HATS: EFFICIENT THRESHOLD-BASED SPARSE ATTENTION VIA HISTORY-GUIDED BLOCK TRAVERSAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context decoding is bottlenecked by repeated access to an ever-growing KV cache. While sparse attention can skip insignificant KV blocks, inexpensive history-based selection is query-agnostic, whereas query-aware selection incurs additional computational overhead. An alternative is to make skip decisions during KV-block traversal, where a threshold determines whether to skip value-side computation right after computing a block's attention score. However, we observe that the achieved sparsity depends not only on the threshold but also on the traversal order: visiting low-scoring blocks before high-scoring ones can retain blocks that would otherwise be skipped, creating a *traversal-induced sparsity gap*. To close this gap, we propose HATS (History-guided Attention with Thresholded Skipping). HATS uses blocks that scored highly at the previous decoding step to guide traversal for the current token, thereby increasing the likelihood of visiting high-scoring blocks as early as possible. History only guides the order of block traversal, while value skipping remains determined by the newly computed current token score and a preset threshold. We prove that, for a given current token and fixed threshold, visiting the globally highest-scoring block first maximizes the attainable sparsity, thereby defining an oracle upper bound on sparsity over the set of all traversal orders. Experiments show that HATS closes 98.6–99.5% of the sparsity gap to the oracle, delivering a 1.98× speedup for the decode-attention kernel at 92.4% sparsity and a 1.26× decode-throughput speedup at 88.2% sparsity.

1 INTRODUCTION

Long-context autoregressive decoding reads an ever-growing KV cache at every generation step. Attention computation and memory traffic therefore increase with context length, making attention a major inference bottleneck. Yet attention often concentrates on a small set of past positions, leaving many KV blocks with little influence on the current output. Sparse attention can exploit this structure by skipping such low-contribution blocks.

Realizing these savings depends on identifying skippable KV blocks at low cost. Existing methods span a trade-off. Methods based on positional structure or on attention observed before the current query restrict future KV access with little selection overhead, but their retained contents are less responsive to changes in the query (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024). Methods that estimate token or page relevance from the current query are more adaptive, but require additional approximate scoring, sorting, or indexing (Tang et al., 2024; Ribar et al., 2024; Singhania et al., 2024). Repeating these selection operations at every decoding step applies them to every generated token, so their cost can offset a substantial part of the savings that sparse attention provides.

A direct alternative folds skip decisions into the existing KV-block traversal, eliminating a separate selection stage. Every block still undergoes QK scoring, and the largest block score seen so far provides the scale for subsequent decisions. If a block falls sufficiently below this maximum according to a preset threshold, its softmax contribution will be small, allowing the corresponding value-side work to be skipped. BLASST is a representative instance of this threshold-based block sparsification strategy (Yuan et al., 2026). The rule, however, depends on when high scores appear in the traversal. If a high-scoring block is visited late, earlier blocks are compared against a lower maximum and may

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.

be retained unnecessarily. Thus, identical current-token block scores, threshold, and skip rule can produce different achieved sparsity under different traversal orders. We call the sparsity lost relative to the ideal order the *traversal-induced sparsity gap*, illustrated in Figure 1.

The gap arises because threshold-based sparsification depends not only on the threshold, but also on how early a sufficiently high current score becomes available. Visiting a block with the global maximum score first immediately establishes the final maximum and attains the greatest sparsity possible under fixed scores and threshold. Its location, however, is unknown until the corresponding QK computation is complete, while an explicit search would reintroduce a separate selection pass. A usable ordering rule must therefore predict that location from information carried over from earlier decoding steps. Prior work has observed that high-attention positions often persist across adjacent decoding steps (Yang et al., 2025c). This persistence can predict which blocks should be visited early, but it cannot determine which blocks the current token should skip because new important regions may still emerge. The gap therefore calls for a rule that reorders blocks without changing the skip test itself.

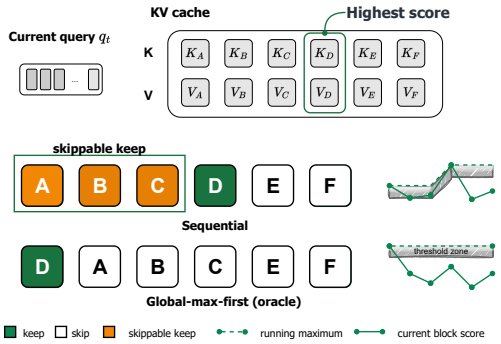


Figure 1: **A minimal traversal-induced sparsity gap under fixed current-token scores, threshold, and skip rule.** In the top traversal the maximum-score block D is visited after A–C, so their value-side work cannot yet be skipped; visiting D first (bottom) lets A–C be skipped. White blocks denote skipped value-side work.

We introduce HATS (History-guided Attention with Thresholded Skipping), a history-guided traversal method for threshold-based block sparsification. HATS uses previous-token block scores to visit a small set of potentially high-scoring KV blocks early, establishing a high current-token maximum sooner. We prove that, for fixed current-token scores and threshold, visiting a globally highest-scoring block first maximizes attainable sparsity and thereby defines an oracle upper bound over traversal orders. The bound depends only on the threshold rule, not on how block scores are produced. We also develop a fused decode-attention kernel that incorporates priority-first traversal and cross-token block-score maintenance into attention execution, translating the additional block sparsity into GPU performance gains. Because the priority order comes from the previous decoding step, HATS adds no selection pass: each block still undergoes a single QK scoring under the original threshold rule, and the added work is only a reordering.

Experiments evaluate HATS at the mechanism, model-quality, and inference levels. Under fixed current-token block scores, threshold, and skip rule, HATS closes 98.6–99.5% of the sparsity gap across models and context lengths, approaching the oracle traversal. At matched achieved sparsity, HATS maintains quality comparable to BLASST across the evaluated benchmarks, while both sparse methods remain close to dense attention at the main operating points. On RULER-32K, the highest achieved sparsity that retains at least 99% of the dense score increases from 79.1% with BLASST to 90.7% with HATS. The additional complete-block skips yield a 1.98× decode-attention kernel speedup at 92.4% sparsity and a 1.26× decode-throughput speedup at 88.2% sparsity.

Our contributions are:

- We identify and formalize the traversal-induced sparsity gap in threshold-based block sparsification, and prove that visiting a globally highest-scoring block first achieves the maximum sparsity attainable under fixed scores and threshold.
- We propose HATS, which uses previous-token block scores to prioritize likely high-scoring blocks while retaining skip decisions based entirely on current-token scores.
- We implement a fused decode-attention kernel and show across models and context lengths that HATS approaches oracle sparsity, maintains model quality, and translates additional sparsity into attention-kernel and decode-throughput speedups.

