fp32 online softmax, GQA-aware, bf16 in/out; keys gathered inside the window band are masked out of the gathered stream so the union is counted once. The `src` map is the schedule fusion point, in which reuse rows read their anchor's indices with no score compute at all. Verified: torch-reference parity (window on/off, reuse map, decode), chunked == whole bit-exact, layer-0 replay on real Qwen3-14B tensors (maxdiff 1.2×10^{-3}), full-model logit maxdiff 0.19 with stable argmax, and a LongBench-v2 (Bai et al., 2025) 35k example reproducing the eager path's generation text exactly. Per-layer timings (K_{sel}=4096, W=2048, bf16):

S	eager	TileLang	speedup
16k	0.222 s	0.173 s	1.3×
32k	0.912 s	0.364 s	2.5×
64k	3.843 s	0.762 s	5.0×
128k	~15 s (extrap.)	1.405 s	~11×

Grouped gathered attention One CTA per ≤LP-row same-anchor block × KV head: K/V are gathered once per ORS block instead of once per row — 276 → 53.8 ms per layer at S=131072, K=1024, W=5, $\hat{m}$=3.3 (5.5× vs the per-row TileLang kernel; 51 TF/s). The plan builder is sync-free (cummax/cumsum/scatter_reduce). Current limiter is Triton 3.3.1's fully synchronous pipeliner on the serial gather loop; a grouped TileLang port measured slower (108 ms).

Schedule-fused prefill. With a prefill schedule active, indexer scores are computed only at anchor rows (the scores kernel's row gather mode, exact vs full-map at 3.6×10^{-6}) and (indices, src_ord) feed the indexed-attention kernel directly; scores run in bf16 with causal-width narrowing and `sorted=False` top-k (62 $\rightarrow$ 28 ms; boundary swaps only within ~2 bf16 ulps of the k-th value). Smoke: LBv2 35k example 30.7 $\rightarrow$ 25.5 s, same generation.

Decode fast path. Per layer-step at S=131072: fused fresh-step scores + exact top-k (4 launches, zero host syncs, ieee dots — TF32 flipped boundary keys) 418 $\rightarrow$ 95 μs; capacity-doubling indexer-K cache 70 $\rightarrow$ 22 μs; the KL gate itself is a handful of flops per position from the tabulated surrogate of §3.1 and adds no measurable time. Reuse steps return the plan tuple (anchor indices, src) directly. End-to-end: 128/128 generated tokens identical to the reference path.

Parallel ORS Fused realization of the §A construction (four kernels: W materialization, doubling, count, enumeration), bit-exact against the serial walk.

C Training-time numerics

C.1 Head-copy initialization

For each layer we measure the concentration of every backbone attention head by the probability mass of its top output keys on a few real long sequences, and copy the query projection blocks of the most concentrated heads into the indexer query heads together with the backbone's trained query-norm weights. The shared key projection is taken from one key-value group of the backbone, with the backbone's trained key-norm weights folded into the copied block, since the indexer's own key-side norm is unparameterized. The folded key path reproduces the teacher's keys up to a per-layer scalar, and a calibration pass over a few sequences absorbs that scalar into the query-side norm by matching the indexer's logit scale to the main attention's. The indexer therefore starts from query heads that already retrieve like the selected backbone heads, with the remaining difference that all indexer heads share a single key head.

C.2 Indexer forward numerics

Indexer energies are computed in fp32 with per-query max-subtraction before `exp` (causal mask applied *before* the max so future keys cannot underflow valid energies). This is exact for KL training (energies renormalize per query) and rank-preserving for top-k; it cured the bf16 overflow/underflow failure modes of Appendix D.3. One subtlety surfaced by real-model validation. The training script's reference max-subtracts over *all* keys (including future), while the kernels use the masked max; the loss values agree to 10^{-4} either way, but at rows with heavy 1e-30-clamp activity the two variants' gradients diverge (0.4–14%, layer-dependent). Bit-parity, if ever wanted, is a one-line change that passes the causal mask to the indexer in the train script, matching the kernels.

The remaining training-dynamics lessons (bf16 `exp` overflow freezing layers, the underflow logit-spread trap cured by fp32 energies, and the zero-avoiding forward KL destroying top-k ranking at concentrated layers, the last cured by the coverage-anchored loss) are catalogued in Appendix D.3.

