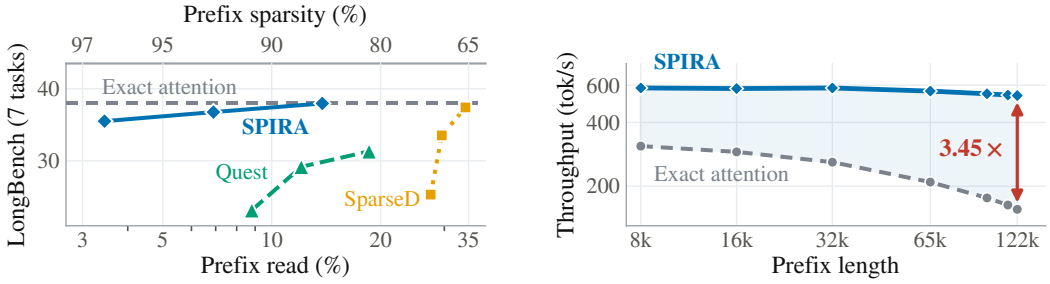
SPIRA: SPARSE PRE-ROPE INFORMED ROUTING ATTENTION FOR DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Block diffusion language models generate a block of tokens at a time by denoising masked positions in parallel. Sparse prefix attention reduces the number of prefix keys attended to during decoding, but selecting those keys is itself a full-prefix operation when each masked query scores the prefix independently. We find that the masked queries of a block share a common direction before RoPE. This structure supports shared candidate generation across the block, while final selection still depends on query-specific scores from the original position-dependent queries. SPIRA exploits this separation: one routing query per query head, constructed from the block's pre-RoPE query mean, scans the prefix once per block to retrieve a candidate set, and the original queries rescore only those candidates to select the final keys. The selected keys are then reused throughout the block, so later denoising steps no longer read the full prefix. Across three block diffusion language models, SPIRA outperforms two sparse baselines on LongBench with 256 retained keys, exceeding the stronger baseline by up to 10.2 points. With 1024 keys, it comes within 0.06 points of exact attention on SDAR-4B-Chat-b64. On the same model, it achieves a 3.45× speedup in per-block decode latency at a 122k-token prefix.

1 INTRODUCTION



(a) quality versus prefix read

(b) decode throughput

Figure 1: Quality–efficiency trade-off and decode throughput on SDAR-4B-Chat-b64. **(a)** LongBench score versus prefix read fraction, excluding selection-only scoring, at three retained-key budgets. **(b)** Decode throughput versus prefix length, excluding prefill.

Block diffusion language models generate tokens in blocks by denoising multiple masked positions in parallel. At each denoising step, every remaining masked position attends to the completed prefix. As the prefix grows, this attention becomes a major decoding cost, motivating sparse attention. Many existing methods for block diffusion focus on reducing work repeated across denoising steps: they reuse sparsity patterns across steps (Wang et al., 2026) or recompute only queries whose hidden states continue to change (Xi et al., 2026). A separate cost remains when the prefix keys to attend to are selected independently for each masked query: each query still scores the full prefix at every layer. This selection cost grows with both the prefix length and the number of masked queries, even though the subsequent attention is sparse. Sharing the full-prefix search across the block would remove this

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.

position-wise repetition, but final selection still requires query-specific information, so the problem is to share the search without discarding that information.

We therefore ask what the masked queries share and where they differ. RoPE (Su et al., 2024) applies a position-dependent rotation to queries and keys. Before RoPE, the masked queries of a block cluster around a common direction, whereas neighbouring unmasked queries, which do not share the mask token, are less aligned. After RoPE, the original queries retain the query-specific differences on which final selection depends. The shared pre-RoPE structure therefore supports shared candidate generation, while the original post-RoPE queries determine the final selection.

Based on this division of roles, we propose SPIRA (Sparse Pre-RoPE Informed Routing Attention). For each query head, a routing query formed from the block's pre-RoPE query mean scans the prefix once per block and retrieves a candidate set; the block's original queries then rescore only these candidates, and their aggregated scores select the final prefix keys. Because the selected keys are reused throughout the block, later denoising steps attend to a fixed subset of the prefix, and their prefix-attention cost no longer grows with the prefix length.

Our contributions are as follows.

- **Shared pre-RoPE query direction.** We characterise the shared direction among a block's masked queries and show that it suffices for candidate generation but not for final selection.
- **Shared routing with candidate rescoring.** We introduce SPIRA, which shares the full-prefix search across the block and uses the original queries to rescore only the retrieved candidates before reusing the selected keys.
- **Improved quality and decoding efficiency.** SPIRA outperforms QUEST (Tang et al., 2024) and SPARSED (Wang et al., 2026) on all three models at the tightest LongBench budget, exceeding the stronger baseline by up to 10.2 points. At the largest budget, it comes within 0.06 points of exact attention on SDAR-4B-Chat-b64 and achieves a $3.45\times$ speedup in per-block decode latency at a 122k-token prefix (Figure 1).

2 BACKGROUND AND RELATED WORK

Block diffusion and prefix attention. Block diffusion models denoise a block of masked positions in parallel, conditioned on a completed prefix (Nie et al., 2025; Arriola et al., 2025). At each denoising step, the block's queries attend both to the cached prefix and to positions inside the block. Let $(\mathbf{K}^P, \mathbf{V}^P)$ denote the prefix keys and values, and $(\mathbf{K}^B, \mathbf{V}^B)$ those of the current block. Collecting the post-RoPE block queries into $\mathbf{Q}^{\text{post}}$, one query head of dimension d computes

$$\mathbf{O} = \text{softmax}\Big(\mathbf{Q}^{\text{post}}[\mathbf{K}^P; \mathbf{K}^B]^\top / \sqrt{d}\Big)\,[\mathbf{V}^P; \mathbf{V}^B], \tag{1}$$

with the model's attention mask omitted. We call this unsparsified computation *exact attention*.

SPIRA sparsifies only prefix attention. It restricts attention over the completed prefix to a selected subset of positions, while attention within the current block remains dense. We denote the size of this subset, the retained-key budget, by K. Under grouped-query attention, several query heads share one key–value head. SPIRA therefore selects one prefix subset per key–value head at each layer.

RoPE (Su et al., 2024) applies a position-dependent rotation to queries and keys, so a direction shared by a block's queries before RoPE becomes position dependent after it. SPIRA forms its routing representation from pre-RoPE queries, while candidate rescoring uses the model's original post-RoPE queries. We use each model's own rotary frequencies and rotating channels.

Reuse across denoising steps. Several methods for block diffusion reduce the work repeated at later denoising steps. Some reuse intermediate state through caching (Liu et al., 2026; Wu et al., 2026b; Ma et al., 2025; Song et al., 2026), while others reuse sparse attention structure across steps. SPARSED (Wang et al., 2026) computes head-specific sparse patterns after an initial fraction of dense steps and reuses them thereafter; we use it as a baseline. MAGE (Kwon et al., 2026b) derives a block-level sparse set from exact attention at the first all-`[MASK]` step and reuses it across later steps. LOSA (Xi et al., 2026) reuses prefix-attention results for queries whose hidden states have settled and recomputes only the rest. FOCUS-DLLM (Long et al., 2026) narrows the active queries

with decoder confidence. FLASHBLOCK (Chen et al., 2026) caches block-external attention outputs. SPIRA instead targets the position-wise repetition in constructing the block's sparse prefix selection.

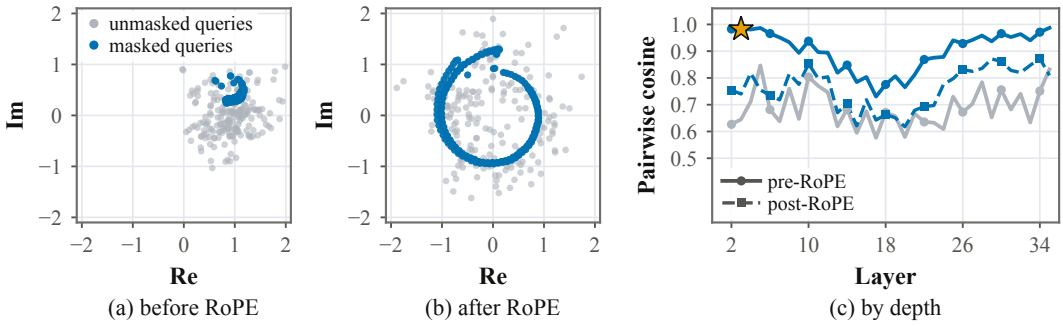
Reducing prefix-selection cost. HERALD (Kwon et al., 2026a) reduces full-prefix retrieval to a representative `[MASK]` query at the centre of the block. BA-ATT (Zhang et al., 2026) instead searches a block-wise downsampled attention space, using norm sorting and a covariance correction to limit the approximation error. Neither method rescores candidates with the block's original position-wise queries before final selection. SPIRA adds this stage: shared routing suffices for candidate generation but not for final selection (Section 3). HERALD additionally targets serving efficiency through CPU–GPU overlap; our focus here is the selection rule.

