RBS-ATTENTION: RADIUS-BOUNDED SPARSE PREFILL FOR LONG-CONTEXT LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context large language model inference is increasingly limited by prefill, where dense self-attention processes the entire prompt before generation begins. Sparse block selection reduces this cost, but a block centroid may hide a highly relevant token among many irrelevant ones. We call this failure mode *mean dilution* and propose RBS-Attention, a training-free sparse-prefill method with two complementary branches. A centroid base branch captures average relevance, while a rescue branch uses the maximum key-block radius and its prompt-, layer-, and head-dependent distribution to identify blocks at risk of underestimation. Independent thresholds and a mask union focus rescue on dispersed blocks while preserving block-sparse FlashAttention execution. On H100 GPUs, RBS-Attention achieves 20.65× standalone prefill-attention speedup and 5.97× end-to-end time-to-first-token speedup at 128K on Qwen3-30B-A3B-Instruct-2507-FP8. On Qwen3-32B, it obtains 88.65 overall RULER accuracy versus 89.52 for dense attention; LongBench-v2, InfiniteBench, and Video-MME provide additional quality evaluation. Matched-density and selector studies further support radius-adaptive rescue as an effective approach to long-context prefill.

1 INTRODUCTION

Large language models (LLMs) increasingly process long contexts for code understanding (Liu et al., 2024b; Jimenez et al., 2024), multi-document question answering (Yang et al., 2018; Liu et al., 2024a), retrieval-augmented generation (Lewis et al., 2020; Izacard & Grave, 2021), and multimodal reasoning (Yue et al., 2024; Fu et al., 2024). Ingesting tens or hundreds of thousands of tokens before generating an answer makes *prefill* a latency bottleneck. Optimized dense-attention kernels improve memory access and hardware utilization (Dao et al., 2022; Dao, 2023), but still perform a quadratic number of query–key interactions.

Sparse-prefill systems reduce this work by inferring attention patterns or selecting coarse key-value blocks before computing token-level attention (Jiang et al., 2024; Fan et al., 2026; Lai et al., 2025; Xu et al., 2025). Their central challenge is to prune aggressively without discarding essential evidence, which may be isolated, irregularly located, or surrounded by irrelevant context.

We study a simple failure mode, *mean dilution*: a block centroid can appear unimportant even when the block contains a critical token. If one relevant key is surrounded by unrelated keys, its contribution to the mean is diluted and the selector may prune the entire block. The affected attention operation then loses that evidence. Our analysis connects this underestimation to the maximum distance of a key from its centroid, or *block radius*. Centroid rank underestimation increases with radius, while the highest-radius quintile contains 39.5% of the top-5% attention blocks (Figure 1). Radius therefore identifies blocks that warrant additional selection capacity.

We propose RBS-Attention, a training-free sparse-prefill method combining *centroid relevance* with *radius-adaptive rescue*. The base branch retains blocks with strong average relevance. The rescue branch augments the centroid score with a radius term, weighted by a coefficient derived from the current prompt's radius distribution for each layer and KV head. Compact blocks receive little uplift; dispersed blocks receive more opportunity for rescue. This allocates additional selection capacity where mean dilution is most likely to hide localized evidence.

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.

RBS applies relative-threshold selection independently to the two branches and unions their masks with the preserved sink, local-window, and recent blocks. The union retains the original relevance path even when radius uplift changes the rescue ranking. Selected block pairs are processed by a block-sparse FlashAttention kernel. Our contribution is the radius-adaptive scoring and coordination of the two branches; the thresholding framework and sparse execution build on existing prefill systems. The method preserves regular GPU execution without requiring training or per-prompt threshold search.

Our primary evaluation covers H100 system performance and quality on dense, MoE, and multimodal Qwen models (Yang et al., 2025; Qwen Team, 2025). At 128K, RBS achieves 20.65× standalone prefill-attention speedup, 11.92× vLLM prefill-attention speedup, and 5.97× end-to-end time-to-first-token (TTFT) speedup on Qwen3-30B-A3B-Instruct-2507-FP8. On Qwen3-32B, it obtains 88.65 overall RULER accuracy versus 89.52 for Dense. LongBench-v2, InfiniteBench, and Video-MME extend the quality evaluation. Supporting experiments examine measured density and isolate the benefit of adaptive rescue. Our contributions are:

- An empirical analysis of mean dilution that links centroid underestimation to key-block dispersion and motivates radius-conditioned rescue.
- A radius-adaptive dual-branch selector that combines base relevance and selective rescue through separate thresholds and a mask union, integrated with regular block-sparse prefill execution.
- An evaluation of long-context acceleration and quality across dense, MoE, and multimodal models, supported by measured-density and selector analyses.

2 Background and Motivation

2.1 Long-context prefill attention

Given query, key, and value matrices $Q, K, V \in \mathbb{R}^{N\times d}$, causal self-attention computes

$$O_i = \sum_{j\le i} \frac{\exp(q_i^\top k_j/\sqrt{d})}{\sum_{\ell\le i}\exp(q_i^\top k_\ell/\sqrt{d})} v_j. \tag{1}$$

Prefill applies this operation to all prompt tokens before generation, incurring quadratic cost in N. Unlike decode, it processes a large triangular attention matrix rather than one new query, making prefill acceleration complementary to KV-cache and decode optimizations.

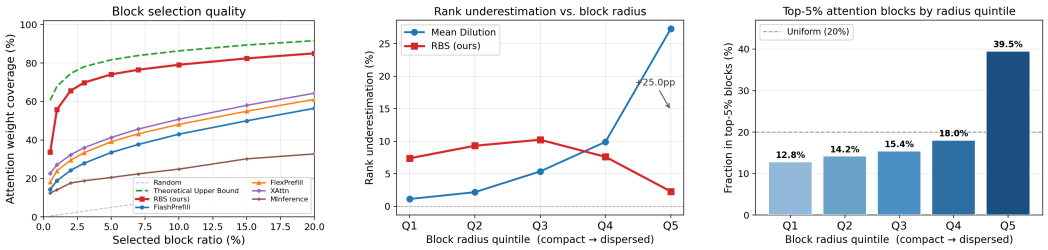
Block-sparse prefill partitions the matrix into query and key blocks of size B and computes token-level attention only inside selected block pairs. This preserves GPU-friendly memory access, but requires a cheap selector that does not discard important evidence.

Existing selection strategies. Pattern-based methods infer sparse layouts from probe queries or partial attention, exploiting regular structures such as local windows and retrieval columns. Block-scoring methods instead summarize each key block and estimate its relevance to query blocks. Centroid summaries are efficient, but assume that the mean represents the tokens inside the block; heterogeneous prompts can violate this assumption.

The two strategies face different sources of uncertainty. A pattern inferred from a small set of probes may vary across layers, heads, tasks, and positions. A centroid can be computed directly from the current keys, but compresses all within-block variation into one vector. Our focus is the latter failure: how to retain the efficiency of block summaries while recognizing when average alignment alone is an unreliable basis for pruning.

2.2 Mean dilution and empirical observation

For a key block K_b with centroid c_b, a mean-only selector uses a score such as $q^\top c_b$. If a critical token $k^\star$ is surrounded by keys pointing in unrelated directions, the block can receive a low mean score even though $q^\top k^\star$ is large. We call this *mean dilution*. Pruning that block removes the relevant



(a) Block selection quality under varying budgets.

(b) Mean-only scoring underestimates dispersed blocks.

(c) Q5 contains 39.5% of top-5% blocks.

Figure 1: Mean dilution and radius-adaptive rescue. **(a)** RBS captures more attention mass at a given selected-block ratio; the dashed curve is a coverage reference, not the logit bound in Eq. (2). **(b)** Mean-only underestimation increases with radius; rescue improves Q4–Q5. **(c)** Q5 contains 39.5% of top-5% attention blocks, versus the uniform reference of 20%.

key and value from the affected attention operation, potentially changing the answer in retrieval or multi-hop reasoning.