D Experimental and training details

D.1 Hardware, frameworks, and hyperparameters

Hardware Indexer training runs on a single 8$\times$H200 node with the backbone layers sharded across ranks, each rank distilling its layer subset independently (§D.2). Kernel-development microbenchmarks ran on A100-40G (the distill-step table of §B.1) and H200. Every quality evaluation and every latency number in this paper is measured on a single H200 (141 GiB).

Frameworks Training is PyTorch with the fused Triton kernels of §B.1 in the loss path, launched with `torchrun` over Hugging Face Transformers backbones. Inference shares one kernel suite across two stacks. The first is the eager HF research harness (TileLang indexed-attention prefill, Triton decode), used for kernel validation and for the 32B latency sheet. The second is the production vLLM 0.26.0 stack with customized plugin, which serves all reported quality evaluations and provides paged

row-granular gather, the split decode kernel, the fused fp32 ORS gate, and, for KimiLinear, the absorbed-layout FlashMLA sparse path. Compute is bf16 throughout; gate and softmax statistics are computed in fp32.

Hyperparameters Table 8 collects the training configuration behind §D.2. Inference runs greedy decoding (chat template, at most 256 new tokens) at indexer top-k 2048, with a $W = 8$ trailing window and anchor age cap 8.

Table 8: Training hyperparameters of the two-stage indexer recipe of §D.2. Steps are optimizer steps; the cosine schedule is counted in optimizer steps as well.

	stage 1 (dense warmup)	stage 2 (context expansion)
context	$\leq$32k windows, native RoPE	128k windows, YaRN $\times 4$
objective	mono-query forward KL + Huber calibration	same
steps	4000	4000
optimizer	AdamW, cosine in optimizer steps, grad-accum 8	same
learning rate	10^{-3}	10^{-3}
ReLU variant	10^{-3} on L0–14, 3×10^{-4} on L15–35	
training block ages	1–8 (matches the deployed age cap)	1–8
τ_h initialization	$\ln 3.5 \approx 1.25$ (log domain, all backbones)	carried from stage 1
coverage term	full key axis	top-2048 target keys per anchor row
precision	fp32 energies, bf16 kernels	same

D.2 Training data and schedule

Training data The corpus is a four-corpus long-document mixture of LongCite-45k (Zhang et al., 2025) (44.6k doc-QA), LongMIT-128k (Chen et al., 2025) (64.4k docs, en+zh), mix-context-post-training-128k (Wang & Chen, 2026) (134.7k docs, re-tokenized from Llama-3 packed rows), and ChatQA2-Long-SFT (Xu et al., 2025) (144k samples, chat-templated at convert time; the other three stay raw document text), totaling 387,695 docs and 14.84B Qwen3-BPE tokens. Two length-filtered shard sets serve the two stages, warmup-32k (docs $\geq$ 4,096 tokens; 358,938 docs / 14.80B tokens / 112 shards, chopped to $\leq$32k windows at build time) and stage2-128k (docs $\geq$ 65,536 tokens, no subsampling of the long end; 78,051 docs / 8.06B tokens / 62 shards, served as 128k windows). The shard format is doc-boundary-preserving (SEG-framed VSL streams), so the same artifacts serve any context length; per-corpus ratios are set by the data glob, and the last shard of each corpus is held out for validation.

Stage 1 (dense warmup, $\leq$32k, native RoPE) The head-copy-initialized indexer is distilled on 32k windows of the warmup corpus with the backbone in its untouched native configuration (`max_position_embeddings = 40960`), for 4000 steps at lr 10^{-3}. The ReLU/DSA variant is the exception, needing a per-layer-group vector (10^{-3} on L0–14, 3×10^{-4} on L15–35) against a metastable deep-layer divergence. The objective is the mono-query forward KL of §3.2 plus the Huber calibration term of §3.2, with training block ages 1 through 8 matching the deployed age cap and τ_h initialized at $\ln 3.5$.