Prefix selection in autoregressive decoding. Early sparse-attention methods for long contexts often used fixed patterns (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020). More recent work also selects the attended positions at run time. In autoregressive decoding, where queries arrive one at a time, one line of work evicts or compresses the key–value cache (Xiao et al., 2024b; Zhang et al., 2023; Li et al., 2024), while another keeps the cache and selects which cache blocks each query attends to (Xiao et al., 2024a; Jiang et al., 2024; Yuan et al., 2025; Lu et al., 2025). QUEST (Tang et al., 2024) belongs to the second line and selects critical KV pages with each decoding query; we use it as a baseline. Concentration among pre-RoPE queries has also been reported in that setting: TRIATTENTION (Mao et al., 2026) derives KV importance from pre-RoPE query and key centres that are stable across positions. SPIRA relies on a different structure: alignment among the masked queries exposed simultaneously within a block, which is weaker among neighbouring unmasked queries (Section 3).

3 OBSERVATIONS

SPIRA relies on two observations about a block's masked queries: they share a direction before RoPE, and this shared structure is sufficient for candidate generation but not for final selection. We test each in turn, then characterize the query-specific information omitted by shared routing.

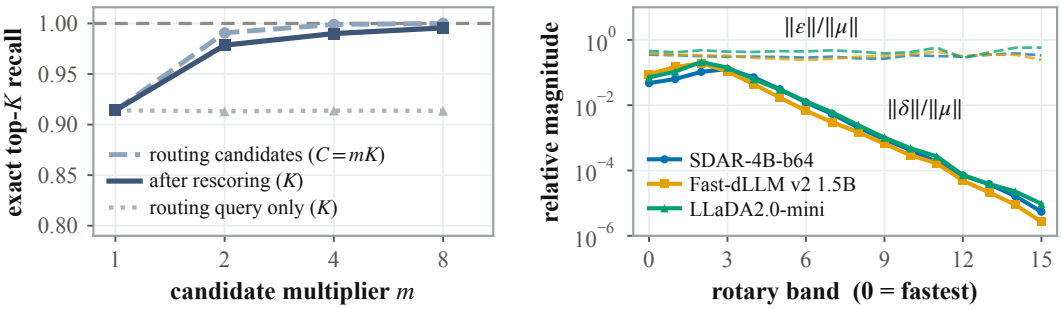
Unless stated otherwise, each probe follows the decoder's actual trajectory and reports medians over 150 documents. For the masked-query curves, we include only positions that remain masked. Additional probe details, including the block-boundary setup used for Fast-dLLM v2, are given in Appendix A.



(a) before RoPE (b) after RoPE (c) by depth

Figure 2: Masked block queries align before RoPE, and RoPE reduces that alignment. SDAR-4B-Chat-b64. **(a, b)** A representative rotary subspace before and after RoPE. **(c)** Mean pairwise cosine across depth, colours as in (a); the star marks the layer shown in (a) and (b).

Masked queries align before RoPE. We measure agreement as the mean pairwise cosine within each query head, averaged over query heads for each document–layer pair. We compare the masked queries before RoPE, the same queries after RoPE, and an equal number of unmasked queries immediately preceding the block. They serve as a control: if the alignment were generic to nearby pre-RoPE queries, they should exhibit comparable concentration. Figures 2a and 2b show one representative document–layer–head cell in a rotary plane, with 3 blocks overlaid. The mean resultant



(a) recall versus candidate multiplier

(b) residual and phase term by band

Figure 3: **(a)** Shared routing retrieves the candidates; rescoring with the original queries raises final top-K recall, and at $m{=}1$ the candidate set is already the selection. SDAR-4B-Chat-b64, $K{=}256$. **(b)** Residual magnitude and phase-induced contribution across rotary bands (sixteen groups of adjacent frequency pairs), 30 documents per model.

length R of the masked queries falls from 0.997 before RoPE to 0.06 after it: the position-dependent rotations disperse most of the pre-RoPE concentration. Before RoPE, the neighbouring unmasked queries reach only 0.87 in the same plane, and the prefix keys (not drawn) 0.88. Figure 2c extends the comparison across depth using mean pairwise cosine, where pre-RoPE agreement is highest at every probed SDAR layer. The same pattern holds for Fast-dLLM v2 and LLaDA2: across the three models, pre-RoPE agreement exceeds post-RoPE agreement in all $11{,}700$ document–layer cells and exceeds the unmasked-query control in $11{,}697$ of them (Appendix A). The alignment is therefore characteristic of the simultaneous masked queries in a block rather than generic concentration in the model's pre-RoPE representations.

Because SPIRA performs selection at every layer, we next ask whether the alignment persists at every depth. Pre-RoPE agreement varies with depth but stays above post-RoPE agreement throughout: across the three models, its minima occur 0.47–0.71 of the way through the network, and it recovers by the last probed layer. At the depth where the pre- and post-RoPE curves come closest, they remain separated by 0.066, 0.034 and 0.063 on the three models, so a shared direction remains identifiable at every depth, not only where agreement is highest (Table 3).

Shared structure is sufficient for candidate generation, but not for final selection. Recall is measured against the K prefix keys carrying the largest aggregated prefix-attention mass (Appendix A), and a multiplier m sets the candidate count C to mK, capped at the prefix length. Figure 3a tests the routing representation first for final selection and then for candidate generation, over $1{,}350$ (document, layer) cells on SDAR (150 documents $\times$ 9 layers). Used on its own to pick the final K, it reaches 0.913 recall. Used instead to return $C = 2K$ candidates, it reaches 0.991 candidate recall: the shared direction retrieves most of the top-K keys, but is not sufficient for final selection. Rescoring those candidates with the original position-dependent queries and keeping the final K recovers recall to 0.979: the shared direction provides a high-recall candidate set, while the original-query scores determine the final selection.

We next ask whether that recovery depends on the positional phase carried by the original queries. We restrict rescoring to a rank-eight basis of the block's query residuals (Appendix A) and vary only whether that basis is taken before or after RoPE. Using the pre-RoPE basis, which carries no RoPE positional phase, instead of the post-RoPE basis reduces top-K recall by 0.0376–0.1211 across the three models and degrades recall in 445, 449 and 450 of 450 cells (50 documents $\times$ 9 layers per model). The recovery therefore depends partly on the original queries' positional phases.

Query-specific variation enters the post-RoPE mean through the rotary phases. To locate the query-specific information that shared routing omits, we decompose each pre-RoPE query as $\mathbf{q}_i = \boldsymbol{\mu} + \boldsymbol{\epsilon}_i$, where $\boldsymbol{\mu}$ is the block mean and $\sum_i \boldsymbol{\epsilon}_i = \mathbf{0}$. For a block of B query positions p_i, let $\mathbf{R}_{p_i}$ denote the RoPE rotation at p_i and $\bar{\mathbf{R}} = \frac{1}{B}\sum_i \mathbf{R}_{p_i}$ their mean. Averaging the post-RoPE queries

gives

$$\frac{1}{B}\sum_i \mathbf{R}_{p_i}\mathbf{q}_i \;=\; \bar{\mathbf{R}}\boldsymbol{\mu} \;+\; \frac{1}{B}\sum_i \left(\mathbf{R}_{p_i} - \bar{\mathbf{R}}\right)\boldsymbol{\epsilon}_i. \tag{2}$$

The first term, $\bar{\mathbf{R}}\boldsymbol{\mu}$, is the routing representation SPIRA uses. The second term captures the contribution of query-specific residuals after each query receives a different rotary phase. We denote it by $\boldsymbol{\delta}$; it is absent from the routing representation by construction. At each rotary frequency, a query contributes in proportion to how far its rotation departs from the mean rotation, so the contribution vanishes when the queries share a rotation and stays small where the block spans little rotary phase.

The rotation-dependent factor is determined by the block positions and rotary frequencies alone, but the resulting contribution also depends on how the residual is distributed across frequencies. If $\boldsymbol{\epsilon}$ were concentrated in the slow (low-frequency) bands, it could still contribute there despite the small rotation-dependent factor. Figure 3b measures that distribution. The residual magnitude itself changes little across the spectrum: on all three models $\|\boldsymbol{\epsilon}\|/\|\boldsymbol{\mu}\|$ varies by a factor of 1.5–2.1. Over the same spectrum $\|\boldsymbol{\delta}\|/\|\boldsymbol{\mu}\|$ varies by a factor of 22,017 to 71,938. The frequency dependence therefore comes from the interaction between the residuals and the position-dependent rotary phases, rather than from larger residuals in particular bands. Over the slow half of the spectrum the shared term preserves 0.9999 of the block mean and $\|\boldsymbol{\delta}\|/\|\boldsymbol{\mu}\|$ stays below 2.5×10^{-3} on every model, against 0.13–0.21 over the fast half.

