MAGE: All-[MASK] Block Already Knows Where to Look in Block Diffusion LLM

Omin Kwon [1] **Yeonjae Kim** [1] **Doyeon Kim** [1] **Minseo Kim** [1] **Yeonhong Park** [2] **Jae W. Lee** [1]

Abstract

Block diffusion LLMs are an emerging paradigm for parallel language generation, but their KV caching makes memory access the dominant bottleneck in long-context inference, motivating sparse attention that attends only to a small KV subset per query. In block diffusion, however, the B tokens of each block must share a single KV subset, and we show this per-block constraint degrades existing sparse KV estimators by up to 25% in recall. We address this by exploiting a property of the block-diffusion training objective: it aligns the block-average query across denoising steps, so the All-[MASK] block at the first step already reveals the per-block KV subset for the entire trajectory. Building on this, MAGE ([MASK]-Guided Sparse Attention) is a training-free method that runs one exact attention pass at the first step and reuses its top-k index sets for all remaining steps within the block. Across three block-diffusion families on LongBench, MAGE matches Exact Attention at k=512 with near-lossless accuracy, achieves up to $6.82\times$ end-to-end speedup at 128K context, and runs up to $3.35\times$ and $2.28\times$ faster than Quest and SparseD, respectively.

1. Introduction

Block diffusion LLMs (Block dLLMs) (Wu et al., 2026a; Cheng et al., 2025; Bie et al., 2025) have emerged as a promising paradigm for parallel language generation: by adapting pretrained autoregressive (AR) models into block-wise diffusion models with exact KV caching, they increase the throughput by multi-token generation while maintaining competitive quality. As context length grows, attention dominates the overall inference computation and becomes bottlenecked by KV cache memory access, motivating sparse attention as a natural remedy. Sparse attention reduces this cost by attending only to a small selected *KV subset*—the top-k KV cache entries that capture most of the attention mass—rather than the full cache. Dynamic estimators such as Quest (Tang et al., 2024) have proven highly effective for AR LLMs, and methods such as SparseD (Wang et al., 2026) have been proposed for fully bidirectional dLLMs—but neither was designed for block diffusion, and transferring them directly reveals a structural mismatch that substantially degrades their accuracy.

The mismatch stems from a constraint unique to block diffusion: the B tokens within a block are forwarded together, so they must share a single *per-block* KV subset rather than each selecting its own per-token subset. Constructing a single shared subset that simultaneously satisfies each of the B tokens' top-k preferences is structurally harder: the union of their per-token subsets averages $4.5\times$ the budget. As a result, when existing estimators are adapted to this regime, their recall drops by up to 25%. We identify an opportunity on a *different axis*: block-diffusion training, by requiring the model to predict each masked token under all surrounding noise levels within a block, aligns the block-average query across the denoising steps. As a consequence, the All-[MASK] block in the first step within a block already reveals the per-block KV subset for the entire trajectory, with per-block top-k recall averaging over 84% at k=512—substantially above the AR pre-training backbone, confirming this anchoring is training-induced.

Building on this observation, we propose MAGE ([MASK]-Guided Sparse Attention), the first sparse attention method tailored to block-diffusion LLMs. MAGE runs a single exact attention pass on the All-[MASK] block at the first step (Step 1), extracts top-k oracle index sets, and reuses them unchanged for all $T-1$ remaining denoising steps—requiring no re-estimation at any subsequent step. To avoid paying the selection cost on the critical path, MAGE executes the index-selection kernels asynchronously on a dedicated CUDA stream, overlapping them with the compute-intensive FFN of the main stream; the amortized non-sparse overhead is $1/T$ of one exact attention pass per step. Across three block-diffusion LLM families on LongBench, MAGE matches Exact Attention at k=512 with near-lossless accu-

[1]Seoul National University, Seoul, Republic of Korea [2]Meta. Correspondence to: Jae W. Lee <jaewlee@gmail.com>.

Proceedings of the ICML 2026 Workshop on AdaptFM: Resource-Adaptive Foundation Model Inference, Seoul, South Korea. 2026.

racy and achieves end-to-end speedups of up to $6.82\times$ at 128K context, running up to $3.35\times$ faster than Quest and $2.28\times$ faster than SparseD.