Stage 2 (sparse context expansion, 128k, YaRN (Peng et al., 2024)) Training resumes from the stage-1 indexer on the YaRN-scaled checkpoint with 128k windows over the long-doc shard set, 4000 steps, same objective, lr 10^{-3}.

Training shards the backbone layers across the node's GPUs, layers independent, no cross-rank backward (§D.1).

D.3 Training findings

Calibration costs nothing on the indexer fit. The calibrated run finishes at mono KL 0.080 against 0.088 for the same recipe without the calibration term, and the trained corrections drift from the $\ln 3.5$ initialization to per-layer-group means of 0.60 to 1.58, with mid layers corrected down, meaning the raw augmented predictor underestimates realized KL there. A plausible explanation for the

free calibration is that the calibration loss carries cross-query information, pairs of query states and their realized reuse error, which the single-query objective does not contain, so it complements the distillation gradient instead of competing with it.

Training-dynamics lessons The remaining lessons were each falsified in turn during development. bf16 exp overflow froze four layers of the first run. bf16 underflow after max-subtraction created a runaway logit-spread trap, cured by fp32 energies. Zero-avoiding forward KL destroyed top-k ranking at the most concentrated layers, cured by anchoring the loss on coverage, which recovered recall from 0.13 and 0.14 to 0.76 and 0.75 on the two dead layers and raised the fresh-index ceiling from 0.57 to 0.95. ReLU head weights died under gate-loss pressure in stage 2, zero being an absorbing state, with six layers affected and up to 69% dead tokens on the worst layer, cured by the softplus parameterization of §3.2. The learning-rate response is family-dependent: the exp indexer trains uniformly at 10^{-3} with a flat response to 10^{-2}, while the ReLU variant is metastable in deep layers and needs a per-layer-group learning-rate vector to finish a clean warmup.

D.4 Coverage law

We measure per-head-normalized softmax-mass coverage $m(S) = H^{-1}\sum_h \sum_{k\in S} \mathrm{softmax}_h(qk)$ on Qwen3-8B and LongBench's passage-retrieval-en subset at 16k context. Our empirical results on 100 samples show that coverage above 70% is lossless on this subset, requiring a median of only 11 keys to reach that coverage. However, coverage at 60% sacrifices half of the retrieval quality, and at 40% the model fails at retrieval. The observation supplies the failure condition behind the sharp corners of §2.

We further measure the coverage costs at up to 128k context on the training corpora across the Qwen3 model family. Table 9 reports the worst-direction percentiles of Keys and Queries of Algorithm 1. We find that the hardness of serving a query decreases with model size while index reusability increases with model size. p50 of Keys decreases as model size increases from 8B to 32B, while the p90 worst case of Queries increases. Together, these results suggest that reusing indices across queries is more effective for larger models. However, the worst-case tails of both Keys and Queries remain comparably extreme across model sizes, requiring careful method design to handle the tail.

Table 9: Worst 50th / 90th / 99th percentiles of Keys and Queries of Algorithm 1 with the local window eliminated ($W{=}0$). Measured on Qwen3-8/14/32B, sequences sampled from our training corpus (32 per cell), all layers pooled. Queries is clipped at block length 16; percentiles are worst-direction (upper tail for Keys, lower tail for Queries). †: with YaRN $2\times$. ‡: with YaRN $4\times$.

model	context	Keys(70%)	Keys(90%)	Queries(70%, 1024)	Queries(70%, 2048)
8B	16k	19 / 159 / 619	362 / 1550 / 3676	16 / 16 / 10	16 / 16 / 16
	32k	26 / 235 / 859	546 / 2618 / 5748	16 / 16 / 2	16 / 16 / 16
	64k†	30 / 353 / 1445	879 / 4202 / 9070	16 / 12 / 0	16 / 16 / 4
	128k‡	51 / 479 / 2275	1687 / 6254 / 14749	16 / 4 / 0	16 / 16 / 0
14B	16k	18 / 245 / 735	371 / 1995 / 4169	16 / 16 / 4	16 / 16 / 16
	32k	25 / 387 / 1191	545 / 3394 / 7378	16 / 16 / 0	16 / 16 / 11
	64k†	26 / 561 / 1790	867 / 5294 / 11078	16 / 5 / 0	16 / 16 / 1
	128k‡	44 / 723 / 2345	1677 / 7938 / 16214	16 / 2 / 0	16 / 12 / 0
32B	16k	8 / 120 / 604	264 / 1515 / 3873	16 / 16 / 5	16 / 16 / 16
	32k	11 / 192 / 1092	416 / 2688 / 6806	16 / 16 / 0	16 / 16 / 15
	64k†	11 / 233 / 1705	566 / 3841 / 10425	16 / 16 / 0	16 / 16 / 1
	128k‡	15 / 296 / 2139	960 / 5481 / 15029	16 / 11 / 0	16 / 16 / 0