The decomposition also relates a post-RoPE block mean to the shared routing term: it equals $\bar{\mathbf{R}}\boldsymbol{\mu} + \boldsymbol{\delta}$, with $\boldsymbol{\delta}$ small in the slow bands and much larger in the fast bands. Shared routing uses the component common to the block's queries, and stage 2 reintroduces query-specific information by rescoring with the original queries. SPIRA builds this shared term from the pre-RoPE mean, for which $\bar{\mathbf{R}}$ has a closed form over consecutive positions, shared across layers (Appendix B).

4 Method

SPIRA implements the division of roles established in Section 3 as two stages that run once per block (Figure 4). Stage 1 uses one routing representation per query head to score the full prefix and retain $C \geq K$ candidates, since routing alone is insufficient for final selection. Stage 2 rescores only these candidates with the original post-RoPE queries and keeps the K keys that the remaining denoising steps of the block reuse.

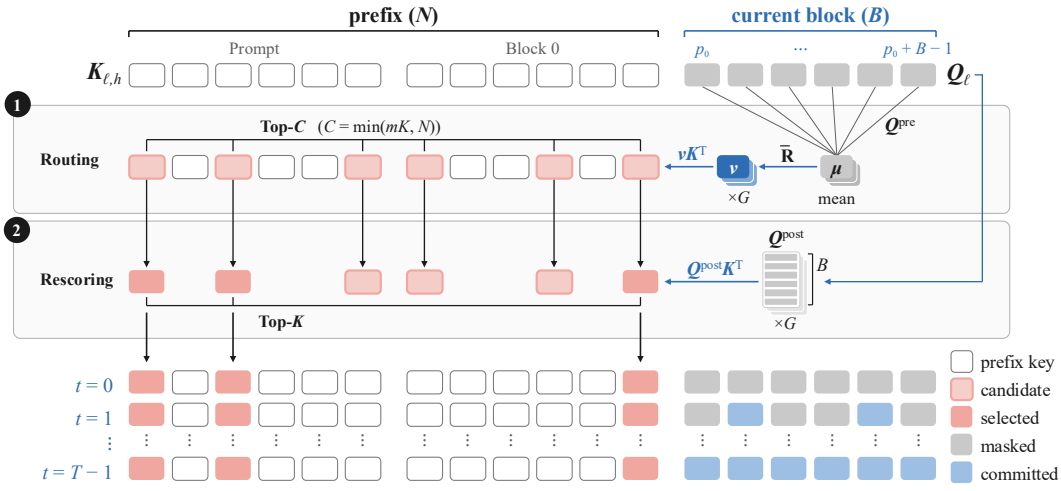


Figure 4: One routing pass proposes C prefix candidates at the block's first denoising step; the block's original queries rescore them to select the K positions that every later step reuses.

At layer ℓ, let b denote a block of B consecutive positions beginning at p_0, N the prefix length, d the head dimension, and T the number of denoising steps a block takes. Under grouped-query attention, let $\text{group}(h)$ denote the G query heads that share key–value head h. Selection is performed

independently for each layer and KV head, and the resulting K prefix positions are shared by the GB block queries attached to that head. We retain G in the definitions below and suppress it only when reporting asymptotic costs.

Stage 1: candidate generation. Stage 1 retrieves $C = \min(mK, N)$ prefix candidates, where $m \geq 1$ is the candidate multiplier, so C never exceeds the prefix length (Appendix C). For each query head g we average the pre-RoPE queries over the block's positions,

$$\boldsymbol{\mu}_{\ell,g} = \frac{1}{B} \sum_{q \in b} \mathbf{Q}_\ell^{\mathrm{pre}}[g, q, :], \qquad \mathbf{v}_{\ell,g} = \bar{\mathbf{R}}\, \boldsymbol{\mu}_{\ell,g}, \qquad \bar{\mathbf{R}}_f = c_f\, \mathbf{R}_{\bar{p},f}, \quad c_f = \frac{\sin(B\theta_f/2)}{B \sin(\theta_f/2)}, \tag{3}$$

where $\bar{p} = p_0 + (B-1)/2$ and θ_f is the f-th rotary frequency. Routing is computed at the block's first denoising step, before any of its masked positions are committed (Appendix A). For consecutive positions the identity for $\bar{\mathbf{R}}$ is exact: the rotary phases at each frequency form an equally spaced sequence whose mean is a rotation to the block centre scaled by c_f. At low frequencies $|c_f|$ stays close to one, while frequencies whose phase varies substantially across the block are suppressed; Appendix B gives the derivation.

Each $\mathbf{v}_{\ell,g}$ scores the cached post-RoPE prefix keys $\mathbf{K}_{\ell,h}$ once, producing $G \times N$ query–key scores per KV head in place of the $G \times B \times N$ a position-wise rule would form. We normalise within a query head before aggregating across the heads that share h:

$$\mathcal{C}_{b,h}^{\ell} = \text{Top-}C\Big(\sum_{g \in \mathrm{group}(h)} \mathrm{softmax}_N\big(\mathbf{v}_{\ell,g} \mathbf{K}_{\ell,h}^{\top} / \sqrt{d}\big) \Big). \tag{4}$$

Normalising before the sum gives each query head unit total mass, so no head dominates the aggregate through its logit scale. Aggregating before Top-C also keeps the candidate budget fixed: selecting a separate set per head and taking their union would leave the gathered set growing with head disagreement, changing both the gather cost and the effective candidate budget, while Equation (4) returns exactly C positions.

Stage 2: candidate rescoring and final selection. The block's original post-RoPE queries, each carrying its own position, rescore the candidates, producing $G \times B \times C$ query–key scores:

$$\mathcal{S}_{b,h}^{\ell} = \text{Top-}K\Big(\sum_{g \in \mathrm{group}(h)} \sum_{q \in b} \mathrm{softmax}_C\big(\mathbf{Q}_\ell^{\mathrm{post}}[g, q, :]\, \mathbf{K}_{\ell,h}[\mathcal{C}_{b,h}^{\ell}]^{\top} / \sqrt{d}\big) \Big). \tag{5}$$

Candidate rescoring uses the original post-RoPE queries and cached keys directly, without an additional projection. Their query–key dot products are evaluated exactly on the C candidates, although the softmax is normalised over the candidate set rather than the full prefix.

The candidate budget must exceed the final budget for stage 2 to change the result: at m=1, C=K and stage 1 alone fixes the selection (the m=1 point of Figure 3a). For $m > 1$, stage 1 proposes more than K candidates, leaving stage 2 to determine which K are kept.

One selection per block. Both stages run only at the first denoising step of a block; Appendix C gives the complete procedure. The index set $\mathcal{S}$ and the prefix keys and values gathered from it are cached until the block completes and reset for the next block, so every later step attends to the same K prefix positions. Selection never removes keys belonging to the current block; intra-block attention keeps the model's own mask and stays dense.

Cost. For one layer and KV head, keeping the grouped-query dimension explicit, exact prefix attention over a block costs $\mathcal{O}(TGBNd)$. SPIRA replaces it with

$$\underbrace{\mathcal{O}(GNd)}_{\text{routing}} + \underbrace{\mathcal{O}(GBCd)}_{\text{rescoring}} + \underbrace{\mathcal{O}(TGBKd)}_{\text{sparse prefix attention}}. \tag{6}$$

Exact attention and SPIRA incur the same dense intra-block term $\mathcal{O}(TGB^2d)$, omitted from Equation (6) because it is common to both. Suppressing G leaves $\mathcal{O}(Nd + BCd + TBKd)$. When $N \geq mK$ we have $C = mK$, and in the long-prefix regime of interest $K, C \ll N$. After the selection step a denoising step costs $\mathcal{O}(GBKd + GB^2d)$ and contains no term proportional to N. The selected indices and the gathered tensor shapes are also fixed for the rest of the block, so every later step repeats the same execution shape. Section 5 evaluates quality at fixed retained-key budgets and measures latency as the prefix length grows.

5 EXPERIMENTAL RESULTS

Setup. We evaluate SDAR-4B-Chat-b64 (Cheng et al., 2026) (B=64), Fast-dLLM v2 1.5B (Wu et al., 2026a) (B=32) and LLaDA2.0-mini (Bie et al., 2025) (B=32) on seven LongBench tasks (Bai et al., 2024) and on all thirteen RULER tasks (Hsieh et al., 2024) at 4k, 8k, 16k and 32k. We sparsify only prefix attention and set m=2 on LongBench and m=4 on RULER. The quality experiments use each model's released confidence threshold for committing tokens; the timing experiments use the fixed denoising schedules of Section 5.2. The prefix-length and latency-breakdown experiments use SDAR at batch 1 on an idle A100-40GB; batch scaling is reported separately. Appendix D gives the baseline configurations and the full per-task results.