2 Background and Problem Formulation

2.1 Threshold-Based Block Sparsification

Threshold-based block sparsification folds keep-or-skip decisions into the tiled online-softmax execution used by modern exact-attention kernels (Dao et al., 2022; Dao, 2024), avoiding a separate selection stage. BLASST is a representative instance of this design (Yuan et al., 2026). At decode step t, KV blocks are visited in an order π_t. For the k-th visited block $j = \pi_t(k)$, the algorithm first computes its current-token block score $b_{t,j}$, defined as the maximum scaled QK logit within the block. The running maximum before this visit is the largest score observed so far:

$$m_{t,k-1} = \max_{1 \le q < k} b_{t,\pi_t(q)}. \tag{1}$$

For $k = 1$, the empty-prefix maximum is defined as $m_{t,0} = -\infty$. Given a threshold $\tau \le 0$, we apply the following skip rule: the value-side computation of block j is skipped when

$$b_{t,j} - m_{t,k-1} < \tau. \tag{2}$$

Thus, every block is scored using the current query, while blocks sufficiently below the available running maximum avoid the corresponding value-side work. Crucially, the running maximum $m_{t,k-1}$ depends on the previously visited blocks. Therefore, even with identical block scores and threshold, different traversal orders may produce different keep-or-skip decisions.

2.2 Traversal-Induced Sparsity Gap

The running maximum depends on which blocks have already been visited. Consequently, traversal order can affect the achieved sparsity even when the current-token block score, threshold, and skip rules remain fixed. If a high-scoring block is visited late, earlier blocks are evaluated against a weaker running maximum and may execute value-side work that a different visit order would have skipped. We call such a block a *skippable keep*.

Figure 2 measures this effect by changing only the traversal order over fixed current-token scores. Random and reverse orders do not close the gap, whereas the global-max-first order increases achieved sparsity by 8.23–12.22 percentage points over sequential traversal.

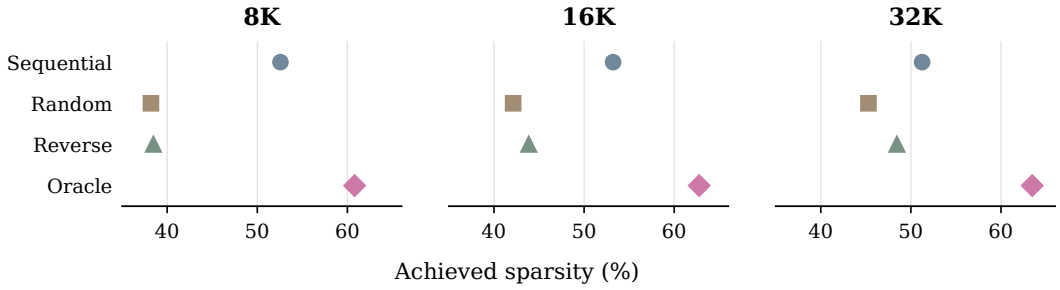


Figure 2: **Traversal order affects achieved sparsity under identical block scores and thresholds, exposing a gap relative to the oracle traversal.**

Let $S_t(\pi, \tau)$ denote the fraction of KV blocks skipped under traversal π and threshold τ. Define the best achievable sparsity over all traversals as

$$S_t^\star(\tau) = \max_{\pi'} S_t(\pi', \tau).$$

We define the *traversal-induced sparsity gap* as

$$G_t(\pi, \tau) = S_t^\star(\tau) - S_t(\pi, \tau). \tag{3}$$

This gap isolates the sparsity lost because the traversal establishes a strong running maximum too late. It quantifies the additional sparsity attainable through traversal reordering alone, with the current-token block scores, threshold, and skip rule held fixed.

2.3 Oracle Traversal

The current token's global maximum block score is

$$m_t^\star = \max_j b_{t,j}. \tag{4}$$

A traversal that visits a block attaining $m_t^\star$ first immediately establishes the strongest possible running maximum. We denote any such order by $\pi_t^\star$ and call it a *global-max-first oracle traversal*.

Proposition 1 (Global-max-first optimality). For fixed current-token block scores, threshold, and skip rule,

$$S_t(\pi, \tau) \le S_t(\pi_t^\star, \tau) \qquad \text{for every traversal } \pi. \tag{5}$$

For any traversal, the running maximum available before a block is visited is at most $m_t^\star$. Every block skipped under that weaker running maximum therefore remains skippable after $m_t^\star$ is established. Hence, $\pi_t^\star$ attains the maximum achievable block sparsity. The complete proof is provided in Appendix A.1. Proposition 1 is stated for a single query head; Section 3 covers query heads that share KV blocks.

The oracle cannot be implemented directly because the maximizing block is unknown before its current-token score is computed. An explicit search would reintroduce a separate selection pass. A practical method must instead establish a strong running maximum early while preserving the original current-token threshold decision.

3 Method

3.1 Design Overview

HATS uses previous-token block scores to prioritize blocks in the current traversal, while every skip decision remains determined by the current-token block score and the threshold rule in Section 2.1.

Figure 3 summarizes HATS. The block scores from step $t-1$ determine an ordered priority list $\mathcal{P}_t$ of H blocks, and these blocks lead the traversal at step t. Each block then receives its current-token score as it is visited, and the threshold rule decides whether its value-side work can be skipped. The scores produced at step t are retained to construct $\mathcal{P}_{t+1}$, so the priority list advances with decoding.

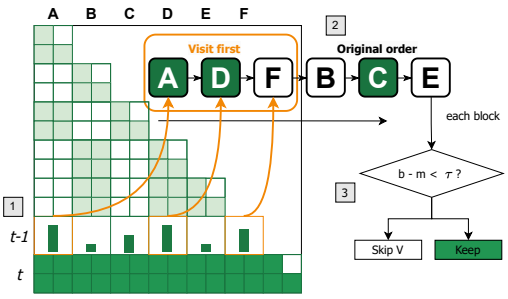


Figure 3: **Overview of HATS.**

3.2 History-Guided Priority Construction

For each group of query heads sharing the same KV blocks, HATS first identifies the highest-scoring block for every head at step $t-1$, then processes the query heads in ascending head-index order and appends each previously unselected block index to $\mathcal{P}_t$ until the list contains H blocks. The remaining positions are filled by ranking the unselected blocks according to their minimum score across the head group, in descending order, so that blocks with consistently high scores across the group are preferred; ties are resolved by block index. A newly appended KV block becomes eligible for the priority list after receiving its first score. This construction reuses block scores already produced by the attention computation and requires neither additional QK computation nor storage of full attention maps. We use a fixed priority budget of $H = 8$ throughout the experiments; its sparsity saturation is reported in Appendix C.2.

3.3 PRIORITY-FIRST THRESHOLD SPARSIFICATION

The ordered list $\mathcal{P}_t$ defines the priority-first traversal used by HATS. Let π_t^{seq} denote the sequential order of all valid KV blocks. The resulting traversal is

$$\pi_t^{\mathrm{HATS}} = \mathcal{P}_t \parallel \left(\pi_t^{\mathrm{seq}} \setminus \mathcal{P}_t\right), \tag{6}$$

where blocks outside $\mathcal{P}_t$ retain their relative order and every valid block appears exactly once.

