CARRYKV: LOGARITHMIC BRANCH ROUTING FOR SPARSE ATTENTION WITH ADDITIVE STATES ON A CARRY FOREST

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient hierarchical sparse attention requires identifying useful regions without scanning the full history. We introduce CarryKV, a training-free retriever with compact, mergeable branch summaries. Each branch partitions post-RoPE keys into cells storing counts and key sums; weighted log-sum-exp over cell means estimates attention mass. States merge additively in a persistent hierarchy, and final attention uses original K/V selected at the leaves. We cast HiP's implicit interval pruning and CarryKV's persistent hierarchy in a common bounded-refinement framework, exposing how branch information determines pruning quality. Multi-cell summaries weakly improve the one-sided mass estimate over a single mean, and lower residuals strengthen a same-frontier retained-mass guarantee. The remaining residual has a reverse-KL interpretation; a merge decomposition separates inherited residual from new readout error. Controlled branch experiments measure the resulting mass estimates and pruning decisions across scales. On Qwen3-8B at native 32K, CarryKV improves RULER by 13.37 points over HiP and 6.80 over InfiniteHiP. On Qwen3-1.7B, it outperforms InfiniteHiP on RULER and PG19 from 4K to 128K. At fixed budgets, routing requires $O(\log N)$ work per query block and sparse-prefill retrieval requires $O(N \log N)$ work over a length-N context.

1 INTRODUCTION

Long-context prefill processes a growing number of query–key interactions before generating the first answer token. IO-aware exact attention reduces data movement through tiling and fusion (Dao et al., 2022); training-free sparse prefill selects part of the history to reduce the interactions themselves. Selection must be cheap enough to preserve the computational benefit of sparsity.

Sparse-prefill methods select patterns, blocks or temporal branches (Jiang et al., 2024; Lai et al., 2025; Lee et al., 2025a). HiP (Lee et al., 2025a) scores each branch through its center key and recursively refines a bounded set of survivors, giving logarithmic routing at fixed budgets. Early decisions determine which subtrees reach finer selection, making the information used to value a branch consequential.

CarryKV augments this bounded search with compact, query-independent summaries.[1] A count and key sum preserve the branch's average query–key logit. Multiple conditional means additionally preserve between-group variation before exponentiation, recovering information about attention mass. This progression from a positional representative to a mean and then a multi-cell estimate improves the information available for pruning.

Our default system instantiation, CarryKV-Hash-4, partitions post-RoPE keys by two fixed direction signs. Each of four cells stores a count and full-dimensional key sum; a count-weighted exponential readout estimates branch mass. Matching child statistics add into a persistent carry forest, where completed subtrees merge as in a base-b counter. Bounded retrieval selects original K/V; up to 64 queries share remote indices and compute independent softmax weights.

[1] Code: `https://anonymous.4open.science/r/exp-27C2`.

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.

Contributions. We cast HiP and CarryKV in a common bounded-refinement framework that isolates branch information as a design choice. We introduce mergeable multi-cell branch summaries and a persistent carry forest that supports $O(\log N)$ routing per query block at fixed budgets. Our one-sided mass analysis links representation residuals to same-frontier retained-mass guarantees, interprets the residual through reverse KL, and decomposes inherited and new merge-readout error.

On Qwen3-8B at native 32K (Yang et al., 2025), our default Hash-4 configuration improves RULER (Hsieh et al., 2024) by 13.37 points over HiP and 6.80 over InfiniteHiP.

2 Related Work

Hierarchical sparse retrieval. HiP (Lee et al., 2025a) uses bounded recursive refinement; InfiniteHiP (Lee et al., 2025b) uses parallel staged screening. HISA (Xu et al., 2026) is a related but distinct indexer which filters pooled blocks and then applies an existing token-level sparse indexer within the retained blocks. CarryKV retains bounded refinement with summary-guided scoring. Appendix B compares the routing costs of HiP, InfiniteHiP, and CarryKV under the evaluated configurations.

Sparse prefill selection. MInference (Jiang et al., 2024) and FlexPrefill (Lai et al., 2025) construct input-dependent patterns; XAttention (Xu et al., 2025) scores antidiagonal samples. FlashPrefill (Fan et al., 2026b) and FlashPrefill V2 (Fan et al., 2026a) use dynamic thresholding for sparse prefill. NSA (Yuan et al., 2025) and MoBA (Lu et al., 2025) learn compression or routing; CarryKV builds summaries from pretrained keys without learning a router.

Additive states and mass summaries. Performer/FAVOR+ (Choromanski et al., 2021) approximates the softmax kernel with additive random-feature sums. COBS (Tian et al., 2026) estimates block mass from means and compressed covariance in NSA. Log-Linear Attention (Guo et al., 2026) organizes recurrent hidden states in a Fenwick hierarchy to improve the representation capacity of linear attention. CarryKV's additive cell statistics index original K/V for sparse softmax attention; Appendix A.7 compares readout and maintenance costs.

Decode retrieval and output correction. Quest (Tang et al., 2024) uses page bounds, RetrievalAttention (Liu et al., 2025a) uses vector search, and ClusterKV (Liu et al., 2025b) uses semantic groups. ParisKV (Qi et al., 2026) combines collision voting and reranking for decode retrieval/offload. For output correction, MagicPIG (Chen et al., 2025) studies sampling, Twilight (Lin et al., 2025) adapts support to cumulative attention weight, and Delta Attention (Willette et al., 2025) propagates sampled dense-minus-sparse residuals. CarryKV's base operator uses uncorrected attention on selected original K/V.

3 Bounded Hierarchical Retrieval

We compare HiP's implicit interval hierarchy and CarryKV's persistent carry forest through a common bounded-refinement framework, focusing on the information available for branch valuation.

3.1 A common bounded-pruning framework

A branch u is a temporal interval. A frontier contains the intervals currently available for selection. A scorer values each branch from the query or query block and the information exposed by its method. A budget B retains a bounded subset; survivors refine until a method-specific terminal resolution supplies original K/V. Direct support (e.g., sink/local positions) joins these regions for causal attention. This abstraction separates search decisions from hierarchy storage.

HiP (Lee et al., 2025a) maintains temporal intervals, bisects survivors, scores the candidates through their center keys, and prunes back to the budget. Bisection generates transient branches without persistent numerical node states. Center-1 (HiP) isolates HiP's center-key branch valuation on fixed candidates; full-system comparisons use HiP. CarryKV augments structural metadata with key summaries, moving from the center representative to Mean-1 and then a multi-cell mass estimate (Figure 1).

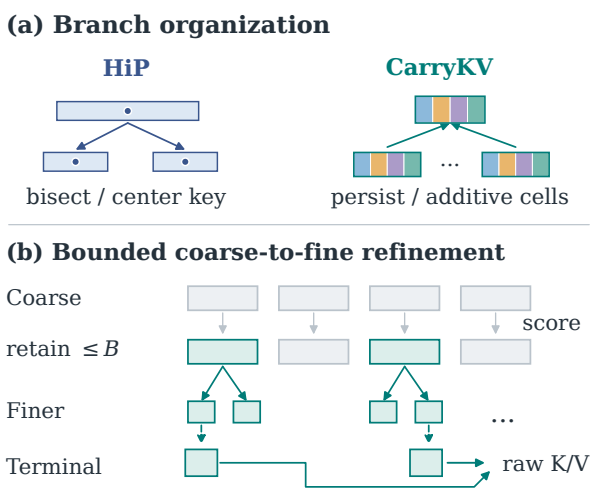


Figure 1: Common bounded hierarchical retrieval. (a) HiP bisects transient intervals and reads center keys; CarryKV persists additive summaries and forms parents by carry merges. (b) At each resolution, branch information scores the frontier, at most B intervals survive, and only survivors refine toward terminal regions and original K/V.

3.2 Branch attention mass as the routing target

For one query at position t, the legal prefix is $\{0, \dots, t\}$. For head dimension d, let $q, k_i \in \mathbb{R}^d$ be post-normalization, post-RoPE queries and keys, and let v_i be the unrotated values. With Q/K normalization, branch valuation is naturally expressed in the resulting query–key score space; we therefore formulate it directly in terms of logits and attention mass. A nonempty legal branch u contains n_u keys with logits $s_i = q^\top k_i/\sqrt{d}$. Its size-normalized log-mass and raw mass are

$$\mathcal{M}(q,u) = \log\left(\frac{1}{n_u}\sum_{i\in u} e^{s_i}\right), \qquad W_u = n_u e^{\mathcal{M}(q,u)}. \tag{1}$$

Raw mass adds across disjoint regions and ranks equal-span branches identically to $\mathcal{M}(q,u)$. Appendix E revisits the distance-locality assumptions underlying center-based representatives in this normalized score regime.

For Q/K-only support selection, retaining more attention probability tightens the worst-case output guarantee. On the legal prefix define $p_i = e^{s_i}/\sum_{r=0}^{t} e^{s_r}$. A nonempty final support S retains probability $P_S = \sum_{i\in S} p_i$; dense and renormalized sparse outputs are $o = \sum_{i=0}^{t} p_i v_i$ and $o_S = \sum_{i\in S}(p_i/P_S)v_i$. With values bounded by $\|v_i\| \le V_{\max}$,

$$\|o - o_S\| \le 2V_{\max}(1 - P_S). \tag{2}$$

The bound is attainable: maximizing retained probability minimizes the worst-case error in this operator class. Actual error also depends on value directions. For $0 < P_S < 1$, let $\bar{v}_S$ and $\bar{v}_{\bar{S}}$ be the conditional attention-weighted value means inside and outside S. Then $o - o_S = (1-P_S)(\bar{v}_{\bar{S}} - \bar{v}_S)$; at $P_S = 1$ the outputs agree. Appendix A.1 supplies the proof.

3.3 Same-frontier selection quality

For a fixed query/head, or a query block/head with a shared branch score, let $\mathcal{C}$ be a frontier of disjoint legal branches with equal spans and token costs, outside any fixed guaranteed support. For a block $\mathcal{Q}$, $W_u = \sum_{q\in\mathcal{Q}}\sum_{i\in u} e^{q^\top k_i/\sqrt{d}}$ is its joint raw mass; a single query recovers Eq. (1). For $1 \le B \le |\mathcal{C}|$, let $S^\star$ select the B largest exact raw masses W_u, and let $\widehat{S}$ be the approximate scorer's top-B set. Their same-frontier mass ratio is

$$Q_{\mathcal{C}} := \frac{\sum_{u\in\widehat{S}} W_u}{\sum_{u\in S^\star} W_u}. \tag{3}$$

$Q_{\mathcal{C}} = 1$ matches exact top-B mass on the same frontier; $1 - Q_{\mathcal{C}}$ is selection loss, computed offline from exact branch masses. For the one-sided Mean/multi-cell estimators below, Section 4.4 gives a retained-mass guarantee in terms of branch residuals; Appendix A.10 also treats arbitrary signed scorer errors.

4 CARRYKV

Within this framework, CarryKV replaces HiP's center-based branch valuation with a mass estimate from compact key statistics. A single mean provides the additive starting point; Hash-4 retains four conditional means and their counts. These states compose in a persistent carry forest, whose bounded traversal selects original K/V for block-shared sparse prefill.

4.1 FROM POSITIONAL REPRESENTATIVES TO MASS-GUIDED SUMMARIES

Mean-1 uses a simple additive summary for branch mass: a count and a full-dimensional key sum. For branch u, these are n_u and $S_u = \sum_{i \in u} k_i$. The arithmetic mean $\bar{k}_u = S_u/n_u$ gives the mean logit $\mu_u = q^\top \bar{k}_u/\sqrt{d} = n_u^{-1} \sum_{i \in u} s_i$. A center representative scores one key; this mean incorporates every key's contribution to the average logit, and disjoint branches combine by adding counts and sums. Averaged over a uniformly sampled positional representative, replacing one position by the branch mean removes the positional-variance term from squared log-mass error (Appendix A.2). This average-case argument motivates mean-based valuation; Section 5.2 evaluates the deterministic HiP center empirically.

Exponentiation makes branch mass sensitive to variation around that average. Define the Jensen gap $J_u = \mathcal{M}(q, u) - \mu_u \geq 0$. It is exactly the log-mass information lost when all logits in the branch are replaced by their mean. This gap motivates retaining several conditional means before exponentiation.

4.2 MULTI-CELL ADDITIVE BRANCH STATE

The default Hash-4 instantiation uses two fixed directions $r_1, r_2 \in \mathbb{R}^d$. The pair of signs of $r_1^\top k_i$ and $r_2^\top k_i$ assigns each post-RoPE key to one of four cells. Directions and sign rules are query-independent and shared across nodes that can merge. In branch u, cell u_j stores count $n_{u,j} = |u_j|$ and full-dimensional key sum $S_{u,j} = \sum_{i \in u_j} k_i$. For nonempty cells, the centroid $\bar{k}_{u,j} = S_{u,j}/n_{u,j}$ is the arithmetic mean, without re-normalization. Let $\mathcal{P}$ denote the cell partition and $w_{u,j} = n_{u,j}/n_u$ its count weights. The weighted log-sum-exp (LSE) readout is

$$\widehat{\mathcal{M}}_{\mathcal{P}}(q, u) = \log \sum_j w_{u,j} \exp\left(q^\top \bar{k}_{u,j}/\sqrt{d}\right). \tag{4}$$

Empty cells contribute zero. Several conditional means preserve between-cell variation that a single mean averages away; the count weights preserve token multiplicity in the estimated exponential mass.

For a parent branch u with disjoint children v, compatible fixed cell labels give the composition rule

$$(n_{u,j}, S_{u,j}) = \sum_{v \in \text{children}(u)} (n_{v,j}, S_{v,j}). \tag{5}$$

Fixed cell identities make Hash-4 directly composable across levels; each parent retains its children's four-cell format. Cells provide scoring resolution inside one candidate; tree children are independently prunable candidates. Figure 2 connects this state to retrieval and raw attention. Matched-capacity partitions isolate grouping allocation (Section 5.2; Appendices A.10 and C.5).

4.3 PERSISTENT CARRY FOREST AND BOUNDED ROUTING

Leaves of span C form a carry forest with branching factor b. Whenever b adjacent completed roots at one level are available, they merge into a parent, leaving at most $b - 1$ residual roots per level. Their counts are the base-b digits of the completed-leaf count. Appending keys closes leaves and propagates merges; completed summaries remain immutable and available. Parallel prefill builds the same closed-node statistics; beam B bounds the retained frontier.

For one head, s queries in block $\mathcal{Q}$ share remote selection. The first query fixes the completed remote prefix legal for every row; all valid queries score a candidate branch u by

$$\widehat{\mathcal{M}}_{\mathcal{Q}}(u) = \log \left[\frac{1}{s} \sum_{q \in \mathcal{Q}} \sum_j \frac{n_{u,j}}{n_u} e^{q^\top \bar{k}_{u,j}/\sqrt{d}} \right]. \tag{6}$$

The sum runs over nonempty cells and all valid queries, including later rows. Equal-span branches rank by estimated joint raw mass; the common factor sn_u cancels. Appendix A.1 gives the joint-index interpretation.

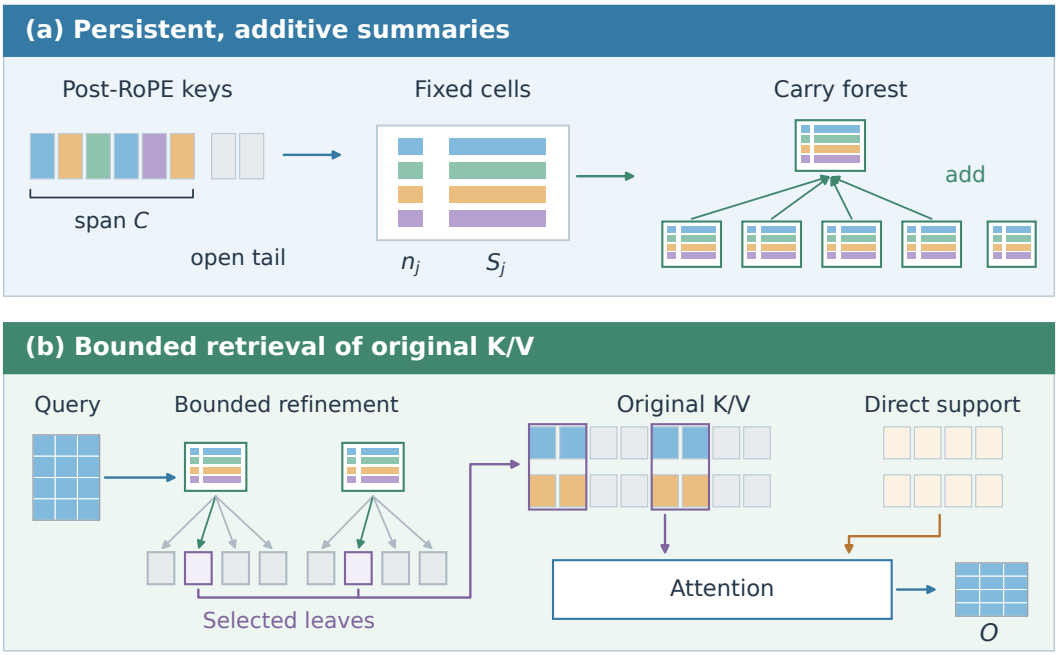


Figure 2: CarryKV overview. Fixed-cell summaries (n_j, S_j) are maintained additively in a persistent forest. A query block uses these summaries for bounded refinement; selected leaves resolve to original K/V and combine with direct support in one row-wise softmax.

Traversal expands survivors and adds residual roots at each level, scoring only frontiers larger than B. Selected leaves and raw tail positions enter the final enumeration, with optional truncation to a configured remote-token cap for budget alignment. Each query row adds its legal sink and local positions and deduplicates the union. Final attention uses a single row-wise softmax over the selected original K/V. Grouped-query attention (GQA) shares KV storage; query heads select independently.

Algorithm 1 in Appendix B.1 gives the traversal and final support assembly for one query block/head.

Complexity. For history length N, the carry forest has $O(\log N)$ levels. At fixed beam, branching factor, cell capacity, head dimension and query-block width, routing costs $O(\log N)$ per block/head and $O(N \log N)$ over a length-N sparse prefill. Summary construction and storage are linear in N. Appendix B derives frontier width, final-support costs, storage and online-update bounds.

4.4 Analysis of the multi-cell branch estimator

The multi-cell estimator admits a direct decomposition of the mass recovered over Mean-1 and the error propagated through merges. For branch u and query q, write exact and estimated log-mass as $\mathcal{M}_u = \mathcal{M}(q,u)$ and $\widehat{\mathcal{M}}_u = \widehat{\mathcal{M}}_{\mathcal{P}}(q,u)$, with mean logit μ_u. Define the mean gap $J_u = \mathcal{M}_u - \mu_u \geq 0$, recovered gap $G_u = \widehat{\mathcal{M}}_u - \mu_u$, and residual $e_u = \mathcal{M}_u - \widehat{\mathcal{M}}_u$. They satisfy

$$J_u = G_u + e_u. \tag{7}$$

Here G_u measures exponential-mass information recovered by grouping, and e_u measures remaining within-cell approximation error. Appendix A.3 relates these measured gaps to query-projected variation.

Proposition 1 (Multi-cell lower bound and refinement). For finite keys,

$$\mu_u \leq \widehat{\mathcal{M}}_{\mathcal{P}}(q,u) \leq \mathcal{M}_u, \qquad 0 \leq e_u \leq J_u. \tag{8}$$

For the same branch and query, grouping weakly reduces underestimation by exactly $G_u = J_u - e_u$. Thus any fixed nonnegative normalized weighting π over branch–query pairs, with finite expectations, satisfies

$$\mathbb{E}_\pi[G_u] = \mathbb{E}_\pi[J_u] - \mathbb{E}_\pi[e_u] \geq 0. \tag{9}$$

For a nested refinement $\mathcal{P}'$, $e_u^{\mathcal{P}'} \leq e_u^{\mathcal{P}}$: this is the theoretical counterpart of the Mean-1/Hash-2/4/8 capacity comparison (proof: Appendix A.7).

Corollary 1 (Residual-to-selection guarantee). On the fixed equal-span, equal-cost frontier of Eq. (3), define exact-mass weights $\alpha_u = W_u / \sum_{v \in S^\star} W_v$ for $u \in S^\star$. Their weighted residuals are $\bar{e}_{\mathcal{C}}^\star = \sum_{u \in S^\star} \alpha_u e_u$ and $\bar{J}_{\mathcal{C}}^\star = \sum_{u \in S^\star} \alpha_u J_u$, with $\bar{e}_{\mathcal{C}}^\star \leq \bar{J}_{\mathcal{C}}^\star$ for multi-cell states. For any one-sided estimator $\widehat{\mathcal{M}}_u = \mathcal{M}_u - e_u$ with $e_u \geq 0$,

$$Q_{\mathcal{C}} \geq \sum_{u \in S^\star} \alpha_u e^{-e_u} \geq \exp\left(- \sum_{u \in S^\star} \alpha_u e_u \right) = e^{-\bar{e}_{\mathcal{C}}^\star}. \tag{10}$$

Lower branch residuals strengthen this same-frontier retained-mass guarantee (proof: Appendix A.10). Since Mean-1 uses $e_u = J_u$, multi-cell summaries give $Q_{\mathcal{C}} \geq e^{-\bar{e}_{\mathcal{C}}^\star} \geq e^{-\bar{J}_{\mathcal{C}}^\star}$; nested refinements strengthen the guarantee again. The same result applies to the joint query-block mass in Eq. (6) (joint-index form: Appendix A.1). In the limiting case of constant logits within each occupied cell, $e_u = 0$ and the full Mean-1 gap is recovered.

Lemma 1 (Cell-uniform projection). Within branch u, let $p_i = e^{s_i} / \sum_{r \in u} e^{s_r}$ for logits $s_i = q^\top k_i / \sqrt{d}$. Let $\mathcal{Q}_{\mathcal{P}}$ contain distributions uniform within each cell, with free cell weights. Then

$$e_u = \min_{r \in \mathcal{Q}_{\mathcal{P}}} \mathrm{KL}(r \| p). \tag{11}$$

The residual is the reverse-KL cost of cell-uniform attention, a specialization of the variational log-partition principle (Blei et al., 2017) (Appendix A.2).