D.5 Performance consistency: distractor-density needle sweeps

Distractor-density needle sweeps reproduce the same ordering at all three model sizes (Table 10). Where the index itself is the bottleneck (8B at the highest density) the learned schedule holds at the index's own ceiling, while FirstAnchor collapses 15 to 16 points below it. At 14B the ORS margin over FirstAnchor reaches 22 points on the densest three-key cell, while being slightly worse on the

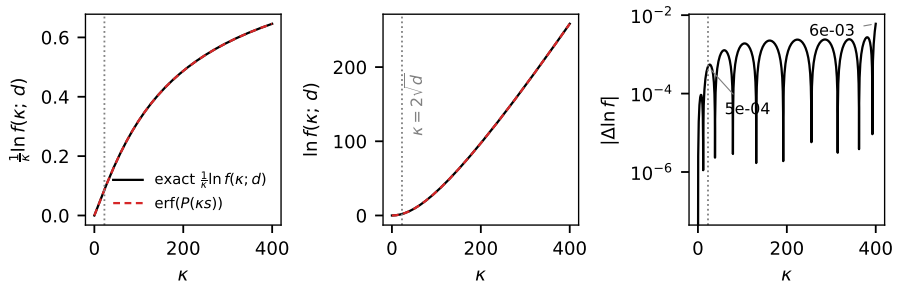


Figure 4: The erf-polynomial surrogate of the sphere kernel at d=128. Left, the normalized log kernel $\frac{1}{\kappa}\ln f(\kappa; d)$ (solid) against $\mathrm{erf}(P(\kappa s))$ (dashed), the empirical form motivating the surrogate. Middle, the resulting $\ln f(\kappa; d)$ against $\kappa\,\mathrm{erf}(P(\kappa s))$. Right, the absolute error $|\Delta \ln f|$ on a log scale. The dotted line marks $\kappa = 2\sqrt{d}$, the range of the original fit. The degree-nine refit stays below 5×10^{-4} inside that range and below 6×10^{-3} over the full reachable range $\kappa \leq 400$.

two-key ladder. At 32B every variant is nearly density-immune and the schedules are indistinguishable. Consistent with this picture, the placement advantage in the main matrix concentrates in the hard retrieval tasks, while the multiquery axis sits near the ceiling at sufficient backbone capacity.

Table 10: Full distractor-density needle sweep at all nine settings (native backbones, top-k 1024, W=8). mk2/mk3 are multi-key retrieval with two/three needles; distractor density grows with the context length and saturates at 32k. mq is multi-query retrieval (four queries, 8/16/64 keys at 32k). Bold marks the higher of the two 50%-reuse arms in each cell (ties bolded in both).

			mk2			mk3			mq		
	reuse		11k	22k	32k	8k	16k	32k	k8	k16	k64
8B	0%	dense	99	99	100	100	100	99	98.75	99	99
	0%	idx	99	95	70	100	99	61	97.75	98.75	97.25
	50%	FirstAnchor	96	74	55	99	97	45	97.75	**98.75**	**96.5**
	50%	$\mathrm{ORS}_{\mathrm{relative}}$+FirstAnchor	**99**	**91**	**68**	**100**	**100**	**58**	**98**	98	96.25
14B	0%	dense	100	100	100	100	100	100	100	100	100
	0%	idx	100	100	95	100	100	93	98.5	99.5	100
	50%	FirstAnchor	**100**	93	**83**	98	96	37	**99.25**	**99.5**	**99.5**
	50%	$\mathrm{ORS}_{\mathrm{relative}}$+FirstAnchor	98	**97**	78	**100**	**99**	**59**	98	**99.5**	**99.5**
32B	0%	dense	100	100	100	100	100	100	100	100	100
	0%	idx	100	100	98	100	100	99	100	100	100
	50%	FirstAnchor	99	95	**97**	99	**100**	89	99.5	99.75	98
	50%	$\mathrm{ORS}_{\mathrm{relative}}$+FirstAnchor	**100**	**99**	94	**100**	**100**	**96**	**100**	**100**	**100**