Along π_t^{HATS}, each query head initializes its running maximum at $-\infty$ and applies the skip rule in Section 2.1 using current-token block scores. Every block therefore undergoes QK scoring, and history changes only the traversal order; sparsity arises from omitted value-side work. Under grouped-query attention (GQA) (Ainslie et al., 2023), query heads sharing the same KV blocks maintain independent running maxima. The complete KV block's value loading, softmax update, and value aggregation are omitted only when every head in the group skips it.

The benefit of this traversal follows from the oracle characterization in Section 2.3. Before a block attaining the current-token global maximum $m_t^\star$ is visited, blocks may be kept although the oracle traversal would skip them, because the running maximum is too low. Once such a block has been processed, all subsequent decisions of the head whose maximum it attains match those under the global-max-first oracle traversal; under grouped-query attention this holds separately for each query head. HATS therefore reduces the prefix in which skippable keeps can occur by prioritizing blocks likely to establish a high running maximum.

3.4 HARDWARE-EFFICIENT GPU IMPLEMENTATION

HATS implements the priority-first traversal within a fused decode-attention kernel. For each visited 128-token KV block, the kernel loads the Key block, computes its QK logits, and reduces them to one block score per query head before applying the threshold decision. The value block is loaded, and blockwise softmax and value aggregation are executed only if at least one query head sharing that KV block retains it. A complete-block skip therefore removes the corresponding value traffic and value-side computation while preserving current-token QK scoring.

The block scores and priority metadata remain device-resident throughout decoding. Scores produced during the current attention step are reused to construct $\mathcal{P}_{t+1}$ on the GPU, and the next decode step consumes this list directly. The production path integrates this metadata construction with the attention execution, requiring neither an additional QK pass nor host-side synchronization.

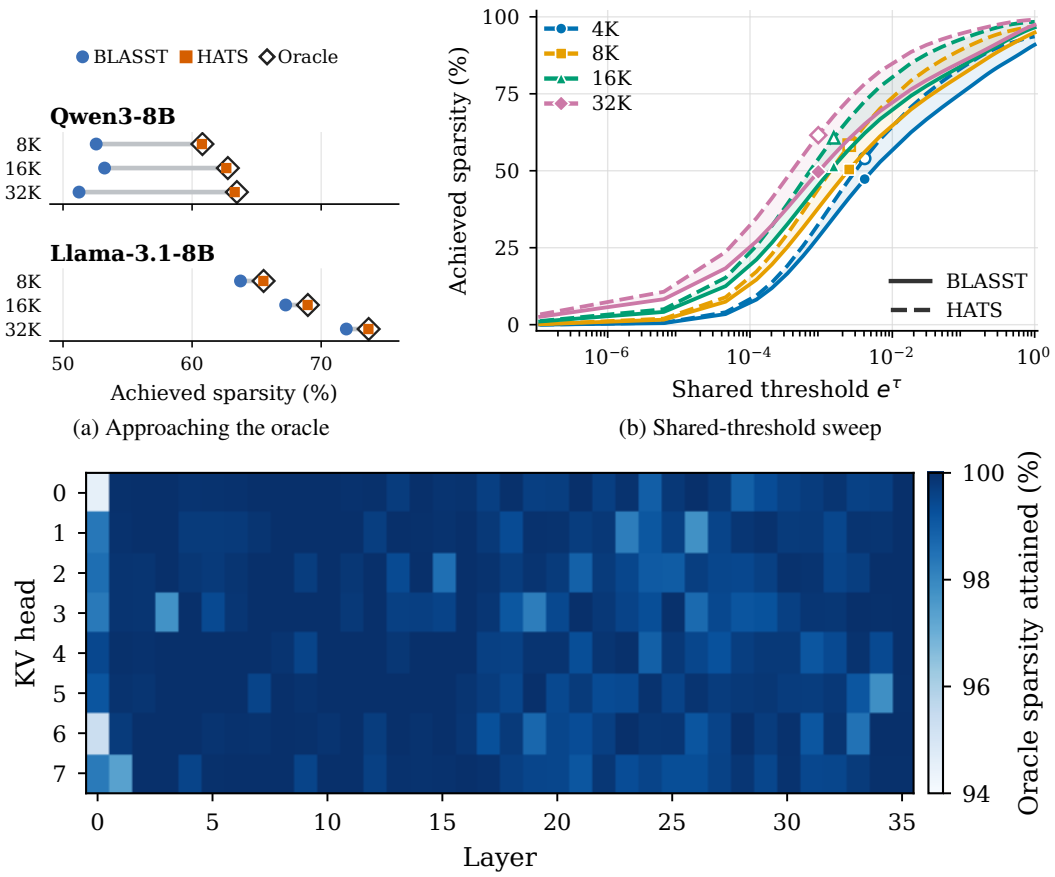
4 EXPERIMENTS

The experiments test whether history-guided traversal closes the sparsity gap, retains task quality at higher sparsity, and improves decode performance. Quality experiments use Qwen3-8B (Yang et al., 2025a) and Llama-3.1-8B (Grattafiori et al., 2024) on RULER, LongBench v2, MATH500, AIME 2024, and GPQA Diamond (Hsieh et al., 2024; Bai et al., 2025; Hendrycks et al., 2021; MAA, 2024; Rein et al., 2023). Prefill remains dense; sparsity applies only to decode. Achieved sparsity is the ratio of complete KV blocks whose value-side work is skipped to the total number of complete KV blocks. Under GQA, a block is skipped only when every query head sharing it satisfies the rule.

Mechanism experiments compare HATS with BLASST's original traversal and the global-max-first oracle under fixed current-token scores and thresholds. For quality, Dense provides the reference and BLASST is the matched-sparsity comparator at approximately 50% sparsity; Quest and RocketKV serve as additional baselines under the configurations recommended in their original papers (Yuan et al., 2026; Tang et al., 2024; Behnam et al., 2025). BLASST and HATS are calibrated separately for quality comparisons, and frontier curves use sparsity measured during each quality run. Appendix B specifies calibration, generation, and scoring protocols.

4.1 CLOSING THE SPARSITY GAP

HATS closes nearly all of the sparsity gap to the oracle on both models at 8K–32K (Figure 4(a)). Under sequential traversal, the Qwen3-8B gap is 8.23–12.22 percentage points, of which HATS closes 98.6–99.5%; the Llama-3.1-8B gap is 1.72–1.79 points, of which HATS closes 98.7–99.0%.



(a) Approaching the oracle

(b) Shared-threshold sweep

(c) Oracle sparsity attained across layers and KV heads

Figure 4: **History-guided traversal closes the sparsity gap.** (a) Achieved sparsity of BLASST, HATS, and the oracle at shared thresholds. (b) Achieved sparsity of both methods at each shared threshold for 4K to 32K; the shaded band is the difference between them, and the paired markers show both methods at one shared operating threshold. (c) HATS/oracle sparsity (%) per layer–KV head at $\tau = -7$.

The same-threshold sparsity advantage holds across the Qwen3-8B sweep: HATS achieves higher mean sparsity at all 440 $(\text{context}, \tau)$ points in Figure 4(b). At the marked operating points, the increase over BLASST grows from 6.71 percentage points at 4K to 11.91 points at 32K. Shared scores and thresholds isolate the effect of traversal order.