Figure 1 links this failure to block dispersion. At the same selected-block ratio, RBS captures more oracle attention mass than the displayed baselines. Mean-only rank underestimation increases across radius quintiles; RBS reduces it in Q4 and Q5, by 25 percentage points in Q5, though not in the lower-radius quintiles. Moreover, Q5 contains 39.5% of top-5% attention blocks, compared with 20% under a uniform distribution. These observations motivate retaining the efficient centroid path while dedicating a rescue path to dispersed blocks.

This distinction matters when a long-context answer depends on a short span buried in unrelated material. A low centroid score does not mean that every token in the block is irrelevant. Yet uniformly preserving all dispersed blocks would spend computation on variation that need not align with the query. The desired selector therefore needs both an ordinary relevance path and a query-dependent way to give at-risk blocks another opportunity for selection.

Radius as a rescue signal. For a simple illustration, suppose a block of size $B \geq 2$ contains one key u and $B-1$ zero keys, with q positively aligned with u. Then $c_b = u/B$ and $r_b = (1-1/B)\|u\|_2$. The centroid score is only $q^\top u/B$, although a token in the block has score $q^\top u$. The full radius term recovers this gap in this aligned example. This makes explicit how averaging can underestimate a localized match, and why within-block deviations carry useful information.

A useful geometric objective is the largest token logit in a key block, $s_b(q) = \max_{k \in K_b} q^\top k$. With centroid c_b and radius $r_b = \max_{k \in K_b} \|k - c_b\|_2$, Cauchy–Schwarz gives

$$s_b(q) \leq q^\top c_b + \|q\|_2 r_b = U_b^{\mathrm{L2}}(q). \tag{2}$$

To see why, write each key as $k = c_b + (k - c_b)$. Its deviation from the centroid score is $q^\top(k - c_b)$, which is at most $\|q\|_2\|k - c_b\|_2 \leq \|q\|_2 r_b$. The centroid and radius therefore describe complementary aspects of a block: its average alignment and the potential size of a hidden token-level contribution.

The radius limits how much a token can exceed the centroid score, but using its full contribution for every block can over-prioritize dispersion unrelated to the query. RBS therefore adapts this contribution to the radius distribution and preserves a separate centroid branch. Appendix G develops the geometry and its relationship to query-aware box bounds (Tang et al., 2024).

3 Method

RBS-Attention retains centroid-based relevance while adding a radius-adaptive rescue branch for blocks susceptible to mean dilution. The two branches select a shared block-sparse mask for prefill execution.

3.1 BLOCK STATISTICS AND RADIUS-ADAPTIVE COEFFICIENT

We partition the sequence into blocks of size B and summarize each key block K_b by its centroid and maximum radius:

$$c_b = \frac{1}{|K_b|} \sum_{k \in K_b} k, \qquad r_b = \max_{k \in K_b} \|k - c_b\|_2. \tag{3}$$

For each prompt, layer, and KV head, let r_{low} and r_{high} be the median and 90th percentile of the block radii. For $r_{\text{high}} > r_{\text{low}}$, define

$$\beta_b = \text{clamp}\left(\frac{r_b - r_{\text{low}}}{r_{\text{high}} - r_{\text{low}}}, 0, 1 \right). \tag{4}$$

Blocks below the median receive no radius uplift; the weight grows continuously to one at the upper quantile. This adapts rescue to the current radius distribution without per-prompt threshold search.

For a compact block, individual keys remain close to the centroid, so the base branch represents their relevance cheaply. A dispersed block can instead mix a few relevant keys with many unrelated ones. Its larger rescue coefficient gives those keys another selection opportunity, while the query-dependent score still determines whether the block passes its threshold.

3.2 DUAL-BRANCH BLOCK SCORING AND SELECTION

For query q, the base branch captures average relevance and the rescue branch adds dispersion-dependent uplift:

$$\ell_{\text{base}}(q, b) = q^\top c_b, \tag{5}$$

$$\ell_{\text{rescue}}(q, b) = q^\top c_b + \|q\|_2 r_b \beta_b. \tag{6}$$

The radius term derives from Eq. (2); with $\beta_b < 1$, it acts as a risk score rather than a strict upper bound (Appendix G).

Query-block aggregation. Let Q_i be query block i and $\mathcal{C}_i$ its causal candidate key blocks. For branch $z \in \{\text{base}, \text{rescue}\}$, aggregate token logits as

$$S^z_{i,b} = \sum_{q_t \in Q_i} \exp\left(\ell_z(q_t, b)/\sqrt{d} - m^z_i \right), \tag{7}$$

where m^z_i is the maximum scaled logit over $Q_i \times \mathcal{C}_i$. This shared shift preserves relative scores across candidate blocks; numerical details are in Appendix A.

Aggregation converts token-query evidence into a decision for an entire query–key tile. Individual queries contribute before their scores are combined, while execution retains a regular block-level mask. The same key summaries are reused across query blocks, and the centroid dot products computed for the base branch are also reused by the rescue branch.

Independent thresholds and mask union. Following prior relative-threshold selection (Fan et al., 2026), each branch uses its own α_z:

$$\mathcal{M}^z_i = \left\{ b \in \mathcal{C}_i : S^z_{i,b} \geq \alpha_z \max_{b' \in \mathcal{C}_i} S^z_{i,b'} \right\}. \tag{8}$$

The final selected set is the union

$$\mathcal{M}^{\text{final}}_i = \mathcal{M}^{\text{base}}_i \cup \mathcal{M}^{\text{rescue}}_i \cup \mathcal{M}^{\text{sink/window/last}}_i. \tag{9}$$

The last set preserves causal sink, local-window, and recent blocks (Xiao et al., 2024; Beltagy et al., 2020); all retained blocks are counted once in actual density. Lower thresholds admit more candidates, but retention remains content-dependent. Thresholds are fixed after calibration: they specify how far below a branch's strongest candidate a block may fall, rather than how many blocks must survive. Branch overlap also changes the final retention rate.

Independent thresholds preserve the base relevance path. Although rescue logits are at least as large as base logits, radius uplift can also raise the rescue branch's maximum, so the two masks need not be nested. Their union preserves every base-selected block while providing additional rescue opportunities.

3.3 SPARSE EXECUTION AND COMPLEXITY

Selected block pairs use a block-sparse FlashAttention kernel (Dao et al., 2022; Dao, 2023), with token-level dot products, causal masking, and softmax over retained keys. Centroids and radii affect selection only: retained tiles use their original token-level keys and values. Approximation comes from omitted block interactions, rather than replacing the attention computation inside a retained tile with its summary. Algorithm 1 gives the complete procedure.

Algorithm 1: RBS-Attention sparse prefill

Input: Queries Q, keys K, values V, block size B, thresholds $\alpha_{\text{base}}, \alpha_{\text{rescue}}$, sink/window/recent rules.
Output: Attention output O.