D.6 Backbone head disagreement under YaRN

The widening idx–dense gap under YaRN in Table 2 has a head-level account. The mono objective of equation 6 distills the indexer against the head-averaged attention row, so its per-key signal degrades as the backbone's heads disagree. We measure this disagreement on the frozen backbones: with p_h the attention row of head h and $\bar{p} = \frac{1}{H}\sum_h p_h$ the pooled distillation target, the head disagreement of a layer is the multivariate Jensen–Shannon divergence $\mathrm{JSD} = \frac{1}{H}\sum_h \mathrm{KL}(p_h \,\|\, \bar{p}) = H(\bar{p}) - \frac{1}{H}\sum_h H(p_h)$, the entropy inflation that head pooling averages over; $\exp(\mathrm{JSD})$ is the resulting multiplicative inflation of the effective support size. Table 11 reports the layer-mean JSD on content-matched held-out long documents.

We have three observations. First, YaRN $4\times$ scaling raises disagreement by about 0.2 nats at fixed 32k context at every model size, and stretching to 128k adds another 0.1, indicating that the indexer trains against a noisier target exactly where Table 2 shows the widened gap, and the degradation is a backbone property rather than a reuse-schedule failure, consistent with the needle-sweep collapse of

Table 11: Head disagreement of frozen backbone attention rows (layer-mean multivariate JSD, nats), on content-matched held-out long documents. YaRN cells run YaRN $4\times$; KimiLinear is NoPE with a native 1M context, so its 128k cell needs no scaling.

backbone	native @32k	YaRN @32k	YaRN @128k
Qwen3-8B	0.868	1.053	1.149
Qwen3-14B	0.913	1.115	1.210
Qwen3-32B	0.981	1.150	1.256
KimiLinear-48B	0.557	—	0.620

idx itself at the densest cells (Appendix D.5). Second, disagreement grows with model size within the family (0.87 to 0.98 nats natively at 32k). Third, KimiLinear's MLA heads stay far below every GQA cell (0.56–0.62 nats) and degrade least with length, probably related to architectural difference between MLA and GQA.

E The acceleration view: reuse pays end-to-end only at small index budgets

The reuse lengths of §2 and Appendix D.4 count only indexing compute displaced by reuse and ignore that sparse main attention costs proportionally to the index budget per query. Writing L for the context length and $\hat{m}(\text{Siz})$ for the realized mean reuse length under index budget Siz — the deployed analogue of Queries in §2 — the end-to-end flop model per attention layer, counting causal MACs for $\text{QK}^\top$ and AV in the sparse pass and amortizing the indexer sweep over the block, is dense $\propto HdL^2$ against sparse $\propto 2Hd(\text{Siz}{+}W)\,L + H_{\text{idx}}\,d\,L^2/\hat{m}(\text{Siz})$. At the deployed design point the trailing window is $W = 8$ and the anchor age cap is $\Delta_{\max} = 8$: the anchor's shared index covers every key up to the anchor position, so a reuse position needs only its intra-block causal tail. The head ratios H_{idx}/H are $4/32$, $6/40$, $8/64$ for 8B, 14B, 32B. The $\hat{m}$ values below are censored at a block length of 5 from the earlier design point; the deployed cap of 8 only lengthens blocks, so the tabulated speedups are conservative. Speedup against dense at increasing Siz:

cell	Siz=1024	Siz=2048	Siz=4096	Siz=8192
8B @16k	6.54×	3.62×	1.90×	0.98×
8B @32k	10.88×	6.58×	3.63×	1.90×
8B @128k	18.63×	15.65×	10.96×	6.60×
14B @32k	10.18×	6.34×	3.56×	1.89×
14B @128k	16.31×	14.15×	10.28×	6.38×
32B @128k	21.16×	16.58×	11.16×	6.63×