Near-oracle sparsity holds across Qwen3-8B's layers and KV heads. Across all 288 layer–KV-head pairs at 32K, the oracle-sparsity ratio $S_{\text{HATS}}/S_{\text{Oracle}}$ ranges from 94.5% to 100.0%, with a median of 99.9% (Figure 4(c)).

4.2 Quality Frontier

HATS reaches higher achieved sparsity while retaining at least 99% of the Dense score on the Qwen3-8B RULER hard subset (Figure 5). At 32K, the highest measured sparsity meeting this quality level rises from 79.1% for BLASST to 90.7% for HATS. At 8K and 16K, HATS's highest measured points also meet the 99% level, at 86.8% and 89.7% achieved sparsity, respectively.

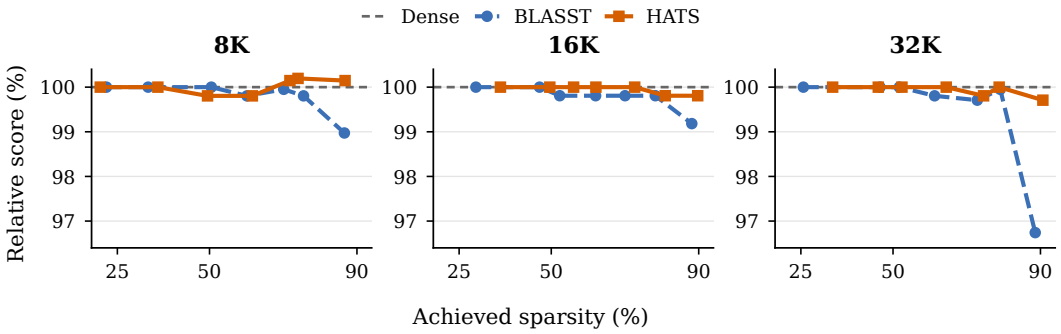


Figure 5: **Near-dense quality at higher achieved sparsity.** Qwen3-8B on the RULER hard subset; scores are normalized to Dense (100%) and all panels share the same vertical range.

4.3 Quality Comparison

At approximately 50% achieved sparsity, HATS remains close to Dense on RULER-32K (93.36 vs. 93.61) and LongBench v2 (32.41 vs. 32.60), and retains high absolute scores on MATH500, AIME 2024, and GPQA (97.40, 76.67, 63.64), covering task breadth at this operating point (Table 1). Table 2 compares the 80% and 90% target-sparsity operating points against Dense for each model–benchmark pair.

Table 1: **Decode quality of Qwen3-8B.** BLASST and HATS target approximately 50% achieved sparsity; Quest and RocketKV use the configurations recommended in their original papers.

Method	RULER-32K	LongBench v2	MATH500†	AIME 2024†	GPQA†
Dense	93.61	32.60	95.87	75.00	61.21
Quest	92.17	34.19	94.18	71.50	60.12
RocketKV	92.41	32.41	95.88	73.54	60.50
BLASST ∼50%	93.29	33.20	96.23	76.50	61.51
HATS ∼50%	93.36	32.41	97.40	76.67	63.64

Note. Columns marked † report values taken from BLASST Table 4 (Yuan et al., 2026) and follow that paper's evaluation protocol; the HATS row is measured in our own runs, with GPQA on the full Diamond split (198 questions). RULER-32K and LongBench v2 are measured in our runs for every row, including Quest and RocketKV.

HATS improves RULER by 0.92 and 3.35 score points at the 80% and 90% operating points, respectively, and matches BLASST on LongBench v2. The Llama-3.1-8B RULER results show smaller gains of 0.28 and 0.63 points. On Llama-3.1-8B LongBench v2, both methods match the Dense score of 30.62 at the 80% target and score 30.22 at the 90% target.

Table 2: **Decode quality at high sparsity.** RULER-32K and LongBench v2 at 80% and 90% target sparsity for both models. Best sparse score within each target is bolded.

Method	Qwen3-8B		Llama-3.1-8B	
	RULER	LongBench v2	RULER	LongBench v2
Dense	93.61	32.60	90.72	30.62
BLASST 80%	92.41	**32.80**	89.40	**30.62**
HATS 80%	**93.33**	**32.80**	**89.68**	**30.62**
BLASST 90%	88.94	**32.80**	88.88	**30.22**
HATS 90%	**92.29**	**32.80**	**89.50**	**30.22**

4.4 Inference Performance

The additional sparsity translates into faster decode attention. Table 3(a) compares identical Qwen3-8B attention inputs at shared thresholds. At $\tau = -2$, HATS achieves 92.4% sparsity and 1.98× speedup over dense FlashAttention-2 (Dao, 2024); BLASST achieves 87.6% sparsity and 1.92× speedup. At matched sparsity, the mean relative kernel-latency difference over the 30–90% range is 1.72%, consistent with most of the shared-threshold gain coming from skipping additional work.

Table 3: **Kernel and decode-throughput speedups.** (a) Kernel speedup at shared thresholds. (b) Decode throughput for Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025a); the first row is Dense. (c) Kernel speedup across 8K–64K contexts, at each batch size and target sparsity level.

(a) Kernel speedup

	BLASST		HATS	
τ	Sparsity	Speedup	Sparsity	Speedup
-8	10.7%	1.05×	18.7%	1.11×
-6	26.6%	1.16×	39.1%	1.26×
-5	40.4%	1.27×	53.4%	1.40×
-4	56.8%	1.45×	68.8%	1.58×
-3	74.6%	1.69×	82.4%	1.78×
-2	87.6%	1.92×	92.4%	1.98×

(b) Decode throughput

Method	Sparsity	Tok/s	Speedup
Dense	0.0%	319.1	1.00×
	38.4%	352.5	1.10×
	49.9%	365.1	1.14×
HATS	61.3%	370.4	1.16×
	76.5%	385.2	1.21×
	88.2%	401.2	1.26×

(c) Across input shapes

Batch	50%	80%	90%
32	1.24–1.27	1.55–1.63	1.71–1.75
64	1.34–1.36	1.71–1.73	1.90–1.92
128	1.43–1.47	1.86–1.89	2.08–2.12

Kernel speedups hold across the tested 8K–64K contexts and batch sizes 32, 64, and 128 (Table 3(c)). At approximately 50%, 80%, and 90% achieved sparsity, HATS reaches 1.24–1.47×, 1.55–1.89×, and 1.71–2.12× speedup, respectively, over shape-matched Dense FlashAttention-2. All measured speedups exceed 1×; within each sparsity setting, they increase with batch size.

The decode-throughput gain persists through projection, feed-forward network (FFN), tensor-parallel communication, and scheduler overhead. Table 3(b) reports Qwen3-30B-A3B decode throughput increasing from 319.1 to 401.2 tokens/s at 88.2% sparsity, a 1.26× speedup. The median batch decode step decreases from 75.2 to 59.8 ms.