CarryKV repeatedly scores nodes formed by merging lower-level summaries, so the local residual must remain interpretable after composition. For one merge, let p be a parent and v range over its disjoint children. Using estimated raw mass $\widehat{W}_u = n_u e^{\widehat{\mathcal{M}}_u}$ for each node, define normalized child-mass weights $\beta_v = \widehat{W}_v / \sum_z \widehat{W}_z$ and the new merge-readout increment $\Delta_p^{\mathrm{merge}} = \log(\sum_v \widehat{W}_v / \widehat{W}_p)$; both sums run over children.

Proposition 2 (Merge error propagation). For compatible fixed cells,

$$e_p = \Delta_p^{\mathrm{merge}} + \log \sum_v \beta_v e^{e_v}. \tag{12}$$

The log-sum term aggregates inherited child residuals; $\Delta_p^{\mathrm{merge}} \geq 0$ is the new merge-readout error. Section 5.2 measures both terms to test how much richer child information survives merging. Appendix A.4 gives the proof, raw-mass deficit, finite-range bounds and full-tree accounting.

5 Experiments

Setup. We evaluate Qwen3-1.7B-Base and Qwen3-8B-Base in BF16 (Yang et al., 2025). Outside controlled representation comparisons, CarryKV defaults to Hash-4, with 64-query blocks (Q64), beam $B = 64$, leaf span $C = 16$, branching factor $b = 4$ and a 1,020-token remote cap, plus four sink tokens and a 320-token local window. HiP (Lee et al., 2025a) uses 32-query blocks (Q32) and two-token key blocks; InfiniteHiP (Lee et al., 2025b) uses landmark-on staged pruning with 64-query blocks. All methods retain the model's Q/K normalization, RoPE and GQA. We use sparse prefill/dense decode (S_p/D) without output correction. Native context is 32,768; 64K/128K use static YaRN RoPE scaling by 2/4 (Peng et al., 2024). Budgets, samples and scoring: Appendix C.

5.1 RULER LONG-CONTEXT EVALUATION

We evaluate retrieval, aggregation and question answering on all 13 RULER tasks (Hsieh et al., 2024) using the official task metrics; higher task-macro scores are better. At 8B/32K, CarryKV gains 13.37 points over HiP (95% paired interval [11.11, 15.69]) and 6.80 over InfiniteHiP ([4.66, 8.99]). Table 1 shows gains over InfiniteHiP at both longer 8B contexts and every 1.7B length. The 1.7B gap grows from 3.08 points at 4K to 21.32 at 64K, then falls to 3.12 at 128K. At 128K, all methods lose accuracy; CarryKV–HiP differences are -0.23 points on 1.7B and -0.04 on 8B, with paired intervals spanning zero.

At 8B/32K, CarryKV gains 9.8 variable-tracking points over InfiniteHiP and loses 12 and 9 points on the second and third single-needle tasks. Appendix C.7 gives all task scores.

Table 1: RULER macro (↑), S_p/D, no output correction. Budget composition: Appendix C.3. All methods share examples: 200/task at 1.7B/4K–16K, 100/task otherwise. 64K/128K use YaRN 2/4. Bold: best displayed sparse result.

Qwen3-Base	Context	Dense	HiP	InfiniteHiP	CarryKV
1.7B	4K	90.04	77.68	81.39	**84.47**
	8K	83.31	61.09	64.75	**67.34**
	16K	75.95	50.51	49.54	**59.89**
	32K	71.45	43.73	30.14	**46.58**
	64K	58.87	27.39	10.09	**31.40**
	128K	35.74	**5.53**	2.18	5.29
8B	32K	88.28	55.80	62.36	**69.16**
	64K	74.32	42.26	45.48	**51.76**
	128K	58.29	**27.44**	24.73	27.41

5.2 BRANCH REPRESENTATION ANALYSIS

Capacity across span. Fixed-window fidelity measures how often an estimated branch ordering agrees with exact mass; each window of length L contains two $L/2$-token branches. Figure 3(a) compares Center-1 (HiP) with CarryKV's Mean-1 and nested Hash-2/4/8 on final-query Dense states from 260 native-32K prompts. At $L = 512$, Mean-1 raises fidelity from 60.70% to 81.22% over Center-1. At $L = 16K$, Mean-1, Hash-2, Hash-4 and Hash-8 achieve 48.91%, 49.83%, 55.10% and 60.89%. Hash-4 gains 6.18 points over Mean-1 with half Hash-8's cell state and centroid dot products. The largest-window Hash-4–Mean-1 gains on 8B are 1.15/1.13/0.50 points at 32K/64K/128K (Appendix A.11). Additional cells improve coarse-window branch ordering, with a smaller gain on the larger model.

Actual recursive frontiers. On competitive Q64 frontiers, $Q_{\mathcal{C}}$ compares selected joint mass to exact top-64 mass on the same candidates. At spans 16/64/256, Mean-1 raises $100Q_{\mathcal{C}}$ over Center-1 by 16.81/16.56/6.84 points. On branches sampled from these frontiers, using the block's last valid query, Hash-4 recovers mean fractions G/J of 15.61%/10.01%/7.20% of Mean-1's gap (Appendix A.8). This measures the branch-level residual reduction in Section 4.4. The one-sided guarantee also applies to joint Q64 mass (Appendix A.1); on that objective, Hash-4 improves $100Q_{\mathcal{C}}$ over Mean-1 by 0.98/0.18/0.16 points, with positive paired intervals (Table 12). Independent complete searches measure the final support's probability P_S over the full legal prefix and output error $\|o_S - o\|_2/\|o\|_2$. Relative to Mean-1, Hash-4 raises P_S by 0.020 percentage points and lowers this normalized error by 0.0086. Grouping therefore improves local choices and final support on these states. Appendix C.2 gives scorer trajectories and target-path survival. Hash-4's mean frontier gain over Mean-1 remains positive at all three spans across eight fixed direction tables (Appendix A.12).

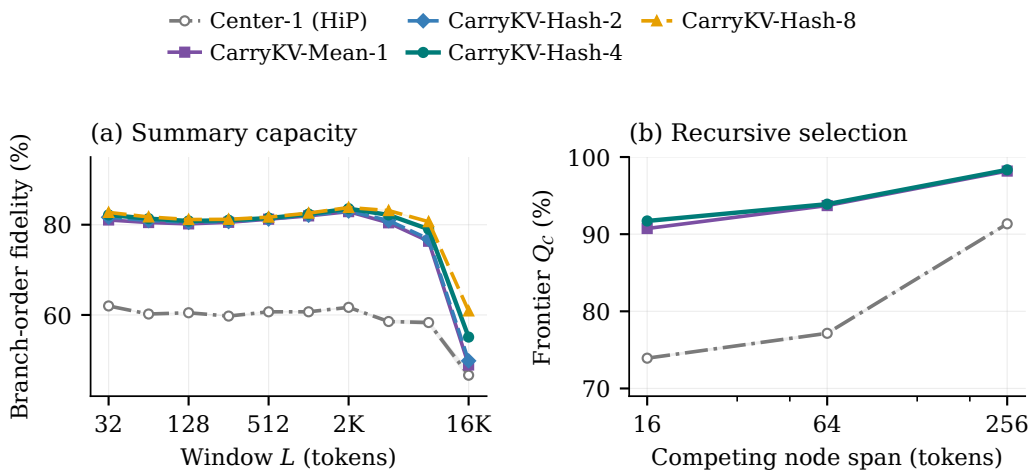


Figure 3: Branch valuation on Qwen3-1.7B Dense-32K states. (a) Fixed-window fidelity on 260 prompts: each window compares two equal halves. (b) Selected/exact top-64 joint Q64 mass, Q_C, on actual Hash-4 frontiers from 234 held-out prompts. Higher is better in both panels. Bands are 95% within-task prompt-bootstrap intervals. Sampling and node/window spans: Appendix C.1.

Information preserved across merges. Proposition 2 separates inherited child error from new merge-readout error. At parent spans 64/256, Hash-4 lowers inherited error relative to Mean-1 by 0.282/0.286 nats, while new merge-readout error increases by 0.079/0.091. Total parent error therefore falls by 0.203/0.195 nats. The paired intervals preserve these signs (Appendix A.8): richer child information survives composition strongly enough to improve the parent estimate.

Allocation at fixed capacity. Temporal-4 groups contiguous quarters; Hash-4 groups key directions. With four cells, Temporal-4 retains more mass on the measured frontiers and exceeds Hash-4's 8B RULER score by 4.26/3.52 points at 32K/64K. Both exceed Mean-1. Hash-4 has lower fixed-state output error on the sampled 1.7B searches, consistent with output error depending on value directions as well as retained mass (Appendix A.1). These comparisons identify capacity and grouping allocation as separate design dimensions (Appendix C.5). Hash-4 keeps post-RoPE directional labels across levels; Temporal-4 uses the four child means at each base-four parent (Appendix A.6).

5.3 Downstream Generation and Language Modeling

We evaluate language modeling on PG19's official 100-book test stream (Rae et al., 2020) using token-weighted perplexity (PPL, lower is better). Following HiP (Lee et al., 2025a), we score shifted labels in disjoint full windows with block-shared selection; later block queries participate in selecting earlier rows' support (Appendix C.4). Table 2 shows lower CarryKV PPL than InfiniteHiP at every 1.7B length and through 64K on 8B. At 8B/128K, InfiniteHiP reaches 10.1509, CarryKV 11.1343 and HiP 11.5548.

LongBench-E provides an auxiliary generation comparison (Bai et al., 2024), with a 1.25-point CarryKV–InfiniteHiP gain on 8B Full inputs and a paired interval spanning zero (Appendix C.4).

Table 2: PG19 on the 100-book test stream and LongBench-E on 20 common examples/task/view. Full inputs: median 4,027, maximum 10,337 tokens. Bold: best sparse result. Protocol: Appendix C.4.

Context / input view	Dense	HiP	InfiniteHiP	CarryKV
Qwen3-1.7B: PG19, PPL ↓				
4K	13.01	13.10	13.09	**13.02**
8K	12.50	12.73	12.73	**12.57**
16K	12.13	12.56	12.56	**12.29**
32K	11.85	12.51	12.57	**12.15**
64K	11.98	13.14	13.67	**12.58**
128K	16.56	26.29	32.06	**24.78**
Qwen3-8B: PG19, PPL ↓				
4K	9.64	9.69	9.67	**9.64**
8K	9.28	9.39	9.36	**9.29**
16K	9.00	9.24	9.21	**9.07**
32K	8.80	9.20	9.23	**8.95**
64K	8.79	9.39	9.49	**9.05**
128K	10.28	11.55	**10.15**	11.13
LongBench-E, 13-task macro ↑				
1.7B, 4K cap	46.38	46.02	44.12	**46.08**
1.7B, 8K cap	48.32	46.43	45.88	**47.04**
1.7B, Full	48.27	46.64	45.89	**47.07**
8B, Full	56.59	54.78	54.85	**56.10**

5.4 Efficiency

We measure complete prefill-attention latency from post-RoPE Q/K/V inputs to the attention output before W_O, including construction, routing, support assembly, and raw attention. Dense uses FlashAttention-4 (FA4) (Zadouri et al., 2026). HiP provides the closest logarithmic hierarchical-kernel comparison, while InfiniteHiP serves as the GPU-efficiency reference. Appendix D specifies all included costs and execution details.

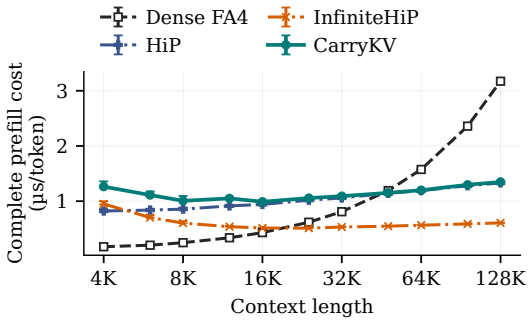


Figure 4: Complete prefill cost: BF16 Qwen3-8B dimensions, batch one, RTX PRO 6000 Blackwell, no sink/local additions or output correction. Medians and 10th–90th percentiles of 36 interleaved calls. Absolute costs and protocol: Appendix D.

At 32K, CarryKV takes 35.75 ms, 3.04% above HiP's 34.70 ms median; InfiniteHiP takes 17.43 ms. CarryKV crosses below Dense between 32K and 48K and stays close to HiP through 128K. InfiniteHiP remains faster throughout.

6 Conclusion

CarryKV uses compact, mergeable multi-cell branch summaries and an additively maintained persistent carry forest for fixed-budget $O(\log N)$ routing per query block, followed by sparse softmax on selected original K/V. Experiments show branch-selection and downstream quality gains; complete-operator timings characterize the current implementation's cost. Our experiments focus on Qwen3 Base and the current GPU stack; broader model coverage would test generality, while tighter retrieval–kernel co-design could further improve systems efficiency.

AI USE STATEMENT

AI tools assisted literature search, research coding, derivation development and checking, experiment orchestration, data analysis and manuscript preparation. The authors verified the scientific and bibliographic content and take responsibility for the final submission.

REPRODUCIBILITY STATEMENT

Proofs and algorithm specifications are provided in Appendices A and B. Appendix C details data processing and evaluation settings; Appendix D describes the implementation and timing protocol. Code is available in the anonymous repository.

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, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6(58):1705–1749, 2005. URL https://jmlr.org/papers/v6/banerjee05b.html.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773. URL https://doi.org/10.1080/01621459.2017.1285773.

Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pp. 380–388, 2002. doi: 10.1145/509907.509965. URL https://doi.org/10.1145/509907.509965.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/6d50d824ae819d5a961c1d8edc15e833-Abstract-Conference.html.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract.html.

Qihang Fan, Huaibo Huang, Zhiying Wu, Bingning Wang, and Ran He. FlashPrefill V2: Block-sparse prefill attention for long-context LLM serving. *arXiv preprint arXiv:2608.19758*, 2026a. URL https://arxiv.org/abs/2608.19758.

Qihang Fan, Huaibo Huang, Zhiying Wu, Juqiu Wang, Bingning Wang, and Ran He. FlashPrefill: Instantaneous pattern discovery and thresholding for ultra-fast long-context prefilling. *arXiv preprint arXiv:2603.06199*, 2026b. URL https://arxiv.org/abs/2603.06199.

Han Guo, Songlin Yang, Tarushii Goel, Eric P. Xing, Tri Dao, and Yoon Kim. Log-linear attention. In *International Conference on Learning Representations*, 2026.

URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/939a9a2604f13729a4ec36ad25d10c60-Abstract-Conference.html.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, 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, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5dfbe6f5671e82c76841ba687a8a9ecb-Abstract-Conference.html.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. FlexPrefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/03645743ea35690f30d795d6bac149a5-Abstract-Conference.html.

Heejun Lee, Geon Park, Youngwan Lee, Jaduk Suh, Jina Kim, Wonyong Jeong, Bumsik Kim, Hyemin Lee, Myeongjae Jeon, and Sung Ju Hwang. A training-free sub-quadratic cost transformer model serving framework with hierarchically pruned attention. In *International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/043f0503c4f652c737add3690aa5d12c-Abstract-Conference.html.

Heejun Lee, Geon Park, Jaduk Suh, and Sung Ju Hwang. InfiniteHiP: Extending language model context up to 3 million tokens on a single GPU. *arXiv preprint arXiv:2502.08910*, 2025b. URL https://arxiv.org/abs/2502.08910.

Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. Twilight: Adaptive attention sparsity with hierarchical top-p pruning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/ad2f57116d62a9d8dcfaba6a168715e4-Abstract-Conference.html.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. RetrievalAttention: Accelerating long-context LLM inference via vector retrieval. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4e36d4049fb0fea195a8267c8dcd0824-Abstract-Conference.html.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. ClusterKV: Manipulating LLM KV cache in semantic space for recallable compression. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7, 2025b. doi: 10.1109/DAC63849.2025.11132479. URL https://doi.org/10.1109/DAC63849.2025.11132479.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19eae75beed66321d62272e794a9c2ac-Abstract-Conference.html.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/874a4d89f2d04b4bcf9a2c19545cf040-Abstract-Conference.html.

Yanlin Qi, Xinhang Chen, Huiqiang Jiang, Qitong Wang, Botao Peng, and Themis Palpanas. ParisKV: Fast and drift-robust KV-cache retrieval for long-context LLMs. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=wxD4wTYQXt.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. SparQ attention: Bandwidth-efficient LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 42558–42583, 2024. URL https://proceedings.mlr.press/v235/ribar24a.html.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/1e027da6bec9ceb2ec37951ceeccae93-Abstract-Conference.html.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Alexander Tian, Aditya Ghai, Sanjit Neelam, Zaal Vasania, and Akshay Mishra. COBS: Cumulant order block sparse attention. *arXiv preprint arXiv:2607.09052*, 2026. URL https://arxiv.org/abs/2607.09052.

Jeffrey Willette, Heejun Lee, and Sung Ju Hwang. Delta attention: Fast and accurate sparse attention inference by delta correction. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0403. URL https://papers.neurips.cc/paper_files/paper/2025/hash/11cc3abb01080339ceed66eff7c93920-Abstract-Conference.html.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. XAttention: Block sparse attention with antidiagonal scoring. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 69819–69831, 2025. URL https://proceedings.mlr.press/v267/xu25ag.html.

Yufei Xu, Fanxu Meng, Fan Jiang, Yuxuan Wang, Ruijie Zhou, Zhaohui Wang, Jiexi Wu, Zhixin Pan, Xiaojuan Tang, Wenjie Pei, Tongxuan Liu, Di Yin, Xing Sun, and Muhan Zhang. HISA: Efficient hierarchical indexing for fine-grained sparse attention. *arXiv preprint arXiv:2603.28458*, 2026. URL https://arxiv.org/abs/2603.28458.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

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, 2025. URL https://aclanthology.org/2025.acl-long.1126/.

Ted Zadouri, Markus Hoehnerbach, Jay Shah, Vijay Thakkar, and Tri Dao. FlashAttention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling. In *Proceedings of Machine Learning and Systems*, volume 8, pp. 912–926, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/ae8b0b5838ba510daff1198474e7b984-Abstract-Conference.html.

APPENDIX

A BRANCH APPROXIMATION AND SELECTION

This appendix links Section 3's target to the information retained by a branch summary. We derive the raw-attention output relation, then prove the partition projection and same-frontier selection results alongside their matching measurements. All fixed-window figures use the protocol in Appendix C.6.

A.1 ATTENTION MASS AND RE-NORMALIZED OUTPUTS

Fix finite Q/K logits on a nonempty legal prefix and let $p_i = e^{s_i}/\sum_r e^{s_r}$. A nonempty support S retains probability $P_S > 0$. Exact sparse softmax gives $\tilde{p}_i = p_i/P_S$ on S and zero elsewhere. Both the on-support and off-support sums of absolute differences equal $1 - P_S$, so total variation is

$$\mathrm{TV}(p, \tilde{p}) = 1 - P_S, \qquad \|o - o_S\| \le 2V_{\max}(1 - P_S) \tag{13}$$

when $\|v_i\| \le V_{\max}$. For $P_S < 1$, writing conditional attention-weighted means as $\bar{v}_S$ and $\bar{v}_{\bar{S}}$ gives $o - o_S = (1 - P_S)(\bar{v}_{\bar{S}} - \bar{v}_S)$. In a nonzero normed value space, choose a unit vector u and set values to $-V_{\max}u$ inside S and $+V_{\max}u$ outside. These values attain the bound in Eq. (2). When $P_S = 1$ or $V_{\max} = 0$, both the error and the bound are zero. For the value matrix $V = (v_i)_i$ and any nonempty admissible family $\mathcal{F}$ of nonempty supports chosen from Q/K alone,

$$\min_{S \in \mathcal{F}} \sup_{\|v_i\| \le V_{\max}} \|o(V) - o_S(V)\| = 2V_{\max}\left(1 - \max_{S \in \mathcal{F}} P_S\right). \tag{14}$$

Eq. (14) shows that maximizing retained mass minimizes worst-case output error in this fixed-state, truncated-and-renormalized operator class. Equal-cost disjoint leaf selection uses the sum of raw mass in each leaf. For a particular value matrix, the conditional value difference determines the realized error. Related output-approximation perspectives include retained-weight pruning (Lin et al., 2025) and value-aware importance sampling (Chen et al., 2025).

Task-specific evidence and shared queries. Retained attention mass is a Q/K-only routing surrogate; task-specific relevance can follow a different distribution. For a Q64 block $\mathcal{Q}$, write $s_{q,i} = q^\top k_i/\sqrt{d}$. Selecting shared remote support S maximizes the joint raw mass $\sum_{q \in \mathcal{Q}} \sum_{i \in S} e^{s_{q,i}}$. Let Z_q be each query's legal softmax normalizer and $P_q(T) = Z_q^{-1} \sum_{i \in T} e^{s_{q,i}}$ its probability on a legal set T. Let U_q be its fixed guaranteed support and S a shared remote support legal for every query and disjoint from each U_q. Set $a_q = Z_q/\sum_{r \in \mathcal{Q}} Z_r$. The weighted sum of per-query output-error bounds is

$$\begin{aligned} &\sum_q a_q \, 2V_{\max}\big[1 - P_q(U_q \cup S)\big] \\ &\qquad = 2V_{\max} - \frac{2V_{\max}}{\sum_q Z_q}\left[\sum_q \sum_{i \in U_q} e^{s_{q,i}} + \sum_q \sum_{i \in S} e^{s_{q,i}}\right]. \end{aligned} \tag{15}$$

The guaranteed-support term is fixed, so maximizing joint raw mass minimizes the weighted bound in Eq. (15). Eq. (6) approximates the variable term with cell counts and means. Uniform query weights give $\sum_q P_q(S)$ as the selection objective. With a common legal key universe and identical final support for every query, the opposing-value construction above attains all per-query bounds simultaneously.

Joint-index projection. For one head, valid queries $\mathcal{Q}$ and a common legal node u, define $p_{q,i} = e^{s_{q,i}}/Z$, with $Z = \sum_{q \in \mathcal{Q}} \sum_{i \in u} e^{s_{q,i}}$, on joint indices. In cell u_j, write $n_j = |u_j|$ and $\mu_{q,j} = n_j^{-1} \sum_{i \in u_j} s_{q,i}$. Partition joint indices into nonempty cells $\{q\} \times u_j$, with $r_{q,i} = a_{q,j}/n_j$, $a_{q,j} \ge 0$ and $\sum_{q,j} a_{q,j} = 1$. Applying the projection calculation of Eq. (11) to joint indices gives

$$\min_r \mathrm{KL}(r\|p) = \log Z - \log \sum_{q,j} n_j e^{\mu_{q,j}} = \mathcal{M}_{\mathcal{Q}} - \widehat{\mathcal{M}}_{\mathcal{Q}}, \tag{16}$$

In Eq. (16), $\mathcal{M}_{\mathcal{Q}} = \log[Z/(|\mathcal{Q}|n_u)]$ is exact joint normalized log-mass and $\widehat{\mathcal{M}}_{\mathcal{Q}}$ is Eq. (6). Both selection bounds in Appendix A.10 apply to this joint raw mass objective, whose query marginals are proportional to their branch exponential sums. Uniformly averaging per-query probabilities gives equal weight to each query.