Partition Q, K, V into blocks.
foreach *key block* K_b **do**
  $c_b \leftarrow \frac{1}{|K_b|} \sum_{k \in K_b} k, \quad r_b \leftarrow \max_{k \in K_b} \|k - c_b\|_2$
Compute $r_{\text{low}} = \text{quantile}(r, 0.5)$ and $r_{\text{high}} = \text{quantile}(r, 0.9)$.
Compute β_b from Eq. (4) for all key blocks b.
foreach *query block* Q_i **do**
  foreach *causal candidate key block* K_b, $b \in \mathcal{C}_i$ **do**
    Compute $\ell_{\text{base}}(q_t, b) = q_t^\top c_b$ for $q_t \in Q_i$
    Compute $\ell_{\text{rescue}}(q_t, b) = q_t^\top c_b + \|q_t\|_2 r_b \beta_b$ for $q_t \in Q_i$
  Aggregate both branches using Eq. (7) with shared scales across candidates
  $\mathcal{M}_i^{\text{base}} \leftarrow \{b \in \mathcal{C}_i : S_{i,b}^{\text{base}} \geq \alpha_{\text{base}} \max_{b' \in \mathcal{C}_i} S_{i,b'}^{\text{base}}\}$
  $\mathcal{M}_i^{\text{rescue}} \leftarrow \{b \in \mathcal{C}_i : S_{i,b}^{\text{rescue}} \geq \alpha_{\text{rescue}} \max_{b' \in \mathcal{C}_i} S_{i,b'}^{\text{rescue}}\}$
  $\mathcal{M}_i^{\text{final}} \leftarrow \mathcal{M}_i^{\text{base}} \cup \mathcal{M}_i^{\text{rescue}} \cup \mathcal{M}_i^{\text{sink/window/last}}$
$O \leftarrow \text{BlockSparseFlashAttention}(Q, K, V, \{\mathcal{M}_i^{\text{final}}\})$
return O

Let $M = \lceil N/B \rceil$ and K_{avg} be the average number of retained key blocks per query block. Centroid/radius computation costs $O(Nd)$; token-to-centroid scoring costs $O(NMd)$ and is shared by both branches; sparse attention costs $O(MK_{\text{avg}}B^2d)$. Appendix A gives numerical details and the accounting of computation and workspace.

Since $M \approx N/B$, centroid scoring requires $O(N^2d/B)$ work rather than evaluating every token–token dot product. Query norms require $O(Nd)$ work, and radius uplift and aggregation add $O(NM)$ scalar operations; thresholding and mask union operate on $O(M^2)$ block scores. The resulting gain depends on retaining substantially fewer than M key blocks per query block. This separates the selector's explicit overhead from the expensive attention interactions it avoids, explaining why longer contexts offer more opportunity to amortize selection.

4 EXPERIMENTAL SETUP

Models and systems. We evaluate system performance on H100 GPUs using Qwen3-30B-A3B-Instruct-2507-FP8 (Yang et al., 2025), vLLM 0.10.0 (Kwon et al., 2023), and FlashAttention 2.8.3. Across 16K–256K contexts, we measure standalone prefill-attention, vLLM attention, and end-to-end time-to-first-token (TTFT), normalizing speedups to Dense under the same model, hardware, length, and serving stack. Quality evaluation additionally includes dense Qwen3-32B and the vision-language model Qwen3-VL-30B-A3B-Thinking-FP8 (Qwen Team, 2025). Supplementary A100 protocols and measurements are detailed in Appendices B and D.

Baselines and benchmarks. We compare with Dense, FlashPrefill (Fan et al., 2026), FlexPrefill (Lai et al., 2025), MInference (Jiang et al., 2024), and XAttn (Xu et al., 2025). The main benchmarks are RULER (Hsieh et al., 2024), covering retrieval and reasoning across context lengths, and LongBench-v2 (Bai et al., 2025), with 503 samples per model. InfiniteBench (Zhang et al., 2024) and Video-MME (Fu et al., 2024) provide additional text and multimodal evaluations; Video-MME

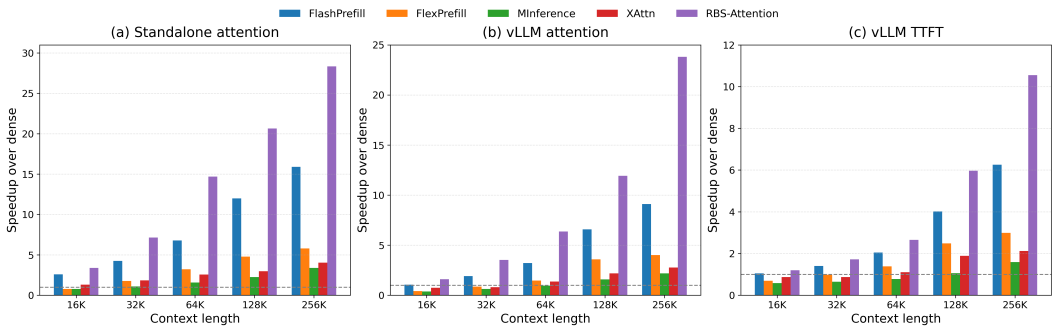


Figure 2: H100 speedups on Qwen3-30B-A3B-Instruct-2507-FP8: **(a)** standalone prefill attention, **(b)** vLLM prefill attention, and **(c)** end-to-end TTFT. Each speedup is relative to Dense at the same context length and execution setup.

uses 1,000 samples and 32 frames/video. Controlled centroid-only, full-L2, and Quest-style (Tang et al., 2024) comparisons isolate selection behavior in a shared prefill backend (Appendix C).

Selection and measured budgets. RBS and the controlled selector comparisons use query/key block size $B = 128$; other baselines retain their native layouts. RBS uses radius quantiles 0.5/0.9 and default thresholds $\alpha_{\text{base}} = 0.22$, $\alpha_{\text{rescue}} = 0.18$. The primary FlashPrefill and XAttn configurations use $\alpha = 0.12$ and stride 16, respectively. Budget sweeps recalibrate selection parameters; thresholds remain fixed after calibration, while radius normalization adapts directly to each prompt. Actual density ρ is the fraction of valid causal block pairs entering the sparse kernel, including the final union and preserved sink/window/recent blocks. We report measured retention because nominal targets and thresholds do not imply equal budgets across methods or prompts. Main-quality density estimates use selector-only samples: 20 examples per LongBench-v2 length category and one example per RULER task at each length. Accuracy retains the full benchmark evaluation; complete density statistics and measurement definitions are in Appendix B.

5 Results

5.1 Long-context acceleration

At 128K context, RBS-Attention achieves 20.65× standalone attention speedup, 11.92× vLLM attention speedup, and 5.97× end-to-end TTFT speedup on H100 (Figure 2). Its standalone gain exceeds FlashPrefill's 11.98×, FlexPrefill's 4.78×, MInference's 2.25×, and XAttn's 2.97×. Gains increase with context length as the saved attention work amortizes selection overhead; shorter contexts yield smaller benefits.

Standalone attention measures kernel-level savings; vLLM attention includes integration costs, and TTFT additionally includes work outside attention. Quality and retained-density measurements below characterize these default configurations, rather than treating their speedups alone as equal-budget comparisons.

5.2 Budget–accuracy trade-off on RULER 128K

Table 1 reports the RULER-128K budget sweep. RBS has the highest sparse accuracy within each nominal-target row, reaching 80.28% at 5.34% actual density and 88.36% at 11.03%, versus Dense's 89.69%. We show actual retention because nominal targets are not exact density matches.

The measured operating points also show that RBS can improve accuracy while retaining fewer blocks. At the nominal 5% target, RBS reaches 80.28 accuracy at 5.34% density, compared with MInference's 76.41 at 5.90%. At the 10% target, the same pattern holds: 88.36 at 11.03% versus 87.18 at 11.45%. The latter RBS point is only 1.33 accuracy points below Dense while retaining approximately one ninth of the causal block pairs.

Table 1: RULER-128K on Qwen3-30B-A3B-Instruct-2507-FP8. Cells give actual density / accuracy (%). Dense: 89.69% accuracy at 100% density. Bold denotes the best sparse accuracy within each nominal-target row.

Target	FlashPrefill	XAttn	FlexPrefill	MInference	RBS-Attention
1%	1.77 / 56.65	1.02 / 44.51	1.17 / 31.64	1.01 / 50.83	1.80 / **60.36**
2%	1.91 / 63.32	1.71 / 50.28	2.25 / 41.41	1.82 / 68.58	2.03 / **70.51**
5%	4.69 / 75.36	3.81 / 60.47	3.59 / 50.72	5.90 / 76.41	5.34 / **80.28**
10%	10.19 / 85.01	7.63 / 71.01	6.69 / 65.87	11.45 / 87.18	11.03 / **88.36**

Table 2: RULER-128K with strictly matched actual-density anchors, on the same model as Table 1. Cells give actual density / accuracy (%); all measured densities are within 0.10 percentage point of their anchor. Dashes denote unmeasured settings.