5 Related Work

KV-cache retention and compression. KV-cache methods reduce decode cost by limiting the states available to future queries. Fixed sink/window policies and history-based eviction retain positions selected by structure or accumulated importance (Xiao et al., 2024; Zhang et al., 2023; Liu et al., 2023). Prompt-observation methods identify important prompt positions from the attention observed before generation (Li et al., 2024), whereas RefreshKV periodically refreshes a smaller cache using full-context attention (Xu et al., 2025). Head-specific methods instead preserve the full cache for retrieval heads while compressing streaming or non-retrieval heads (Xiao et al., 2025; Tang et al., 2025). ThinKV further adapts quantization and eviction to the importance of reasoning segments (Ramachandran et al., 2026). These methods reduce the KV states retained or made available to future queries; our setting retains the full cache and targets decode computation rather than cache capacity.

Sparse-attention selection and approximation. Early sparse-attention architectures replace dense connectivity with fixed, local, global, or hash-based patterns (Child et al., 2019; Kitaev et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020). These designs reduce asymptotic attention cost but commonly require architectural changes or training with the sparse pattern. Recent inference-time methods instead select tokens or pages from the current query using per-page metadata, partial dimensions, low-rank representations, or frequency components (Tang et al., 2024; Ribar et al., 2024; Singhania et al., 2024; Behnam et al., 2025; Wang et al., 2026b; Mu et al., 2025). Other methods adapt the selection budget to estimated attention mass (Zhu et al., 2025; Lin et al., 2025; Zhu et al., 2026). MagicPIG uses importance-sampling estimators, while vAttention combines top-k selection

with random sampling and provides user-specified approximation guarantees (Chen et al., 2025; Desai et al., 2026). RESA estimates the aggregate contribution omitted by sparse attention (Yang et al., 2026). This line of work reduces attention computation through explicit selection or a modified estimator; our setting avoids a separate selector and retains the original attention estimator.

Selection reuse and execution-integrated sparsification. TidalDecode reuses positions selected by occasional full-attention layers across neighboring Transformer layers, whereas LServe reuses page-selection results across adjacent decode tokens (Yang et al., 2025b;c). PRR predicts likely blocks from previous selections, overlaps speculative attention with the current selector, and repairs blocks missed by the prediction (Wang et al., 2026a). TidalDecode and LServe amortize explicit selection overhead, whereas PRR overlaps selection with attention computation and repairs missed blocks.

Modern attention kernels process the KV cache in tiles to reduce memory traffic (Dao et al., 2022; Dao, 2024). BLASST integrates threshold-based sparsification into this traversal: it scores each block with the current query, conditionally skips its value-side work, and evaluates fixed sequential and reverse orders (Yuan et al., 2026). More generally, decisions based on the running maximum make traversal order an algorithmic design dimension for this class of methods. Our work formalizes the resulting sparsity gap and studies history-guided traversal within this broader design space.

6 Limitations

HATS benefits from traversal reordering only when the original traversal leaves a meaningful sparsity gap. In models with pronounced attention sinks, such as Llama-3.1-8B, high-scoring blocks often already occur near the beginning of the traversal, leaving less room for improvement. Abrupt shifts in high-scoring blocks across adjacent tokens can also weaken the history signal and reduce the resulting sparsity gain.

HATS retains the full KV cache and computes QK scores for every block. It only skips value loading, softmax updates, and value aggregation for blocks rejected by the threshold rule. HATS targets value-side computation and is complementary to methods that reduce KV-cache storage or QK computation.

7 Conclusion

Threshold-based block sparsification is intrinsically sensitive to traversal order: when high-scoring blocks are visited late, the running maximum remains weak and retains value-side work that a stronger traversal could omit. HATS shrinks this traversal-induced sparsity gap by using previous-token block scores to prioritize likely high-scoring KV blocks. Global-max-first traversal characterizes the maximum sparsity attainable under fixed current-token scores and threshold. Across the evaluated settings, HATS closely approaches this bound, extends the observed quality–sparsity frontier, and converts the resulting complete-block skips into decode-attention kernel and decode-throughput gains. These results establish traversal order as a design dimension for threshold-based sparse attention.

AI Use Statement

Mathematical claims and proofs. Generative AI tools were used to assist in formulating and proving the mathematical claims in this paper, specifically Proposition 1 and its proof in Appendix A.1. AI-assisted proof steps were manually verified line-by-line by the authors, and the final mathematical arguments were checked for logical soundness and consistency with the stated assumptions.

Method implementation. Generative AI tools were used to assist in implementing the method and in writing the experiment scripts. Generated code was reviewed and tested by the authors.

Writing and editing. Generative AI tools were used to polish the writing of portions of this manuscript and to check its LaTeX consistency. The human authors supplied the scientific outline, the core arguments, and the experimental protocols, and reviewed all AI-assisted text for accuracy.

We did not use generative AI tools for designing or providing feedback on the experimental methodology, for interpreting the results, for generating synthetic datasets, for cleaning or reformatting datasets,

for developing the conceptual framework of this work, or for proposing or refining hypotheses; the remaining required disclosure tasks are not applicable to this work.

We have reviewed all AI-assisted work. The authors take responsibility for the final content of this work, including all text, claims, citations, results, and artifacts produced with the aid of generative AI.

Reproducibility Statement

Code and implementation. The HATS implementation is the fused decode-attention kernel described in Section 3.4. The supplementary material provides this kernel source, including the priority-first traversal logic and the threshold-based skip decisions, together with the observation and evaluation code used to produce the reported measurements. All experiments use publicly available models (Qwen3-8B, Qwen3-30B-A3B-Instruct-2507, and Llama-3.1-8B) and benchmarks (RULER, LongBench v2, MATH500, AIME 2024, and GPQA Diamond).

Experimental protocol. All quality runs use dense prefill and sparse autoregressive decode. Within each benchmark, all methods share frozen prompts, tokenizer rendering, chat templates, generation settings, and scorer. Thresholds are method-specific, and achieved sparsity is measured by the complete-block audit of the corresponding run. Calibration and quality inputs are disjoint, as detailed in Appendix B.

Reproducibility artifacts. The mechanism and quality artifacts record resolved configurations, input identities, software and hardware metadata, and reduced outputs. The mechanism study fixes its panel and expectations before replay.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov. Rocketkv: Accelerating long-context llm inference via two-stage kv cache compression, 2025. URL https://arxiv.org/abs/2502.14051.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

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 Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 44169–44190, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/6d50d824ae819d5a961c1d8edc15e833-Paper-Conference.pdf.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.),

International Conference on Learning Representations, volume 2024, pp. 35549–35562, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/98ed250b203d1ac6b24bbcf263e3d4a7-Paper-Conference.pdf.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359. Curran Associates, Inc., 2022. doi: 10.52202/068431-1189. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

Aditya Desai, Kumar Agrawal, Shuo Yang, Alejandro Cuadron Lafuente, Luis Gaspar Schroeder, Matei Zaharia, Joseph E Gonzalez, and Ion Stoica. vattention: Verified sparse attention via sampling. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 52253–52284, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/55cb562b1f5af71f6707f3ff3c7941e6-Paper-Conference.pdf.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu,

Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

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?, 2024. URL https://arxiv.org/abs/2404.06654.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020. URL https://arxiv.org/abs/2001.04451.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/28ab418242603e0f7323e54185d19bde-Paper-Conference.pdf.