The Z_q weights in Eq. (15) are theoretical normalizers; routing evaluates Eq. (6) directly from summaries.

A.2 Mean, partitions and their information content

Mean and partition gaps. For a fixed branch, suppress its subscript: $n = n_u$, $n_j = n_{u,j}$, $w_j = n_j/n$, $\mathcal{M} = \mathcal{M}(q,u)$ and $\widehat{\mathcal{M}}_{\mathcal{P}} = \widehat{\mathcal{M}}_{\mathcal{P}}(q,u)$. Let $\bar{s} = n^{-1}\sum_i s_i$ be the branch mean logit and $J = \mathcal{M} - \bar{s} \geq 0$ its Jensen gap. Within cell j, let μ_j be the mean logit and $J_j = \log \mathbb{E}_j e^{s_i} - \mu_j$ the cell gap, with uniform within-cell expectation $\mathbb{E}_j$. Grouping exponential sums yields

$$\mathcal{M} = \log \sum_j w_j e^{\mu_j + J_j}, \quad \bar{s} \leq \widehat{\mathcal{M}}_{\mathcal{P}} \leq \mathcal{M}, \quad 0 \leq \mathcal{M} - \widehat{\mathcal{M}}_{\mathcal{P}} \leq \max_j J_j. \tag{17}$$

For the last bound in Eq. (17), normalize $w_j e^{\mu_j}$ to weights a_j and use $\mathcal{M} - \widehat{\mathcal{M}}_{\mathcal{P}} = \log \sum_j a_j e^{J_j}$. In Eq. (17), Jensen within cells gives $\widehat{\mathcal{M}}_{\mathcal{P}} \leq \mathcal{M}$, and Jensen across cell means gives $\bar{s} \leq \widehat{\mathcal{M}}_{\mathcal{P}}$. Refinement is covered by Proposition 1; the three four-cell grouping rules are generally non-nested.

For a uniformly sampled branch position I, expanding $s_I - \mathcal{M} = (s_I - \bar{s}) - J$ gives

$$\mathbb{E}_I(s_I - \mathcal{M})^2 = \mathrm{Var}_i(s_i) + J^2, \qquad (\bar{s} - \mathcal{M})^2 = J^2. \tag{18}$$

By Eq. (18), the mean removes the sampled position's variance term. This comparison uses uniform positional sampling; a fixed midpoint has its own error. For two branches A and B with exact margin $\gamma = \mathcal{M}_A - \mathcal{M}_B$, the mean-based margin is $\gamma - (J_A - J_B)$.

Proof of Lemma 1. Fix finite logits on a nonempty branch and omit empty partition cells. Set $Z = \sum_i e^{s_i}$, $Z_{\mathcal{P}} = \sum_j n_j e^{\mu_j}$ and $\widehat{a}_j = n_j e^{\mu_j}/Z_{\mathcal{P}}$. Every distribution in $\mathcal{Q}_{\mathcal{P}}$ has $r_i = a_j/n_j$ in cell j. With $0 \log 0 = 0$,

$$\begin{aligned} \mathrm{KL}(r\|p) &= \sum_j a_j(\log a_j - \log n_j - \mu_j) + \log Z \\ &= \mathrm{KL}(a\|\widehat{a}) + \log Z - \log Z_{\mathcal{P}}. \end{aligned} \tag{19}$$

The minimum is attained uniquely at $a = \widehat{a}$. Subtracting the common $\log n$ proves $e_{\mathcal{P}} = \mathcal{M} - \widehat{\mathcal{M}}_{\mathcal{P}}$. The count factor n_j arises from uniformity over tokens inside each cell. A single cell gives the uniform distribution over its n tokens. Under refinement, a parent-cell weight a_j can be distributed to subcells in proportion to their sizes, so $\mathcal{Q}_{\mathcal{P}} \subseteq \mathcal{Q}_{\mathcal{P}'}$ and the minimum KL decreases. This is a cell-uniform specialization of variational log-partition bounds (Blei et al., 2017). The optimal distribution interprets the routing score; the sparse core uses selected original K/V. Center-1's log-mass approximation error can have either sign.

Mean-1 indistinguishability. The projection result characterizes the residual for a fixed summary state; we next identify branch information that a count-and-mean state cannot distinguish. Let $i^\star$ be a target token outside fixed sink/local support and S_{final} the final raw-token support. Its recall is $Q_N = \Pr(i^\star \in S_{\mathrm{final}})$, with N eligible completed remote tokens. Consider $L = N/C$ completed remote leaves with even $C \geq 2$ and $0 \leq B \leq L$. Fix a nonzero query in dimension at least two. Choose equal-norm vectors v, w with $q^\top w = 0$ and $q^\top v/\sqrt{d} = A > 0$. Ordinary leaves consist of $(w, -w)$ pairs. Replace one pair in one special leaf with $(v, -v)$. These pairs preserve each node's count and zero key sum. The completed count/mean states are therefore identical for every location of the special leaf, whose positive-A token is the unique maximum.

Assume the router uses only the fixed query, positions, tree structure, and completed-node count/mean states before selecting at most B leaves. For a uniform special-leaf index J, independent of router randomness, the summary transcript and selected leaf set $\mathcal{S}_{\mathrm{leaf}}$ are independent of J. Therefore

$$Q_N \leq \Pr(J \in \mathcal{S}_{\mathrm{leaf}}) = \mathbb{E}[|\mathcal{S}_{\mathrm{leaf}}|/L] \leq B/L = BC/N. \tag{20}$$

This input-family limit holds even when every count/mean summary is scanned. The two-logit branches $(0, 0)$ and $(a, -a)$ illustrate the lost information: both means are zero, while their raw masses are 2 and $2\cosh a$. Partitioning can separate opposing directions and expose this difference. Residual collisions remain possible inside a cell. When the query has a nonzero projection onto the hash-nullspace, choose an interior key x and small nullspace perturbations v, w with $|q^\top v| \neq |q^\top w|$. Pairs $x \pm v$ and $x \pm w$ share the cell and mean x, but their exponential sums differ by the corresponding cosh factors. This gives an information limit for these summaries on finite input families.

A.3 Recovered information and finite residuals

Having identified this information limit, we quantify what grouping recovers and what residual remains within cells. Within one branch of n finite logits, cell j contains n_j logits with arithmetic mean μ_j and count weight $w_j = n_j/n$. The branch mean is $\mu = \sum_j w_j \mu_j$; the readout assigns exponential weight $a_j = w_j e^{\mu_j} / \sum_r w_r e^{\mu_r}$ to cell j. The total Jensen gap J is the log-mass lost by a single mean, G is the part recovered by grouping, and $e = \mathcal{M} - \widehat{\mathcal{M}}$ is the remaining error:

$$G = \widehat{\mathcal{M}} - \mu = \log \sum_j w_j e^{\mu_j - \mu}, \qquad J = \mathcal{M} - \mu = G + e. \tag{21}$$

Strict Jensen gives $G \geq 0$, with equality exactly when occupied cell-mean logits agree, and $e \geq 0$, with equality exactly when logits are constant within every occupied cell. These gaps quantify log-mass information in nats.

Curvature. Let c_j be cell-mean keys, $\bar{k} = \sum_j w_j c_j$ the branch mean, m_{occ} the occupied-cell count, and $\Sigma_B = \sum_j w_j (c_j - \bar{k})(c_j - \bar{k})^\top$ the between-cell covariance. Differentiating the finite log-partition gives

$$\nabla_q^2 \widehat{\mathcal{M}}(q) = \mathrm{Cov}_{a(q)}(c_j)/d, \quad \operatorname{rank} \nabla_q^2 \widehat{\mathcal{M}} \leq \min(d, m_{\text{occ}} - 1), \quad \nabla_q^2 \widehat{\mathcal{M}}(0) = \Sigma_B/d. \tag{22}$$

The rank bound in Eq. (22) follows from the weighted linear dependence of the centered cell means. For fixed query direction and membership, define $V_B = \mathrm{Var}_w(\mu_j)$, $V_T = \mathrm{Var}_i(s_i)$. As the query scale $\lambda \to 0$, Taylor expansion yields

$$G(\lambda q) = \lambda^2 V_B/2 + O(\lambda^3), \qquad J(\lambda q) = \lambda^2 V_T/2 + O(\lambda^3). \tag{23}$$

Consequently, when $V_T > 0$,

$$\lim_{\lambda \to 0} \frac{G(\lambda q)}{J(\lambda q)} = \rho = \frac{V_B}{V_T}. \tag{24}$$

These are standard log-partition derivatives (Blei et al., 2017); covariance-based block estimates also use second-order information (Tian et al., 2026).

For a uniformly random reassignment of $n > 1$ fixed logits with $V_T > 0$ to m cells of fixed positive sizes,

$$\mathbb{E}_{\text{perm}} \rho = \frac{m - 1}{n - 1}. \tag{25}$$

Indeed, sampling without replacement gives

$$\mathrm{Var}(\mu_j) = \frac{V_T (n - n_j)}{n_j (n - 1)}. \tag{26}$$

Weighting Eq. (26) by n_j/n and summing proves Eq. (25). This supplies the local-curvature reference for the occupancy-matched experiment; Eq. (28) controls the finite-scale residual. For a Q64 query block $\mathcal{Q}$, define the joint Mean-1 readout $\mathcal{M}_{\text{mean},\mathcal{Q}} = \log(|\mathcal{Q}|^{-1} \sum_q e^{\mu_q})$, where μ_q is each query's branch mean logit. The gaps are $J_{\mathcal{Q}} = \mathcal{M}_{\mathcal{Q}} - \mathcal{M}_{\text{mean},\mathcal{Q}}$ and $G_{\mathcal{Q}} = \widehat{\mathcal{M}}_{\mathcal{Q}} - \mathcal{M}_{\text{mean},\mathcal{Q}}$. Subtracting the joint-index Taylor series cancels between-query mean variance, giving

$$\lim_{\lambda \to 0} \frac{G_{\mathcal{Q}}(\lambda \mathcal{Q})}{J_{\mathcal{Q}}(\lambda \mathcal{Q})} = \frac{\sum_q V_{B,q}}{\sum_q V_{T,q}} \tag{27}$$

when the denominator is positive; $V_{B,q}$ and $V_{T,q}$ are the between-cell and total logit variances for query q, and $\lambda \mathcal{Q} = \{\lambda q : q \in \mathcal{Q}\}$ scales all queries together.

Finite residual control. For $r_i = s_i - \mu_j$ in cell j, put $J_j = \log \mathbb{E}_j e^{r_i}$, with uniform within-cell expectation. Grouping the exact exponential sum proves

$$e = \log \sum_j a_j e^{J_j}. \tag{28}$$

If cell logits have range h_j, Hoeffding's lemma gives

$$J_j \le h_j^2/8, \tag{29}$$

controlling the remaining gap through within-cell variation. A short proof sets $f_j(t) = \log \mathbb{E}_j e^{t r_i}$: $f_j(0) = f_j'(0) = 0$ and $f_j''(t) = \mathrm{Var}_{j,t}(r_i) \le h_j^2/4$ under exponential tilting. Integrating twice gives Eq. (29) and the exact identity

$$J_j = \int_0^1 (1-t)\,\mathrm{Var}_{j,t}(s_i)\,dt. \tag{30}$$

For $R = \max_i |r_i|$ and $S_2 = \frac{1}{2}\sum_j a_j \mathbb{E}_j r_i^2$, Taylor's integral remainder, using $\mathbb{E}_j r_i = 0$, also yields

$$e^{-R} S_2 \le e^e - 1 \le e^R S_2, \qquad |e^e - 1 - S_2| \le \frac{e^R}{6} \sum_j a_j \mathbb{E}_j |r_i|^3. \tag{31}$$

Eq. (31) motivates the variance and positive-tail features in the residual study.

Rescues and harms. Consider competing equal-span branches A and B, with exact normalized log-masses $\mathcal{M}_A$ and $\mathcal{M}_B$. Their exact margin is $\gamma = \mathcal{M}_A - \mathcal{M}_B$. Define $\Delta J = J_A - J_B$ as the difference in information lost by their single means, and $\Delta G = G_A - G_B$ as the difference in information recovered by grouping. The mean and partition margins are $\gamma - \Delta J$ and $\gamma - \Delta J + \Delta G$, respectively.

A rescue changes an incorrect mean-based choice into a correct partition choice; a harm changes a correct choice into an incorrect one. For $\gamma > 0$, a strict rescue requires $\Delta J > \gamma$ and $\Delta G > \Delta J - \gamma$. A strict harm requires $\Delta J < \gamma$ and $\Delta G < \Delta J - \gamma$. Exchange branch labels for negative γ and apply the prescribed tie rule at equality. Selecting the lower-mass branch incurs oracle-relative mass loss $1 - e^{-|\gamma|}$. The measured rates count rescues and harms on the same branch pairs, relating recovered information to changed decisions.

A.4 Merge-time readout deficit

For a parent p with disjoint complete children v, fixed compatible cell labels and $\widehat{W}_u = \sum_j n_{u,j} e^{\mu_{u,j}}$, define $D_p = \sum_v \widehat{W}_v - \widehat{W}_p$. This specializes convex information decomposition (Banerjee et al., 2005) to branch readout.

Proof of Proposition 2. In occupied parent cell j, write $\pi_{v,j} = n_{v,j}/n_{p,j}$. Its mean is $\mu_{p,j} = \sum_v \pi_{v,j}\mu_{v,j}$, so Jensen gives $n_{p,j} e^{\mu_{p,j}} \le \sum_v n_{v,j} e^{\mu_{v,j}}$.

Equality holds iff occupied child-cell means agree. Define child mass weights $\beta_v = \widehat{W}_v / \sum_z \widehat{W}_z$ and merge increment $\Delta_p^{\mathrm{merge}} = \log(\sum_v \widehat{W}_v / \widehat{W}_p)$, with sums over children. Summing proves $D_p \ge 0$; exact mass additivity yields

$$W_p - \widehat{W}_p = \sum_v (W_v - \widehat{W}_v) + D_p, \tag{32}$$

$$e_p = \log \sum_v \beta_v e^{e_v} + \Delta_p^{\mathrm{merge}}. \tag{33}$$

Eq. (33) follows by substituting $W_v = \widehat{W}_v e^{e_v}$ into $e_p = \log(W_p/\widehat{W}_p)$. Repeated substitution in Eq. (32) gives the sum of all leaf residuals plus all internal D_u in the subtree. State addition is exact; the nonlinear readout loses child identity.

Let $H_{p,j}$ be the range of occupied child-cell mean logits in cell j, and $a_{p,j} = n_{p,j} e^{\mu_{p,j}} / \widehat{W}_p$. Eq. (29) gives

$$\Delta_p^{\text{merge}} \le \log \sum_j a_{p,j} e^{H_{p,j}^2/8}. \tag{34}$$

Equal child-cell means make $D_p = 0$ even with changing cell proportions, a useful sufficient regime for fixed cell labels. More generally, with $A_u = \widehat{W}_u^{\text{partition}} - \widehat{W}_u^{\text{mean}} \ge 0$,

$$D_p^{\text{mean}} - D_p^{\text{partition}} = A_p - \sum_v A_v, \tag{35}$$

which can have either sign. Temporal-4 rebuilds parent quarters from child totals; its relabeling is handled by that construction rather than matching-cell addition.

A.5 Query-weighted within-cell geometry

The merge decomposition isolates where error enters during composition; we next examine the query-projected geometry that controls the information left unresolved within each cell. Projected within-cell variation describes the information left after grouping: variation along a query direction contributes to its remaining mass gap. For fixed finite keys and a fixed partition, let c_j be cell j's mean. Define the within-cell key covariance Σ_W and its query-projected variance $V_W(q)$ by

$$\Sigma_W = \frac{1}{n} \sum_j \sum_{i \in u_j} (k_i - c_j)(k_i - c_j)^\top, \qquad V_W(q) = \frac{q^\top \Sigma_W q}{d}. \tag{36}$$

$V_W(q)$ averages within-cell logit variances with token-count weights.

Local residual expansion. For fixed q, as a scalar $\lambda \to 0$,

$$e_{\mathcal{P}}(\lambda q) = \frac{\lambda^2}{2d} q^\top \Sigma_W q + O(\lambda^3). \tag{37}$$

Proof. Both finite log-partition functions are analytic near zero, vanish there, and have the same first derivative, the mean logit. Their second derivatives are total logit variance and between-cell mean-logit variance. The population variance decomposition leaves $V_W(q)$ on subtraction; Taylor's theorem gives the remainder. Here λ scales the query, leaving keys and membership fixed. For a finite query set $\mathcal{Q}$, define its second-moment matrix $A_{\mathcal{Q}} = |\mathcal{Q}|^{-1} \sum_q q q^\top$. Then

$$\frac{1}{|\mathcal{Q}|} \sum_q V_W(q) = \frac{1}{d} \operatorname{tr}(A_{\mathcal{Q}} \Sigma_W). \tag{38}$$

Eq. (38) follows from $q^\top \Sigma_W q = \operatorname{tr}(q q^\top \Sigma_W)$ for each fixed query. For the joint Q64 log-partition gap, the same second-derivative subtraction cancels the between-query variance of mean logits, giving the average $V_W(q)$. At native query scale the joint exponential weighting remains distinct from an average of per-query gaps.

Finite-scale control and angular cells. Let $J_j = \log \mathbb{E}_j e^{s_i - \mu_j}$ be the within-cell Jensen gap and $a_j = w_j e^{\mu_j} / \sum_r w_r e^{\mu_r}$ the exponential cell weight. Then $e = \log \sum_j a_j e^{J_j}$. If cell j has logit range h_j, the finite range bound in Eq. (29) yields

$$0 \le e \le \log \sum_j a_j e^{h_j^2/8} \le \max_j h_j^2/8. \tag{39}$$

Eq. (39) identifies projected within-cell variation as the relevant geometry. For two nonzero vectors with angle θ, a random spherical hyperplane normal has equal-sign probability $1 - \theta/\pi$ (Charikar, 2002, Section 3). In their plane, the normal's angle is uniform and disagreeing wedges occupy fraction θ/π; collinear cases follow directly. CarryKV fixes the sampled table before use. Angular agreement thus motivates post-RoPE directional grouping. Finite-scale control additionally depends on actual projected ranges: equal signs can contain widely separated directions and arbitrarily different norms.

A fixed-subspace extension. Loki (Singhania et al., 2024) reports layer/head-dependent low-dimensional key structure before and after RoPE on its evaluated models; its transformed scoring returns to full-dimensional attention on selected keys. SparQ (Ribar et al., 2024) uses large-magnitude query coordinates for selective key access. These motivate studying the alignment between query directions and within-cell variation in Qwen3. An untested extension would fix an r-dimensional subspace with orthonormal basis $U \in \mathbb{R}^{d\times r}$ and assign signs by $g^\top U^\top k$, where $g \in \mathbb{R}^r$ is fixed. Counts, full-dimensional key sums and the $1/\sqrt{d}$ readout remain unchanged. For $\Delta k = k_i - c_j$,

$$|q^\top \Delta k| \le \|U^\top q\|\,\|U^\top \Delta k\| + \|(I_d - UU^\top)q\|\,\|(I_d - UU^\top)\Delta k\|. \tag{40}$$

Here I_d is the $d \times d$ identity. Orthogonal decomposition and Cauchy–Schwarz prove Eq. (40); both retained and discarded directions matter. A fixed U preserves cell identities across merges, whereas query-specific coordinate choices would require changing the persistent partition. Since $g^\top U^\top k = (Ug)^\top k$, this selects directions within the same ambient hyperplane family at unchanged bit capacity. Explicit projection costs $O(dr)$ per key and storage for U, or the normals can be precomposed as Ug. Full key sums for m cells still cost $O(md)$.

A.6 Partition families and capacity

Table 3 distinguishes the three four-cell families. Their readout is the same count-weighted exponential sum; the grouping and parent construction differ. With $T_{u,j} = (n_{u,j}, S_{u,j})$,

$$T^{\mathrm{Hash}}_{p,j} = \sum_v T^{\mathrm{Hash}}_{v,j}, \qquad T^{\mathrm{Temporal}}_{p,v} = \sum_j T^{\mathrm{Temporal}}_{v,j}. \tag{41}$$

In Eq. (41), the first rule adds matching labels; the second makes child v the parent's vth quarter. A Temporal-4 parent is scored by four child means, without giving its cells independent beam slots. Position hashing also adds matching labels. The Hash-2/4/8 capacity study uses nested prefixes of one fixed direction table; it changes cell count within one family.

Table 3: Controlled branch representations. All cell means retain the full key dimension.

Representation	Cell rule / information	Parent construction
CarryKV-Hash-4	Two post-RoPE direction signs	Sum matching cells
CarryKV-Position-Hash-4	Fixed absolute-position hash	Sum matching cells
CarryKV-Temporal-4	Four contiguous equal quarters	Child totals as quarters
CarryKV-Mean-1	One count and full key sum	Sum child totals
Center-1 (HiP)	Lower-midpoint raw key	Interval-derived address

CarryKV-*variant* identifies the summary; analyses use short names. Hash-4 is the default system instantiation. Center-1 isolates HiP's center-key valuation; complete HiP comparisons include its search and block organization.

A.7 Summary design and recursive maintenance

Table 4 separates storing a useful estimate from maintaining it recursively. Performer feature sums are additive; full count/sum/second-moment statistics are additive as well. CarryKV's fixed cells combine exact composition with a deterministic one-sided readout and bounded candidate access. COBS compresses covariance using low-rank factors, query subspaces and quantization; those choices have their own approximation and construction costs (Tian et al., 2026).

Table 4: Summary design and recursive maintenance. State sizes are per summarized set; r, m and d denote feature count, cell count and head dimension.