Our contributions are summarized as follows:

- We identify a *per-block shared KV selection* challenge unique to block diffusion, under which existing estimators lose up to 25% recall (§3.1).
- We discover the All-[MASK] anchoring property induced by block-diffusion training that makes Step 1 an accurate oracle anchor (§3.2).
- We propose MAGE, a training-free sparse attention method whose asynchronous dual-stream design adds only $+1.9$–6.4% Step-1 overhead (§4).
- We show near-lossless accuracy at k=512 with up to $6.82\times$ speedup at 128K, beating Quest by $3.35\times$ and SparseD by $2.28\times$ (§5).

2. Background

2.1. Block Diffusion Language Models

Diffusion LLMs generate text by iteratively denoising sequences of [MASK] tokens, enabling parallel multi-token generation and breaking the left-to-right dependency of autoregressive (AR) models (Nie et al., 2026; Ye et al., 2025a). Earlier fully bidirectional dLLMs (Nie et al., 2026; Ye et al., 2025a) recompute the entire sequence at every denoising step, incompatible with KV caching and yielding substantial latency (Kim et al., 2025); subsequent approximate caching schemes (Wu et al., 2026b; Ma et al., 2026) accumulate errors across steps.

More recent *block diffusion LLMs* (Wu et al., 2026a; Cheng et al., 2025; Bie et al., 2025) adapt pretrained AR models into block-wise diffusion, enabling *exact* block-level KV caching without approximation. This shifts the primary bottleneck from compute to KV cache memory access, particularly in long-context settings where the cache size grows substantially. We focus on block dLLMs and target this memory bottleneck.

2.2. Sparse Attention for Long-Context Inference

Sparse attention alleviates this bottleneck by accessing only the small subset of KV entries on which the attention mass concentrates—ideally the *oracle top-k*, defined as the k prefix indices that carry the bulk of the mass in the exact attention score distribution. Obtaining the oracle requires the very score matrix that full attention would produce; existing methods therefore perform sparse attention through *lightweight estimators*. We measure how well any KV subset serves a given token via *estimation accuracy*—the fraction of that token's oracle top-k covered by the selected subset:

$$\mathrm{EstAcc}_{q,h}^{(t)} = \frac{\left|\mathcal{S}_h^{(t)} \cap \mathcal{S}_{q,h}^{*(t)}\right|}{k}, \tag{1}$$

where h indexes the KV head, $\mathcal{S}_h^{(t)}$ is the KV subset being evaluated, and $\mathcal{S}_{q,h}^{*(t)}$ is the oracle for token q. No such estimator has yet been designed for block dLLMs; we review existing methods below.

Methods for AR LLMs. For AR LLMs, beyond fixed sparsity patterns (Xiao et al., 2024; Beltagy et al., 2020) and one-shot eviction (Zhang et al., 2023; Li et al., 2024), *dynamic* per-token estimators (Tang et al., 2024; Ribar et al., 2024; Singhania et al., 2024) achieve substantially higher accuracy by tracking the oracle on the fly. Quest (Tang et al., 2024) is a representative example: it partitions the prefix KV cache into contiguous *pages* of size p, summarizes each page by its per-channel max/min keys, scores each page against the query, and selects the top-scoring pages as the sparse subset. Smaller p yields finer-grained selection at higher scoring cost—a granularity-cost trade-off. Follow-ups refine this recipe via cheaper scoring (Ribar et al., 2024; Singhania et al., 2024) or adaptive per-layer budgets (Cai et al., 2025; Zhu et al., 2026; Lin et al., 2026).

Methods for Fully Bidirectional Diffusion LLMs. In fully bidirectional dLLMs, SparseD (Wang et al., 2026) performs exact attention during the first 20% of the total denoising steps across all blocks, extracts each token's oracle top-k at the final exact-attention step, and reuses these per-token attention masks across the remaining 80% of the steps. MaskKV (Huang et al., 2025) instead drops less critical prompt tokens after prefill, reducing the sequence length processed in subsequent steps. In both, the sparse pattern is captured once per inference and never refreshed, which can degrade recall as the trajectory unfolds.

3. Challenges and Opportunities of Sparse Attention in Block Diffusion LLMs

We identify the structural constraint that sparse attention on block dLLMs must operate under, and a block-diffusion-specific property that recovers estimation accuracy within this constraint along a different axis.