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 D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 119406–119432. Curran Associates, Inc., 2025. doi: 10.52202/085713-3986. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ad2f57116d62a9d8dcfaba6a168715e4-Paper-Conference.pdf.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 52342–52364. Curran Associates, Inc., 2023. doi: 10.52202/075280-2279. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a452a7c6c463e4ae8fbdc614c6e983e6-Paper-Conference.pdf.

MAA. American invitational mathematics examination (AIME), 2024. URL https://maa.org/maa-invitational-competitions/.

Junlin Mu, Hantao Huang, Jihang Zhang, Minghui Yu, Tao Wang, and Yidong Li. Sals: Sparse attention in latent space for kv cache compression. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 384–403. Curran Associates, Inc., 2025. doi: 10.52202/085713-0013. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/00a0ebcad584c59dbc439c2af8793638-Paper-Conference.pdf.

Akshat Ramachandran, Marina Neseem, Charbel Sakr, Rangharajan Venkatesan, Brucek Khailany, and Tushar Krishna. Thinkv: Thought-adaptive kv cache compression for efficient reasoning models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 110072–110106, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/b2b502c3629beadda06311386d2c6f73-Paper-Conference.pdf.

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

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient llm inference, 2024. URL https://arxiv.org/abs/2312.04985.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 16692–16723. Curran Associates, Inc., 2024. doi: 10.52202/079017-0532. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1e027da6bec9ceb2ec37951ceeccae93-Paper-Conference.pdf.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Danning Ke, Shikuan Hong, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient kv cache compression through

retrieval heads. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 16632–16646, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/2a98af4fea6a24b73af7b588ca95f755-Paper-Conference.pdf.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference, 2024. URL https://arxiv.org/abs/2406.10774.

Tianyu Wang, Gourav Rattihalli, Aditya Dhakal, Junbo Li, Zhiwei Ren, Dejan Milojicic, and Longfei Shangguan. Predict, reuse, and repair: Accelerating dynamic sparse attention for long-context llm decoding, 2026a. URL https://arxiv.org/abs/2606.30389.

Yifei Wang, Yueqi Wang, Zhenrui Yue, Huimin Zeng, Yong Wang, Ismini Lourentzou, Zhengzhong Tu, Xiangxiang Chu, and Julian McAuley. Fasa: Frequency-aware sparse attention, 2026b. URL https://arxiv.org/abs/2602.03152.

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

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 37228–37253, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5c1ddd2e59df46fd2aa85c833b1b36ed-Paper-Conference.pdf.

Fangyuan Xu, Tanya Goyal, and Eunsol Choi. RefreshKV: Updating small KV cache during long-form generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24878–24893, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1211. URL https://aclanthology.org/2025.acl-long.1211/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Lijie Yang, Zhihao Zhang, Zhuofu Chen, Zikun Li, and Zhihao Jia. Tidaldecode: Fast and accurate llm decoding with position persistent sparse attention. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 5823–5844, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/11440c427f0f76f191ac06b50d7a2517-Paper-Conference.pdf.

Shang Yang, Junxian Guo, Haotian Tang, Qinghao Hu, Guangxuan Xiao, Jiaming Tang, Yujun Lin, Zhijian Liu, Yao Lu, and Song Han. Lserve: Efficient long-sequence llm serving with unified sparse attention. In M. Zaharia, G. Joshi, and Y. Lin (eds.), *Proceedings of Machine Learning and Systems*, volume 7. MLSys, 2025c. URL https://proceedings.mlsys.org/paper_files/paper/2025/file/cc8c6b9d89f7a898a29f58869b238e46-Paper-Conference.pdf.

Weihao Yang, Hao Huang, Ningke Li, Shihao Wang, Darong Yang, Yanqi Pan, Wen Xia, Shiyi Li, and Xiangyu Zou. Resa: Bringing back what sparse attention ignores with residual estimation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 85544–85564, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/89bd6217280d1417370c89ee493ba3c7-Paper-Conference.pdf.

Jiayi Yuan, Cameron Shinn, Kai Xu, Jingze Cui, George Klimiashvili, Guangxuan Xiao, Perkz Zheng, Bo Li, Yuxin Zhou, Zhouhai Ye, Weijie You, Tian Zheng, Dominic Brown, Pengbo Wang, Markus Hoehnerbach, Richard Cai, Julien Demouth, John D. Owens, Xia Hu, Song Han, Timmy Liu, and Huizi Mao. Blasst: Dynamic blocked attention sparsity via softmax thresholding. In A. Chowdhery and Z. Jia (eds.), *Proceedings of Machine Learning and Systems*, volume 8, pp. 843–859. MLSys, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/file/c6ee784cbe46d854843e4c883a3321ef-Paper-Conference.pdf.

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

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

Kan Zhu, Tian Tang, Qinyu Xu, Zhan Jin, Yile Gu, Zhichen Zeng, Rohan Kadekodi, Liangyu Zhao, Ang Li, Arvind Krishnamurthy, and Baris Kasikci. Tactic: Adaptive sparse attention with clustering and distribution fitting for long-context llms. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 30626–30644, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/33f94d79acf71051d6a27f4d8889e20e-Paper-Conference.pdf.

Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Xiao Chuanfu, Dahua Lin, and Chao Yang. Sampleattention: Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. In M. Zaharia, G. Joshi, and Y. Lin (eds.), *Proceedings of Machine Learning and Systems*, volume 7. MLSys, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/file/2d04d97593c8c33d415337f408ed0e1b-Paper-Conference.pdf.

A Theoretical Analysis

A.1 Global-Max-First Optimality

The result in Proposition 1 concerns the sparsity induced by a fixed skip rule. Fix a decode step t, the N_t current-token block scores $\{b_{t,j}\}_{j=1}^{N_t}$ and a threshold $\tau \leq 0$. For a traversal π, define

$$m_{t,k}(\pi) = \max_{1 \leq r \leq k} b_{t,\pi(r)}, \qquad m_{t,0}(\pi) = -\infty, \tag{7}$$

and the corresponding skip set

$$\mathcal{K}_t(\pi, \tau) = \left\{ \pi(k) : b_{t,\pi(k)} - m_{t,k-1}(\pi) < \tau \right\}. \tag{8}$$

The first visited block is retained because its pre-visit running maximum is $-\infty$. The achieved sparsity is $S_t(\pi, \tau) = |\mathcal{K}_t(\pi, \tau)|/N_t$.

Let $m_t^\star = \max_j b_{t,j}$, and let u_t be a block attaining this maximum. The global-max-first traversal $\pi_t^\star$ visits u_t first. For any block $j \in \mathcal{K}_t(\pi, \tau)$ visited at position k under an arbitrary traversal π,

$$b_{t,j} - m_t^\star \le b_{t,j} - m_{t,k-1}(\pi) < \tau. \tag{9}$$

The skipped block cannot attain $m_t^\star$: a maximizing block has a nonnegative difference from every pre-visit running maximum, while the skip inequality is strict and $\tau \le 0$. Under $\pi_t^\star$, block j is therefore visited after u_t and is skipped by Eq. 9. Hence

$$\mathcal{K}_t(\pi, \tau) \subseteq \mathcal{K}_t(\pi_t^\star, \tau), \tag{10}$$

which proves Proposition 1.