Design	Stored state	Parent construction	Query access
HiP	Positional raw keys	Interval addressing	$O(\log N)$ recursion
InfiniteHiP	Searched representatives	Staged chunk search	Parallel screening
Performer	Random-feature sums	Exact feature addition	$O(Nrd)$ evaluation
Full moments	Count, sum, $kk^\top$: $O(d^2)$	Exact moment addition	Covariance readout
COBS	Mean, compressed covariance	Compression-specific	Block descriptors
CarryKV	Fixed cells: $O(m(d+1))$	Exact cell-wise addition	$O(\log N)$ routing

HiP, InfiniteHiP and CarryKV are training-free; Performer uses kernel attention, COBS targets NSA selection, and full moments are generic. Routing costs use fixed budgets; Performer reports sequence evaluation. CarryKV's routing costs are per block/head, giving $O(N \log N)$ over a sparse prefill. Section 5.4 measures the complete prefill operator.

Proof of Proposition 1. The fixed-cell state in Table 4 is closed under additive merging. Proposition 1 characterizes the readout of this recursively maintained state. For disjoint children, u_j is the disjoint union of their cell-j tokens; therefore both counts and key sums add without accessing the raw keys. Within-cell Jensen gives $\widehat{\mathcal{M}}_{\mathcal{P}} \le \mathcal{M}$; Jensen across cell means gives $\mu \le \widehat{\mathcal{M}}_{\mathcal{P}}$. For a refinement, each coarse mean is a count-weighted average of its finer means; another within-cell Jensen application proves $\widehat{\mathcal{M}}_{\mathcal{P}} \le \widehat{\mathcal{M}}_{\mathcal{P}'}$. Consequently $0 \le e_u^{\mathcal{P}'} \le e_u^{\mathcal{P}} \le J_u$. For any fixed normalized nonnegative weighting π over branch–query pairs with finite expectations, averaging $G_u = J_u - e_u \ge 0$ yields $\mathbb{E}_\pi[G_u] = \mathbb{E}_\pi[J_u] - \mathbb{E}_\pi[e_u] \ge 0$. Strict convexity of the exponential gives equality $\widehat{\mathcal{M}}_{\mathcal{P}} = \mathcal{M}$ exactly when all logits within each occupied cell coincide, for the fixed query. This is the zero-residual case $e_u = 0$, with recovered gap $G_u = J_u$. Differentiating the finite weighted exponential sum yields

$$\nabla_q^2 \widehat{\mathcal{M}}(q) = \mathrm{Cov}_{a(q)}(\bar{k}_j)/d, \qquad a_j(q) = \frac{w_j e^{q^\top \bar{k}_j/\sqrt{d}}}{\sum_r w_r e^{q^\top \bar{k}_r/\sqrt{d}}}. \tag{42}$$

At $q = 0$, $a_j = w_j$ and this is Σ_B/d. There are m scalars and m dimension-d sums. The state stores no within-cell covariance; Appendix A.3 derives the between/within decomposition and local-query expansion.

A.8 Gap recovery and merge measurements

We test three consequences of the preceding analysis: whether between-cell variation explains recovered mass information, whether within-cell geometry explains the remaining residual, and whether the recovered information survives additive merges. Sampling and predictor fitting on branches encountered during recursive search are specified in Appendix C.1.

Curvature and directional grouping. The recovered fraction G/J measures how much of the single-mean gap is restored by grouping. Its small-query limit is $\rho = V_B/V_T$, the ratio of between-cell to total logit variance (Appendix A.3). Scaling queries by $\lambda \in \{1/16, 1/8, 1/4, 1/2, 1\}$ preserves cell membership. At $\lambda = 1/16$, mean $|G/J - \rho|$ on held-out branches is .00331/.00202/.00137 for spans 16/64/256, versus .04682/.03463/.02738 at native scale. Hash-4 recovers .1561/.1001/.0720 of Mean-1's gap, compared with .0905/.0300/.0090 under 16 random membership permutations preserving cell sizes. Paired gains are .0656 [.0636,.0675], .0701 [.0686,.0716] and .0630 [.0618,.0642]. Directional grouping therefore preserves information beyond the multiple-group finite-sample effect.

Remaining residual error. Residual e is the log-mass still missing after grouping. A second-order estimate replaces each within-cell Jensen gap by half its logit variance; its absolute difference from e measures second-order discrepancy. We predict these two quantities from within-cell variance, with or without the largest positive deviation from a cell mean. Adding this tail feature reduces held-out residual-e RMSE by .08091 [.07859,.08324] nats and second-order-discrepancy RMSE by .00491 [.00387,.00594] nats (Table 5). The tail feature is more informative for remaining log-mass error

than for the local second-order discrepancy. These predictors characterize the information remaining within cells.

Table 5: Held-out prediction error on 234 prompts. Entries are RMSE in nats (lower is better), with 95% within-task prompt-bootstrap intervals conditional on the fitted predictors. Predictor fitting and sample split: Appendix C.1.

Features	Residual e	Second-order discrepancy
Variance	0.5696 [0.5627,0.5767]	0.5182 [0.4994,0.5370]
Variance + positive max	0.4887 [0.4825,0.4951]	0.5133 [0.4947,0.5317]

Inherited information and merge loss. At parent spans 64/256, Hash-4 reduces held-out parent error relative to Mean-1 by .2029 [.1971,.2091] and .1947 [.1890,.2005] nats. Inherited error falls by .2822 [.2749,.2897] and .2858 [.2786,.2932], outweighing a larger merge-readout increment Δ_p^{merge} of .0793 [.0758,.0829] and .0911 [.0872,.0951] (paired 95% intervals). Low/high-drift mean increments are .0713/.7446 and .0797/.7930 nats: smaller differences between corresponding child-cell mean logits identify low-loss merges. Thus richer child information outweighs the additional parent readout loss in Proposition 2's decomposition.

Position-matched parent/four-child groups also cover spans 1K/2K/4K/8K. Hash-4 has lower total parent error than Mean-1 throughout. Its new log-readout loss is higher at 1K–4K and lower at 8K (.925 versus .992 nats), while normalized raw merge deficit is higher at every span. As in the finer recursive merges, retained child information and new merge loss contribute separately to parent accuracy.

A.9 Fixed-window gap recovery

The recursive measurements above follow deployed search trajectories; fixed windows complement them by isolating representation behavior across a wider range of spans. On the 260-prompt native 32K pool, Hash-4 at $L = 16K$ recovers 0.900 nats of Mean-1's mean 6.853-nat Jensen gap. The mean per-branch recovered fraction is 7.30%, a different statistic from the ratio of means. Rescues and harms, each divided by all exact-mass-nontied pairs, are 7.24% [7.15,7.32] and 1.05% [1.00,1.11]; at $L = 512$ they are 3.17% [3.15,3.19] and 2.92% [2.91,2.94]. Against Center-1 at $L = 512$, Hash-4 rescues 28.82% and harms 8.04%. The measurement protocol is in Appendix C.6.

Capacity measurements. Figure 5 resolves the coarse-window regime with additional measured windows. Hash counts and means use nested prefixes of the same direction table; the exact branch pairs are common within each sampling regime. At $L = 16K$, adding the third and fourth cells improves fidelity by 5.27 points; four further cells give 5.80 points.

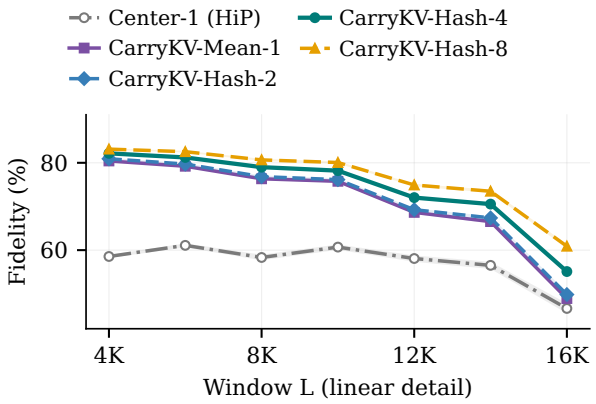


Figure 5: Capacity across windows on the 260-prompt native-32K 1.7B pool. Seven additionally measured coarse windows on a linear axis. Each window compares two $L/2$-token branches using the final query of each layer/head; ties select left. Bands are 95% prompt-bootstrap intervals within tasks. Hash-2/4/8 use nested cells.

A.10 SAME-FRONTIER GUARANTEES, SIGNED ERRORS AND ALLOCATION

One-sided residual guarantee (Corollary 1). Fix a query/head with finite logits and a nonempty frontier $\mathcal{C}$ of disjoint legal branches of common span n and equal token cost, outside fixed guaranteed support. Let $1 \le B \le |\mathcal{C}|$, $\mathcal{M}_u = \log(W_u/n)$, and let $S^\star$ maximize the sum of exact raw masses W_u over B candidates. A one-sided estimator $\widehat{\mathcal{M}}_u = \mathcal{M}_u - e_u$, $e_u \ge 0$, gives $\widehat{W}_u = ne^{\widehat{\mathcal{M}}_u} = W_u e^{-e_u}$. Let $\widehat{S}$ be its top-B set, and define

$$Q_{\mathcal{C}} = \frac{\sum_{u\in\widehat{S}} W_u}{\sum_{u\in S^\star} W_u}, \qquad \alpha_u = \frac{W_u}{\sum_{v\in S^\star} W_v} \quad (u \in S^\star). \tag{43}$$

All masses are positive, the weights sum to one, and $0 < Q_{\mathcal{C}} \le 1$. One-sided underestimation and the optimality of $\widehat{S}$ for estimated mass give, respectively,

$$\sum_{u\in\widehat{S}} W_u \ge \sum_{u\in\widehat{S}} \widehat{W}_u \ge \sum_{u\in S^\star} \widehat{W}_u = \sum_{u\in S^\star} W_u e^{-e_u}. \tag{44}$$

Dividing by the exact top-B mass proves the first inequality in Eq. (10). Jensen's inequality for the convex function $x \mapsto e^{-x}$ proves the second:

$$Q_{\mathcal{C}} \ge \sum_{u\in S^\star} \alpha_u e^{-e_u} \ge \exp\left(-\sum_{u\in S^\star} \alpha_u e_u\right). \tag{45}$$

Tied top-B sets also maximize the relevant mass sums, so the proof permits any tie resolution consistent with the corresponding scores.

Mean-1 has $e_u = J_u$. Every multi-cell partition has $0 \le e_u \le J_u$, so the exact-top-B mass-weighted residuals satisfy

$$\bar{e}^\star_{\mathcal{C}} := \sum_{u\in S^\star} \alpha_u e_u \le \sum_{u\in S^\star} \alpha_u J_u =: \bar{J}^\star_{\mathcal{C}}, \qquad Q_{\mathcal{C}} \ge e^{-\bar{e}^\star_{\mathcal{C}}} \ge e^{-\bar{J}^\star_{\mathcal{C}}}. \tag{46}$$

By Eq. (46), both bounds in Eq. (10) weakly improve on Mean-1's for the same frontier, exact set and weights. Nested refinement strengthens them again. If all residuals on $S^\star$ vanish, the lower bound in Eq. (10) equals one, hence $Q_{\mathcal{C}} = 1$. These conclusions concern the guaranteed mass; realized decisions depend on comparisons with all competing branches.

Generic signed-error guarantee. Keep the same frontier and exact set, and let an arbitrary scorer assign $\widetilde{\mathcal{M}}_u = \mathcal{M}_u - \delta_u$, where δ_u may have either sign. Here $\widehat{S}$ denotes this scorer's top-B set. Define $\delta_- = \min_{u\in\mathcal{C}} \delta_u$, $\delta_+ = \max_{u\in\mathcal{C}} \delta_u$, and $\Omega_{\mathcal{C}} = \delta_+ - \delta_-$. Top-$B$ estimated values maximize the sum of their exponentials, giving

$$\begin{aligned} \sum_{u\in\widehat{S}} e^{\mathcal{M}_u} &\ge e^{\delta_-} \sum_{u\in\widehat{S}} e^{\widetilde{\mathcal{M}}_u} \ge e^{\delta_-} \sum_{u\in S^\star} e^{\widetilde{\mathcal{M}}_u} \\ &\ge e^{\delta_- - \delta_+} \sum_{u\in S^\star} e^{\mathcal{M}_u}. \end{aligned} \tag{47}$$

The common span in $W_u = ne^{\mathcal{M}_u}$ cancels, proving

$$Q_{\mathcal{C}} \ge e^{-\Omega_{\mathcal{C}}}. \tag{48}$$

For $B < |\mathcal{C}|$, define the exact cutoff margin $\gamma_{\text{cut}} = \mathcal{M}_{(B)} - \mathcal{M}_{(B+1)}$, with masses sorted in descending order. Every exact-selected/rejected pair has estimated margin at least $\gamma_{\text{cut}} - \Omega_{\mathcal{C}}$. If this is positive, the selected support is identical; equality admits ties.

Eq. (10) exploits one-sided Mean/multi-cell underestimation and supplies the paper's main link from representation to pruning quality. Eq. (48) also covers Center-1 and removes common log-bias through the error range; it is a generic worst-case diagnostic. Neither bound uniformly dominates the other: constant positive errors give a unit spread bound, while error concentrated on an exact-rejected branch can weaken the spread bound even when exact-selected residuals vanish. Both use exact branch masses for offline analysis.

Common log-bias cancels. Pointwise improvement can increase error spread: for exact values $(0.5, 0)$, errors $(1, 1)$ preserve order, whereas smaller errors $(1, 0)$ reverse it. Under unequal spans, use log raw mass $\mathcal{M}_u + \log n_u$ and its estimate. Unequal costs or overlaps alter the selection problem. Fixed sink/local support should be removed before defining uncovered candidate mass. Eq. (48) uses extrema within one candidate pool. Quantiles and ranges pooled across queries/heads describe different statistical objects.

Margins. For two branches A, B, define exact margin $\gamma = \mathcal{M}_A - \mathcal{M}_B$ and estimated margin $\widehat{\gamma} = \widetilde{\mathcal{M}}_A - \widetilde{\mathcal{M}}_B$. With the signed errors above,

$$\widehat{\gamma} = \gamma - (\delta_A - \delta_B). \tag{49}$$

A common additive bias cancels; the differential error determines how the margin moves. Strict inversion occurs exactly when $\gamma[\gamma - (\delta_A - \delta_B)] < 0$; equality follows the tie rule. Empirical inversion rates depend on the joint distribution of margins and differential errors.

Fixed-capacity comparisons. Position-Hash-4 assigns four cells with a fixed hash of absolute token position. Temporal-4 uses contiguous quarters; Hash-4 uses two direction signs of post-RoPE keys. Each stores four counts and four key sums. Figures 6 and 7 use common branches and the exact mass reference. Temporal grouping leads at many fine windows; Hash-4 leads at the largest one. At $L = 16K$, Temporal-4 has lower average projection error, while Hash-4 has lower average pair spread and lower observed selection loss. At $L = 512$, Temporal-4 improves all three. These outcomes distinguish accurate branch values from accurate comparisons between branches.

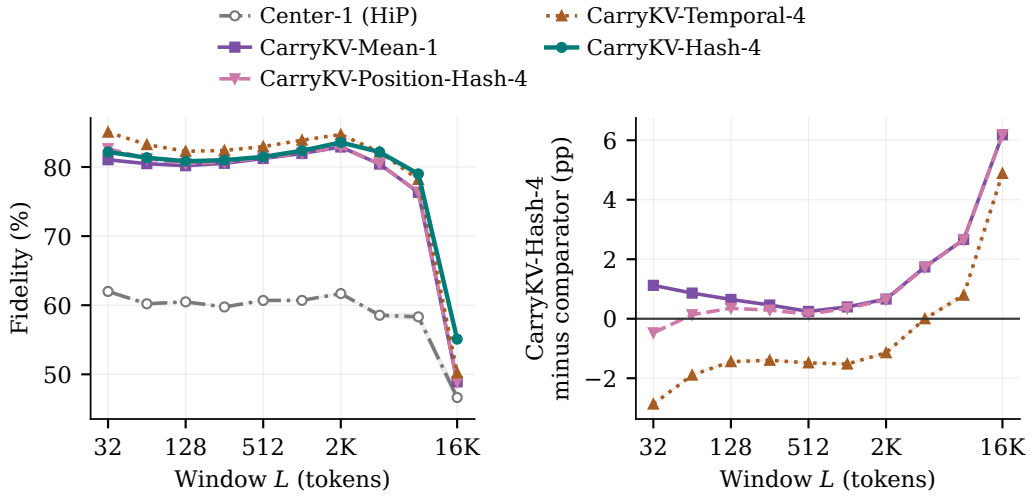


Figure 6: Four-cell allocation, separate from the Hash-2/4/8 capacity study. Left: branch-order fidelity for Center-1, Mean-1, Position-Hash-4, Temporal-4 and Hash-4. Right: Hash-4 minus the partition comparators, with paired 95% prompt-bootstrap intervals. The 260-prompt fixed-window protocol, exact normalized-mass reference and left ties are common. Temporal-4 groups contiguous quarters; Position-Hash-4 hashes absolute positions; Hash-4 groups post-RoPE key directions.

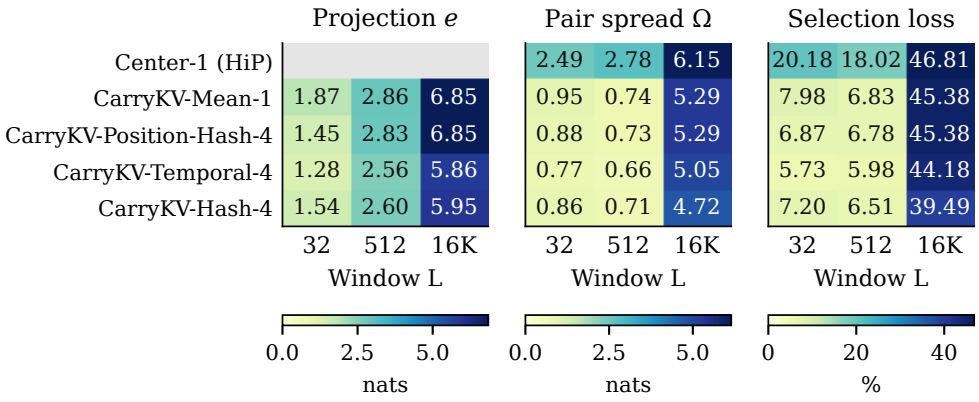


Figure 7: Value error, error variation and selection at three measured windows on the same 260-prompt pool. Each panel has method rows and $L = 32, 512, 16K$ columns; cell labels give means. Center-1 is omitted from projection error e and retained for generic spread $\Omega = |\delta_A - \delta_B|$ and loss $100(1 - Q)$, where Q is selected mass divided by exact-best binary mass. Exact-mass ties are excluded. Table 6 gives selection quality, bound and slack; Figure 6 gives paired uncertainty in fidelity.

Empirical bound tightness. For each equal-span branch pair, Q is the selected branch's mass divided by the exact binary choice's mass; a value of one is optimal. The differential error spread is $\Omega = |\delta_A - \delta_B|$, where δ_A, δ_B are the two log-mass estimation errors. Eq. (48) gives the per-pair lower bound $Q \geq e^{-\Omega}$. Slack, $Q - e^{-\Omega}$, measures how far the observed quality exceeds this guarantee. Table 6 averages these pair-specific quantities.

At $L = 16K$, Hash-4 has higher mean Q than Temporal-4 (.6051 versus .5582), while its mean lower bound is smaller (.1698 versus .1762). Their mean slacks are .4354 and .3819. Eq. (48) is valid but loose in this population. Its average ranks these non-nested partitions differently from their observed selection quality.

Table 6: Binary-selection bound utility on 260 prompts. Each pair contains two equal-span branches and selects one. Q is selected mass divided by exact-choice mass; Ω is the pair's scorer-error spread. Slack is $Q - e^{-\Omega}$; certification requires exact margin $|\gamma| > \Omega$. Bounds use exact errors measured offline. Exact-mass ties are excluded; averages are taken within layer/head, prompt and task.

Window	State	Mean Q	Mean $e^{-\Omega}$	Mean slack	Certified (%)
32	CarryKV-Mean-1	0.9202	0.5633	0.3569	70.8
32	CarryKV-Position-Hash-4	0.9313	0.5793	0.3520	71.8
32	CarryKV-Temporal-4	0.9427	0.6247	0.3181	75.5
32	CarryKV-Hash-4	0.9280	0.5762	0.3518	71.1
512	CarryKV-Mean-1	0.9317	0.6335	0.2982	68.7
512	CarryKV-Position-Hash-4	0.9322	0.6349	0.2973	68.8
512	CarryKV-Temporal-4	0.9402	0.6630	0.2772	71.2
512	CarryKV-Hash-4	0.9349	0.6328	0.3021	68.3
16K	CarryKV-Mean-1	0.5462	0.1622	0.3839	45.9
16K	CarryKV-Position-Hash-4	0.5462	0.1624	0.3838	46.0
16K	CarryKV-Temporal-4	0.5582	0.1762	0.3819	47.2
16K	CarryKV-Hash-4	0.6051	0.1698	0.4354	49.5

A.11 Model size and position extrapolation

Model and position regimes. Figure 8 compares Hash-4 with Mean-1 on matched Dense query/key states across model and position regimes. The static YaRN factors, rotary amplitudes and unscaled context match Appendix C.3. Additional full-range results retain the small-window controls; this grid shows the recorded coarse windows. Table 7 resolves the paired largest-window increments: Hash-4 improves over Mean-1 in every case, with smaller gains on 8B.

Table 7: Hash-4 minus Mean-1 fidelity at $L = N/2$, in points with paired 95% prompt-bootstrap intervals within tasks. Figure 8 specifies the input pools and other measured coarse windows.