5.1 QUALITY

On LongBench, SPIRA comes within 0.06 points of exact attention on SDAR and 0.12 on LLaDA2 at the largest budget, and stays within 0.83 on Fast-dLLM v2. At the tightest budget it outperforms both sparse baselines on all three models, exceeding the stronger baseline by up to 10.2 points on the seven-task mean, and at the largest budget it still leads both on SDAR and Fast-dLLM v2. Table 1 reports the per-task results.

Table 1: LongBench, seven tasks, three models; QA F1 except QMSum (ROUGE-L). Equal K is not equal computation: the baselines retain additional dense computation (Appendix D). Bold, best sparse result at that budget; blue, SPIRA.

			K=256			K=512			K=1024		
Model	Task	Exact	SPIRA	Quest	SparseD	SPIRA	Quest	SparseD	SPIRA	Quest	SparseD
SDAR 4B	Hotpot	48.66	**45.18**	30.57	33.43	**49.40**	40.89	42.54	**51.09**	43.00	48.66
	QMS	15.17	**17.06**	7.56	11.07	**16.50**	9.15	14.40	15.78	10.69	**16.19**
	2Wiki	34.66	**31.64**	20.65	27.60	**32.32**	26.69	32.21	**33.97**	29.20	33.03
	Trivia	85.04	**80.71**	53.74	53.03	**81.03**	66.09	76.83	84.63	69.67	**86.51**
	Narr.	24.52	**22.41**	9.91	6.23	**22.79**	13.99	14.57	**24.28**	16.01	21.66
	Qasp.	22.98	**20.45**	15.51	20.37	**22.77**	19.48	22.12	22.88	18.90	**22.92**
	MFQA	35.17	**31.09**	24.17	25.51	**32.54**	27.93	32.05	**33.13**	31.79	32.92
	Avg.	38.03	**35.50**	23.16	25.32	**36.76**	29.18	33.53	**37.97**	31.32	37.41
Fast-dLLM v2 1.5B	Hotpot	20.07	14.27	10.00	**14.52**	**16.14**	10.71	15.95	**17.72**	14.06	17.14
	QMS	22.46	**21.52**	19.77	20.22	**21.85**	20.81	21.20	**22.47**	21.68	21.77
	2Wiki	14.68	**13.04**	11.51	11.87	**14.07**	11.17	13.06	**14.73**	12.26	14.15
	Trivia	53.81	**47.68**	32.23	46.79	**53.25**	42.40	49.38	**55.07**	49.19	50.62
	Narr.	12.01	**11.26**	9.64	9.49	**12.17**	10.14	10.41	**12.07**	9.79	11.50
	Qasp.	21.86	**16.82**	13.57	13.14	**18.74**	15.90	17.52	20.13	20.33	**20.79**
	MFQA	38.73	31.03	26.13	**32.71**	33.37	29.61	**36.39**	35.57	33.80	**39.00**
	Avg.	26.23	**22.23**	17.55	21.25	**24.23**	20.11	23.42	**25.40**	23.02	25.00
LLaDA2.0 mini	Hotpot	58.17	**56.52**	50.14	55.14	57.50	56.59	**58.39**	57.24	**60.27**	59.03
	QMS	22.95	**23.07**	17.02	20.75	**22.80**	19.65	22.57	22.36	22.12	**22.50**
	2Wiki	51.72	**50.85**	45.36	42.30	50.08	49.41	**50.12**	49.44	47.12	**51.51**
	Trivia	87.38	**88.51**	81.95	84.21	87.75	**88.82**	87.42	87.75	**88.52**	88.14
	Narr.	25.17	**24.47**	20.09	23.11	**26.37**	22.00	23.68	**26.67**	23.82	24.90
	Qasp.	46.46	**44.27**	41.53	40.60	45.41	42.52	**45.87**	**47.05**	46.84	45.64
	MFQA	52.78	**52.31**	50.40	48.89	52.36	**53.38**	52.24	53.25	**54.35**	52.70
	Avg.	49.23	**48.57**	43.79	45.00	**48.90**	47.48	48.61	49.11	49.01	**49.20**

Because the methods do different selection work beyond the retained-key budget K, we report the prefix read fraction separately from measured latency. It is the fraction of prefix positions read by the attention operator, averaged over layers and denoising steps; a dense layer or a dense step counts as reading the whole prefix, and selection-only scoring is excluded.

At K=256, the tightest budget in Figure 1a, SPIRA reads 3.45% of the prefix on SDAR against SPARSED's 27.51% and scores 10.2 points higher on LongBench. At matched quality rather than

matched budget, SPARSED reads $9.2\times$ as much of the prefix as SPIRA, interpolating in $\log K$ between SparseD's measured points.

Retrieval. RULER's retrieval category complements LongBench by testing retrieval of information inserted into a long context. Table 6 reports its eight needle-in-a-haystack tasks. At the tightest budget $K{=}256$, SPIRA at $m{=}4$ leads the strongest sparse baseline at both 4k and 8k, by $+18.7$ to $+19.8$ points on SDAR and $+10.3$ to $+15.0$ on LLaDA2. On Fast-dLLM v2 the strongest baseline stays ahead.

Table 5 reports the thirteen-task average up to 32k. At 32k and a common $K{=}1024$, SPIRA reads 3.2% of the prefix on all three models, compared with 8.6 to 12.9% for QUEST and 18.0 to 38.8% for SPARSED, so the baselines read 2.7–$12.1\times$ as much of the prefix as SPIRA. SPIRA scores above both baselines on SDAR and LLaDA2. Figure 6 summarises this trend across all four context lengths.

5.2 SPEED

Both stages run once at the start of a block. The K prefix positions returned by stage 2, and the key and value tensors gathered from them, are reused by every later denoising step, so each of those steps reads a fixed-size prefix set, and the per-step latency remains nearly flat as the prefix length grows: 11.10 ms at 32k and 11.17 ms at 65k, against exact attention's 18.0 ms rising to 25.7 ms over the same range. This yields a $3.45\times$ speedup in per-block decode latency at a 122k prefix (Figure 5a).

We measure per-block decode latency as the incremental decode time per output block, with prefill excluded, since prefill is identical across methods. Exact attention uses FLASHINFER (Ye et al., 2025), the faster of the two dense backends we measured, in the prefix-length experiments and FLASHATTENTION in the batch experiments. Latency is estimated from the slope between runs generating two and six blocks. Every method runs its decoding loop as a captured CUDA graph, and the denoising schedule is held equal across methods so that a content-dependent step count does not enter the ratio. We report a short schedule of 8 steps per block and a long one of 32. Appendix E gives the protocol, the two backends and the end-to-end path in full.

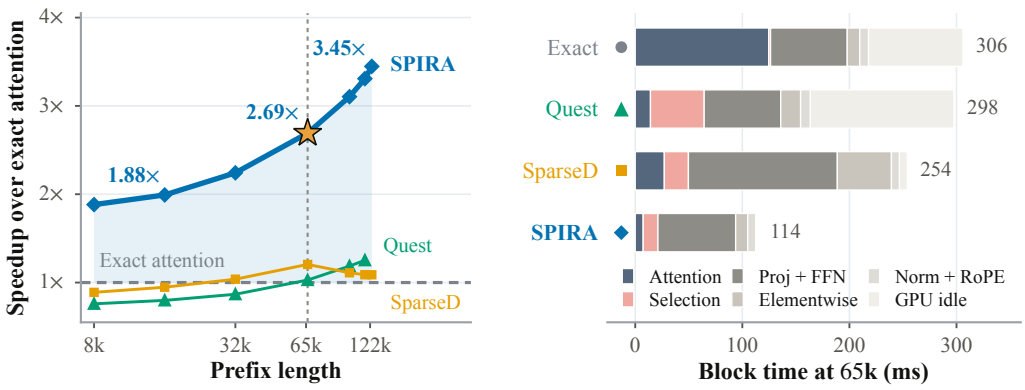
Prefix-length scaling. SPIRA's per-step cost stays nearly flat as the prefix grows, while the selection cost is paid once per block; Table 2 gives the split. The reused indices also fix the shape of the gathered key and value tensors for the remaining denoising steps, which lets the sparse path reuse its CUDA graph across the block (Figure 7).