The argument also covers multiple maximizers. Every maximizing block is retained under every traversal, and choosing one of them first establishes $m_t^\star$. Defining the skippable-keep set as

$$\mathcal{C}_t(\pi, \tau) = \mathcal{K}_t(\pi_t^\star, \tau) \setminus \mathcal{K}_t(\pi, \tau) \tag{11}$$

gives the identity

$$G_t(\pi, \tau) = \frac{|\mathcal{C}_t(\pi, \tau)|}{N_t}. \tag{12}$$

Thus, the traversal-induced sparsity gap is exactly the fraction of blocks that the oracle skips but the evaluated traversal keeps.

B Calibration and Evaluation Protocols

B.1 HATS Threshold Calibration

HATS uses a held-out RULER calibration panel to choose an initial threshold for a requested sparsity. The panel contains 324 Qwen3-8B sequences: 96 at each of 4K, 8K, and 16K and 36 at 32K. Calibration sample indices are below 32 and do not overlap the quality indices 32–56. Dense prefill is followed by 32 teacher-forced decode steps, and the replay counts only complete KV-block skips.

Let $\lambda = e^\tau$, let L denote the actual KV length, and let S denote achieved sparsity as a fraction. Across the prespecified calibration interval $S \in [0.25, 0.80]$, the HATS measurements are fit by

$$\lambda L = \alpha \exp(\beta S), \qquad \alpha = 0.2401, \quad \beta = 7.8411, \tag{13}$$

with $R^2 = 0.9891$. Figure 6 shows the collapse across context lengths. The fitted formula proposes an initial threshold.

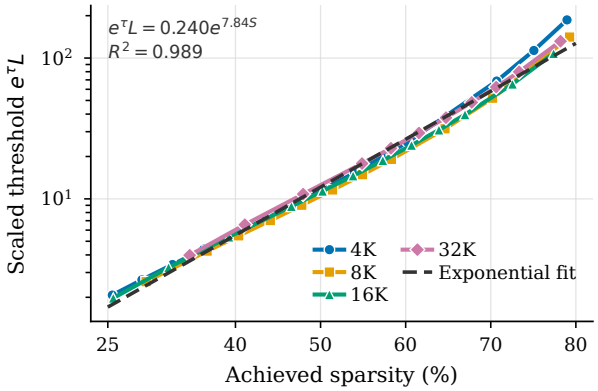


Figure 6: **Scaled thresholds collapse across context lengths.** HATS is calibrated over the 25–80% interval; the dashed curve is the exponential fit in Eq. 13.

B.2 Frozen Inputs, Generation, and Scoring

All quality runs use dense prefill and sparse autoregressive decode. Within a benchmark, methods share frozen prompts, tokenizer rendering, chat template, generation settings, and scorer. Thresholds are method specific, and achieved sparsity comes from the corresponding run's complete-block audit.

The complete RULER-32K panel contains 13 tasks with 25 cases each and reports the official string-matching macro average. The quality–sparsity frontier uses 25 cases from each of seven harder tasks at 8K, 16K, and 32K, for 175 cases per context. It uses greedy decoding, official RULER generation lengths, and disabled thinking. LongBench v2 uses all 503 examples, greedy decoding, token-space middle truncation at 32K, at most 128 generated tokens, and disabled thinking. Calibration and quality inputs are disjoint.

The quality–sparsity frontier contains one dense point and seven sparse targets per method at each context, for 45 accepted measurements. Every point is plotted at its measured sparsity. Dense scores are 97.38, 99.05, and 97.90 at 8K, 16K, and 32K, respectively. Table 4 lists the numerical values underlying Figure 5.

Table 4: **Complete Qwen3-8B RULER hard-subset sweep.** Each entry is measured achieved sparsity (%) / score relative to Dense; column headings are calibration targets.

Context	Method	25%	40%	50%	60%	70%	80%	90%
8K	BLASST	22.0/1.000	33.3/1.000	50.5/1.000	60.2/0.998	70.1/1.000	75.5/0.998	86.6/0.990
8K	HATS	20.4/1.000	36.0/1.000	49.5/0.998	61.6/0.998	71.8/1.001	74.0/1.002	86.8/1.001
16K	BLASST	29.6/1.000	46.8/1.000	52.3/0.998	62.0/0.998	70.0/0.998	78.2/0.998	88.1/0.992
16K	HATS	36.2/1.000	49.6/1.000	56.1/1.000	62.1/1.000	72.6/1.000	80.9/0.998	89.7/0.998
32K	BLASST	25.7/1.000	46.5/1.000	51.7/1.000	61.3/0.998	72.9/0.997	79.1/1.000	88.6/0.967
32K	HATS	33.6/1.000	46.0/1.000	52.4/1.000	64.4/1.000	74.6/0.998	78.8/1.000	90.7/0.997

C Mechanism Controls and Design Analysis

C.1 Fixed-Score Replay and Order Controls

The mechanism study uses 36 frozen Qwen3-8B BF16 RULER sequences: four tasks, three sequences per task, and 8K, 16K, and 32K contexts. Dense prefill is followed by 32 teacher-forced decode steps. Each trace records FP32 current-token block scores for every model layer, KV head, and 128-token KV block. Statistics start at decode token 1 so that a previous-token priority list exists. Replays hold the current-token scores, threshold, forced-retain policy, and complete-block denominator fixed.

Table 5 changes only traversal order. Random and reverse orders can reduce sparsity relative to the original traversal; arbitrary reordering therefore does not close the traversal-induced gap.

Table 5: **Only history-guided and oracle orders exceed the original traversal.** Achieved complete-block sparsity under fixed-score, shared-threshold replay.

Context	τ	BLASST	Random	Reverse	HATS	Oracle
8K	-6.0	52.57%	38.20%	38.47%	60.76%	60.80%
16K	-6.5	53.22%	42.13%	43.86%	62.65%	62.77%
32K	-7.0	51.24%	45.29%	48.45%	63.30%	63.47%

The broader Qwen3-8B replay evaluates 110 shared thresholds at each of 4K, 8K, 16K, and 32K. At each of the 440 context–threshold settings, HATS has higher mean achieved sparsity. Shuffling the previous-token score-to-block mapping reduces final-maximum coverage, whereas the true history reaches all current query-head maxima within the first eight visits for 92.9–96.6% of the layer–KV-head–token units. Directly reusing the previous token's skip mask would incorrectly skip 7.0–8.0% of blocks that the current-token oracle retains. These controls support using history for traversal order, not for current-token skip decisions.