Model	32K native	64K YaRN	128K YaRN
1.7B	6.18 [6.08, 6.29]	5.27 [5.16, 5.39]	3.22 [3.06, 3.37]
8B	1.15 [1.06, 1.24]	1.13 [1.02, 1.23]	0.50 [0.40, 0.60]

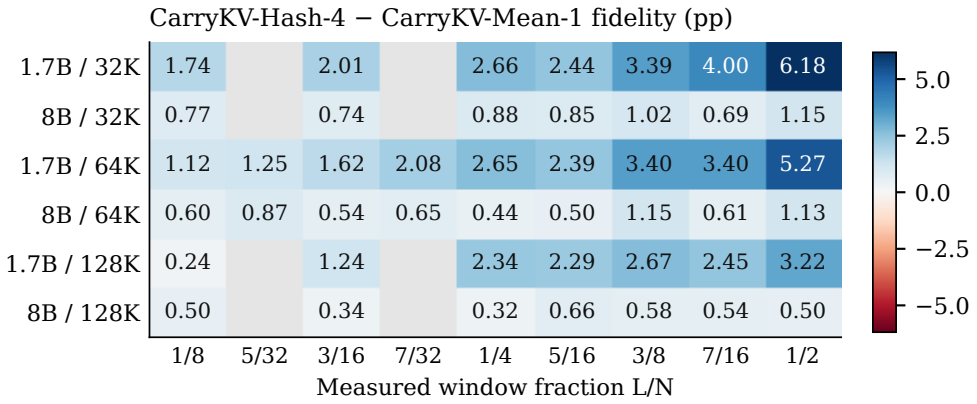


Figure 8: Hash-4 minus Mean-1 branch-order fidelity (pp). Rows identify model/context: native 32K, YaRN2/64K and YaRN4/128K; columns give window fraction L/N. Each window compares equal halves, with exact-mass reference and left ties. Prompts are shared between models: 260/130/130 by length. Grey cells are unmeasured. Means weight tasks equally; Table 7 gives paired 95% prompt-bootstrap intervals at $L = N/2$. Sampling is specified in Appendix C.6.

A.12 Fixed-direction sensitivity

We evaluate eight fixed normalized Gaussian direction tables, including the deployed table, and change only the random draw. The four cells use the first two rows of a three-direction table, preserving the construction, nested-prefix convention and shared scope. All tables use the same 234 held-out states and 12,636 deployed frontiers. We also let each table control a complete search on the same Q/K/V.

Figure 9 separates sampling uncertainty from variability across direction tables. Each table's mean Hash–Mean frontier effect is positive at spans 16/64/256; 23 of 24 paired intervals are positive and one includes zero. The ranges are 0.485–1.477, 0.054–0.522 and 0.095–0.273 points, respectively. Complete-search leaf mass relative to exact-tree selection improves for all eight tables, as does normalized output error $\mathrm{NL2} = \|o_S - o\|_2/\|o\|_2$. Full-prefix retained probability has a positive mean change for seven tables and a positive paired interval for five; the other three intervals include zero. Its smallest mean change is -0.0061 points.

The fixed-window counterpart uses the final query at the same three layers and two heads, with at most 64 uniformly spaced nonoverlapping windows per size. It is a distinct coordinate sample from the all-layer/all-head study. At windows 32/512, all eight tables improve mean fidelity over Mean-1;

at 4K/8K/16K, the signs are positive for 3/4/6 tables, respectively. For the deployed table, these three coarse contrasts are -0.449, $+0.285$ and -0.855 points. Thus direction and coordinate choice affect coarse pairwise behavior even though the actual-frontier increments retain their sign across all eight tables.

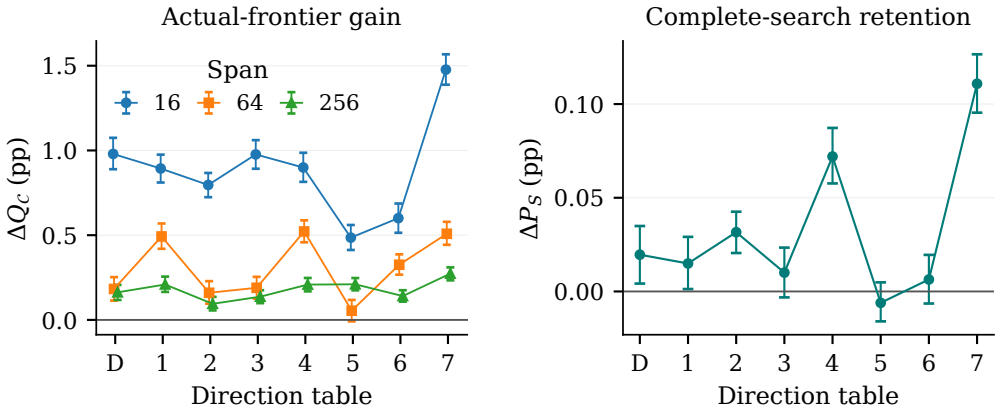


Figure 9: Eight fixed direction tables on the same 234 held-out Qwen3-1.7B Dense-32K states. D denotes the deployed table, followed by seven additional fixed tables. Left: same-frontier joint-mass contrasts. Right: complete-search mean per-query retained-probability contrasts, using the full legal-prefix denominator. Error bars resample prompts within the 13 tasks (95% paired intervals); table-to-table variability is a separate source of variation.

Table 8: Direction-table differences from Mean-1 (pp; NL2 is dimensionless). The ΔQ_C columns specify branch spans. Q_{tree} divides selected remote mass by exact-node search mass; P_S uses full-prefix mass. Median, quartiles (Q25/Q75) and range summarize eight tables. D rank is ascending, with no table reselection.

Table	ΔQ_C span 16	ΔQ_C span 64	ΔQ_C span 256	ΔQ_{tree}	ΔP_S	ΔNL2
D	+0.9795	+0.1828	+0.1627	+0.9076	+0.0196	-0.0086
1	+0.8934	+0.4924	+0.2090	+1.2426	+0.0149	-0.0007
2	+0.7963	+0.1601	+0.0950	+0.9265	+0.0316	-0.0054
3	+0.9766	+0.1896	+0.1371	+1.0286	+0.0100	-0.0046
4	+0.8994	+0.5220	+0.2085	+1.0483	+0.0720	-0.0054
5	+0.4855	+0.0545	+0.2102	+0.5264	-0.0061	-0.0063
6	+0.6005	+0.3257	+0.1407	+0.7581	+0.0064	-0.0053
7	+1.4769	+0.5083	+0.2732	+1.6059	+0.1108	-0.0098
Median	+0.8964	+0.2576	+0.1856	+0.9775	+0.0173	-0.0054
Q25	+0.7474	+0.1771	+0.1398	+0.8702	+0.0091	-0.0069
Q75	+0.9773	+0.4963	+0.2093	+1.0968	+0.0417	-0.0052
Min	+0.4855	+0.0545	+0.0950	+0.5264	-0.0061	-0.0098
Max	+1.4769	+0.5220	+0.2732	+1.6059	+0.1108	-0.0007
D rank	7	3	4	3	5	2

A.13 Layer and Head Structure

Direction-table sensitivity varies the partition globally; we next resolve where the Hash–Temporal differences occur across fixed layers and heads. Figure 10 resolves the same 260-prompt branch pairs by layer and attention head at three windows. A head-cell value averages prompts within each task and then the 13 tasks equally. Layer curves first average heads within each prompt. Intervals resample prompts within tasks; heads are treated as fixed model coordinates.

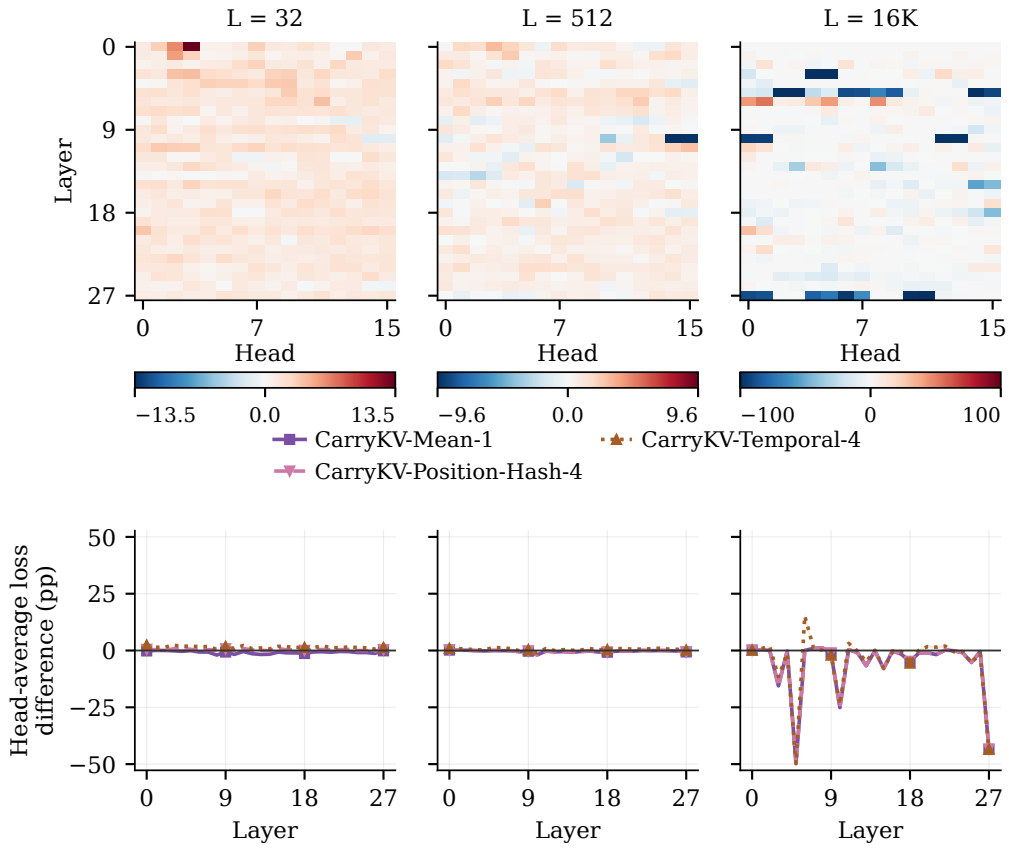


Figure 10: Layer/head structure in binary mass loss $1-Q$, on 260 Qwen3-1.7B Dense-32K prompts. Top: Hash-4 minus Temporal-4; blue favors Hash-4. Each window labels its symmetric color range; grey is missing. Bottom: head-averaged Hash-4 contrasts on a common scale, with paired 95% within-task prompt-bootstrap intervals. Exact-mass ties are excluded.

At window 16K, Hash-4 reduces loss in 16 of 28 layer means; 14 paired layer intervals lie below zero. Among 448 layer/head cells, 187 improve, 108 tie and 153 worsen relative to Temporal-4. The magnitude is concentrated in a subset of layers, visible in Figure 10's lower-right plot. At windows 32 and 512, respectively 14 and 74 cells favor Hash-4, consistent with Temporal-4's advantage at fine scales. These counts describe heterogeneity across fixed model coordinates. Full five-state layer/head values accompany the results.

B Hierarchy Construction, Updates, and Complexity

The state in Section 4.2 supports compatible parent construction and bounded traversal. We first specify the state and traversal, then establish compatible parent construction before counting routing, storage, and update work.

Table 9: Training-free retrieval designs and routing work at fixed dimensions, blocks and budgets.

Method	Branch information / access	Per block	Prefill
HiP	Center keys; bounded refinement	$O(\log N)$	$O(N \log N)$
InfiniteHiP	Representatives; parallel first-stage scan	$O(N)$	$O(N^2)$
CarryKV	Cell counts/sums; bounded refinement	$O(\log N)$	$O(N \log N)$

Per-block and prefill costs count query-dependent routing for context length N. CarryKV additionally constructs its state in $O(N)$ work.

B.1 State and traversal specification

Token positions are zero-based and intervals are half-open. For query-block width b_q, full-prompt prefill partitions positions into consecutive blocks $[jb_q, \min((j+1)b_q, N))$, $j = 0, \ldots, \lceil N/b_q \rceil - 1$; only the final block is padded, and padded rows do not score branches. Let a be the remote-tree origin, w the local-window length, and q_0 the first valid query in a block, at position t_0. A level-ℓ node with index $u \geq 0$ covers $[a + uCb^\ell, a + (u+1)Cb^\ell)$, with leaf span C and branching factor b. Its children have indices $bu, \ldots, bu + b - 1$. The first query's remote region is $[a, r_0)$, with $r_0 = \max(a, t_0 + 1 - w)$. Its completed prefix is $[a, e)$, where

$$e = a + C \left\lfloor \frac{r_0 - a}{C} \right\rfloor, \qquad L_{\mathcal{Q}} = (e - a)/C. \tag{50}$$

Its residual roots are the canonical base-b decomposition of $L_{\mathcal{Q}}$ complete leaves. Writing $r_\ell = \lfloor L_{\mathcal{Q}}/b^\ell \rfloor \bmod b$, their node indices are

$$\mathcal{R}_\ell = \left\{ b \left\lfloor L_{\mathcal{Q}}/b^{\ell+1} \right\rfloor + j : 0 \leq j < r_\ell \right\}. \tag{51}$$

The roots in Eq. (51) partition the prefix defined by Eq. (50). Later nodes remain stored for subsequent query blocks. For a query q at position t, direct support is $D_q = [0, \min(n_{\text{sink}}, t+1)) \cup [\max(n_{\text{sink}}, t+1-w), t+1)$, where n_{sink} is the sink count; empty intervals contribute no tokens. Quality evaluation uses $a = n_{\text{sink}} = 4$ and $w = 320$; query-block boundaries remain aligned to position zero, independently of a. The raw tail $I_{\text{tail}} = [e, r_0)$ is the leaf-alignment remainder. It joins the expanded leaf tokens for optional token-level top-K_{rem} selection using q_0. The resulting remote set is shared by the block; each row adds D_q and applies its causal mask. With $w \geq b_q$, causal keys within the current query block are already covered by D_q.

Construction and addressing. The prefill builder assigns fixed sign labels, reduces complete leaves and combines corresponding cells up the banks. Append publishes a closed leaf and carries across complete groups without changing old node states. Native KV-head storage is shared under GQA, but each query head keeps its own beam. Algorithm 1 specifies the traversal; root lookup is level-indexed. Raw-tail tokens join the final enumeration; an optional remote-token cap supports budget alignment.

In Algorithm 1, level zero contains leaves and $\ell_{\max}$ is the highest eligible level (zero for an empty prefix). Inputs are valid queries $\mathcal{Q}$, immutable summary banks $\{\mathcal{B}_\ell\}$ for Eq. (6), eligible-prefix residual roots $\{\mathcal{R}_\ell\}$, raw-tail indices I_{tail}, original arrays K, V, beam B, optional remote cap $K_{\text{rem}} \in \mathbb{N} \cup \{\infty\}$, and each row's legal sink/local support D_q. Setting $K_{\text{rem}} = \infty$ disables the cap. $\mathcal{F}_\ell$ and $\mathcal{T}_\ell$ denote the candidate and retained sets at level ℓ; I_{rem} is the remote enumeration and I_q the final support of row q. Expand enumerates leaf tokens; unions identify tokens by their absolute indices. TopB and TopK return the indices with the largest indicated scores. The query-block branch score $\widehat{\mathcal{M}}_{\mathcal{Q}}$ in Eq. (6) uses all valid queries. If the remote enumeration exceeds a finite cap, token-level selection retains the K_{rem} largest raw QK scores for q_0, once per block. $\mathrm{LegalUnique}$ deduplicates the union with sink/local support and keeps causally legal indices for row q.

Algorithm 1 Sparse prefill retrieval and attention (one head, one query block).

$\mathcal{T}_{\ell_{\max}+1} \leftarrow \emptyset$
for $\ell = \ell_{\max}, \dots, 0$ **do**
  $\mathcal{F}_\ell \leftarrow \mathrm{children}(\mathcal{T}_{\ell+1}) \cup \mathcal{R}_\ell$
  if $|\mathcal{F}_\ell| > B$ **then**
    $\mathcal{T}_\ell \leftarrow \mathrm{TopB}(\mathcal{F}_\ell; \widehat{\mathcal{M}}_{\mathcal{Q}})$
  else
    $\mathcal{T}_\ell \leftarrow \mathcal{F}_\ell$
  end if
end for
$I_{\mathrm{rem}} \leftarrow \mathrm{Expand}(\mathcal{T}_0) \cup I_{\mathrm{tail}}$
if $|I_{\mathrm{rem}}| > K_{\mathrm{rem}}$ **then**
  $I_{\mathrm{rem}} \leftarrow \mathrm{TopK}_{K_{\mathrm{rem}}}\left(I_{\mathrm{rem}};\, i \mapsto q_0^\top k_i/\sqrt{d}\right)$
end if
for all $q \in \mathcal{Q}$ **do**
  $I_q \leftarrow \mathrm{LegalUnique}(I_{\mathrm{rem}} \cup D_q; q)$
  $a_q \leftarrow \mathrm{softmax}\left(qK[I_q]^\top/\sqrt{d}\right)$
  $o_q \leftarrow a_q V[I_q]$
end for
return $O = \{o_q : q \in \mathcal{Q}\}$

B.2 Compatible summaries and changes of grouping

For fixed cells, a token belongs to the same cell in every ancestor; parent counts and sums therefore add corresponding child statistics. For Temporal-4 in a base-4 tree, each parent quarter is one complete child, whose total count and sum are recovered by summing its four cells. Both schemes support parent construction.

Temporal-to-angular state conversion. Recursive maintenance requires compatible cell identities for exact matching-cell addition across levels. Changing the grouping can require information that the child states no longer retain. Consider two non-collinear unit hash directions r_1, r_2 and define unit keys

$$v = \frac{r_1 + r_2}{\|r_1 + r_2\|}, \qquad w = \frac{r_1 - r_2}{\|r_1 - r_2\|}. \tag{52}$$

The pair $(v, -v)$ occupies angular cells $(++), (--)$, whereas $(w, -w)$ occupies $(+-), (-+)$. Both pairs have the same count, zero sum and unit key norms. Place the pair inside one temporal quarter and keep all other keys unchanged. Every child Temporal-4 count and sum is identical in the two inputs, but the angular cell counts of the parent differ. Consequently, no function of these temporal child states alone can reconstruct the exact parent Hash-4 state for all keys. With compatible angular cells, the parent instead follows directly from componentwise addition.

A mixed-depth implementation can obtain the missing information through auxiliary angular accumulation or a raw-key reduction at the transition. These are additional maintenance paths that can use transition-local accumulators. Their buffer lifetime and execution cost depend on the implementation. Uniform Hash-4 gives the implementation one compatible state format and one additive update path from leaves to roots.

B.3 Bounded frontier and logarithmic routing

This section counts work within one attention layer. Routing is counted per query head; storage is first counted per distinct KV stream. Let N denote the history length, C the leaf span, $b \geq 2$ the branching factor, $B \geq 1$ the beam bound, m the number of cells, and d the head dimension. A query block contains $1 \leq s \leq b_q$ valid rows, with $b_q = 64$ in the evaluated configuration. Let $L_{\mathrm{leaf}} \leq \lfloor N/C \rfloor$ be the number of complete leaves in the indexed history and $H = 1 + \lfloor \log_b \max(1, L_{\mathrm{leaf}}) \rfloor$ the number of possible levels. The roots visible to a query can cover a shorter completed remote prefix.

Figure 11 separates the number of levels from the width processed at each level. The base-b digits of the eligible prefix's completed-leaf count give at most $b-1$ residual roots at each level. At level ℓ, let $\mathcal{R}_\ell$ be residual roots, $\mathcal{T}_\ell$ retained nodes, and $\mathcal{F}_\ell$ candidates. Algorithm 1 initializes $\mathcal{T}_{\ell_{\max}+1} = \emptyset$. Its level update obeys

$$\mathcal{F}_\ell = \text{children}(\mathcal{T}_{\ell+1}) \cup \mathcal{R}_\ell, \qquad |\mathcal{F}_\ell| \le bB + b - 1 =: F_{\max}. \tag{53}$$

$F_{\max}$ bounds candidate count. Eq. (53) follows by induction: both branches of the algorithm keep $|\mathcal{T}_\ell| \le B$. Each retained node has b children at the next level, and $|\mathcal{R}_\ell| = r_\ell \le b-1$. Thus at every level $|\mathcal{F}_\ell| \le b|\mathcal{T}_{\ell+1}| + |\mathcal{R}_\ell| \le bB + b - 1$, including scoreless levels. Residual roots and descendants of higher roots cover disjoint intervals.

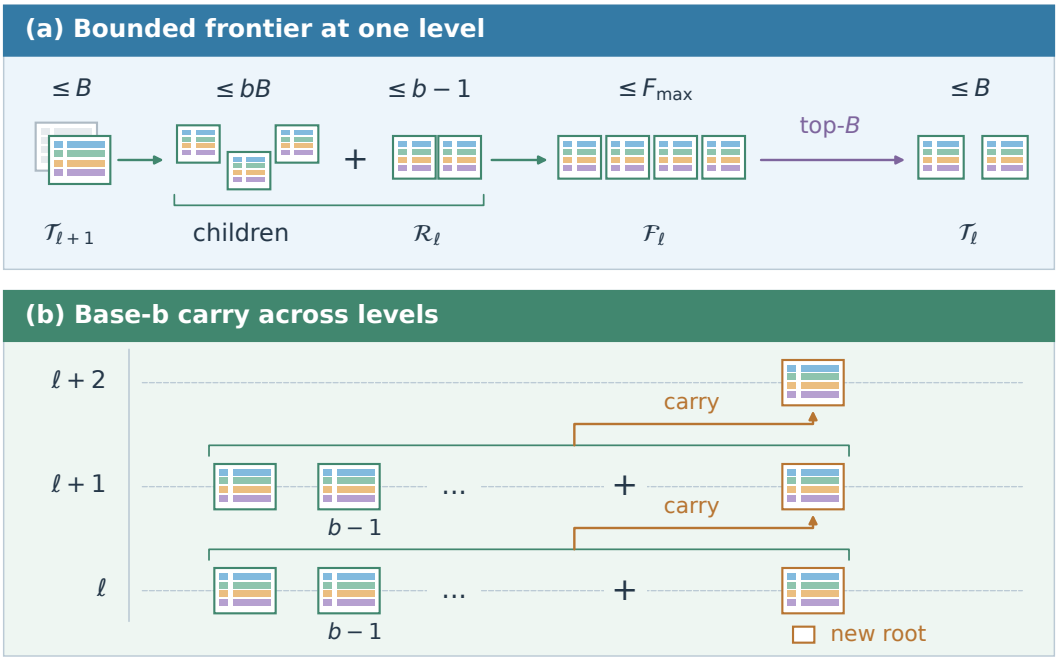


Figure 11: Frontier-width invariant and carry maintenance. (a) $\mathcal{T}_{\ell+1}$ expands into at most bB children; adding $\mathcal{R}_\ell$ gives $|\mathcal{F}_\ell| \le F_{\max} = bB + b - 1$ before retaining $\mathcal{T}_\ell$ of size at most B. (b) Each merge replaces b roots by one parent, reducing the total root count by $b-1$; this supports the amortized update count in Appendix B.4. Orange outlines mark new roots; glyphs denote groups schematically.