Anchor	FlashPrefill	MInference	XAttn	FlexPrefill	RBS-Attention
2.03%	2.038 / 67.13	1.944 / 68.40	1.802 / 48.40	2.101 / 52.40	2.030 / **70.51**
5.34%	5.300 / 74.73	5.286 / 76.87	5.311 / 66.77	5.352 / 58.56	5.340 / **80.28**
11.03%	11.035 / 79.62	10.950 / 83.44	11.025 / 78.31	10.973 / 72.99	11.030 / **88.36**

Table 3: LongBench-v2 quality (503 samples/model) and estimated mean density $\bar{\rho}$ (%). Density uses 20 selector-only samples per length category; p50/p90 and protocols are in Appendix B. Bold denotes the best sparse accuracy.

	Qwen3-32B					Qwen3-30B-A3B (FP8)				
Method	Short	Med.	Long	Total	$\bar{\rho}$	Short	Med.	Long	Total	$\bar{\rho}$
Dense	0.450	0.367	0.380	0.394	100	0.422	0.358	0.388	0.388	100
FlashPrefill	0.378	0.349	0.361	0.362	14.149	0.400	0.340	0.315	0.356	11.060
XAttn	**0.439**	0.340	**0.370**	**0.376**	29.197	0.383	0.316	0.352	0.348	26.129
FlexPrefill	0.422	0.293	0.352	0.352	11.397	0.328	**0.350**	0.343	0.340	16.544
MInference	0.417	0.335	0.315	0.360	55.763	**0.406**	0.302	0.352	0.348	51.851
RBS-Attention	0.411	**0.353**	0.361	**0.376**	9.719	**0.406**	0.340	**0.370**	**0.370**	7.912

Strictly matched actual budgets. Table 2 recalibrates competing methods to three measured RBS density anchors, each within 0.10 percentage point. RBS has the highest measured accuracy at every anchor. Relative to MInference, its gains are 2.11, 3.41, and 4.92 accuracy points at 2.03%, 5.34%, and 11.03%, respectively; the corresponding gains over FlashPrefill are 3.38, 5.55, and 8.74 points. The central anchor also includes XAttn and FlexPrefill. Thus, the quality advantage persists across multiple closely aligned retained budgets, supporting more effective selection rather than simply retaining more blocks. These accuracy-only recalibrations are separate from the configurations used for system timing.

5.3 Main quality results: RULER and LongBench-v2

On LongBench-v2 (Table 3), RBS matches the best sparse total on Qwen3-32B at 0.376, with 9.719% estimated mean density, versus 29.197% for XAttn at the same score. On Qwen3-30B-A3B-Instruct-2507-FP8, RBS obtains 0.370 at 7.912% estimated density, compared with Dense's 0.388. These results support quality preservation on both dense and MoE architectures.

The length groups provide complementary evidence beyond the overall scores. On the MoE model, RBS obtains 0.370 on the long subset, compared with 0.315 for FlashPrefill and 0.352 for XAttn. On the dense model, RBS achieves the best sparse medium-length score, 0.353, and ties the best sparse total. The benefit is therefore not restricted to synthetic retrieval settings. The mean-density estimates characterize each method's overall operating configuration, rather than an individual length subset.

On Qwen3-32B RULER (Table 4), RBS scores 88.65 overall versus Dense's 89.52. At 128K, it reaches 80.57 versus Dense's 80.70 while retaining an estimated 6.028% of causal block pairs. MInference reaches 80.96 at an estimated 47.968% density, illustrating why accuracy should be

Table 4: Qwen3-32B RULER accuracy (%) and estimated 128K density ρ_{128} (%). Density uses one example/task/length; full density statistics are in Appendix B. Bold denotes the best sparse accuracy.

Method	4K	8K	16K	32K	64K	128K	Overall	ρ_{128}
Dense	93.30	91.90	92.00	89.70	89.50	80.70	89.52	100
FlashPrefill	91.00	89.20	88.80	81.50	81.80	68.10	83.40	10.075
XAttn	92.40	**91.30**	90.90	86.80	87.20	79.20	87.97	29.196
FlexPrefill	72.30	75.10	78.30	79.00	77.70	73.60	76.00	11.719
MInference	**92.61**	91.24	90.41	87.59	88.96	**80.96**	88.63	47.968
RBS-Attention	91.10	89.90	**91.20**	**89.50**	**89.60**	80.57	**88.65**	6.028

Table 5: InfiniteBench on Qwen3-32B (unit-interval scores). Task abbreviations follow the benchmark; Avg. is the macro average. Bold denotes the best sparse score.

Method	Ret. KV	Ret. PassKey	Ret. Num	En. Sum	En. MC	En. QA	Zh. QA	En. Dia	Math. Find	Code Debug	Avg.
Dense	0.244	1.000	1.000	0.289	0.105	0.087	0.115	0.360	0.451	0.066	0.372
FlashPrefill	0.148	**1.000**	**1.000**	0.291	0.100	0.075	0.106	0.170	**0.463**	0.028	0.338
XAttn	0.194	**1.000**	**1.000**	0.272	**0.105**	**0.084**	0.109	0.310	0.413	0.030	0.352
RBS	**0.204**	**1.000**	0.998	**0.296**	0.092	0.083	**0.112**	**0.325**	0.459	**0.053**	**0.362**

considered together with the retained budget. RBS also obtains 89.50 and 89.60 at 32K and 64K, respectively. Appendix B reports retention across all lengths, clarifying how fixed thresholds yield different actual budgets as context grows.

The context-length breakdown further clarifies the quality trend. RBS and FlashPrefill are close at 4K, with a 0.10-point accuracy difference, but the gap grows to 8.00 points at 32K and 12.47 points at 128K. At 128K, RBS also has lower estimated retention: 6.028% versus 10.075%. This pattern is consistent with the motivation that preserving a small number of important blocks becomes increasingly consequential as evidence is embedded in longer contexts.

5.4 Additional results: InfiniteBench and Video-MME

On InfiniteBench (Table 5), RBS has the best sparse macro average, 0.362 versus 0.338 for FlashPrefill and 0.352 for XAttn. Gains extend beyond passkey retrieval: Ret.KV reaches 0.204 and En.Dia reaches 0.325, compared with FlashPrefill's 0.148 and 0.170. On Video-MME (Table 6), RBS obtains the highest sparse total, 65.98 versus Dense's 66.16, and the highest sparse long-video score, 62.1. This extends the quality evidence to multimodal prefill; the H100 text-model speedups do not measure vision-language inference.

5.5 Supporting evidence for adaptive rescue

Table 7 isolates the selection policy with the same model, prefill backend, $B = 128$, and 130 RULER-128K examples. At approximately 5.34% actual density, RBS has the highest point estimate. Using the unattenuated L2 bound alone does not reproduce this result, consistent with allocating rescue selectively while protecting the base relevance path. Paired difference confidence intervals include zero; these diagnostic results are descriptive rather than statistically conclusive. Full confidence intervals, settings, and task-group breakdowns are in Appendix C.

The task-group scores connect this aggregate behavior to localized-evidence retrieval. RBS reaches 90.00 on retrieval and 93.33 on multi-key tasks, compared with 87.81 and 90.00 for full-L2 selection. The comparison evaluates the complete adaptive-union policy: radius attenuation moderates dispersion-driven uplift, while the separate base branch protects centroid-relevant blocks. This is consistent with the complementary roles of the two branches.

Additional A100 system, sensitivity, calibration, and memory results appear in Appendices D–F.