Table 2: Prefix-length and batch scaling on SDAR-4B-Chat-b64 ($K{=}1024$, $m{=}2$; prefill excluded); blue, SPIRA. *Once a block*: fixed cost paid once per block. Exact attention uses FLASHINFER for the prefix-length half and FLASHATTENTION for the batch half. Batch 16 exceeds GPU memory; end-to-end measurements agree within 1.3% (Appendix F).

	once a block (ms)		per step (ms)			decode throughput (tok/s) at an 8k prefix				
Prefix	Exact	SPIRA	Exact	SPIRA	Batch	Exact	QUEST	SPARSED	SPIRA	× exact
8k	107.3	**21.4**	12.4	**11.06**	1	299	235	278	**581**	1.94
16k	105.0	**21.5**	14.4	**11.14**	2	502	332	417	**813**	1.62
32k	102.3	**21.1**	18.0	**11.10**	4	747	433	586	**1,031**	1.38
65k	100.3	**24.4**	25.7	**11.17**	8	805	507	714	**1,172**	1.46

Latency breakdown. At a 65k prefix, SPIRA takes 113.7 ms per block, against 253.9 ms for SPARSED, 297.6 ms for QUEST and 306.0 ms for exact attention (Figure 5b). QUEST cuts the attention component as well, but pays selection at every denoising step where SPIRA pays it once. It therefore issues roughly 9200 GPU operations per block, against 2000 for SPIRA. A substantial fraction of QUEST's block time is GPU-idle.

Throughput. SPIRA's decode throughput stays nearly flat from 8k to 65k while exact attention's falls by a factor of 1.53. In the complete-generation measurements, at 65k it delivers $2.83\times$ the decode throughput of exact attention on FLASHATTENTION. End-to-end throughput reaches $1.62\times$ at 8k and remains $1.33\times$ at 32k. The difference between the decode and end-to-end gains reflects



(a) speedup versus prefix length

(b) per-block latency breakdown

Figure 5: SDAR, $K{=}1024$, $m{=}2$, batch 1, 8 denoising steps per block. **(a)** Speedup over exact attention; the star marks the prefix shown in (b). **(b)** Kernel-level latency breakdown at that prefix, GPU idle included.

work unaffected by the selector: projections and feed-forward take 72.5 ms at 65k, nearly two thirds of SPIRA's block, and prefill takes a larger share as the context grows.

Batch. At batch 8, SPIRA reaches 1,172 decode tokens per second against exact attention's 805, a same-batch gain of $1.46\times$, compared with $1.94\times$ at batch 1; selection is performed independently for each sequence rather than amortised across the batch. Among the sparse methods tested, only SPIRA is faster than exact attention at every tested batch size (Table 2); Appendix F gives the full results.

5.3 Ablations and scope

At $m{=}1$, $C{=}K$, so stage 2 cannot change the selected set: at $K{=}1024$ this lowers the LongBench score by 0.95, 0.58 and 0.31 points on SDAR, Fast-dLLM v2 and LLaDA2, respectively (Table 8). Raising m from 2 to 4 improves the mean LongBench score by 0.18 points and RULER by 3.08 points with less than 0.6% change in block time, and outperforms $m{=}2$ in eight of the twelve checkpoint–budget cells, while raising K narrows the gap to exact attention at all four SDAR checkpoints (Figure 9), making the retained-key budget the more consistent parameter for improving quality.

Fast-dLLM v2 is the only model whose first block is not entirely masked. Without trimming its prompt, pre-RoPE agreement over 3,874 cells still exceeds both the post-RoPE and the unmasked control, and restricting the routing mean to the remaining masked positions changes candidate recall by 0.0003 (Appendix A): the shared direction survives a partially masked block, and the routing mean is insensitive to the inclusion of unmasked positions. Our selector-fidelity probes show that SPIRA's selected set captures attention mass within 0.0004 of the oracle at $K{=}1024$ on 2.7–6.2k prefixes (Appendix A).

6 Conclusion

The masked queries of a block share a common direction before RoPE, providing shared structure for candidate generation while final selection remains query-specific. SPIRA turns this division of roles into a block-level selector: one routing query per query head scans the full prefix once per block, the original queries rescore only the retrieved candidates, and the selected keys are reused throughout the block. At the tightest LongBench budget, SPIRA outperforms both sparse baselines on all three models by up to 10.2 points; at the largest budget, it comes within 0.06 points of exact attention on SDAR, where it also reaches a $3.45\times$ speedup in per-block decode latency at a 122k-token prefix.

AI USE STATEMENT

We used generative AI tools to assist with writing and editing the text of this paper, and with writing analysis and plotting scripts used to produce the figures and tables. We did not use generative AI tools to generate research ideas, to design the method, or to produce or interpret any experimental result; every number reported here comes from a measurement we ran. Retrieval and discovery of related work were carried out by the authors. We have reviewed all AI-assisted work and take full responsibility for the content of this paper.

ETHICS STATEMENT

This work studies inference-time attention sparsity for publicly released language models on the public LongBench and RULER benchmarks. It introduces no new data collection, no human subjects and no model training. Every model and dataset is used under the terms of its public release. SPIRA reduces the cost of long-context decoding; as an efficiency method it inherits, but does not amplify, whatever capabilities and failure modes the underlying models already have.

REPRODUCIBILITY STATEMENT

Section 5 states the models, benchmarks and hardware. Appendix D gives the baseline configurations and the full LongBench and RULER cells; Appendix F gives the latency and throughput results; Appendix G reports the budget and candidate-multiplier sweeps. The method is specified in full by Equations (3)–(5) and Algorithm 1, the mean-rotation identity is derived in Appendix B, and Appendix A gives the probe protocol behind Figure 2. Source code will be released.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2025.

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

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

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. LLaDA2.0: Scaling up diffusion language models to 100B. *arXiv preprint arXiv:2512.15745*, 2025.

Zhuokun Chen, Jianfei Cai, and Bohan Zhuang. FlashBlock: Attention caching for efficient long-context block diffusion. In *International Conference on Machine Learning (ICML)*, 2026.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*, 2024.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Omin Kwon, Doyeon Kim, Jongseok Park, Seung Yul Lee, Ion Stoica, and Jae W. Lee. HERALD: High-throughput block diffusion LLM serving via CPU–GPU cooperative KV cache retrieval. *arXiv preprint arXiv:2606.21633*, 2026a.

Omin Kwon, Yeonjae Kim, Doyeon Kim, Minseo Kim, Yeonhong Park, and Jae W. Lee. MAGE: All-[MASK] block already knows where to look in block diffusion LLM. *arXiv preprint arXiv:2602.14209*, 2026b.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. dLLM-Cache: Accelerating diffusion large language models with adaptive caching. In *International Conference on Machine Learning (ICML)*, 2026.

Lingkun Long, Yushi Huang, Shihao Bai, Ruihao Gong, Jun Zhang, Ao Zhou, and Jianlei Yang. Focus-dLLM: Accelerating long-context diffusion LLM inference via confidence-guided context focusing. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dKV-Cache: The cache for diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Weian Mao, Xi Lin, Wei Huang, Yuxin Xie, Tianfu Fu, Bohan Zhuang, Song Han, and Yukang Chen. TriAttention: Efficient long reasoning with trigonometric KV compression. In *International Conference on Machine Learning (ICML)*, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yuerong Song, Xiaoran Liu, Ruixiao Li, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Sparse-dLLM: Accelerating diffusion LLMs with dynamic cache eviction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

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

Zeqing Wang, Gongfan Fang, Xinyin Ma, Xingyi Yang, and Xinchao Wang. SparseD: Sparse attention for diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion LLM. In *International Conference on Learning Representations (ICLR)*, 2026a.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *International Conference on Learning Representations (ICLR)*, 2026b.

Haocheng Xi, Harman Singh, Yuezhou Hu, Coleman Hooper, Rishabh Tiwari, Aditya Tomar, Minjae Lee, Wonjun Kang, Michael Mahoney, Chenfeng Xu, Kurt Keutzer, and Amir Gholami. LoSA: Locality aware sparse attention in diffusion language models. In *International Conference on Machine Learning (ICML)*, 2026.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024b.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. FlashInfer: Efficient and customizable attention engine for LLM inference serving. In *Proceedings of Machine Learning and Systems (MLSys)*, 2025.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Wenhu Zhang, Yiming Wu, Huanyu Wang, Yaoyang Liu, Huanzhang Dou, Senqiao Yang, Sitong Wu, Hanbin Zhao, and Jiaya Jia. Efficient long-context modeling in diffusion language models via block approximate sparse attention. In *Findings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A QUERY AGREEMENT AND SELECTOR FIDELITY

Probe protocol. All probes are collected on the decoder's own generation path, at the first denoising step of a block, using the same prompt template as the evaluation runner. We use 150 documents per model and exclude layers 0 and 1, where the masked queries are still nearly identical. Figure 2a and 2b show one SDAR cell in the plane of a single rotary frequency pair, picked by a fixed rule. Layer 3 is the layer of largest pre/post contrast among layers 2 and deeper, taken over all 150 documents. Within that layer, head 18 has the median pre/post gap on the first document. Frequency pair 11 carries the most query magnitude among pairs whose block rotation spans at least half a turn. Each series is divided by its own mean magnitude, leaving the angles untouched. Figure 2c reports every probed layer.