One candidate requires sm centroid dot products of length d, followed by cell and query log-sum-exp reductions. With candidate-indexed summary access, scoring costs $O(smd|\mathcal{F}_\ell|)$. A comparison sort gives the conservative selection bound $O(|\mathcal{F}_\ell| \log |\mathcal{F}_\ell|)$; a linear-selection primitive can reduce this term. Level-indexed root lists and compact frontiers add $O(|\mathcal{F}_\ell| + b)$ bookkeeping. Thus

$$T_{\text{route,block}} = O(H\,[smdF_{\max} + F_{\max} \log F_{\max}])\,. \tag{54}$$

Here $T_{\text{route,block}}$ counts arithmetic and indexing work for one query block and head; s, m, d are its valid-query count, cell count and head dimension, respectively. The final partial block is padded to b_q rows in execution, so allocation and timing counts replace s by b_q there. Prefill adds fewer than b_q padded rows in total, preserving Eq. (56). For fixed C, m, b, B, d, b_q, Eq. (54) gives

$$T_{\text{route,block}} = O(\log N). \tag{55}$$

The disjoint prefill blocks partition the N valid queries, so their valid-query counts sum to N, while the number of blocks is $N_{\text{blk}} = \lceil N/b_q \rceil$ for full-block width b_q. Therefore

$$T_{\text{prefill,route}} = O(H\,[NmdF_{\max} + N_{\text{blk}} F_{\max} \log F_{\max}]) = O(N \log N) \tag{56}$$

at fixed budgets. Sharing amortizes traversal and selection; all s query rows still contribute to branch valuation.

Final support and attention. At most B leaves supply at most BC token candidates. The final enumeration also includes raw-tail tokens; an optional cap limits the remote-token count for budget alignment. For a fixed head and query q at position t, Algorithm 1 returns the unique legal support I_q, with $K_{\text{eff}}(t) = |I_q|$. A conservative pre-deduplication allowance is $C-1$ leaf-alignment positions and b_q-1 block-completion positions, in addition to at most $n_{\text{sink}}+w$ sink/local positions. Overlaps count once in I_q. With a fixed padding allowance p_{pad}, let $g = C + b_q - 2 + n_{\text{sink}} + w + p_{\text{pad}}$ and $K_{\text{idx}} \le BC + g$ be the stored index width, including padding. Thus $K_{\text{eff}}(t) \le K_{\text{idx}}$, and both widths are bounded independently of N at fixed settings. Optional raw-token rescoring uses q_0 once per block and visits only expanded leaf tokens and the alignment remainder. Rescoring and selection cost at most $O((BC+g)[d+\log(BC+g)])$ using a comparison sort; row-wise support assembly costs $O(s(BC+g))$. The streaming attention kernel processes the final slots in $O(sK_{\text{idx}}d)$ work per block; only valid slots contribute attention weight. These operations use bounded selected-index arrays. Dense decode in the primary S_p/D deployment retains $O(Nd)$ attention work per query head.

B.4 Construction, storage and online updates

Let V_{node} count completed nodes across all levels, including stored descendants of current roots. Then

$$V_{\text{node}} = \sum_{\ell=0}^{H-1} \left\lfloor \frac{L_{\text{leaf}}}{b^\ell} \right\rfloor \le \frac{b}{b-1} L_{\text{leaf}}. \tag{57}$$

Every node stores m counts and m length-d key sums, or an equivalent span-normalized representation. Routing storage is consequently $O(V_{\text{node}} m(d+1))$ scalars per KV head; raw K/V additionally require $2Nd$ scalars per KV head. Auxiliary node means and metadata add $O(V_{\text{node}} d)$ and $O(V_{\text{node}})$ storage. With h_{kv} KV heads, these storage terms are multiplied by h_{kv} for KV-head-indexed storage; scoring is multiplied by the query-head count. The execution stores one bank per KV head and maps each query head to that bank when gathering candidates.

For sign cells, assigning a key costs $O(d \log m)$ and updating its assigned sum costs $O(d)$. Initializing leaf states costs $O(L_{\text{leaf}} m(d+1))$. Each internal node merges b child states in $O(bm(d+1))$. There are $V_{\text{node}} - L_{\text{leaf}}$ such internal nodes. At fixed b, C, m, d, h_{kv}, Eq. (57) therefore gives

$$T_{\text{build}} = O(N), \qquad S_{\text{route}} = O(N), \tag{58}$$

where T_{build} is construction work and S_{route} is routing-state storage. No parent construction rescans the raw keys covered by that parent.

For online construction starting from an empty forest, let T leaves close. A level-ℓ merge occurs once per b^ℓ leaves, so the total internal merges satisfy

$$\sum_{\ell \ge 1} \left\lfloor \frac{T}{b^\ell} \right\rfloor \le \frac{T}{b-1}. \tag{59}$$

Here T is the number of leaf closures over the entire update sequence; the summand counts level-ℓ merges. One closure can trigger $O(H)$ merges; Eq. (59) gives at most $1/(b-1)$ amortized merges per completed leaf. The implementation stages C raw keys and reduces a leaf in $O(Cmd)$ work. Assuming constant-time indexed writes to preallocated storage, or bounded-size chunks allocated without copying existing banks, the append bound is

$$O(Cmd + bm(d+1)H) \tag{60}$$

in the worst case and

$$O\left(md + \frac{bm(d+1)}{C(b-1)}\right) \tag{61}$$

amortized per token. At fixed dimensions and budgets, Eqs. (60) and (61) give $O(\log N)$ worst-case and $O(1)$ amortized updates. Updating a node never rescans its covered raw keys: compatible child statistics determine its state. For a prebuilt prefix, let r be the total number of residual roots at the append endpoint and use potential $\Phi = r/(b-1)$ in units of merge work. An update closing one leaf and performing k merges has $\Delta\Phi = 1/(b-1) - k$, so $k + \Delta\Phi = 1/(b-1)$. The initial $O(H)$ credit is charged to construction. These roots belong to the full append state; $\mathcal{R}_\ell$ in Algorithm 1 can describe an earlier, query-eligible prefix.

C Evaluation Protocols and Complete Results

This appendix specifies what each measurement evaluates. Benchmark examples measure complete model behavior; branch comparisons use fixed Dense states. Their sampling and uncertainty units are given separately.

C.1 Recursive measurements and estimation protocol

Fixed-window sampling and aggregation are specified in Appendix C.6; here we define recursive-frontier metrics and the common fitting, validation, and held-out protocol. Fixed windows compare two predetermined $L/2$-token branches using one final query per layer/head (Appendix C.6). Fidelity is the fraction whose estimated larger branch agrees with exact normalized mass, with left ties. Binary mass quality divides the selected branch's mass by the larger branch mass. These windows sample representation across span independently of a search trajectory.

Recursive measurements use the actual Q64 forest and competitive frontiers, with node span Cb^{ℓ}, $C = 16$, $b = 4$, $B = 64$. The deployed Hash-4 trajectory supplies identical candidates to all scorers in the matched-frontier comparison. Every scorer sees the same candidate nodes, valid query rows and top-B rule. The candidate set may contain children of several retained parents and residual roots. Its quality $Q_{\mathcal{C}}$ divides selected *joint Q64 raw mass* by exact top-64 mass in that same set. At 32K the competing spans are 256, 64 and 16; a window $L = 512$ has two span-256 children but neither the population nor the decision budget of a span-256 frontier. Tables 10 and 11 supply the recursive data alongside the fixed-window figures.

Sampling and estimation protocol. The common Qwen3-1.7B Dense-32K pool has 20 prompts per task across 13 tasks. Of these, 13 prompts fit predictors and define feature transforms and margin/drift bins; 13 provide initial validation; the remaining 234 form the held-out evaluation. Each uses layers 2/14/26, heads 0/8 (zero-based), and four Q64 blocks at fractions 1/8, 1/2, 7/8 and 1 of the available block-index range, rounded down: 24 cases per prompt. The held-out set contains 5,616 cases and 12,636 competitive frontiers: 5,616/4,212/2,808 at spans 16/64/256. Means first average within prompts and then equally across tasks; intervals use 4,000 paired prompt resamples within tasks. Pooled summaries include all 260 prompts, including the fitting subset, and are descriptive.

For gap and merge measurements, eight evenly spaced candidate slots per competitive frontier supply branches, scored with the block's last valid query. Selection metrics use all valid Q64 rows. The grouping control uses 16 random membership permutations preserving cell sizes. Residual predictors use standardized $\log(1 + x)$ features, ridge penalty one and an unpenalized intercept. The two feature sets are within-cell variance alone and variance plus the largest positive within-cell residual. Feature transforms, coefficients and bins are fixed on the fitting subset; held-out intervals condition on those fitted models.

Matched-frontier comparisons evaluate all scorers on identical frozen states and candidate sets. Independent searches then allow each scorer to determine subsequent frontiers. Exact-node search uses exact joint mass in each visited node; the global leaf oracle directly selects the best 64 equal-cost remote leaves. The former is a greedy policy, the latter the reference for Q_{leaf}. Sink/local/tail unions and causal masks are common. These frozen-state analyses retain up to 64 complete span-16 remote leaves, with sink/local/tail positions added separately.

Table 10: Same-frontier selection on 234 held-out 1.7B Dense-32K prompts. Entries are $100Q_{\mathcal{C}}$ and 95% prompt-bootstrap intervals. All scorers retain 64 nodes from identical Hash-4 frontiers; noncompetitive cases are excluded.

Summary	Span 16	Span 64	Span 256
Center-1 (HiP)	73.93 [73.76,74.10]	77.15 [76.95,77.35]	91.34 [91.17,91.50]
CarryKV-Mean-1	90.73 [90.63,90.83]	93.71 [93.60,93.82]	98.18 [98.11,98.25]
CarryKV-Position-Hash-4	91.96 [91.87,92.06]	93.92 [93.81,94.02]	98.20 [98.13,98.27]
CarryKV-Temporal-4	93.42 [93.32,93.51]	94.55 [94.46,94.65]	98.48 [98.41,98.55]
CarryKV-Hash-4	91.71 [91.61,91.81]	93.89 [93.79,93.99]	98.34 [98.29,98.40]

Table 11: Complete independent searches on the same 5,616 frozen-state cases. Q_{leaf} uses the global top-64 leaf mass and P_S the full-prefix mass as denominator. NL2 is $\|o_S - o\|_2/\|o\|_2$, evaluated per query before aggregation. Entries include 95% prompt-bootstrap intervals. Exact-node is the greedy reference.

Summary	$100Q_{\mathrm{leaf}} \uparrow$	$100P_S \uparrow$	Output NL2 $\downarrow$
Center-1 (HiP)	67.28 [67.07,67.49]	82.39 [82.30,82.49]	0.2197 [0.2184,0.2209]
CarryKV-Mean-1	87.41 [87.27,87.55]	84.67 [84.58,84.76]	0.1906 [0.1896,0.1916]
CarryKV-Position-Hash-4	88.52 [88.39,88.65]	84.69 [84.61,84.78]	0.1886 [0.1876,0.1896]
CarryKV-Temporal-4	90.43 [90.30,90.56]	84.89 [84.81,84.97]	0.1874 [0.1864,0.1884]
CarryKV-Hash-4	88.31 [88.17,88.46]	84.69 [84.60,84.77]	0.1821 [0.1811,0.1831]
Exact-node	99.88 [99.87,99.88]	85.43 [85.35,85.51]	0.1744 [0.1733,0.1754]

C.2 Recursive selection and target-path survival

We move from local same-frontier selection to complete independent searches, then to cumulative target survival across successive competitive stages. Mean-1 substantially improves on the Center-1 positional representative. Hash-4's smaller gains over Mean-1 have positive paired intervals at all three competitive spans; Temporal-4 has higher $Q_{\mathcal{C}}$ throughout. Table 12 reports paired selection effects. In complete searches, Hash-4 improves Q_{leaf}, P_S and output NL2 over Mean-1. Temporal-4 exceeds Hash-4's P_S by 0.203 [0.191,0.215] points, while Hash-4's NL2 is lower by 0.00531 [0.00501,0.00561]. The value-direction term in Appendix A.1 distinguishes mass retention from realized output error.

A fixed target population through every stage. For each case let $\mathcal{L}^*$ be the global top-64 remote leaves, W_ℓ their exact joint Q64 mass, and A_h the leaves still reachable after stage h. Target-mass survival is

$$R_h = \frac{\sum_{\ell \in \mathcal{L}^* \cap A_h} W_\ell}{\sum_{\ell \in \mathcal{L}^*} W_\ell}. \tag{62}$$

Higher R_h means more of the initial oracle target mass remains reachable. All 5,616 cases appear before selection and after each of spans 256/64/16. Pending roots survive until injected; stages without competition carry forward the previous value. Thus successive values have the same denominator and population, including short-prefix cases. Only actual support-discarding competitions can reduce R_h. Table 13 shows that Hash-4 retains 79.105% of target mass, Mean-1 78.286%, and Temporal-4 82.238%. Hash-4 loses 8.103% above leaves and another 12.792% at leaf competition.

Table 12: Recursive selection on the held-out set and full pool. Mean, Center, Hash and Temporal denote Mean-1, Center-1, Hash-4 and Temporal-4. All Q_C differences are percentage points; the node span follows the metric. Intervals use 4,000 paired prompt resamples within 13 fixed tasks. Each entry gives an estimate above its interval. Sample splits and pooling: Appendix C.1.

Contrast / metric	Fit + validation (26)	Held-out (234)	Full pool (260)
Mean–Center	17.281	16.807	16.855
Q_C, span 16	[17.081,17.476]	[16.643,16.977]	[16.707,17.002]
Mean–Center	16.249	16.557	16.526
Q_C, span 64	[15.901,16.596]	[16.375,16.747]	[16.349,16.704]
Mean–Center	6.291	6.836	6.782
Q_C, span 256	[5.914,6.671]	[6.675,7.002]	[6.628,6.939]
Hash–Mean	0.939	0.980	0.975
Q_C, span 16	[0.751,1.123]	[0.888,1.070]	[0.887,1.061]
Hash–Mean	0.031	0.183	0.168
Q_C, span 64	[-0.186,0.247]	[0.114,0.252]	[0.098,0.236]
Hash–Mean	0.084	0.163	0.155
Q_C, span 256	[-0.015,0.186]	[0.117,0.204]	[0.113,0.197]
Hash–Temporal	-1.609	-1.704	-1.695
Q_C, span 16	[-1.855,-1.367]	[-1.797,-1.609]	[-1.784,-1.606]
Hash–Temporal	-0.784	-0.664	-0.676
Q_C, span 64	[-0.949,-0.615]	[-0.733,-0.593]	[-0.747,-0.609]
Hash–Temporal	-0.217	-0.134	-0.143
Q_C, span 256	[-0.320,-0.111]	[-0.181,-0.087]	[-0.187,-0.097]
Hash–Mean	0.806	0.901	0.892
Q_{leaf} (pp)	[0.570,1.033]	[0.813,0.989]	[0.804,0.976]
Hash–Mean	0.009	0.020	0.019
P_S (pp)	[-0.038,0.054]	[0.005,0.035]	[0.004,0.033]
Hash–Mean	-0.00786	-0.00856	-0.00849
Output NL2	[-0.00865,-0.00707]	[-0.00885,-0.00827]	[-0.00878,-0.00821]

Its terminal advantage over Mean-1 is 0.819 [0.723,0.918] points; the disadvantage to Temporal-4 is 3.133 [3.017,3.250] points. Exact-node's near-unit Q_{leaf} measures the recoverable scoring gap in these states; its selected target intersection, and hence survival, is a distinct ratio.

Table 13: Fixed-population target-mass survival (↑), in percentages with 95% prompt-bootstrap intervals. All 5,616 cases share the initial global top-64 leaf target as denominator throughout their trajectories; initial survival is 100%.

Summary	After span 256	After span 64	After span 16
Center-1 (HiP)	95.33 [95.16,95.48]	78.97 [78.72,79.23]	54.51 [54.27,54.74]
CarryKV-Mean-1	98.48 [98.41,98.55]	91.42 [91.27,91.57]	78.29 [78.11,78.46]
CarryKV-Position-Hash-4	98.50 [98.43,98.57]	91.60 [91.45,91.75]	79.64 [79.46,79.81]
CarryKV-Temporal-4	98.75 [98.68,98.82]	92.50 [92.37,92.64]	82.24 [82.07,82.40]
CarryKV-Hash-4	98.60 [98.54,98.67]	91.90 [91.76,92.04]	79.11 [78.92,79.29]
Exact-node	99.88 [99.87,99.89]	99.14 [99.11,99.18]	99.14 [99.11,99.18]

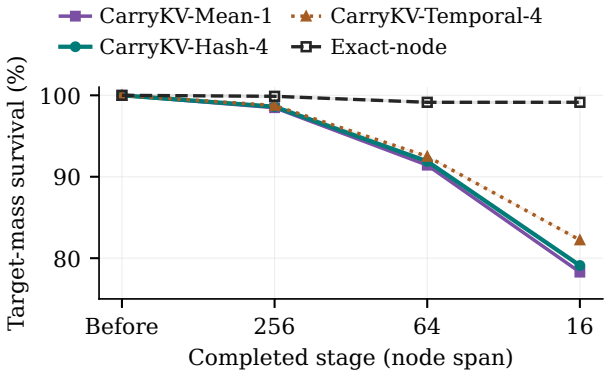


Figure 12: Cumulative target-mass survival through independent searches on the same 5,616 cases. Each stage uses the initial global top-64 leaf target mass as denominator; pending roots and noncompetitive cases remain alive. Bands are paired 95% within-task prompt-bootstrap intervals. Table 13 reports all five summaries and Exact-node; this view compares Mean-1 with directional and temporal grouping.

Conditional path identity. For a target outside guaranteed support, let D count actual irreversible competitions, E_h mean survival through the hth, and E_0 be the certain event. Nested events give

$$\Pr(E_D) = \prod_{h=1}^{D} \Pr(E_h \mid E_{h-1}) \tag{63}$$

when conditioning events have positive probability; an earlier zero probability already forces final recall to zero. Expanding every token in selected leaves gives target recall $Q_N = \Pr(E_D)$; further token filtering can reduce it. If each conditional retention is at most $1 - \varepsilon$ for fixed $0 < \varepsilon < 1$, then

$$Q_N \le (1 - \varepsilon)^D, \tag{64}$$

without cross-level independence. In a complete b-ary tree of height H_{tree} with $N = Cb^{H_{\text{tree}}}$, beam $B = b^r$ for a nonnegative integer r, and $N > BC$, the number of competitive levels is $D = H_{\text{tree}} - r$, giving

$$Q_N \le (BC/N)^{-\log_b(1-\varepsilon)}. \tag{65}$$

The persistent conditional error is an assumption of Eq. (65). Fixed-window fidelity is a different sampling probability; the empirical survival in Eq. (62) uses actual stages.

Cutoff margins and selection error. To examine individual competitions along these paths, we stratify frontiers by their exact selection margin. The low/middle/high margin bins defined in Appendix C.1 yield mean rescue/harm counts of 3.399/3.253, 3.260/2.914 and 2.335/1.852 oracle nodes per frontier. Rescued/harmed oracle-normalized masses are .04091/.03742, .03948/.03256 and .02585/.01972; the net gains have paired intervals [.168,.521], [.580,.806] and [.476,.753] points. Hash-4's mean $e^{-\Omega_C}$ is .117/.110/.120 at spans 16/64/256, versus $Q_C = .917/.939/.983$; no frontier meets the strict cutoff certificate. Eq. (48) is loose on this population.

C.3 Models, positions and attention interfaces

Qwen3-1.7B-Base and Qwen3-8B-Base have native context 32,768, RoPE base 10^6, head dimension 128 and eight KV heads. The models respectively have 28/36 layers and 16/32 query heads. The native representation experiment uses the longest unscaled checkpoint context; the exact and approximate scores share the same post-RoPE states. Extended evaluations use external static YaRN factors 2/4 at 64K/128K, with amplitudes 1.0693/1.1386. The Qwen3 report distinguishes Base pretraining from its post-trained long-context evaluations (Yang et al., 2025); our comparisons use the Base checkpoints throughout.

Fixed direction table. Hash cells use the first $\log_2 m$ rows of a three-by-128 FP32 table. A CPU PyTorch generator with seed 20260829 draws independent standard Gaussian entries; each row is divided by its Euclidean norm, without orthogonalization. A nonnegative projection sets the corresponding bit. Drawing all three rows before selecting a prefix makes Hash-2/4/8 nested. The table is shared across nodes, layers and heads, and the same seed and head dimension specify both Qwen3 scales and their position regimes. Centroids themselves remain arithmetic means without renormalization. The experiments use this fixed table; direction-table sensitivity is a separate source of variation from prompt uncertainty.

Baseline settings. The HiP and InfiniteHiP implementations come from the same public repository.[2] All methods receive post-normalization, post-RoPE Q/K and unrotated V, preserve GQA head mapping, and reset the contiguous KV cache between prompts. All sparse methods use sparse prefill followed by dense decode. Prefill uses sparse attention in all 28/36 layers, and decode is dense in all layers. The comparisons use uncorrected outputs and the selectors' own support, with auxiliary AVD/VD masks disabled.

HiP's original branch valuation (Lee et al., 2025a) uses the interval's center key; Center-1 isolates that rule in the controlled analyses. The evaluated block configuration uses Q32, key blocks of two and default lower-resolution search. Original HiP retains approximately K/b_k search intervals, where K is the remote token budget and b_k is key-block size; this count differs from CarryKV's beam of span-16 leaves.

InfiniteHiP uses landmark-on screening. Its 256/32/8-token chunk stages use query strides 4/4/1 within Q64 blocks, so the first two stages sample 16 rows and the last uses all 64. Scores aggregate query/landmark comparisons by a maximum. Intermediate budgets cover 32K and 8K tokens, followed by the final nominal token budget of 1,024.

Budget composition. All three methods use four sink tokens and a 320-token local window. CarryKV caps remote tokens at 1,020; HiP and InfiniteHiP use 1,024, aligned to multiples of 64. Sink and local tokens are additional.

The sparse full-prompt pass produces the first generated-token logits; subsequent one-token calls use dense attention. This is the same S_p/D phase boundary for all three sparse methods.

C.4 Benchmark pools and scoring

We first define the shared benchmark pools and uncertainty units, then specify scoring for RULER, LongBench-E, and PG19. All task macros weight the 13 tasks equally. Paired benchmark intervals use 5,000 resamples of examples within each fixed task, preserving the method pairing. They quantify example uncertainty on this task set.