Table 6: Video-MME on Qwen3-VL (1,000 samples, 32 frames/video). Scores (%); bold: best sparse score.

Method	Short	Med.	Long	Total
Dense	73.6	62.5	58.6	66.16
FlashPrefill	71.2	**61.5**	60.9	65.26
XAttn	**72.8**	61.0	61.2	65.76
FlexPrefill	71.0	57.9	56.3	62.82
MInference	71.4	60.5	59.8	64.72
RBS	72.7	61.2	**62.1**	**65.98**

Table 7: Selector diagnostic (130 RULER-128K examples, Qwen3-30B-A3B FP8). Scores/density (%); Ret./Multi: retrieval/multi-key. Quest-style is a prefill adaptation.

Selector	Density	Overall	Ret.	Multi
Centroid-only	5.300	73.63	88.75	90.00
Full-L2 bound	5.295	72.41	87.81	90.00
Quest-style	5.334	73.71	88.75	90.00
RBS union	5.350	**76.67**	**90.00**	**93.33**

6 Related Work

Efficient attention kernels. FlashAttention and FlashAttention-2 accelerate dense attention through tiling and reduced memory traffic (Dao et al., 2022; Dao, 2023). RBS complements these kernels by selecting fewer block pairs and applying FlashAttention-style execution to the retained causal keys.

Sparse prefill attention. MInference, FlexPrefill, XAttn, and FlashPrefill infer attention structure, sample the attention matrix, or construct coarse block scores (Jiang et al., 2024; Lai et al., 2025; Xu et al., 2025; Fan et al., 2026). RBS focuses on mean dilution in centroid summaries. It builds on relative-threshold selection (Fan et al., 2026), adding radius-conditioned rescue and independent branch masks to retain localized evidence without abandoning centroid relevance.

Query-aware KV-page selection. Quest (Tang et al., 2024) scores KV-cache pages using coordinate-wise min/max bounds and retrieves Top-K pages for decoding. RBS instead selects prefill block pairs with an attenuated radius score and a separate centroid branch; relative thresholds induce variable density rather than fixed cardinality. The distinction is the selection policy and execution stage, not a claim that the L2 bound always dominates the box bound (Appendix G). Our Quest-style experiment isolates scoring within a shared prefill backend.

Heterogeneity-aware sparse attention. SpargeAttention (Zhang et al., 2025) protects blocks failing a self-similarity test from first-stage pruning, then uses an online softmax-aware filter and quantized attention. RBS uses continuous radius-quantile weighting and a separately thresholded rescue branch: dispersion raises a block's rescue score but does not automatically retain it. These are different policies for handling unreliable summaries.

7 Limitations and Conclusion

Limitations. RBS targets prefill; decode requires separate implementation and evaluation, and short contexts offer less opportunity to amortize selector overhead. Radius indicates possible mean dilution, not guaranteed query relevance. Quality and realized density depend on the model, prompt, block size, and thresholds; broader transfer remains to be tested. Our speed claims concern the measured MoE setup, not unmeasured dense or multimodal speedups. The supplementary analyses report sampling uncertainty, selector workspace, and full-model memory separately.

Conclusion. RBS-Attention addresses mean dilution through centroid relevance and radius-adaptive rescue, independently thresholded and combined for block-sparse execution. It achieves 20.65× standalone prefill-attention speedup and 5.97× TTFT speedup at 128K on H100, with quality evaluated across long-context text and multimodal tasks. The results support combining average relevance with selective use of block dispersion for efficient, training-free prefill.

AI USE STATEMENT

Generative AI tools assisted with literature discovery and comparisons, interpreting author-provided experimental results. They also assisted with translation, manuscript drafting, revision and organization, and LaTeX, table, and reference formatting. The authors remain responsible for verifying originality, citations, code, results, and all claims, and for the final content of this work, including material produced with AI assistance.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 specify block statistics, radius normalization, selection, and sparse execution. Section 4 lists models, benchmarks, software versions, baseline configurations, and default hyperparameters. Appendix A provides numerical details and complexity accounting, while Appendix G gives the geometric arguments. Appendix B describes hardware configurations, system metrics, density definitions, and sampling protocols. Controlled selector settings and confidence intervals appear in Appendix C; sensitivity and calibration measurements appear in Appendices E and F. Primary H100 measurements and supplementary A100 experiments are reported separately, with actual retained budgets alongside accuracy to clarify comparison settings.

REFERENCES

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

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

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. URL https://arxiv.org/abs/2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *arXiv preprint arXiv:2205.14135*, 2022. URL https://arxiv.org/abs/2205.14135.

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*, 2026.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. *arXiv preprint arXiv:2405.21075*, 2024.

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? *arXiv preprint arXiv:2404.06654*, 2024.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 874–880, 2021. doi: 10.18653/v1/2021.eacl-main.74. URL https://doi.org/10.18653/v1/2021.eacl-main.74.

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. *arXiv preprint arXiv:2407.02490*, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. Association for Computing Machinery, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. FlexPrefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *The Thirteenth International Conference on Learning Representations*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a. doi: 10.1162/tacl_a_00638. URL https://doi.org/10.1162/tacl_a_00638.

Tianyang Liu, Canwen Xu, and Julian McAuley. RepoBench: Benchmarking repository-level code auto-completion systems. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=pPjZIOuQuF.

Qwen Team. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

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.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

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 of *Proceedings of Machine Learning Research*, pp. 69819–69831. PMLR, 2025. URL https://proceedings.mlr.press/v267/xu25ag.html.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL https://doi.org/10.18653/v1/D18-1259.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9556–9567, 2024. doi: 10.1109/CVPR52733.2024.00913. URL https://doi.org/10.1109/CVPR52733.2024.00913.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76397–76413. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhang25ch.html.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. InfiniteBench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.814. URL https://aclanthology.org/2024.acl-long.814/.

A Numerical Details and Implementation Accounting

This appendix supplements Algorithm 1 in Section 3 with equivalent stable score evaluations and a breakdown of computation and storage.

A.1 Comparable scores and numerical rescaling

For branch z and query block Q_i, define the unscaled aggregate and a shared shift as

$$A^z_{i,b} = \sum_{q_t \in Q_i} \exp\left(\ell_z(q_t, b)/\sqrt{d}\right), \tag{10}$$

$$m^z_i = \max_{b \in \mathcal{C}_i} \max_{q_t \in Q_i} \ell_z(q_t, b)/\sqrt{d}. \tag{11}$$

Equation (7) evaluates $S^z_{i,b} = \exp(-m^z_i) A^z_{i,b}$. Since the multiplicative factor is shared by all candidate key blocks, it cancels in Eq. (8). This provides stable exponentiation while preserving the relative-threshold rule.

An equivalent evaluation can form a local maximum $m^z_{i,b} = \max_{q_t \in Q_i} \ell_z(q_t, b)/\sqrt{d}$ and then restore the shared scale:

$$S^z_{i,b} = \exp(m^z_{i,b} - m^z_i) \sum_{q_t \in Q_i} \exp\left(\ell_z(q_t, b)/\sqrt{d} - m^z_{i,b}\right). \tag{12}$$

The factor $\exp(m^z_{i,b} - m^z_i)$ is needed for cross-candidate comparison. These expressions describe equivalent mathematical evaluations; they do not prescribe whether a kernel materializes intermediate logits or performs tiled reductions.

A.2 Computation and workspace

Let $M = \lceil N/B \rceil$ denote the number of blocks. Computing centroids and radii costs $O(Nd)$, followed by quantile computation over M radii for each layer and KV head. The token-query aggregation in Eq. (7) scores all query tokens against key centroids, costing $O(NMd) = O(N^2 d/B)$ up to boundary and causal constant factors. Both branches share these dot products. Query norms cost $O(Nd)$, radius uplift and token aggregation add $O(NM)$ scalar operations, and relative thresholding and mask union operate on $O(M^2)$ block scores.