3.1. The Challenge: Sparse KV Subset Estimation Is Harder in Block Diffusion LLMs

Per-Block KV Subset Forced by Shared Memory Access. Within a block, the B tokens pass through the model in a single forward pass with shared prefix KV reads, computed as a single GEMM. Unlike AR LLMs and bidirectional dLLMs that select *per-token* subsets, a block dLLM requires the entire block to share a single *per-block* subset. Combined with the existing G-head sharing under GQA, the per-block top-k

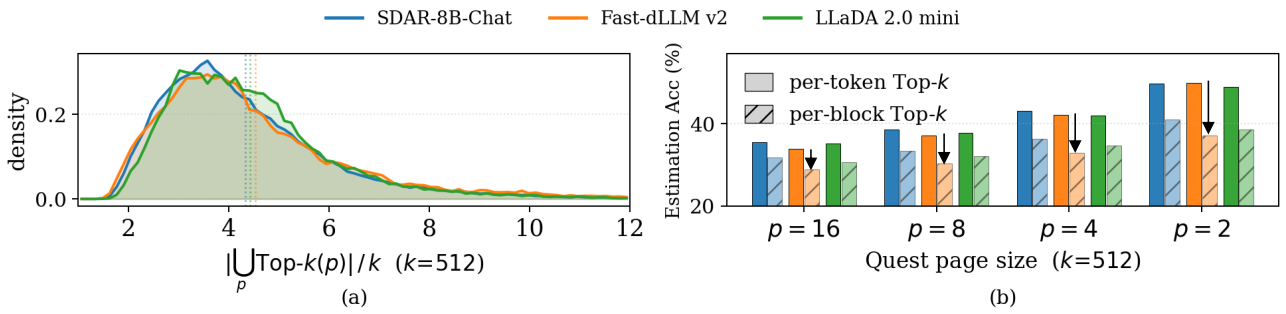

Figure 1. Sparse attention challenge on block dLLMs. *(a)* Distribution of $|\bigcup_p \text{Top-}k(p)|/k$ at k=512, t=1. *(b)* Quest (Tang et al., 2024) estimation accuracy at k=512, t=1, page sizes $p \in \{16, 8, 4, 2\}$.

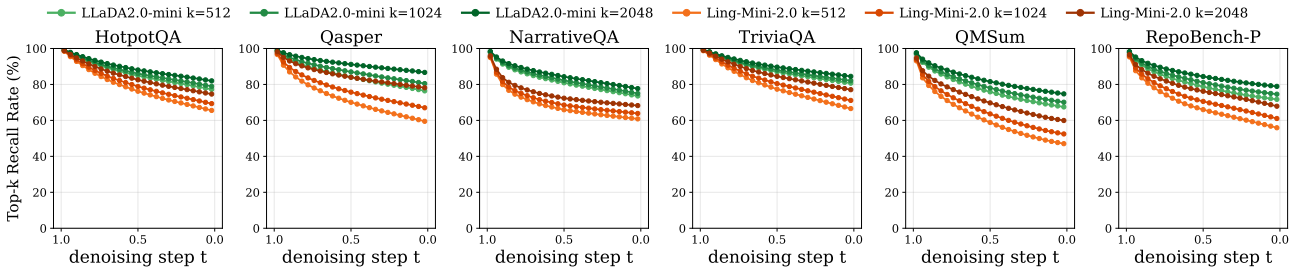

Figure 2. Per-block top-k recall along the denoising trajectory on LongBench. *Trained*: LLaDA2.0-mini; *Backbone*: pre-training checkpoint Ling-Mini-base-2.0.

averages scores over both the G heads and the B tokens:

$$\mathcal{S}^{(t)}_{q,h} = \underset{j}{\text{Top-}k}\Big(\tfrac{1}{G}\sum_{g\in\text{group}(h)} s^{(t)}_{g,q,j}\Big) \quad \text{(per-token)}$$
$$\mathcal{S}^{(t)}_{b,h} = \underset{j}{\text{Top-}k}\Big(\tfrac{1}{GB}\sum_{g\in\text{group}(h)}\sum_{q\in b} s^{(t)}_{g,q,j}\Big) \quad \text{(per-block)} \tag{2}$$

where $j \in [N]$ indexes a prefix KV position. With s as the exact attention score, the result is the *oracle per-block subset* $\mathcal{S}^{*(t)}_{b,h}$; with a lightweight approximation (e.g., Quest), it is the *estimated per-block subset* $\hat{\mathcal{S}}^{(t)}_{b,h}$.