Table 14: Benchmark samples shared by Dense and sparse methods within each model and context/input view. Counts concern benchmark examples.

Result	Examples	Aggregation
1.7B, 4K–16K	13×200	Task macro
1.7B, 32K–128K	13×100	Task macro
8B, 32K–128K	13×100	Task macro
LongBench-E, each view	13×20	Task macro
PG19, each model	100-book stream	Total NLL / scored tokens

RULER uses the official task metrics and greedy full generation,[3] with the same prompt, generation and stopping rules across methods. The 1.7B sparse 32K and 64K paired CarryKV–InfiniteHiP intervals are [14.50,18.33] and [19.46,23.15] percentage points; the full six-length intervals are given in Table 15. The 8B common-pool intervals are in Table 16.

[2] https://github.com/DeepAuto-AI/hip-attention

[3] https://github.com/NVIDIA/RULER

Table 15: Qwen3-1.7B paired RULER differences, CarryKV minus InfiniteHiP. Intervals bootstrap examples within the 13 fixed tasks.

Context	Difference (pp)	95% interval
4K	+3.08	[1.99,4.18]
8K	+2.59	[1.31,3.83]
16K	+10.35	[9.03,11.71]
32K	+16.44	[14.50,18.33]
64K	+21.32	[19.46,23.15]
128K	+3.12	[2.09,4.18]

Table 16: Paired 8B differences. RULER uses native 32K and 100 examples/task; LongBench-E uses Full and 20/task. Intervals resample examples within the fixed 13 tasks.

Benchmark	Contrast	Difference (pp)	95% interval
RULER	CarryKV minus HiP	13.37	[11.11, 15.69]
RULER	CarryKV minus InfiniteHiP	6.80	[4.66, 8.99]
RULER	CarryKV minus Dense	-19.11	[-21.10, -17.23]
LongBench-E	CarryKV minus HiP	1.32	[-0.50, 3.26]
LongBench-E	CarryKV minus InfiniteHiP	1.25	[-1.28, 3.85]
LongBench-E	CarryKV minus Dense	-0.49	[-2.20, 1.16]

Table 17: CarryKV minus HiP on common examples. RULER has 200 examples/task at 1.7B/4K–16K and 100 otherwise; LongBench-E has 20/task. Intervals use 5,000 paired example resamples within the fixed 13 tasks.

Benchmark	Model	Context / view	Difference (pp)	95% interval
RULER	1.7B	4K	6.79	[5.60, 7.96]
RULER	1.7B	8K	6.25	[5.08, 7.41]
RULER	1.7B	16K	9.39	[8.12, 10.64]
RULER	1.7B	32K	2.86	[1.02, 4.72]
RULER	1.7B	64K	4.02	[2.46, 5.60]
RULER	1.7B	128K	-0.23	[-1.32, 0.83]
RULER	8B	64K	9.50	[7.32, 11.64]
RULER	8B	128K	-0.04	[-1.80, 1.70]
LongBench-E	1.7B	4K cap	0.06	[-2.23, 2.32]
LongBench-E	1.7B	8K cap	0.61	[-2.13, 3.48]
LongBench-E	1.7B	Full	0.43	[-2.30, 3.27]

LongBench-E. We use common source examples across methods and input caps, with the LongBench-E task metrics.[4] Full inputs have median 4,027 and maximum 10,337 Qwen3 tokens; five exceed 8,192. Thus 16K and 32K caps produce the same inputs and predictions as Full. The 1.7B CarryKV–InfiniteHiP 95% paired intervals are [-0.11,4.09], [-1.02,3.29] and [-1.01,3.30] for 4K cap, 8K cap and Full. The 8B Full comparison is generated from the same formal experiment as its task breakdown in Table 18.

[4] https://github.com/THUDM/LongBench

Table 18: Qwen3-8B LongBench-E Full, 20 examples per task. Common IDs and uncorrected sparse prefill/dense decode.

Task	Dense	HiP	InfiniteHiP	CarryKV
qasper	38.92	30.45	29.26	**31.66**
multifieldqa_en	61.10	57.96	57.18	**61.26**
hotpotqa	59.58	57.92	57.92	**60.95**
2wikimqa	54.58	55.77	54.35	**58.48**
gov_report	34.28	32.72	34.37	**34.91**
multi_news	27.30	27.22	**28.15**	27.35
trec	55.00	**65.00**	60.00	60.00
triviaqa	97.50	**97.50**	**97.50**	**97.50**
samsum	36.30	34.50	34.52	**34.64**
passage_count	40.00	25.00	20.00	**35.00**
passage_retrieval_en	100.00	95.00	**100.00**	**100.00**
lcc	69.30	68.85	**73.15**	66.65
repobench-p	61.85	64.20	**66.60**	60.90
Macro	56.59	54.78	54.85	**56.10**

PG19. The official 100-book test stream[5] is joined by two newlines and tokenized once, giving 9,974,638 tokens; both models share this tokenizer. Consecutive nonoverlapping full windows of N tokens are evaluated, discarding the final incomplete window at each length. Labels at positions 1 through $N-1$ are scored, and vocabulary projection is processed in 512-row chunks. PPL is $\exp(\sum \mathrm{NLL}/\sum \text{scored tokens})$. Window counts at 4K/8K/16K/32K/64K/128K are 2,435/1,217/608/304/152/76. All methods use the same windows at each length.

This is full-window teacher forcing with block-shared selection: CarryKV/InfiniteHiP use Q64 and HiP uses Q32. Within a block, selection can depend on queries after the scored position. These PPLs characterize that block approximation, whereas strict prefix-by-prefix likelihood requires independently prefix-available selection. The protocol is held fixed across the reported windows. Different lengths resegment the test stream and omit different incomplete suffixes.

C.5 Representation families in complete generation

The branch analyses separate summary capacity from grouping allocation; here we test the same representation substitutions inside the complete CarryKV generation pipeline. Table 19 replaces the summary within the same 8B CarryKV retrieval framework. Mean-1 preserves one conditional mean; Hash-4 uses four fixed angular cells; Temporal-4 uses four contiguous quarters. All share Q64, B64, C16, base four, the raw support rule and uncorrected sparse-prefill/dense-decode protocol. The samples are the common examples at 32K and 64K, 100 examples per task across 13 RULER tasks.

Table 19: Summary substitution within CarryKV on Qwen3-8B: 13 RULER tasks with 100 common examples/task and length, using the main uncorrected S_p/D configuration. Upper entries are macros; lower entries are paired differences (pp) and 95% within-task example-bootstrap intervals. Hash-4 matches Table 1.

Summary / contrast	32K native	64K YaRN2
CarryKV-Mean-1	63.11	45.36
CarryKV-Hash-4	69.16	51.76
CarryKV-Temporal-4	73.43	55.28
CarryKV-Hash-4	+6.06	+6.39
minus CarryKV-Mean-1	[4.39,7.70]	[4.69,8.08]
CarryKV-Temporal-4	+4.26	+3.52
minus CarryKV-Hash-4	[2.56,5.98]	[1.74,5.27]

[5] https://github.com/google-deepmind/pg19

Table 20: Complete 8B RULER task breakdown for Table 19; 100 common examples per task and length. M/H/T denote Mean-1/Hash-4/Temporal-4.

Task	32K M	32K H	32K T	64K M	64K H	64K T
cwe	77.80	86.20	86.60	31.10	47.20	46.20
fwe	95.67	95.33	95.00	90.00	89.33	90.00
niah_multikey_1	51.00	61.00	65.00	21.00	33.00	35.00
niah_multikey_2	17.00	23.00	32.00	4.00	6.00	9.00
niah_multikey_3	15.00	25.00	25.00	0.00	2.00	1.00
niah_multiquery	87.25	90.25	93.50	53.00	60.75	68.00
niah_multivalue	93.25	93.75	96.25	61.00	68.75	73.25
niah_single_1	100.00	100.00	100.00	100.00	100.00	100.00
niah_single_2	51.00	72.00	80.00	43.00	50.00	60.00
niah_single_3	57.00	69.00	82.00	39.00	54.00	70.00
qa_1	45.00	52.00	57.00	32.00	43.00	42.00
qa_2	39.00	40.00	46.00	28.00	32.00	33.00
vt	91.40	91.60	96.20	87.60	86.80	91.20

Hash-4 improves Mean-1 by 6.06 [4.39,7.70] and 6.39 [4.69,8.08] points at 32K and 64K. Temporal-4 further improves Hash-4 by 4.26 [2.56,5.98] and 3.52 [1.74,5.27] points. Table 20 retains every task. These comparisons examine summary capacity and grouping allocation within the same hierarchy (Section 5.2; Appendix A.10). Cross-method system comparisons use Hash-4.

Temporal-4 uses balanced, nonempty contiguous quarters for complete nodes. It retains temporal score structure before the exponential readout and can therefore benefit when nearby keys have similar query projections. The fixed-window and recursive comparisons show where this grouping is effective. Determining how much of that advantage comes from occupancy, temporal dependence or query geometry requires separate interventions. Hash-4 uses post-RoPE directional labels that remain fixed across merges; Temporal-4 reconstructs each parent quarter from a child total.

C.6 Representation sampling and aggregation

This section specifies fixed-window representation sampling; recursive frontiers and predictor fitting follow Appendix C.1. The native-32K pool contains 260 prompts, 20 per RULER task stratum, shared between models. Each YaRN64K/128K pool contains 130 prompts, ten per task, also shared between models. Each prompt supplies its final Dense query from every layer and attention head. All summary methods in a case see the same query/key states and exact branch pairs.

Table 21: Representation inputs, separate from benchmark sample counts. Each prompt contributes fixed final-query states from every listed layer/head; uncertainty resamples source prompts, not heads.

Model	Context / positions	Prompts	Layers $\times$ heads
1.7B	32K / native	260	28×16
	64K / YaRN2	130	28×16
	128K / YaRN4	130	28×16
8B	32K / native	260	36×32
	64K / YaRN2	130	36×32
	128K / YaRN4	130	36×32

Window L denotes two adjacent branches of $L/2$ tokens. The full-range native sweep uses geometric windows from 32 through 16K; up to 64 evenly positioned nonoverlapping windows are sampled per final-query head. The 32K/128K coarse sweeps use seven windows from $N/8$ through $N/2$ in increments of $N/16$; the 64K sweep additionally samples 10K and 14K, giving nine points. Each samples complete nonoverlapping windows. These regimes are plotted separately. At $L = N/2$ the recorded pair begins at position zero, so its spatial coverage differs from smaller windows. This window coordinate differs from a deployment node span $16 \times 4^{\ell}$ and from key-pair distance Δ.

Center-1 uses the lower midpoint; Mean-1 is an arithmetic mean. Hash-2/4/8 use nested prefixes of one direction table. Position-Hash-4 uses a fixed integer hash of the absolute position; Temporal-4 uses four contiguous quarters. Centroids are not re-normalized. Exact normalized log-mass on raw QK is the reference. Branch-order fidelity retains its left-tie rule. The fixed-window gap, rescue and harm measurements use stored FP32 logits and 5,000 paired within-task prompt resamples. For error/selection metrics, exact-mass ties are excluded with a common mask across all states. In each remaining pair, $\Omega = |\delta_A - \delta_B|$ and $Q = \exp(\mathcal{M}_{\text{selected}} - \max(\mathcal{M}_A, \mathcal{M}_B))$. Means average windows within layer/head, then layer/heads within prompt, then prompts within each equally weighted task. Bands bootstrap prompts within tasks. Paired-gain intervals use paired prompt effects, not differences between independent CI endpoints.

Position-Hash-4 labels. Both fixed-window and recursive analyses hash the absolute token position i, without resetting at a branch or subtracting the tree origin. With seed $\sigma = 20260829$, the four-cell label is

$$\begin{aligned} x_0 &= i + \sigma, \\ x_1 &= \mathrm{s64}\big((x_0 \oplus (x_0 \gg 30))c_1\big), \\ x_2 &= \mathrm{s64}\big((x_1 \oplus (x_1 \gg 27))c_2\big), \\ \mathrm{cell}(i) &= (x_2 \oplus (x_2 \gg 31)) \,\&\, 3, \end{aligned} \tag{66}$$

where $c_1 = -4658895280553007687$ and $c_2 = -7723592293110705685$. Here $\oplus$ is bitwise XOR, $\gg$ is an arithmetic right shift, and $\mathrm{s64}$ wraps modulo 2^{64} and interprets the result as a signed two's-complement integer. The final bitwise AND selects the lowest two bits. The same label is used at every level and head.

C.7 Complete task results

This section reports task-level results underlying the main-table macros and paired contrasts. The following tables retain the full 13-task RULER and 1.7B LongBench-E breakdowns; Table 18 gives 8B LongBench-E. They include the single-needle and code-completion results for each method. Where bold is used, it marks the largest measured sparse score in that row. The main PPL table contains every evaluated length for both models.

Table 22: Qwen3-1.7B RULER task scores across contexts. All listed sparse methods share 200 examples/task at 4K–16K and 100/task at 32K–128K; all 13 tasks are retained.

Task	Method	4K	8K	16K	32K	64K	128K
niah_single_1	HiP	100.00	100.00	100.00	100.00	97.00	10.00
	InfiniteHiP	100.00	100.00	97.50	72.00	30.00	10.00
	CarryKV	100.00	100.00	100.00	100.00	100.00	16.00
niah_single_2	HiP	100.00	99.50	86.50	64.00	18.00	2.00
	InfiniteHiP	100.00	98.50	64.50	25.00	3.00	0.00
	CarryKV	100.00	100.00	91.50	57.00	28.00	2.00
niah_single_3	HiP	100.00	98.50	95.00	85.00	73.00	0.00
	InfiniteHiP	100.00	100.00	99.50	83.00	11.00	0.00
	CarryKV	100.00	100.00	100.00	94.00	61.00	0.00
niah_multikey_1	HiP	93.00	71.00	45.50	24.00	17.00	4.00
	InfiniteHiP	98.00	68.00	51.00	15.00	5.00	1.00
	CarryKV	95.00	82.00	67.00	32.00	21.00	3.00
niah_multikey_2	HiP	52.50	19.50	10.50	5.00	1.00	0.00
	InfiniteHiP	84.50	44.50	12.50	3.00	0.00	0.00
	CarryKV	79.50	45.00	18.50	6.00	0.00	0.00
niah_multikey_3	HiP	64.50	16.50	3.50	0.00	0.00	0.00
	InfiniteHiP	47.50	11.50	1.00	0.00	0.00	0.00
	CarryKV	70.00	18.50	4.50	0.00	0.00	0.00
niah_multivalue	HiP	96.62	95.88	84.75	67.50	27.25	1.00
	InfiniteHiP	98.12	96.00	85.00	43.25	3.25	0.25
	CarryKV	97.50	96.50	91.88	72.00	17.00	0.25
niah_multiquery	HiP	97.25	93.50	88.25	61.25	21.25	1.75
	InfiniteHiP	97.62	91.25	64.25	38.00	4.00	0.00
	CarryKV	96.62	94.12	86.62	59.50	23.25	2.25
vt	HiP	46.10	40.60	16.80	33.20	6.60	0.00
	InfiniteHiP	78.40	69.10	42.10	0.00	0.00	0.00
	CarryKV	90.80	66.90	69.60	55.80	59.40	0.00
cwe	HiP	85.20	17.30	1.60	1.20	1.60	1.10
	InfiniteHiP	82.70	27.75	9.80	2.90	0.90	0.70
	CarryKV	86.15	21.55	7.15	4.30	1.60	1.00
fwe	HiP	81.67	82.83	79.17	82.33	66.33	23.00
	InfiniteHiP	73.67	74.17	73.33	66.67	47.00	7.33
	CarryKV	80.00	82.33	78.33	79.00	63.00	22.33
qa_1	HiP	54.00	28.00	19.00	16.00	13.00	9.00
	InfiniteHiP	57.00	28.50	23.00	20.00	9.00	3.00
	CarryKV	60.50	34.00	30.50	19.00	15.00	9.00
qa_2	HiP	39.00	31.00	26.00	29.00	14.00	20.00
	InfiniteHiP	40.50	32.50	20.50	23.00	18.00	6.00
	CarryKV	42.00	34.50	33.00	27.00	19.00	13.00
Macro	HiP	77.68	61.09	50.51	43.73	27.39	5.53
	InfiniteHiP	81.39	64.75	49.54	30.14	10.09	2.18
	CarryKV	84.47	67.34	59.89	46.58	31.40	5.29

Table 23: Qwen3-8B RULER task scores across contexts. Common examples within each length; all 13 tasks are retained. Each length uses 100 examples/task, with the same inputs for the measured methods.

Task	Method	32K	64K	128K
niah_single_1	Dense	100.00	100.00	100.00
	HiP	100.00	100.00	100.00
	InfiniteHiP	98.00	99.00	98.00
	CarryKV	100.00	100.00	100.00
niah_single_2	Dense	100.00	96.00	81.00
	HiP	43.00	43.00	26.00
	InfiniteHiP	84.00	71.00	12.00
	CarryKV	72.00	50.00	19.00
niah_single_3	Dense	100.00	98.00	85.00
	HiP	53.00	34.00	36.00
	InfiniteHiP	78.00	49.00	27.00
	CarryKV	69.00	54.00	24.00
niah_multikey_1	Dense	94.00	84.00	58.00
	HiP	34.00	26.00	15.00
	InfiniteHiP	57.00	31.00	10.00
	CarryKV	61.00	33.00	9.00
niah_multikey_2	Dense	96.00	70.00	37.00
	HiP	8.00	2.00	2.00
	InfiniteHiP	15.00	1.00	1.00
	CarryKV	23.00	6.00	1.00
niah_multikey_3	Dense	90.00	20.00	0.00
	HiP	17.00	4.00	0.00
	InfiniteHiP	12.00	1.00	0.00
	CarryKV	25.00	2.00	0.00
niah_multivalue	Dense	95.00	89.00	66.00
	HiP	83.25	50.00	16.00
	InfiniteHiP	89.25	64.75	22.50
	CarryKV	93.75	68.75	19.25
niah_multiquery	Dense	95.75	92.50	77.75
	HiP	82.25	54.50	9.00
	InfiniteHiP	87.25	50.00	8.75
	CarryKV	90.25	60.75	8.25
vt	Dense	98.60	97.60	95.80
	HiP	50.60	47.20	25.60
	InfiniteHiP	81.80	86.40	60.60
	CarryKV	91.60	86.80	53.40
cwe	Dense	73.90	30.10	3.50
	HiP	86.60	38.00	0.50
	InfiniteHiP	51.10	11.70	0.60
	CarryKV	86.20	47.20	0.70
fwe	Dense	94.33	88.00	81.67
	HiP	94.67	91.67	81.67
	InfiniteHiP	90.33	70.33	40.00
	CarryKV	95.33	89.33	76.67
qa_1	Dense	61.00	58.00	39.00
	HiP	40.00	30.00	21.00
	InfiniteHiP	35.00	29.00	21.00
	CarryKV	52.00	43.00	22.00
qa_2	Dense	49.00	43.00	33.00
	HiP	33.00	29.00	24.00
	InfiniteHiP	32.00	27.00	20.00
	CarryKV	40.00	32.00	23.00
Macro	Dense	88.28	74.32	58.29
	HiP	55.80	42.26	27.44
	InfiniteHiP	62.36	45.48	24.73
	CarryKV	69.16	51.76	27.41

Table 24: Qwen3-1.7B LongBench-E task scores. Each input view uses the same 20 source examples per task; views apply different input caps.

Task	Method	4K cap	8K cap	Full
qasper	Dense	37.40	36.48	36.48
	HiP	35.84	37.08	37.08
	InfiniteHiP	36.15	31.41	31.41
	CarryKV	42.41	42.60	42.60
multifieldqa_en	Dense	67.72	64.64	64.64
	HiP	70.17	66.70	66.70
	InfiniteHiP	68.06	67.96	67.96
	CarryKV	65.68	62.42	62.42
hotpotqa	Dense	57.07	61.54	61.54
	HiP	58.09	54.58	54.58
	InfiniteHiP	50.59	53.37	53.37
	CarryKV	58.08	53.37	53.37
2wikimqa	Dense	26.57	34.31	34.31
	HiP	18.73	29.76	29.76
	InfiniteHiP	19.37	33.89	33.89
	CarryKV	27.50	28.14	28.14
gov_report	Dense	35.50	34.99	34.99
	HiP	32.79	33.90	33.90
	InfiniteHiP	35.53	35.10	35.10
	CarryKV	34.48	34.68	34.68
multi_news	Dense	26.50	26.60	26.60
	HiP	26.26	26.59	26.59
	InfiniteHiP	27.04	27.05	27.05
	CarryKV	26.74	26.39	26.39
trec	Dense	65.00	65.00	65.00
	HiP	70.00	65.00	65.00
	InfiniteHiP	55.00	60.00	60.00
	CarryKV	60.00	65.00	65.00
triviaqa	Dense	93.21	93.21	93.21
	HiP	87.86	88.21	88.21
	InfiniteHiP	88.21	88.21	88.21
	CarryKV	88.21	88.21	88.21
samsum	Dense	32.42	31.02	31.02
	HiP	31.78	30.89	30.89
	InfiniteHiP	32.85	33.11	33.11
	CarryKV	32.38	33.33	33.33
passage_count	Dense	5.00	5.00	5.00
	HiP	5.00	5.00	5.00
	InfiniteHiP	5.00	5.00	5.00
	CarryKV	5.00	5.00	5.00
passage_retrieval_en	Dense	60.00	80.00	80.00
	HiP	50.00	60.00	60.00
	InfiniteHiP	55.00	55.00	55.00
	CarryKV	50.00	65.00	65.00
lcc	Dense	45.90	45.90	45.90
	HiP	56.15	56.15	56.15
	InfiniteHiP	52.05	52.05	52.05
	CarryKV	55.50	55.50	55.50
repobench-p	Dense	50.65	49.45	48.85
	HiP	55.65	49.70	52.40
	InfiniteHiP	48.65	54.30	54.45
	CarryKV	53.05	51.90	52.20
Macro	Dense	46.38	48.32	48.27
	HiP	46.02	46.43	46.64
	InfiniteHiP	44.12	45.88	45.89
	CarryKV	46.08	47.04	47.07

D Implementation and Efficiency

CarryKV's query-time computation centers on two principal GPU kernels over global-memory-resident state: fused branch scoring and indexed raw attention. The scorer loads active summary rows and reduces tiled cell/query contributions to one score per candidate. Selection and support assembly supply indices for the raw kernel, which loads original K/V and performs tiled online attention. Figure 13 shows these global-memory-to-on-chip transfers; the following subsections specify storage, computation and measurement.