With an average of K_{avg} selected key blocks per query block, sparse attention costs $O(MK_{\text{avg}}B^2 d)$. The computation reduction comes from retaining $K_{\text{avg}} \ll M$ while amortizing the selector. The algorithm retains quadratic worst-case dependence on context length.

Persistent key summaries use $O(Md)$ storage per KV head: a centroid vector and two scalars, r_b and β_b, per key block. The selected block-index list uses $O(MK_{\text{avg}})$ entries. Temporary selector workspace is a separate quantity: materialized block scores or masks use $O(M^2)$ space, while materializing all token-to-centroid logits uses $O(NM)$ space. Tiled reductions can reduce the latter. Thus, compact radius metadata alone does not imply a small total selector footprint; the supplementary system analysis reports workspace separately from full-model peak memory.

B Extended Evaluation Protocols and Budget Measurements

B.1 Hardware and system metrics

The supplementary controlled experiments use A100 GPUs. Qwen3-30B-A3B-Instruct-2507-FP8 uses tensor parallelism of two (TP=2), while Qwen3-32B uses TP=4. The FP8 checkpoint loader in vLLM 0.10.0 imposes a shard-size divisibility constraint that prevents TP=4 for the MoE model; all methods for that model therefore use the same two-GPU configuration. These experiments cover actual-density matching, selector alternatives, sensitivity, calibration, throughput, and memory. Their latency measurements are separate from the H100 scaling results in Figure 2.

TTFT is measured from request arrival until the first output token is available. Selector time includes block statistics, normalization, scoring, thresholding or Top-K, mask union when applicable,

and index compaction. Sparse-attention time covers attention execution after selection. Prompt throughput is the number of processed prompt tokens divided by batch wall-clock time, normalized to Dense at the same context and batch size. Peak allocated memory is the maximum of `torch.cuda.max_memory_allocated` over TP ranks. Persistent block statistics, temporary workspace, and full-model peak memory are reported separately because they describe different storage costs.

B.2 Actual-density definition and sampling

We define actual density as

$$\rho = \frac{\#\{\text{selected valid causal block pairs}\}}{\#\{\text{all valid causal block pairs}\}}. \tag{13}$$

The numerator counts the final pairs entering the sparse-attention kernel, including the base/rescue union and preserved sink, local-window, or recent blocks. We first compute density per sample and then report its mean, median (p50), and 90th percentile (p90), where available. A target budget is a calibration target rather than a guarantee of identical retention across methods and prompts. The fixed system-prompt density and accuracy-set density are therefore listed separately in Table 12.

For LongBench-v2, the density measurements use a fixed-seed selector-only sample of 20 examples from each of the short, medium, and long categories. Table 8 reports the complete distribution statistics for the original configurations. Their accuracy scores in Table 3 use the full 503-sample evaluation; the density estimates do not.

For RULER, density is measured on one example per task at each context length. Table 9 gives the mean-density estimates for all lengths, complementing the complete accuracy results in Table 4. The overall column is the reported average across lengths. These measurements characterize the budgets of the original configurations without replacing the original accuracy evaluation.

Table 8: Complete LongBench-v2 density estimates for the configurations in Table 3. Mean, p50, and p90 are percentages over the fixed-seed selector-only sample of 20 examples per length category.

Method	Mean	p50	p90
Qwen3-32B			
Dense	100	100	100
FlashPrefill	14.149	9.197	28.047
XAttn	29.197	26.711	43.858
FlexPrefill	11.397	10.203	16.260
MInference	55.763	44.204	87.840
RBS-Attention	9.719	5.515	20.172
Qwen3-30B-A3B-Instruct-2507-FP8			
Dense	100	100	100
FlashPrefill	11.060	7.269	22.325
XAttn	26.129	24.297	37.965
FlexPrefill	16.544	15.863	23.124
MInference	51.851	40.397	81.776
RBS-Attention	7.912	4.638	16.367

Table 9: Estimated mean actual density (%) for the Qwen3-32B RULER configurations in Table 4. Measurements use one example per task at each context length.

Method	4K	8K	16K	32K	64K	128K	Overall
Dense	100	100	100	100	100	100	100
FlashPrefill	81.091	59.781	40.970	26.754	16.872	10.075	39.257
XAttn	69.518	59.482	51.776	43.645	36.201	29.196	48.303
FlexPrefill	30.000	23.689	20.140	16.915	14.999	11.719	19.577
MInference	100.000	99.990	97.353	86.940	70.251	47.968	83.750
RBS-Attention	73.030	48.465	30.185	18.224	10.720	6.028	31.109

C Controlled Selector Accuracy and Uncertainty

To examine the role of radius-adaptive allocation, we compare centroid-only scoring, the unattenuated full-L2 bound, and Quest-style selection in a shared prefill backend. Quest-style uses per-dimension key minima/maxima and query-dependent block selection (Tang et al., 2024); this experiment evaluates the adapted selection policy rather than the official decoding system. RBS combines radius attenuation with separately thresholded base and rescue branches. All four policies use Qwen3-30B-A3B-Instruct-2507-FP8, the same backend, $B = 128$, and the first ten examples from each of the 13 RULER-128K tasks (130 examples in total).

All actual densities are within 0.10 percentage point of 5.34%. RBS has the highest point estimate at 76.67, compared with 73.63 for centroid-only, 72.41 for the full-L2 selector, and 73.71 for Quest-style selection (Table 10). This pattern is consistent with the role of adaptive base/rescue allocation: applying the unattenuated radius contribution to all blocks need not be the best use of a limited retained budget. The paired bootstrap difference intervals between RBS and each comparator include zero, however, so this diagnostic set does not establish statistical significance. The confidence intervals in the table characterize the individual accuracy estimates, not between-method differences. Task-group breakdowns appear in Table 11.

Table 10: Supporting selector comparison on 130 RULER-128K examples with Qwen3-30B-A3B-Instruct-2507-FP8. All selectors share the prefill backend and $B = 128$. Accuracy and density are percentages. Intervals are sample-bootstrap 95% confidence intervals for each accuracy estimate, not intervals for between-method differences. Quest-style denotes a controlled prefill adaptation.

Selector	Actual density	Accuracy	95% CI
Centroid-only	5.300	73.63	[68.00, 81.21]
Full-L2 bound	5.295	72.41	[66.32, 79.79]
Quest-style	5.334	73.71	[66.76, 80.47]
RBS adaptive union	5.350	**76.67**	[69.01, 82.15]

C.1 Selector settings and task-group diagnostics

The centroid-only and full-L2 selectors use selection thresholds of 0.194 and 0.022, respectively, and Quest-style selection uses its density parameter 0.0405. Since these parameters have different meanings across policies, the comparison is aligned using measured density.

The retrieval and multi-key breakdowns in Table 11 agree with the aggregate trend, but the limited diagnostic sample warrants interpreting the differences descriptively. These experiments do not establish a universal ordering of the mathematical tightness of box and L2 bounds.

Table 11: Task-group diagnostics for the same 130-example selector comparison as Table 10. Scores are percentages.

Selector	Retrieval accuracy	Multi-key accuracy
Centroid-only	88.75	90.00
Full-L2 bound	87.81	90.00
Quest-style	88.75	90.00
RBS adaptive union	90.00	93.33

D Additional A100 System Measurements

D.1 Complete 128K sweep

Table 12 reports the additional 128K system sweep on Qwen3-30B-A3B-Instruct-2507-FP8 using A100 GPUs and TP=2. The system measurements use a fixed prompt; the accuracy entries are companion evaluation results. Since the prompt distribution and selection configurations affect retention, the two measured densities are shown separately. These nominal-target companion points are distinct from the strict recalibrations in Table 2 and the 130-example selector ablation in Table 10.