Estimation Accuracy Drop Under Per-Block Shared Selection. A single per-block subset shared by B tokens cannot satisfy each token's individual oracle. Forming the union of the B per-token oracle top-k subsets, we find at k=512 the union averages $4.5\times$ the per-token budget across three block-diffusion models (Fig. 1(a), heavy tail beyond $10\times$). Consequently, Quest's per-block estimation accuracy falls below the per-token baseline by up to 25% across page sizes $p \in \{16, 8, 4, 2\}$ (Fig. 1(b)).

3.2. The Opportunity: Accurate Estimation via All-[MASK] Block Anchoring

That $4.5\times$ union covers all B=32 tokens implies a substantial intersection of KV positions exists; identifying it within the k-budget is the central question. We discover that this intersection—the oracle per-block subset—barely drifts across the denoising trajectory, rooted in the block-diffusion training mechanism. This stability lets us invest one exact-attention pass at the first step to identify the oracle, then reuse it across the remaining steps.

Observation: Oracle Per-Block Subset Barely Drifts Across Denoising Steps. We quantify stability as the fraction of the all-[MASK] oracle (t=1) recovered at an arbitrary noise level $\tau \in [0, 1)$: $r^{(\tau)}_{b,h} = |\mathcal{S}^{*(1)}_{b,h} \cap \mathcal{S}^{*(\tau)}_{b,h}|/k$. LLaDA2.0-mini averages 85%/86%/89% recall at k=512/1024/2048, uniform across all 6 LongBench tasks (Fig. 2). The same measurement on its AR pre-training checkpoint (Ling-Mini-base-2.0) yields only 76%/79%/83%, pointing to training as the source.

Mechanism: Anchoring as a Direct Consequence of the Block-Diffusion Objective. Block-diffusion training keeps the prefix clean while corrupting the block at a random noise level $\mathbf{m}$ and predicting the same ground-truth target under every $\mathbf{m}$. This forces activations of [MASK] and revealed positions to align across noise levels, propagating from predictions through hidden states and queries to the per-block top-k subset. We verify this five-stage chain against AR pre-training backbones: training (**S1**) collapses cross-noise predictions (D_{KL} ↓43–63%); aligns (**S2**) hidden states (8–19%), (**S3**) per-position queries via $q = W_Q h$ (13–27%), and (**S4**) block-average queries, which by Jensen

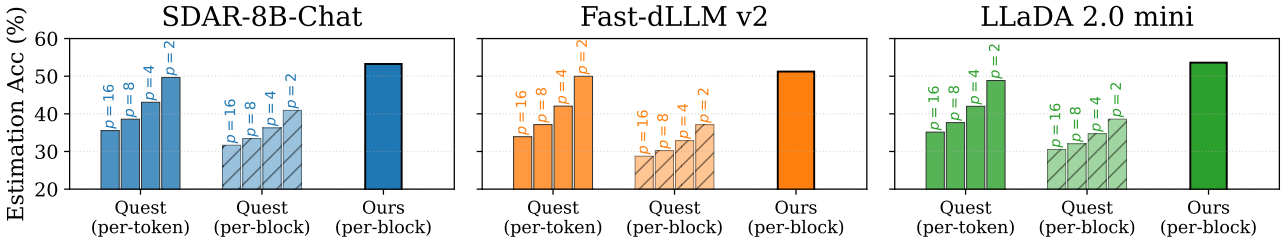

Figure 3. Estimation accuracy of different KV subset estimators at k=512 on LongBench.

$\|\mathbb{E}_{\mathbf{m}}[\bar{q}^{(\mathbf{m}^*)} - \bar{q}^{(\mathbf{m})}]\| \le \frac{1}{B}\sum_i \|\bar{\delta}_i\|$ stay below per-position (col. 4 ¡ col. 3 in every family); finally (**S5**) stabilizes the per-block top-k (recall at k=512 ↑5–11 points). Full per-stage measurements and loss decomposition are in Appendix A (Table 2).

Our Proposal: One Exact Attention Pass at Step 1 Closes the Estimation Gap. We run one exact-attention pass on the all-`[MASK]` block at t=1, extract $\mathcal{S}^{*(1)}_{b,h}$, and reuse it for all remaining steps ($\hat{\mathcal{S}}^{(t)}_{b,h} = \mathcal{S}^{*(1)}_{b,h}$ for $t < 1$). Because the subset is computed exactly rather than approximated, our method removes the estimator-side error that Quest accumulates on top of the per-block constraint. Our estimation accuracy surpasses Quest per-block at every page size and matches or exceeds even Quest per-token at p=2 (Fig. 3).