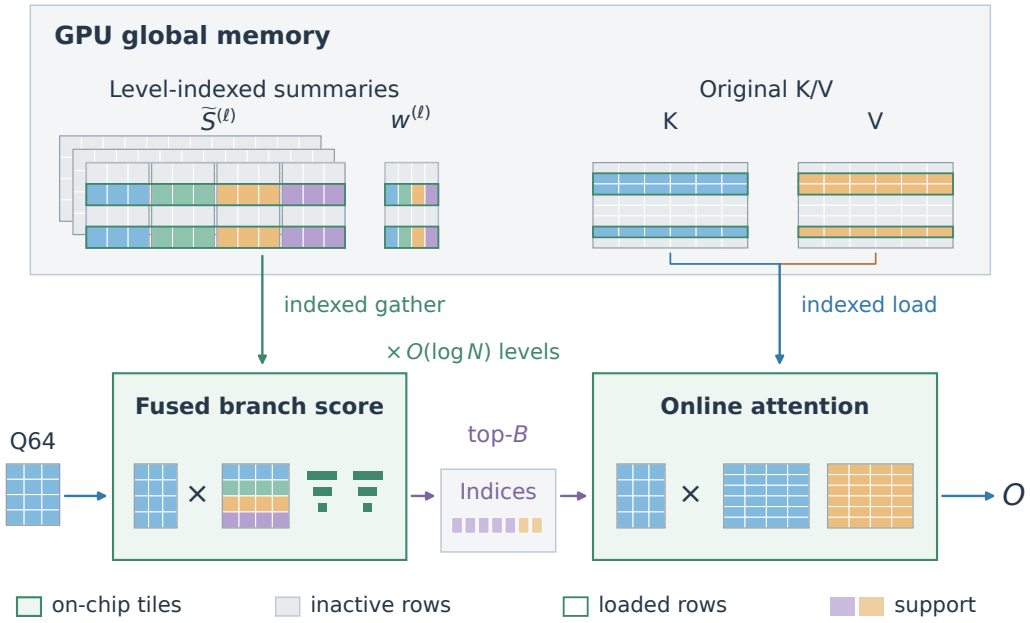


Figure 13: Physical GPU dataflow for CarryKV. Global storage supplies active summary rows to fused branch scoring and original K/V to indexed online attention. Selection and support indices connect the two compute stages. Green compute boxes mark kernel boundaries and contain on-chip tiles; solid arrows show data movement, and the level badge denotes hierarchy iteration. The logical query–candidate–cell scoring tensor is reduced on chip. Two 32-query attention tiles share each Q64 routing decision.

D.1 Persistent banks and candidate access

Each KV head owns level-indexed summary banks and separate original K/V arrays in global memory. The GQA map addresses the corresponding KV bank for each query head; beams remain independent across query heads. For complete level-ℓ nodes, $n_u = Cb^\ell$. The stored quantities are the span-normalized sums $\widetilde{S}_{u,j} = S_{u,j}/n_u$ and cell fractions $w_{u,j} = n_{u,j}/n_u$. Omitting batch,

$$\widetilde{S}^{(\ell)} \in \mathbb{R}^{h_{kv} \times L_\ell \times m \times d}, \qquad w^{(\ell)} \in \mathbb{R}^{h_{kv} \times L_\ell \times m}, \tag{67}$$

where h_{kv} counts KV heads and L_ℓ counts complete nodes at the level. Equal-span children merge these stored quantities by averaging, equivalent to adding their unnormalized counts and sums. A nonempty cell's centroid is $\widetilde{S}/w$. Candidate indices gather only active rows from these banks into scoring tiles.

Level-indexed root lists provide at most $b-1$ residual roots directly. The first competitive frontier is constructed directly, avoiding traversal of higher levels where every node survives. Candidate management compacts valid nodes in order and keeps padding separate. These indexed gathers realize the frontier-width bound in Eq. (53): a query block/head accesses a bounded number of nodes per level even though the persistent bank grows with the history.

D.2 Scoring and indexed raw attention

For a Q64 routing block, fused scoring loads active summaries in small candidate groups, reduces first over cells and then over valid query rows, and writes one score per branch. Its logical scoring tensor has axes query, candidate and cell:

$$Z_{q,u,j} = q^\top \bar{k}_{u,j}/\sqrt{d}, \qquad Z \in \mathbb{R}^{s\times F\times m}. \tag{68}$$

Here $s \le 64$ is the valid-query count, $F \le bB+b-1$ the candidate count, and $m = 4$ the cell count. This full tensor is not materialized. Each batch entry is an independent query block; reductions stay within each entry, and all valid rows contribute to Eq. (6). Top-B survivor indices feed the next hierarchy level; frontiers of size at most B bypass scoring. Selection uses `torch.topk`; exact-score ties follow that routine's ordering.

Expanded leaf tokens and the first query's leaf-alignment remainder form the remote candidate pool. When budget alignment is enabled, the first query's raw QK scores select top-K_{rem} tokens, shared by the Q64 block. Each row then unions and deduplicates its remote and sink/local support. The raw kernel reads token indices of padded width K_{idx} in 32-query by 128-key tiles, applies each row's causal/padding mask, and computes QK, stable online softmax and PV with one normalizer per query row. Two 32-query attention tiles share each Q64 remote selection.

Model Q/K/V use BF16; summaries and output accumulations use FP32.

D.3 Measurement protocol and costs

The hardware is an NVIDIA RTX PRO 6000 Blackwell Server Edition (SM120). The optimized build uses PyTorch 2.9.1+CUDA13.0 and Triton3.4.0. Dense FA4 uses an SM120-compatible CuTe DSL FlashAttention build with native GQA and causal masking.[6] Measurements use an exclusive GPU with no competing workload and no heavy profiler. Warm-up and compilation are outside steady-state timing; required state construction, numerical checks, support management, allocation, layout conversion and output writes remain inside the call. CUDA-event time and synchronized wall time are recorded separately.

Quality is measured with the reference implementations; latency uses optimized Hash-4/Q64/B64/C16 without output correction. The latency curves disable sink and local additions for all three sparse methods; quality evaluation uses the budgets in Appendix C.3.

Each of 13 lengths uses four warm-ups and three rounds of 12 interleaved calls for all four methods, with mode switching outside timing. The 32K CUDA-event medians are 35.75 ms for CarryKV, 34.70 ms for HiP, 17.43 ms for InfiniteHiP and 26.51 ms for Dense FA4. Reported medians and quantiles include all repeated calls on the same input. Native states supply lengths through 32K, static YaRN2 supplies 48K/64K, YaRN4 supplies 96K/128K and YaRN8 supplies 192K/256K. These longer timing inputs extend the systems measurement beyond the quality table. HiP (Lee et al., 2025a) provides the closest logarithmic hierarchical-kernel comparison, while InfiniteHiP (Lee et al., 2025b) serves as the GPU-efficiency reference through staged parallel screening with less synchronization. Table 25 reports their complete measured costs.

Measured structural work. Routing-only measurements isolate bounded-search work from construction and raw-attention costs included in complete latency. Instrumented traversals count valid competitive candidates separately from unscored expansion. Across 4K–256K, the maximum frontier is 259, and the mean number of scored candidates per query block/head rises from 97.78 to 846.54. Logical cell accesses are four times candidate counts. Bank construction remains linear in context, with 81–5,454 parent nodes per KV-head bank over these endpoints. A separate prebuilt-bank measurement of one Q64 block and all 32 heads grows from 0.857 to 2.190 ms; the construction, numerical checks and raw attention are outside this routing-only boundary. Figure 14 separates this timing from work counts.

[6] `https://github.com/Dao-AILab/flash-attention`

Table 25: Complete-prefill CUDA-event time in milliseconds, median [p10,p90] over all 36 interleaved calls per method and length. The boundary matches Figure 4. The last two rows extend the systems measurement beyond the quality range.

Context	Dense FA4	HiP	InfiniteHiP	CarryKV
4K	0.72 [0.70,0.73]	3.37 [3.35,3.51]	3.89 [3.85,4.09]	5.17 [5.13,5.57]
6K	1.25 [1.24,1.31]	5.15 [5.10,5.30]	4.32 [4.28,4.55]	6.83 [6.68,7.23]
8K	2.03 [2.02,2.10]	7.01 [6.99,7.26]	4.94 [4.90,5.23]	8.24 [8.20,8.97]
12K	4.14 [4.13,4.17]	11.23 [11.19,11.29]	6.61 [6.58,6.80]	12.91 [12.85,13.25]
16K	7.05 [7.04,7.07]	15.43 [15.42,15.47]	8.42 [8.38,8.55]	16.21 [16.18,16.62]
24K	15.21 [15.20,15.23]	25.01 [24.98,25.05]	12.57 [12.55,12.62]	25.97 [25.89,26.33]
32K	26.51 [26.50,26.54]	34.70 [34.67,34.79]	17.43 [17.40,17.76]	35.75 [35.69,36.40]
48K	58.55 [58.54,58.58]	56.23 [56.22,56.25]	26.88 [26.85,27.02]	56.59 [56.51,56.88]
64K	103.20 [103.16,103.25]	78.06 [78.00,78.16]	37.01 [36.96,37.05]	78.57 [78.34,78.98]
96K	231.92 [230.70,232.88]	125.79 [125.70,125.84]	57.87 [57.81,58.02]	127.50 [126.53,128.26]
128K	415.84 [413.10,418.54]	174.37 [174.29,174.45]	79.52 [79.44,79.70]	176.75 [175.90,177.94]
192K	941.54 [928.84,946.24]	281.35 [280.61,281.50]	122.27 [122.14,122.47]	278.21 [277.46,279.72]
256K	1765.64 [1693.09,2737.10]	387.84 [387.57,411.32]	166.84 [166.52,199.04]	385.89 [383.54,526.40]

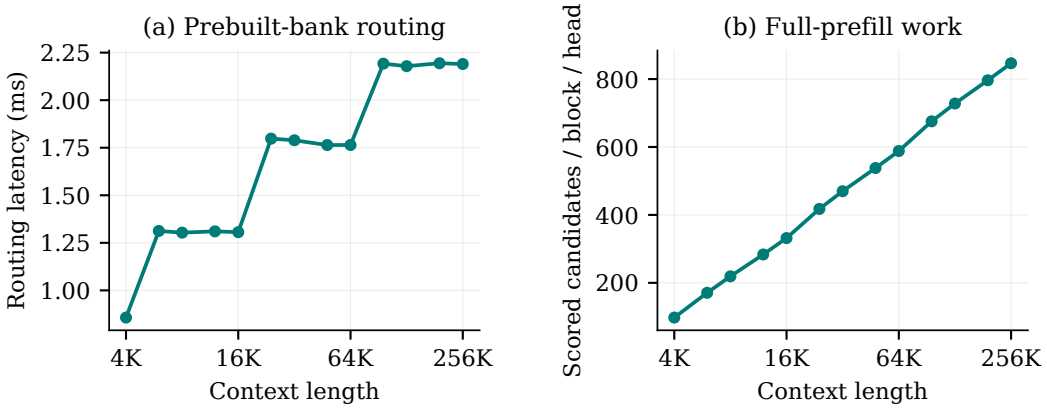


Figure 14: Routing cost and structural work. Left: median latency for the final query block and 32 heads with prebuilt summary banks. Right: mean number of scored candidates per query block/head over full prefill. Complete prefill latency, including construction and raw attention, appears in Figure 4.

D.4 Memory and state lifetimes

Latency characterizes execution cost; we finally separate persistent and temporary memory lifetimes. Table 26 measures isolated operator allocations. Each method uses a fresh process, with the baseline reset after warm-up. Common native Q/K/V storage is 0.375/0.750/1.500 GiB and returned output storage is 0.250/0.500/1.000 GiB at 32K/64K/128K; outputs do not alias inputs. CarryKV's extra allocated peak above its recorded resident-input baseline exceeds Dense's by 13.88%, 10.05% and 10.05% at 32K/64K/128K. Reserved memory, allocated peaks and persistent storage have different lifetimes; the measurements retain these quantities separately.

In the current implementation, persistent routing state remains resident across layers during dense decode. At capacity 32,768, native BF16 raw KV occupies 128 MiB (134.2 decimal MB) per layer, or 4.50 GiB (4.83 decimal GB) across 36 layers. Routing state adds 56.17 MB per layer for Hash-4 versus 11.19 MB for Mean-1/Temporal-4. The latter can form temporal quarters from child totals. These are component allocations at the forest capacity, not a measured whole-model serving peak.

For a 1,536-token prompt, temporary prefill state occupies 0.51 MB for mean/radius and 4.10 MB for four-cell temporal/hash banks per builder call. These temporaries are released layer by layer.

Persistent route cells remain allocated during dense decode; the reported costs include their storage and updates.

Table 26: Isolated complete-operator CUDA allocation, batch-one 8B shapes. A/R means allocated/reserved GiB; increment is peak minus the method-specific resident-input baseline. Post-call metadata (MiB) counts adapter tensors retained after this call; temporary summary banks contribute to the peak. Model weights are excluded. Model-level persistent state is reported separately in Appendix D.4.

Context	Method	Baseline A/R	Increment A/R	Peak A/R	Post-call metadata
32K	Dense FA4	0.375/0.502	0.500/0.500	0.875/1.002	0.00
32K	HiP	0.375/0.502	0.879/0.766	1.254/1.268	0.00
32K	InfiniteHiP	0.375/0.502	0.969/1.549	1.345/2.051	16.38
32K	CarryKV	0.383/0.502	0.569/0.631	0.953/1.133	0.00
64K	Dense FA4	0.750/1.002	1.000/1.000	1.750/2.002	0.00
64K	HiP	0.750/1.002	1.754/1.516	2.504/2.518	0.00
64K	InfiniteHiP	0.750/1.002	1.923/2.861	2.673/3.863	32.75
64K	CarryKV	0.758/1.006	1.101/1.004	1.859/2.010	0.00
128K	Dense FA4	1.500/2.002	2.000/2.000	3.500/4.002	0.00
128K	HiP	1.500/2.002	3.504/3.016	5.004/5.018	0.00
128K	InfiniteHiP	1.500/2.002	3.830/5.973	5.331/7.975	65.50
128K	CarryKV	1.508/2.004	2.201/2.008	3.710/4.012	0.00

E Distance-Conditioned Score Differences

HiP's theoretical justification for center representatives uses a distance-conditioned locality model. In Appendix A.1 of Lee et al. (2025a), Axiom 1 models score differences at separation $\Delta > 0$ as zero-centered Gaussian variables with increasing scale $\sigma(\Delta)$. In their two-branch top-1 setting with a uniform maximizer location and independent difference variables, the authors use Theorem 1 to show that midpoint representatives improve correct branch selection over random token selection. Theorem 2 in Appendix A.2 gives midpoint optimality under the additional condition $\sigma''(\Delta)\sigma(\Delta) < [\sigma'(\Delta)]^2$. Appendix A.3 measures scale, shape and dependence across models and heads, and relates logarithmic scale fits to this condition.

The evaluated Qwen3 models apply Q/K normalization before RoPE. We examine the corresponding distance-conditioned statistics in this normalized score regime. Fix a model, prompt, layer, head and query, with realized logits s_i. Sampling a pair (I, J) at separation Δ defines

$$D_\Delta = s_I - s_J, \qquad |I - J| = \Delta, \qquad \sigma_\Delta = \sigma(\Delta) = \sqrt{\mathrm{Var}(D_\Delta)}. \tag{69}$$

The distribution in Eq. (69) is over sampled pairs at the specified distance.

E.1 Distance-conditioned scale and shape

Figure 15 summarizes both 260-prompt Dense-32K pools. Each task/layer/head/distance stratum pools 20 prompts and 256 position pairs per prompt. The resulting 87,360 and 224,640 moment rows characterize heterogeneity across task–layer–head strata.

The median paired $\sigma(16K)/\sigma(4K)$ ratios are 1.012 and 1.014; their 10th–90th percentile intervals are [0.875,1.155] and [0.907,1.181]. Across all 4K/8K/16K distances, 60.8%/61.9% of strata stay within 10% of their 4K scale, and 82.6%/85.1% within 20%. Maximum relative drift has medians 7.27%/7.48% and 90th percentiles 26.08%/25.17%. These finite-range contrasts coexist with predominantly positive fitted log-scale trends and substantial head heterogeneity. Standardizing the differences separates distributional shape from raw scale.

Centering, shape and the distance model. Table 27 examines centering and shape alongside a logarithmic scale fit on the normalized Qwen3 scores.

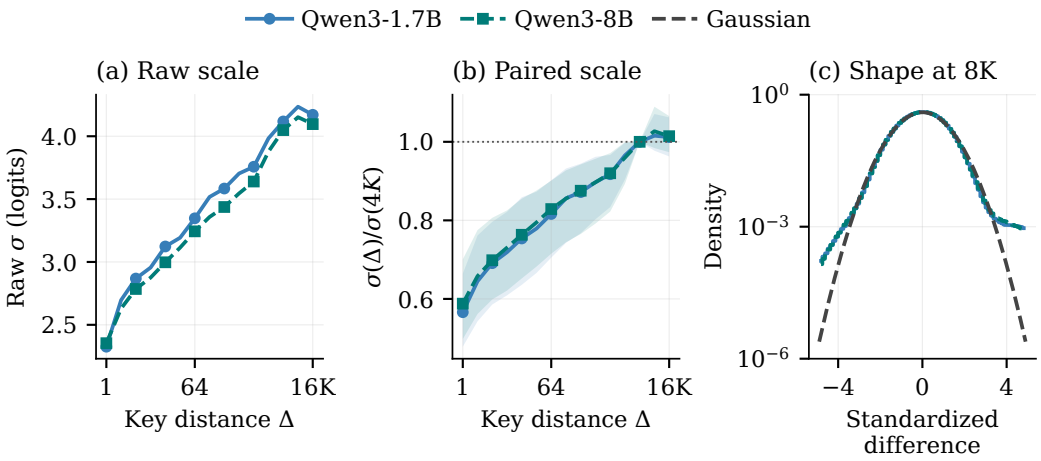


Figure 15: Distance-conditioned differences. (a) Median raw standard deviation versus key-pair distance. (b) Scale relative to 4K: median and IQR across strata. (c) Standardized 8K differences versus a Gaussian; clipped outer-tail bins are omitted and zero-density bins masked on the log axis.

Table 27: Distance-conditioned statistics: 260 Dense-32K prompts/model, 20 per task–layer–head stratum. Shape/centering use $\Delta = 8K$; fits use all 15 distances. NRMSE divides residual RMSE by mean measured σ within each stratum; IQR/percentiles summarize stratum heterogeneity.

Statistic	Qwen3-1.7B	Qwen3-8B
Median $\lvert\mathbb{E}D_\Delta\rvert/\sigma_\Delta$	.214	.190
Median excess kurtosis	.834	.721
Excess kurtosis IQR	[.157,2.072]	[.144,1.740]
Strata with positive excess kurtosis	81.7%	81.6%
Log-scale fit NRMSE, median / p90	.0612 / .1554	.0570 / .1419
Strata with positive fitted slope	99.90%	99.96%
Strata with an adjacent-distance decrease	97.6%	98.2%

Positive excess kurtosis is common, and nonzero centering gives a separate departure from the zero-centered Gaussian description. The fitted model is

$$\sigma_\Delta = \alpha \log \Delta + \beta, \tag{70}$$

with slope α and intercept β. Eq. (70) omits HiP's input-shift parameter. Positive fitted slopes coexist with local adjacent-distance decreases and weak relative contrast at large distances. These observations concern marginal differences sampled by distance. Representative ordering additionally conditions on the extremum and visited branch. The measured marginal difference distribution therefore does not determine the conditional representative-ordering distribution used by HiP's analysis.

E.2 Relative locality signal and branch valuation

For $x, g \in \mathbb{R}^d$, learned-gain RMS normalization $N_g(x) = (g \odot x)/\sqrt{\|x\|^2/d + \varepsilon}$ uses elementwise gains g and stabilizer $\varepsilon \geq 0$. A positive denominator gives $\|N_g(x)\| \leq \sqrt{d}\|g\|_\infty$. With query/key gains g_q, g_k and orthogonal RoPE,

$$|q^\top k/\sqrt{d}| \leq \sqrt{d}\|g_q\|_\infty \|g_k\|_\infty; \tag{71}$$

the difference bound is twice this, and positional amplitudes multiply it. Eq. (71) bounds score magnitude without identifying a saturation distance.

Consider the logarithmic scale model

$$\sigma(x) = a\log(x+b) + c > 0, \qquad a > 0, \quad x + b > 0, \tag{72}$$

with slope a, input shift b and offset c. Any fixed distance ratio $r > 1$ gives

$$\frac{\sigma(rx)}{\sigma(x)} = 1 + \frac{a\log[(rx+b)/(x+b)]}{a\log(x+b) + c} \xrightarrow[x\to\infty]{} 1. \tag{73}$$

Thus logarithmic growth can remain increasing while the relative scale contrast between fixed-ratio distances weakens (Eq. (73)). Within HiP's locality model, this reduces the scale separation available to representative ordering, even when monotonicity remains. A positive logarithmic scale also satisfies the curvature condition of HiP's Theorem 2.

A common-mode example uses n random logits $S_i = Z + \epsilon_i$, with a shared finite-variance component Z independent of iid centered, finite-variance noise ϵ_i. For distinct positions, difference variance is constant, whereas the branch mean $\bar{S} = n^{-1}\sum_i S_i$ retains the common component:

$$\mathrm{Var}(S_i - S_j) = 2\,\mathrm{Var}(\epsilon_1), \qquad \mathrm{Var}(\bar{S}) = \mathrm{Var}(Z) + \mathrm{Var}(\epsilon_1)/n. \tag{74}$$

Eq. (74) shows that pairwise differences do not determine branch-level averages. These observations motivate evaluating branch valuation directly rather than inferring it from distance alone. CarryKV's analysis characterizes the information retained by branch summaries and its effect on same-frontier mass selection; Section 5.2 measures center, mean and multi-cell scorers on actual competitive frontiers.

Distance-conditioned pair statistics therefore provide only part of the information relevant to recursive branch valuation. Our branch analysis works directly with finite realized logits (Appendix A.2), while recursive path survival is measured on the actual competitive stages in Appendix C.2.