Agreement across the three models. Pre-RoPE agreement exceeds post-RoPE agreement in all 11,700 document–layer cells across the three models. It also exceeds the unmasked control of Figure 2c in 11,697 of them; the 3 exceptions come from one SDAR document at layers 16 to 18. Only Fast-dLLM v2 needs its prompt trimmed to make the probe's first block entirely masked. Both comparisons also hold without that trimming: over its 3,874 cells (149×26), pre-RoPE agreement still exceeds post-RoPE agreement and the unmasked control, including blocks with only 3 of 32 positions masked. SDAR and LLaDA2 begin with all-mask blocks under their released evaluation setup.

Table 3: Query agreement across depth, 150 documents each. *Valley* is the layer of lowest pre-RoPE agreement and *depth* that layer divided by the model's total layer count; *closest* is the smallest pre/post gap at any layer. The first two layers are excluded, so *probed* is two fewer than the total, and each model is probed at its released block length.

			valley			last layer		
Model	B	probed	layer	depth	pre	pre	post	closest
SDAR-4B-Chat-b64	64	34	17	0.47	0.730	0.987	0.810	0.066
Fast-dLLM v2 1.5B	32	26	20	0.71	0.938	0.960	0.908	0.034
LLaDA2.0-mini	32	18	12	0.60	0.837	0.937	0.767	0.063

Partial blocks. Fast-dLLM v2's first block also holds unmasked positions. To test whether they bias the routing mean, we recompute it from the remaining masked positions alone: candidate recall changes by 0.0003 and the task score does not improve, although the two means separate as the block empties (cosine 0.886 at $1/32$ masked). On SDAR and LLaDA2, whose first block is entirely masked, the two means coincide.

Selector fidelity. The *oracle* consists of the K prefix keys with the largest aggregated prefix-attention mass. For each KV head, we compute attention from the block's B post-RoPE queries to the prefix keys, normalise over the prefix, and sum the resulting weights over the B queries and the G query heads sharing that KV head. At $K{=}1024$, SPIRA's selected set captures attention mass within 0.0004 of the oracle on all three models, over 450 document–layer cells per model (50 documents $\times$ 9 evenly spaced layers). The oracle itself leaves about 6% of the attention mass outside its own K keys. These probes use 2,688–6,162 token prefixes, where $K{=}1024$ is a larger fraction of the context than at evaluation length.

Residual-basis probe. For each query head, we form a $B \times d$ residual matrix and take its top eight left singular vectors, which define a rank-eight subspace over block positions. The pre-RoPE arm uses $\mathbf{Q}^{\mathrm{pre}} - \boldsymbol{\mu}$ and the post-RoPE arm $\mathbf{Q}^{\mathrm{post}} - \bar{\mathbf{R}}\boldsymbol{\mu}$; the original post-RoPE queries are projected onto that subspace along the position dimension before rescoring, on the same 450 cells.

B THE ROTARY DECOMPOSITION

Table 4 reports the two quantities plotted in Figure 3b over the slow and fast halves of the rotary spectrum. The instrumentation is read-only and leaves selection unchanged. It passes three sanity

checks: $\boldsymbol{\delta}$ falls to 1.4×10^{-7} of the block mean when $\boldsymbol{\epsilon} = \mathbf{0}$ and to 6.5×10^{-8} when all queries share one position, and unrotated channels are excluded under partial rotary.

Two quantities are reported per band. The per-band magnitude $|\bar{\mathbf{R}}|$ depends only on the block length and the rotary schedule, while $\|\bar{\mathbf{R}}\boldsymbol{\mu}\|/\|\boldsymbol{\mu}\|$ also reflects how query magnitude is distributed across the rotary bands, and corresponds to the shared term used in stage 1. They agree to within 0.02 over most bands, and differ by up to 0.06 at the transition between the slow and fast halves of the spectrum. Under the routing-recall probe of Section 3, substituting the post-RoPE block mean for the shared term changes top-K recall by less than 0.002 on all three models.

Table 4: Consensus and residual terms by rotary band (Equation 2): $\bar{\mathbf{R}}\boldsymbol{\mu}$ is the shared term used in stage 1, while $\boldsymbol{\delta}$ captures the contribution of query-specific residuals. Each half reports the *worst* of its 8 bands (16 in total), taken over per-band medians across (document-block, layer) records, 30 documents per model. The last two columns give the largest-to-smallest ratio across the spectrum; LLaDA2's bands cover its rotary channels only.

			slow half		fast half		spread over bands	
Model	B	pairs	$\|\bar{\mathbf{R}}\boldsymbol{\mu}\|/\|\boldsymbol{\mu}\|$	$\|\boldsymbol{\delta}\|/\|\boldsymbol{\mu}\|$	$\|\bar{\mathbf{R}}\boldsymbol{\mu}\|/\|\boldsymbol{\mu}\|$	$\|\boldsymbol{\delta}\|/\|\boldsymbol{\mu}\|$	$\boldsymbol{\epsilon}$	$\boldsymbol{\delta}$
SDAR-4B-Chat-b64	64	64	0.9999	2.1×10^{-3}	0.0371	0.128	1.5×	2.3×10^{4}
Fast-dLLM v2 1.5B	32	64	1.0000	1.5×10^{-3}	0.0682	0.193	1.8×	7.2×10^{4}
LLaDA2.0-mini	32	32	0.9999	2.4×10^{-3}	0.0592	0.210	2.1×	2.2×10^{4}

The mean-rotation identity. For the frequency-f pair, RoPE rotates position p by $p\theta_f$. Averaging over the block's B consecutive positions $p_0, \ldots, p_0 + B - 1$ gives

$$\frac{1}{B}\sum_{r=0}^{B-1} e^{i(p_0+r)\theta_f} \;=\; e^{i\bar{p}\theta_f}\,\frac{\sin(B\theta_f/2)}{B\sin(\theta_f/2)}, \qquad \bar{p} = p_0 + \tfrac{B-1}{2}, \tag{7}$$

which is Equation (3), and it holds for whatever frequencies θ_f the model actually rotates by. We therefore evaluate the mean rotation at the model's own inverse frequencies rather than at a base RoPE parameter. LLaDA2 rotates only the first half of each head, so the identity applies to those channels and the remaining ones use the identity transform.

C THE SELECTION PROCEDURE

Algorithm 1 is the per-layer procedure, including the guards the implementation applies: the retained budget is clipped below the prefix length where possible, and the candidate pool cannot exceed the prefix.

Algorithm 1 SPIRA selection at one layer and one KV head h, whose query heads are $\mathrm{group}(h)$. Selection runs when $\mathcal{S}$ is unset; later denoising steps of the block reuse the cached set.

state $\mathcal{S}$, gathered $(\mathbf{K}_{\mathcal{S}}, \mathbf{V}_{\mathcal{S}})$ ▷ unset at the start of each block
if $\mathcal{S}$ is unset **then**
  $K \leftarrow \max(1, \min(K, N-1))$; $C \leftarrow \min(mK, N)$
  for all $g \in \mathrm{group}(h)$ **do**
    $\boldsymbol{\mu}_g \leftarrow \frac{1}{B}\sum_{q\in b} \mathbf{Q}^{\mathrm{pre}}[g, q, :]$ ▷ pre-RoPE block mean
    $\mathbf{v}_g \leftarrow \bar{\mathbf{R}}\,\boldsymbol{\mu}_g$ ▷ fold in the block's mean rotation
  end for
  $\mathcal{C} \leftarrow \text{Top-}C \sum_g \mathrm{softmax}_N\big(\mathbf{v}_g \mathbf{K}^\top/\sqrt{d}\big)$ ▷ one query per head scans the prefix
  $\mathcal{S} \leftarrow \text{Top-}K \sum_g \sum_{q\in b} \mathrm{softmax}_C\big(\mathbf{Q}^{\mathrm{post}}[g, q, :]\,\mathbf{K}[\mathcal{C}]^\top/\sqrt{d}\big)$ ▷ all B queries, candidates only
  $(\mathbf{K}_{\mathcal{S}}, \mathbf{V}_{\mathcal{S}}) \leftarrow \mathrm{gather}(\mathbf{K}, \mathbf{V}, \mathcal{S})$
end if
return $\mathrm{Attention}\big(\mathbf{Q}^{\mathrm{post}}, [\mathbf{K}_{\mathcal{S}}; \mathbf{K}^b], [\mathbf{V}_{\mathcal{S}}; \mathbf{V}^b]\big)$ ▷ the block's own keys are never budgeted

D LongBench and RULER, in full

Prefix read fraction. Every selector also scans the prefix, SPIRA once per block and Quest once per denoising step, but that scan is selection-only and is not counted. SparseD's read fraction is dominated by the two dense steps it keeps per block (see *Baseline configuration*); on its sparse steps it reads K/N, as SPIRA does at every step.