RBS reaches 5.29×, 4.47×, and 3.05× TTFT speedup at the 2%, 5%, and 10% targets, respectively. The corresponding companion accuracy values are 70.51, 80.28, and 88.36. At the central target, the other measured TTFT speedups range from 2.84× to 3.58×. This provides supporting system evidence at low measured density. Because actual densities and accuracy evaluation sets are reported separately, the table should not be interpreted as an exactly equal-density Pareto frontier.

Table 12: Supplementary A100 128K sweep on Qwen3-30B-A3B-Instruct-2507-FP8 (TP=2). Sys./acc. are actual densities on the system prompt and companion accuracy evaluation, respectively; densities and accuracy are percentages. Selector and sparse-attention shares sum to 100% of their reported combined path, not TTFT. Dense's 86.31 is the controlled-subset reference, not the primary full-benchmark 89.69. Companion accuracies retain their source evaluation protocols; the additional 2%/10% points for XAttn, FlexPrefill, and Quest-style use 130 examples.

Method	Target	Sys./acc. density	Selector	Sparse attn.	TTFT gain	Accuracy
Dense	100	100 / 100	—	100.00	1.00×	86.31
FlashPrefill	2	1.826 / 1.910	23.87	76.13	4.89×	63.32
	5	5.071 / 4.690	12.39	87.61	2.90×	75.36
	10	11.147 / 10.190	5.99	94.01	2.04×	85.01
MInference	2	1.568 / 2.022	41.11	58.89	4.84×	68.54
	5	4.184 / 5.056	26.61	73.39	3.58×	76.09
	10	8.609 / 9.995	16.61	83.39	2.93×	82.82
RBS-Attention	2	1.830 / 2.030	31.68	68.32	5.29×	70.51
	5	4.792 / 5.340	16.28	83.72	4.47×	80.28
	10	10.517 / 11.030	7.73	92.27	3.05×	88.36
XAttn	2	1.957 / 1.971	72.14	27.86	5.01×	52.62
	5	4.105 / 3.810	53.07	46.93	3.55×	60.47
	10	10.588 / 10.970	33.38	66.62	1.74×	76.51
FlexPrefill	2	2.908 / 2.041	45.12	54.88	4.51×	41.95
	5	5.269 / 3.590	29.00	71.00	3.21×	50.72
	10	10.762 / 11.065	19.87	80.13	2.32×	77.90
Quest-style	2	1.851 / 1.972	52.38	47.62	4.22×	62.32
	5	4.653 / 4.807	34.73	65.27	2.84×	70.56
	10	10.953 / 11.098	20.64	79.36	2.06×	78.63

D.2 PROMPT THROUGHPUT

Table 13 reports prompt throughput for the 5% target setting. Context denotes the nominal length bucket; the Dense column gives actual total prompt tokens and wall time. Throughput gains are computed against Dense separately for every context and batch size. RBS reaches 4.47× throughput at 128K with batch size one and 3.85× at batch size four. Shorter contexts show smaller gains, consistent with the cost of computing a sparse mask and dispatching sparse kernels.

The MInference and FlexPrefill kernels used in this experiment accept a single sequence. For batch size four, we invoke them sequentially on four independent causal sequences and include dispatch costs in batch wall time. Consequently, their batched results characterize this adaptation rather than an optimized native batching implementation. No requests are concatenated into one causal sequence.

Table 13: Supplementary A100 prompt throughput at the 5% target, using Qwen3-30B-A3B-Instruct-2507-FP8 (TP=2). The upper panel provides absolute Dense measurements; the lower panel gives method tokens/s divided by Dense tokens/s for the same row. FP = FlashPrefill; MI = MInference; Flex = FlexPrefill. Quest denotes the controlled prefill adaptation.

Context	Batch	Dense wall time (ms)	Total prompt tokens	Dense tokens/s
32K	1	1,920.68	31,286	16,289
64K	1	5,734.91	62,486	10,896
128K	1	19,709.12	125,486	6,367
32K	4	6,996.10	118,097	16,880
64K	4	21,265.25	239,377	11,257
128K	4	72,127.35	478,727	6,637

Context	Batch	FP	MI	RBS	XAttn	Flex	Quest
32K	1	1.25×	1.36×	1.27×	0.98×	0.95×	1.29×
64K	1	2.05×	2.24×	2.06×	1.49×	1.69×	2.12×
128K	1	2.90×	3.58×	4.47×	3.55×	3.21×	2.84×
32K	4	1.35×	0.68×	1.36×	0.96×	0.99×	0.99×
64K	4	2.12×	1.24×	2.12×	2.00×	1.79×	2.19×
128K	4	2.83×	2.05×	3.85×	2.39×	3.57×	3.18×

E CONFIGURATION SENSITIVITY

E.1 BLOCK SIZE

Table 14 compares $B \in \{64, 128, 256\}$ after recalibrating each setting to approximately 5.34% actual density. At the reported operating points, $B = 128$ gives the highest RBS accuracy. Smaller blocks increase the number of block summaries and candidate pairs to process, while larger blocks coarsen the grouping of relevant and irrelevant tokens. The observed accuracy depends on the selector as well as the granularity: RBS is below centroid-only at $B = 64$ and $B = 256$ in this sweep. Thus, the evidence supports $B = 128$ as the selected configuration rather than a claim of robustness to arbitrary block size. These sensitivity scores are separate from the 130-example selector comparison in Table 10.

E.2 THRESHOLDS AND MODEL TRANSFER

Table 15 applies the same three pairs of thresholds to the MoE and dense Qwen3 models. The *conservative* and *aggressive* labels refer to observed retention, not guaranteed accuracy: the conservative setting retains more blocks, and the aggressive setting retains fewer. Fixed thresholds yield different actual densities across models. In particular, the default pair produces 4.997% mean density on Qwen3-30B-A3B and 6.029% on Qwen3-32B in this diagnostic. This is expected for content-dependent selection and motivates reporting measured density with every configuration.

Accuracy need not vary monotonically with density when both branch thresholds change: they alter which blocks are selected as well as how many. The cross-model diagnostic demonstrates this

Table 14: Block-size sensitivity on RULER-128K with Qwen3-30B-A3B-Instruct-2507-FP8. Each setting is recalibrated to the 5.34% density anchor. Density and accuracy are percentages. The reference RBS $B = 128$ row is the primary budget-sweep operating point.

Selector	Block size	Calibration parameter	Actual density	Accuracy
Centroid-only	64	0.100	5.302	74.17
	128	0.194	5.300	74.73
	256	0.400	5.276	72.15
RBS union	64	0.096	5.343	71.86
	128	Reference	5.340	80.28
	256	0.382	5.361	70.73

distinction and should be interpreted within its own evaluation rather than compared directly with the full primary benchmark scores.

Table 15: Supplementary cross-model threshold sensitivity on RULER-128K. Density and accuracy are percentages. The same threshold pairs are used for both models; the radius normalization adapts to each prompt at runtime.

Model	Setting	α_{base}	α_{rescue}	Mean density	p90 density	Accuracy
Qwen3-30B-A3B FP8	Conservative	0.16	0.24	6.478	7.282	75.31
	Default	0.22	0.18	4.997	5.441	71.86
	Aggressive	0.24	0.16	4.676	5.048	75.12
Qwen3-32B	Conservative	0.16	0.24	7.983	8.845	65.05
	Default	0.22	0.18	6.029	6.561	63.90
	Aggressive	0.24	0.16	5.600	6.078	61.74

F Calibration, Runtime Adaptation, and Memory

F.1 Offline calibration and online selection

Table 16 quantifies calibration cost on 50 prompts of length 128K. RBS uses three configurations, taking 0.443 wall-clock hours and 0.885 GPU-hours with TP=2. The reported maximum error from its 5% target is 0.0078 percentage point. The median/90th-percentile choices for the radius normalization remain fixed throughout calibration.