The indexing-only view says Siz=4096 is the knee of the $\hat{m}(\text{Siz})$ curve at 128k; the end-to-end view reverses it, and Siz=4096 is 1.59 to 3.46 times slower than Siz=1024 in every cell: raising Siz lifts the censored $\hat{m}$ only from 3.29 to 4.32 while the main-attention term quadruples from 1032 to 4104 attended keys. Deployment should therefore use Siz ≤ 2048, and the index budget assumed in training must match the deployment budget, since reuse targets computed under a 4096-key index are invalid under a 2048-key deployment index. This is a flop proxy: gathered attention is bandwidth-bound while the indexer sweep is GEMM-like, so the wall-clock gap is likely wider than 1.59 to 3.46 times, not narrower. The fused kernels of Appendix B close the gap between this model and wall clock.

Wall-clock costs across context lengths Figure 5 measures this model on the production stack at Siz=1024. The left panel adds the gathered attention to the indexer forward of Figure 2 of the main text: the grouped kernel holds the sparse layer nearly flat against a dense layer that grows linearly in context, so the per-layer margin of the 75%-reuse ORS variant over dense widens from $1.6\times$ at 32k to $5.7\times$ at 128k and about $18\times$ at 1M. The right panel multiplies out to the prefill wall: the ORS variant beats dense from 32k onward, and the margin grows to about $9\times$ at 1M, while at matched reuse ORS and fixed-period grouping differ only by the fp32 gate.

Measured acceleration Table 12 gives the measured counterpart of Figure 5 at the deployed corner Siz=1024, W=8. The rows through 128k are solo end-to-end measurements, and the longer rows are extrapolated from per-layer microbenchmarks validated within 10% against them. The 75%-reuse attention layer undercuts dense attention from 32k on ($2.46\times$) and reaches $95\times$ at 1M; multiplied out, the 75% ORS variant wins the prefill wall from 32k ($1.08\times$) to 1M ($\sim 9\times$), while decode attention stays at 41 µs per token against the dense cost of 141 µs at 128k and 1037 µs at 1M. At 128k its prefill margin over dense is $2.05\times$, $1.88\times$, and $1.66\times$ at Siz 1024, 2048, and 4096.

Table 12: Measured acceleration of Qwen3-32B on the production stack (single H200, bf16, top-k 1024, $W = 8$ trailing window, age cap 8, fixed reuse thresholds; realized reuse 0.47–0.50 for the 50% ORS variant and 0.73–0.75 for the 75% variant). Attention columns give per-layer prefill cost (50% $\approx 1.7\times$ 75%, the measured ratio). Prefill columns give end-to-end wall time; rows through 128k are measured end-to-end, ~ marks cells extrapolated from per-layer microbenchmarks validated within 10% against the measured rows. Decode attention per token (not shown): dense grows 141 µs at 128k to 1037 µs at 1M, the ORS sparse plan stays 41 µs flat.

	attention (ms/layer)			prefill wall (s)			
L	dense	idx	ORS 75%	dense	idx	ORS 50%	ORS 75%
32k	29.0	24.1	11.8	5.97	6.43	6.13	5.52
64k	127	48.3	23.4	15.90	14.18	12.83	11.46
128k	511	98.9	46.2	49.59	31.40	28.58	24.22
256k	2264	201.4	93.1	181.9	~78.8	~62.6	~49.9
512k	9424	408.4	193.9	677.2	~209.9	~152.3	~117.4
1M	38754	823.8	407.5	2628.3	~629.9	~411.9	~291.0

Pooled anchors: comparable speed at matched reuse, preserving quality at high-reuse Table 13 measures the indexer forward of the pooled-anchor variants (block-JSD admission) at their production corner. At matched reuse the pooled indexer forward equals the first-row one up to the gate; the gathered attention of Table 12 is anchor-count- and window-driven, so its attention and prefill columns carry over to the pooled variants at matched reuse (substituting the pooled totals moves the end-to-end wall by less than 1%), and decode is untouched. The pooled JSD gate itself costs 1.9–4.7 ms per layer through 128k (33 ms at 1M), 1.4–1.5$\times$ the first-row fp32 gate at $L \geq 128$k and cheaper than it at 32k. What pooling adds to the acceleration picture is the high-reuse condition: at 87.5% reuse the indexer forward drops a further 1.5–1.9$\times$ below the 75% variant, reaching 4.4$\times$ and 6.9$\times$ under full indexing at 128k and 1M, so the regime where the pooled anchor's quality advantage lives (Table 2) is also the cheapest to serve.