Baseline configuration. Quest uses KV page size 16, max aggregation over the block's queries, and keeps its first two layers dense, as its released code does (`layer_id < 2`). SparseD uses KV page size 128 and a fixed key budget rather than a page ratio, so the budget column means the same thing for every method. Because its released dense-step schedule assumes a fixed step count and threshold committing does not produce one, we keep two dense steps per block throughout, which gives SparseD more dense computation than a strict budget would. All methods are evaluated at $K \in \{256, 512, 1024\}$.

Table 5: RULER's thirteen-task average at every context length, 2,600 documents per model. Each cell gives the score and, beneath it, the prefix read fraction; SPIRA at m=4, baselines as in Appendix D. Q, Quest; SD, SparseD; SP, SPIRA. Bold, best sparse; blue, SPIRA. †, stage 1 returns every key.

Model	K	4k Q	4k SD	4k SP	8k Q	8k SD	8k SP	16k Q	16k SD	16k SP	32k Q	32k SD	32k SP
SDAR 4B	Exact		73.0			61.4			54.7			40.4	
	256	43.1 .12	33.9 .32	**58.9** .07	25.9 .09	18.5 .25	**43.4** .03	17.4 .07	15.1 .20	**34.6** .02	7.0 .06	7.7 .16	**20.6** .01
	512	52.8 .19	56.2 .41	**62.7** .14	37.6 .12	42.3 .32	**45.8** .07	26.8 .09	30.8 .22	**39.8** .03	12.7 .07	13.1 .16	**23.4** .02
	1024	63.4 .32	**67.8** .52	65.9† .28	46.4 .18	**52.4** .37	50.4 .13	35.6 .12	**43.9** .27	43.3 .06	21.2 .09	24.5 .18	**25.8** .03
Fast-dLLM v2 1.5B	Exact		71.3			65.6			55.5			40.9	
	256	**50.2** .14	33.4 .33	46.5 .07	**31.9** .10	26.1 .30	31.5 .03	18.1 .09	**26.7** .29	18.5 .02	8.8 .08	**14.5** .26	8.2 .01
	512	58.6 .20	**60.1** .41	54.9 .14	44.0 .13	**47.5** .34	41.0 .07	25.4 .10	**39.5** .32	24.8 .03	12.9 .09	**21.1** .26	11.8 .02
	1024	64.3 .33	**68.9** .52	63.0† .28	51.9 .19	**58.5** .39	50.5 .13	33.3 .13	**45.6** .35	34.7 .06	16.7 .10	**26.5** .28	17.8 .03
LLaDA2 mini	Exact		92.6			91.5			89.4			83.2	
	256	78.7 .16	55.1 .29	**87.5** .07	73.4 .13	58.2 .27	**85.5** .03	68.0 .11	60.3 .37	**81.3** .02	55.6 .11	51.4 .32	**69.3** .01
	512	85.5 .23	82.0 .39	**90.1** .14	78.9 .16	79.9 .36	**86.6** .07	75.7 .13	76.4 .42	**84.0** .03	65.7 .11	67.8 .37	**75.1** .02
	1024	88.6 .35	**90.9** .51	90.9† .28	84.5 .22	**88.8** .42	88.5 .13	82.0 .16	80.8 .45	**86.4** .07	71.0 .13	71.6 .39	**77.3** .03

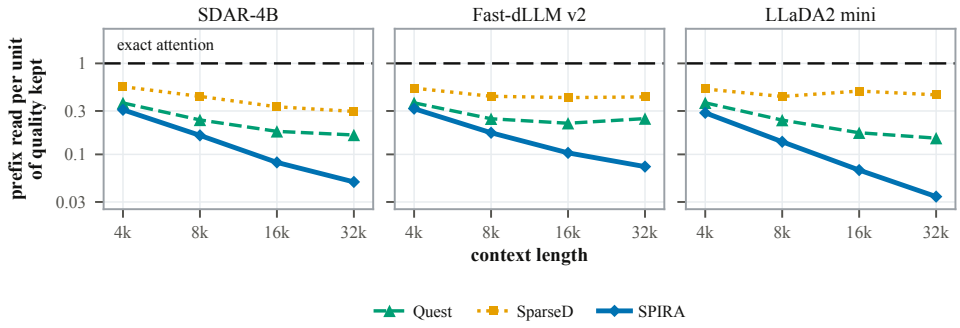


Figure 6: Prefix read fraction divided by the quality retained, against context length; K=1024, m=4, RULER's thirteen tasks. Quality is normalised by each model's own exact-attention score, so exact attention sits at 1 and lower is better.

Table 6: RULER's eight needle-in-a-haystack tasks at 4k and 8k, 800 documents per model. Each cell gives the score and, beneath it, the prefix read fraction; SPIRA's two scores are m=2 and m=4, which share a read fraction. Baselines as in Appendix D. Bold, best sparse at m=4; blue, SPIRA. †, stage 1 returns every key.

		4k tokens			8k tokens		
Model	K	Quest	SparseD	SPIRA m=2→4	Quest	SparseD	SPIRA m=2→4
SDAR 4B	Exact	83.2			67.7		
	256	51.9 .12	36.2 .32	65.4 → **70.6** .07	29.5 .09	16.0 .25	45.3 → **49.3** .03
	512	62.3 .19	66.0 .41	71.8 → **75.1** .14	43.9 .12	49.3 .32	49.8 → **51.6** .07
	1024	75.6 .32	**79.2** .52	76.4 → 77.9† .28	52.0 .18	**60.8** .37	53.2 → 56.5 .13
Fast-dLLM v2 1.5B	Exact	84.6			79.8		
	256	**60.9** .14	38.5 .33	47.6 → 55.8 .07	**40.1** .10	28.4 .30	29.6 → 38.7 .03
	512	**72.1** .20	71.0 .41	60.8 → 64.2 .14	54.5 .13	**58.6** .34	42.4 → 49.6 .07
	1024	77.5 .33	**81.3** .52	69.6 → 73.4† .28	63.8 .19	**69.4** .39	56.4 → 60.8 .13
LLaDA2 mini	Exact	99.5			99.6		
	256	88.2 .16	62.2 .29	97.9 → **98.5** .07	81.9 .13	66.8 .27	96.4 → **96.9** .03
	512	93.2 .23	95.9 .39	99.8 → **99.4** .14	85.8 .16	96.9 .36	98.2 → **98.2** .07
	1024	96.8 .35	**99.8** .51	99.5 → 99.1† .28	91.7 .22	99.0 .42	99.0 → **99.1** .13

E TIMING PROTOCOL

The prefix-length and latency-breakdown measurements use SDAR at batch 1 on one A100-40GB; batch scaling is reported separately. The longest prefix reported for SDAR is 122k; beyond this point its evaluated execution paths exceed the available 40 GB of GPU memory.

For the batch-1 prefix-length measurements we use the faster of the two dense backends. At a 32k prefix FLASHINFER takes 246.4 ms per block against 259.1 ms for FLASHATTENTION (Dao et al., 2022; Dao, 2024), so Table 2's left half and Figure 5a use FLASHINFER. Its right half uses FLASHATTENTION, the backend available to SPIRA above batch 1; the end-to-end cells in Appendix F use it for all three models.

Prefill is excluded because it is the same work under every method; at 32k it costs 3.3 s against 1.2 s for five blocks of decoding. Table 2's batch panel derives throughput from the measured block time while Appendix F measures it over a complete generation; at 8k and batch 1 the two agree to within 1.3%.

Methods are timed in one process up to 98k. At 114k and above each method is timed in its own process. QUEST at 122k is omitted because its repeated runs disagreed and one run failed to record all the required CUDA graphs.

All methods are timed with CUDA graphs. Baseline configurations are as in Appendix D. In the latency breakdown, device time is bucketed by kernel name, and selection covers the scoring and gather kernels of whichever selector is running.

F LATENCY AND THROUGHPUT

Table 7 reports time to first token, decode time and end-to-end latency. Prefill dominates time to first token and is unchanged across methods.