Calibration is an offline operating-point choice. At inference time, the thresholds are fixed and the prompt/layer/head-specific radius distribution directly determines β_b. Table 17 reports total selector time at the 128K system operating point, including statistics, normalization, and selection. The RBS selector takes 694.81 ms, including its two-branch scoring and mask construction; this work is part of the measured execution path. Relative to centroid-only selection, the extra work implements the adaptive rescue policy evaluated in the quality experiments.

Table 16: Offline calibration on 50 prompts of length 128K using the A100 main-model setup (TP=2). GPU-hours are the reported aggregate device time; small rounding differences from twice the wall time are retained. Maximum density error is in percentage points (pp). RBS fixes the radius quantiles at 0.5/0.9 while calibrating its selection thresholds.

Method	Configurations	Wall hours	GPU-hours	Target density	Max. error (pp)
RBS-Attention	3	0.443	0.885	5%	0.0078
FlashPrefill	3	0.418	0.837	5%	0.1625
MInference	3	0.232	0.464	2/5/10%	0.0913

Table 17: Total per-request selector time at the 128K A100 system operating point. Times include block statistics, normalization, score computation, threshold/Top-K decisions, and index construction; RBS additionally unions its branch masks. There is no per-request threshold search.

Selector	Total selector time (ms)
FlashPrefill	349.89
Full-L2	279.21
RBS-Attention	694.81
Quest-style	1,195.88

F.2 Persistent statistics, workspace, and model peak

Table 18 reports storage for block statistics and the temporary selector workspace. With the reported accounting dimensions of 1,024 blocks, head dimension 128, and four KV heads, centroid-only statistics occupy 1.000 MiB per layer. Adding a FP32 radius and a FP32 adaptive coefficient increases RBS's statistics to 1.031 MiB per layer, whereas the two BF16 vectors used by Quest-style selection require 2.000 MiB. This is a comparison of block-statistic representations, not of total allocator usage.

The temporary workspace is substantially larger than these statistics. RBS uses 1,155.601 MiB in the reported implementation, compared with 515.499 MiB for centroid-only and 428.231 MiB for Quest-style selection. Thus, the small persistent radius overhead does not imply an equally small total selector footprint. Table 19 measures the separate full-model quantity: at 128K, the RBS run peaks at 60.486 GiB allocated, compared with 61.552 GiB for Dense. These measurements include the complete serving configuration and depend on allocator and workspace lifetimes; they are not a claim that sparse prefill compresses or evicts the KV cache.

Table 18: Selector storage under the reported accounting dimensions: 1,024 blocks, head dimension 128, four KV heads, and 48 active layers. Statistics are reported per layer; temporary workspace is reported separately and should not be multiplied by the layer count. Vectors use BF16 and the radius/β scalars use FP32.

Selector	Statistics per block	MiB/layer	Workspace (MiB)
Centroid-only	128-vector	1.000	515.499
Full-L2	128-vector + radius	1.016	515.324
RBS-Attention	128-vector + radius + β	1.031	1,155.601
Quest-style	Two 128-vectors	2.000	428.231

Table 19: Full-model peak memory on A100 GPUs with Qwen3-30B-A3B-Instruct-2507-FP8 (TP=2). Each cell reports **max-rank allocated / reserved** memory in GiB. The largest tested length in this supplementary memory experiment is 128K; it is not a measured hardware capacity limit.

Method	32K	64K	128K
Dense	55.710 / 56.650	57.328 / 59.279	61.552 / 65.510
Centroid-only	55.248 / 57.010	56.865 / 58.918	61.090 / 63.441
RBS-Attention	54.644 / 56.357	56.262 / 58.266	60.486 / 64.016
Quest-style	59.894 / 61.541	61.512 / 63.449	61.330 / 63.496

G Geometric Bounds and the Dual-Branch Selector

This appendix separates the geometric motivation for the radius from the selection policy that uses it. Throughout, K_b is a finite set of key vectors, c_b is its mean, $r_b = \max_{k \in K_b} \|k - c_b\|_2$, and q is a query. We omit the common attention scale $1/\sqrt{d}$ because it does not affect the inequalities.

G.1 THE FULL L2 BOUND AND THE ATTENUATED SCORE

For every $k \in K_b$,

$$q^\top k = q^\top c_b + q^\top (k - c_b) \tag{14}$$
$$\leq q^\top c_b + \|q\|_2 \|k - c_b\|_2 \tag{15}$$
$$\leq q^\top c_b + \|q\|_2 r_b. \tag{16}$$

Taking the maximum over k proves Eq. (2). Geometrically, this is the maximum linear score over a ball containing all keys in the block.

The actual rescue score replaces r_b by $\beta_b r_b$. When $\beta_b < 1$, the smaller ball need not contain the keys. For example, take $K_b = \{-u, u\}$ with $\|u\|_2 = 1$ and $q = u$. Then $c_b = 0$, $r_b = 1$, and the largest token logit is 1, whereas the rescue logit is β_b. This provides a direct counterexample to interpreting the attenuated score as a universal upper bound. The radius supplies a geometric risk signal; quantile-based attenuation controls how strongly that signal influences selection.

G.2 COMPARISON WITH A COORDINATE-WISE BOX

For coordinate j, let $a_{b,j} = \min_{k \in K_b} k_j$ and $u_{b,j} = \max_{k \in K_b} k_j$. The page score used by Quest (Tang et al., 2024) is

$$U_b^{\text{box}}(q) = \sum_{j=1}^{d} \max(q_j a_{b,j}, q_j u_{b,j}). \tag{17}$$

Each coordinate of every key lies in its corresponding interval, so $\max_{k \in K_b} q^\top k \leq U_b^{\text{box}}(q)$. Equivalently, if $h_b = (u_b - a_b)/2$ and $v_b = (u_b + a_b)/2$, then $U_b^{\text{box}}(q) = q^\top v_b + |q|^\top h_b$. The box center v_b need not equal the key mean c_b.

The two bounds have no uniform ordering. Two-dimensional examples make this explicit:

- Let $K = \{e_1, -e_1, e_2, -e_2\}$ and $q = (1, 1)^\top$. The centroid is zero and the radius is one, giving $U^{\text{L2}}(q) = \sqrt{2}$ and $U^{\text{box}}(q) = 2$. Here the L2 bound is tighter: the box contains the corner $(1, 1)$ even though no key lies there.
- Let $K = \{e_1, -e_1, \epsilon e_2, -\epsilon e_2\}$ for $0 < \epsilon < 1$, and let $q = e_2$. The radius is again one, giving $U^{\text{L2}}(q) = 1$ and $U^{\text{box}}(q) = \epsilon$. Here the box bound is tighter because it captures the narrow range in the second coordinate.

These examples illustrate the trade-off between a coordinate-wise envelope and an isotropic one. Under a common orthogonal transformation of keys and queries, the centroid dot product and L2 radius are unchanged. A coordinate-wise box can change under the same transformation. Rotation invariance is a geometric property, not a guarantee that the L2 selector will have better accuracy or a tighter bound on a given input.

G.3 WHY SEPARATE THRESHOLDS MATTER

An upper bound characterizes one block, not global selection quality: a loose bound can rank an irrelevant block above a relevant one. Neither bound-based ranking nor the attenuated score guarantees preservation of dense-attention mass; we evaluate the resulting masks empirically.

RBS separately selects candidates by centroid relevance and radius-supported rescue. Uplift can increase the rescue maximum and hence its relative threshold, removing other candidates even though their individual logits did not decrease. The independent base mask prevents this loss for centroid-selected blocks. Its union with rescue can therefore differ from both centroid-only and single full-L2 selection.

Thresholds control relative scores, not cardinality. For fixed scores, decreasing a threshold cannot shrink that branch's mask. Across prompts and layers, score distributions and branch overlap vary, so fixed thresholds need not yield equal density. Nominal-target and calibrated actual-density comparisons therefore answer different questions.