C.2 Priority Budget

Figure 7 varies the priority budget on the same 36 frozen traces. The $H = 0$ point is BLASST with its original traversal. Most of the gain appears by $H = 4$, while $H = 8$ closes 98.6–99.5% of the oracle gap. Increasing the budget from 8 to 16 adds only 0.01–0.10 percentage points across 8K–32K. We therefore use $H = 8$ as an empirical saturation point rather than a theoretically optimal budget.

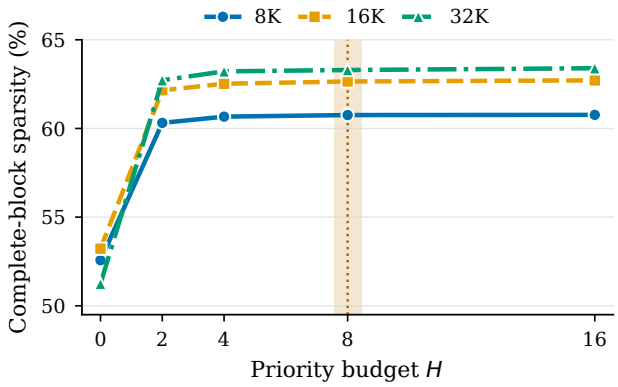


Figure 7: **Priority-budget saturation.** Achieved complete-block sparsity under shared thresholds on 36 Qwen3-8B RULER traces. The highlighted point marks the fixed $H = 8$ setting used in the paper.

Table 6 isolates the implementation cost of the longer priority prefix. It fixes the Qwen3-8B decode-attention shape at 32K context and batch 64 on one A100 80GB GPU, sets $\tau = -100$ to retain every block, and reports median CUDA-event latency over 100 repeats. Increasing H from 2 to 8 adds 26.62 μs (0.44%), while $H = 16$ adds 61.44 μs (1.02%). Together with the saturation in Figure 7, this fixed-work control supports $H = 8$: it captures nearly all of the sparsity gain without the longer-prefix overhead of $H = 16$.

Table 6: **Fixed-work kernel latency versus priority budget.** Zero complete-block skips; values are median CUDA-event latency over 100 repeats.

H	Skip	Latency (μs)	Δ vs. $H = 2$
2	0%	5995.52	+0.00 μs (+0.00%)
4	0%	6000.64	+5.12 μs (+0.09%)
8	0%	6022.14	+26.62 μs (+0.44%)
16	0%	6056.96	+61.44 μs (+1.02%)

C.3 Per-Head Top-1 Prefix

The construction in Section 3.2 spends the first positions of the priority list on the previous-token Top-1 block of every query head in the group, then ranks the remaining blocks by their minimum score across the group. Table 7 compares this per-head Top-1 prefix with group-min ranking alone, holding $H = 8$, the scores, thresholds, and replay decisions fixed. The per-head Top-1 prefix consistently increases complete-block sparsity, although the improvement is small.

Table 7: **The per-head Top-1 prefix raises complete-block sparsity.** Shared thresholds as in Table 5; differences are in percentage points, and confidence intervals are paired bootstrap 95% intervals over frozen cases.

Context	Group-min	+ Per-head Top-1	Difference	95% CI
8K	60.71%	60.76%	+0.04	[0.02, 0.07]
16K	62.49%	62.65%	+0.16	[0.09, 0.23]
32K	62.94%	63.30%	+0.35	[0.27, 0.43]

Across the range where group-min sparsity is 25–80%, the per-head Top-1 prefix gives higher sparsity at all 11, 12, and 11 replayed thresholds for 8K, 16K, and 32K, respectively.

C.4 History Staleness

Figure 8 tests how far previous-token scores can be carried forward before their ordering becomes less useful. At 32K, it rebuilds the $H = 8$ priority list from $t-1$, $t-2$, $t-4$, and $t-8$ over the same 24 decode steps per trace ($t \geq 8$). For Qwen3-8B, the share of groups with a changed top block increases from 41.75% to 63.18%. Among these groups, the list still contains 92.45% of the current top blocks and attains 92.35% of the oracle sparsity gain at $t-8$. Llama-3.1-8B changes less often (19.68% to 28.22%) and attains 97.08% of the gain at $t-8$. Thus, older history weakens the ordering signal gradually rather than invalidating it when a top block changes.

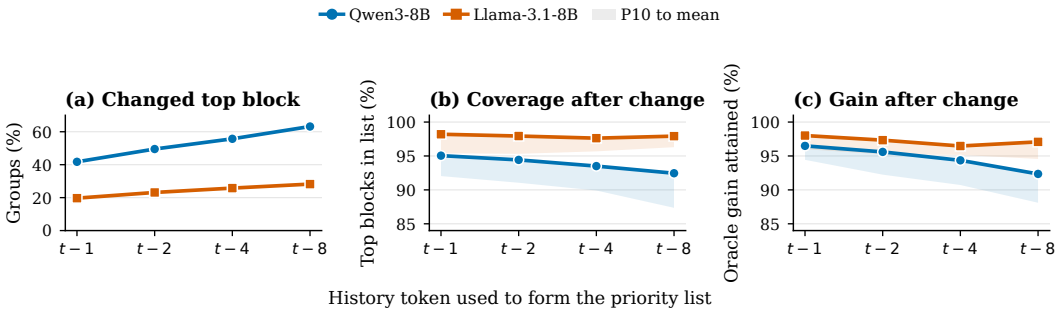


Figure 8: **History eight steps old still guides traversal.** (a) Share of GQA groups whose current top-scoring block differs from the selected history block. (b) Fraction of those groups whose $H = 8$ priority list still contains the current top block. (c) Fraction of the oracle's additional skipped blocks attained over sequential traversal by HATS.

D GPU Implementation and Reproducibility

D.1 Device Execution

The Ampere implementation supports BF16 single-query decode on SM80 and SM86. For Qwen3-8B, it uses 32 query heads, eight KV heads, head dimension 128, 128-token KV blocks, and $H = 8$. The priority list remains on the GPU across decode steps.

For every visited block, the production path loads Key, computes the fresh QK logits, and reduces them to one block score per query head. Each query head maintains its own running maximum. Value loading, blockwise softmax, and value aggregation are omitted only when every query head sharing the KV block skips it. The current block scores are retained on device and used to emit the next token's priority list. This path requires neither an additional QK pass nor host synchronization between decode steps.

D.2 Performance Measurement Boundaries

The kernel experiment uses the Qwen3-8B attention geometry on one A100 80GB GPU at 32K context and batch 64. Dense FlashAttention-2, BLASST, and HATS receive identical Q/K/V inputs. Sparsity is collected by an untimed audit call, and latency is the CUDA-event median after 20 warmup iterations and 50 measurement repetitions. The matched-sparsity diagnostic interpolates both sparse curves over their common 30–90% range; its mean relative absolute latency difference is 1.72%.

The decode-throughput experiment uses Qwen3-30B-A3B-Instruct-2507 on two A100 80GB GPUs with tensor parallelism 2, 32K context, and batch 24. Prefix-cache fill precedes the timed window. Five warmup iterations are followed by five repetitions of ten decode steps. Throughput is batch output tokens divided by decode time. These measurements cover steady decode after the KV cache is available; they do not measure prefill, TTFT, or complete request latency.