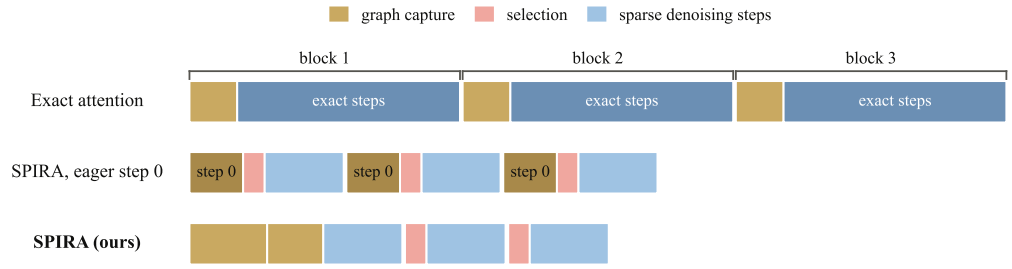


Figure 7: Timeline of three blocks at a 65k prefix; bar widths are measured milliseconds. SPIRA records two CUDA graphs per generation, one for the selection step and one for the reuse steps, and replays them in every block. Middle row: SPIRA with graph reuse disabled.

Table 7: Latency and throughput, exact attention (FLASHATTENTION) against SPIRA at K=1024, m=2, batch 1 on an idle A100-40GB. SDAR generates 576 tokens and the other two 160, so rows are not comparable across models.

		TTFT (s)		decode throughput (tok/s)		end to end (tok/s)		
Model	prefix	Exact	SPIRA	Exact	SPIRA	Exact	SPIRA	ratio
SDAR-4B-Chat-b64	8k	0.67	0.63	301.0	**574.0**	242.6	**393.2**	1.62×
	32k	3.25	3.17	247.3	**572.1**	108.3	**143.7**	1.33×
	65k	10.27	9.75	197.1	**557.1**	44.8	**54.3**	1.21×
Fast-dLLM v2 1.5B	8k	0.71	0.82	88.3	**193.9**	74.1	**108.1**	1.46×
	32k	3.68	3.70	58.4	**97.7**	27.3	**31.9**	1.17×
	65k	12.25	12.38	52.2	**123.1**	10.9	**11.9**	1.09×
LLaDA2.0-mini	8k	1.37	1.44	45.6	**134.7**	38.3	**66.9**	1.75×
	32k	3.02	3.01	44.4	**87.1**	27.1	**35.7**	1.32×
	65k	8.68	8.58	46.4	**83.1**	14.0	**15.8**	1.13×
	131k	29.95	28.32	31.8	**91.4**	4.7	**5.4**	1.14×

G BUDGET, MULTIPLIER AND CHECKPOINT SWEEPS

We vary the retained budget K and the candidate multiplier m, with $C = \min(mK, N)$ candidates in stage 1. Table 8 reports every (K, m) setting on the three models. Figure 8 isolates the effect of m, while Figure 9 repeats the sweep across four SDAR checkpoints with different block sizes, 1,350 documents each. Across those checkpoints, exact attention itself falls by about 6 points from B=8 to B=64, so each arm is compared against its own checkpoint's exact attention. We use m=2 on LongBench and m=4 on RULER. The latency experiments use K=1024.

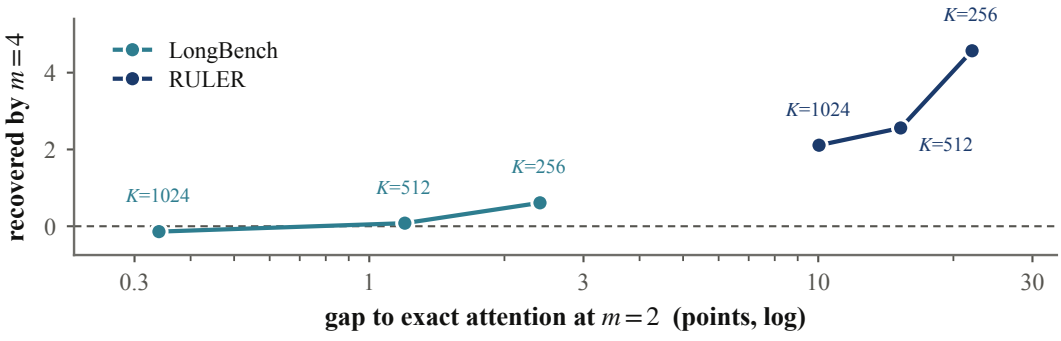


Figure 8: Effect of raising the candidate multiplier from m=2 to m=4. One point per benchmark–budget pair: x is the gap m=2 leaves to exact attention, y what m=4 recovers.

Table 8: SPIRA candidate multiplier sweep. In this table $m = 1$ is evaluated at $K = 1024$ only, where the candidate set is the selection and stage 2 cannot change it; the checkpoint sweep of Figure 9 also covers $m = 1$ at $K = 256$ and 512. QA F1, except QMSum (ROUGE-L), 0–100, on the full test sets: 200 documents per task and 150 for MultiFieldQA-en, 1,350 per model. Avg. is the unweighted mean of the seven tasks. Bold, best m at each model and budget; blue, $m = 2$.

Model	m	K	Hotpot	QMS	2Wiki	Trivia	Narr.	Qasp.	MFQA	Avg.
SDAR 4B	—	Exact	48.66	15.17	34.66	85.04	24.52	22.98	35.17	38.03
	1	1024	49.80	15.61	32.87	82.53	23.34	22.31	32.70	37.02
	2	256	45.18	**17.06**	**31.64**	80.71	22.41	20.45	31.09	35.50
		512	49.40	16.50	**32.32**	81.03	**22.79**	22.77	**32.54**	36.76
		1024	**51.09**	15.78	**33.97**	**84.63**	**24.28**	**22.88**	33.13	**37.97**
	4	256	**48.83**	16.56	**31.64**	**81.29**	**23.82**	**22.62**	**33.06**	**36.83**
		512	**50.02**	**16.56**	31.01	**83.21**	21.91	**22.80**	32.12	**36.80**
		1024	48.79	**15.91**	33.35	84.44	24.19	**22.88**	**33.19**	37.54
Fast-dLLM v2 1.5B	—	Exact	20.07	22.46	14.68	53.81	12.01	21.86	38.73	26.23
	1	1024	16.19	22.06	**15.32**	54.37	12.21	19.88	33.69	24.82
	2	256	14.27	21.52	**13.04**	47.68	11.26	16.82	31.03	22.23
		512	**16.14**	21.85	**14.07**	53.25	**12.17**	**18.74**	33.37	24.23
		1024	17.72	**22.47**	14.73	**55.07**	12.07	20.13	35.57	25.40
	4	256	**14.92**	**21.73**	13.00	**49.94**	**11.64**	**17.12**	**31.33**	**22.81**
		512	16.05	**22.23**	13.66	**53.43**	12.13	18.58	**35.87**	**24.57**
		1024	**18.78**	22.46	14.51	54.64	**13.00**	**20.43**	**36.27**	**25.73**
LLaDA2.0 mini	—	Exact	58.17	22.95	51.72	87.38	25.17	46.46	52.78	49.23
	1	1024	57.56	22.46	**49.64**	87.54	25.95	45.40	53.03	48.80
	2	256	56.52	**23.07**	**50.85**	**88.51**	24.47	**44.27**	**52.31**	**48.57**
		512	57.50	**22.80**	**50.08**	87.75	**26.37**	**45.41**	**52.36**	**48.90**
		1024	57.24	22.36	49.44	**87.75**	**26.67**	**47.05**	**53.25**	**49.11**
	4	256	**57.88**	22.96	50.04	88.21	**24.76**	43.98	51.53	48.48
		512	**58.50**	22.54	49.41	**88.46**	25.70	45.10	51.65	48.77
		1024	**57.82**	**22.64**	49.30	87.38	25.71	46.36	52.24	48.78

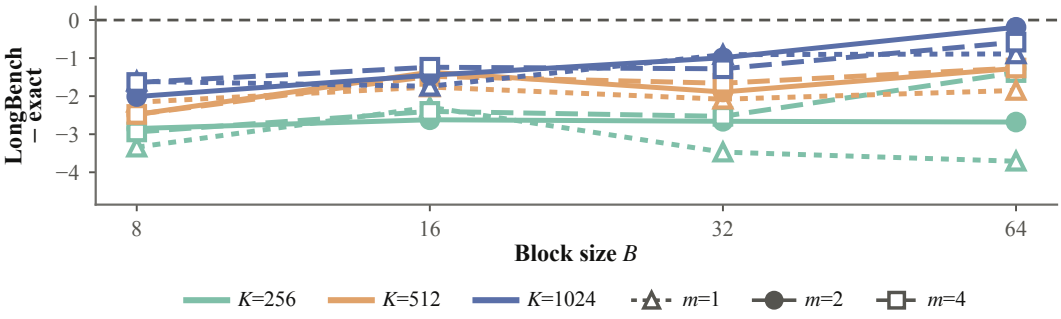


Figure 9: Effect of K and m across SDAR checkpoints, against *each checkpoint's own* exact attention. Increasing K narrows the gap at every checkpoint; m has a smaller and less consistent effect. Block size is confounded with checkpoint.