Table 13: Measured indexer forward per layer of the pooled-anchor variants (ORS+PIVOT-reuse, block-JSD admission), Qwen3-32B, single H200, bf16, top-k 4096, block cap 16, fixed production thresholds; iso-reuse arms bisected to realized reuse 0.50 / 0.75 / 0.875 within ± 0.003. At matched reuse the pooled indexer forward equals the first-row one up to the gate (42.8 against 43.2 ms at 128k, 75% reuse), so the attention and prefill columns of Table 12 carry over to the pooled variants at matched reuse (end-to-end shift below 1%); the 87.5% column prices the high-reuse regime the pooled anchor is built for.

	indexer forward (ms/layer)				speedup vs idx	
L	idx	ORS+PIVOT 50%	ORS+PIVOT 75%	ORS+PIVOT 87.5%	75%	87.5%
32k	11.0	8.8	6.5	5.5	1.7×	2.0×
64k	34.6	22.8	15.1	11.0	2.3×	3.1×
128k	122.5	71.6	42.8	28.0	2.9×	4.4×
256k	460	251	140.0	83.3	3.3×	5.5×
512k	1763	935	499	277	3.5×	6.4×
1M	6908	3614	1891	1005	3.7×	6.9×

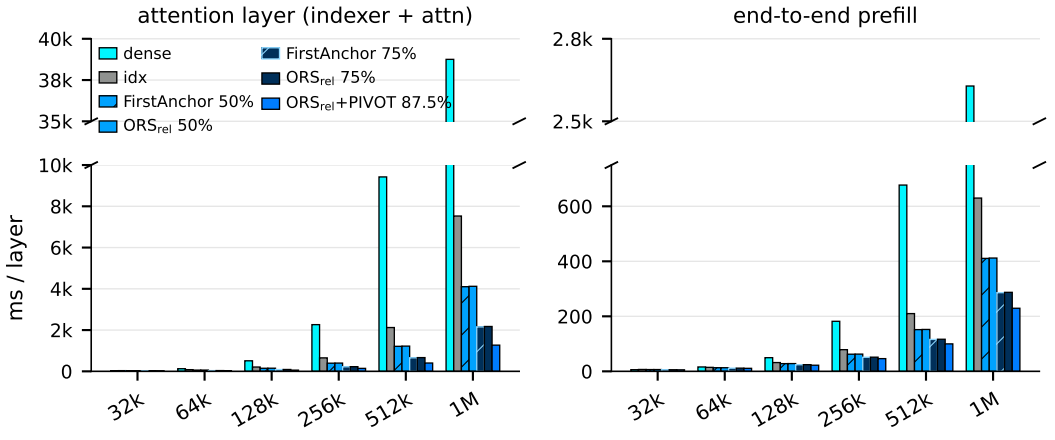


Figure 5: Attention-layer and end-to-end prefill costs of Qwen3-32B across context lengths on the production stack (single H200, bf16, top-k 1024, W=8 trailing window, age cap 8, fixed reuse thresholds). Left: indexer forward plus gathered attention per layer. Right: prefill wall; dense is measured, and the indexed variants are backbone + 64 $\times$ layer computed from the measured per-layer components, tracking the measured end-to-end buckets within 5% at 32k–128k. Cells at 256k and beyond are extrapolated from per-layer microbenchmarks validated within 10% against the measured rows. The y-axes are folded (10k–35k on the left, 750–2500 on the right) to fit the dense 1M column. The ORS$_{rel}$+PIVOT 87.5% series prices the pooled-anchor operating point: its indexer term is measured (block-JSD gate; the indexer totals are top-k-weak), its attention term derives from the measured 50%/75% rows by anchor-fraction scaling, and at matched reuse the pooled variants coincide with the corresponding ORS$_{rel}$ bars.