4. MAGE: [MASK]-Guided Sparse Attention

We present MAGE (`[MASK]`-Guided Sparse Attention), the first sparse attention method tailored to block-diffusion LLMs (Algorithm 1). Building on the one-shot All-`[MASK]` selection of §3.2, MAGE introduces an asynchronous dual-stream execution that hides the selection cost behind the layer's FFN, leaving the main forward pass as MAGE's sole non-sparse work. The procedure is training-free and applies to any pretrained block-diffusion LLM.

Phase 1: Oracle Estimation at Step 1. At the start of each block's trajectory the B block positions are all `[MASK]`, and Phase 1 splits work across two CUDA streams (Algorithm 1): a MAIN stream runs a full forward pass (FlashAttention + FFN) without materializing the score matrix, while a SELECT stream independently re-derives the prefix attention scores. At each layer ℓ, the MAIN stream hands off $\mathbf{Q}^{(1)}_\ell$ right after the QKV projection and proceeds to $\mathrm{FlashAttn}_\ell$ and FFN_ℓ; the SELECT stream computes $\mathbf{A}^{(1)}_\ell = \mathrm{softmax}(\mathbf{Q}^{(1)}_\ell \mathbf{K}^\top/\sqrt{d}) \in \mathbb{R}^{G\times B\times N}$ (N the prefix length, G the query heads per KV group) and, for each KV head h, averages over the GB queries that share the head to select the top-k prefix positions:

$$\mathcal{S}^{*\ell}_{b,h} \;=\; \text{Top-}k\Big(\tfrac{1}{GB}\sum_{g\in\mathrm{group}(h)}\sum_{q\in b}\mathbf{A}^{(1)}_\ell[g,q,:]\Big). \quad (3)$$

Algorithm 1 MAGE block inference

Require: Prefix KV cache $\mathcal{C}$, block size B, budget k, steps T
Ensure: Generated block $\mathbf{x}$
  Initialize block: B `[MASK]` tokens
  // Phase 1: oracle estimation at Step 1; two streams run concurrently
  Main Stream
  for layer $\ell = 1,\dots,L$ **do**
    $(\mathbf{Q}^{(1)}_\ell, \mathbf{K}^{(1)}_\ell, \mathbf{V}^{(1)}_\ell) \leftarrow \mathrm{QKVProj}_\ell(\mathbf{h}^{(1)}_{\ell-1})$
    enqueue $\mathbf{Q}^{(1)}_\ell \rightarrow$ SELECT ▷ non-blocking
    $\mathbf{h}^{(1)}_\ell \leftarrow \mathrm{Attn}_\ell(\mathbf{Q}^{(1)}_\ell, \mathbf{K}^{(1)}_\ell, \mathbf{V}^{(1)}_\ell, \mathcal{C})$
    $\mathbf{h}^{(1)}_\ell \leftarrow \mathrm{FFN}_\ell(\mathbf{h}^{(1)}_\ell)$
  end for
  Select Stream
  for layer $\ell = 1,\dots,L$ **do**
    wait for $\mathbf{Q}^{(1)}_\ell$
    $\mathbf{A}^{(1)}_\ell \leftarrow \mathrm{softmax}(\mathbf{Q}^{(1)}_\ell \mathbf{K}^\top/\sqrt{d})$
    for each KV head h **do**
      $\bar{\alpha}_{\ell,h} \leftarrow \frac{1}{GB}\sum_{g,q}\mathbf{A}^{(1)}_\ell[g,q,:]$
      $\mathcal{S}^{*\ell}_{b,h} \leftarrow \text{Top-}k(\bar{\alpha}_{\ell,h})$
    end for
  end for
  synchronize(MAIN, SELECT)
  // Phase 2: sparse denoising for Step $s = 2,\dots,T$
  for $s = 2$ to T **do**
    for each layer ℓ **do**
      $(\mathbf{Q}_\ell, \mathbf{K}_\ell, \mathbf{V}_\ell) \leftarrow \mathrm{QKVProj}_\ell(\mathbf{h}_{\ell-1})$
      for each KV head h **do**
        $\mathbf{o}_{\ell,h} \leftarrow \mathrm{SparseAttn}(\mathbf{Q}_{\ell,h}, \mathbf{K}_{\ell,h}, \mathbf{V}_{\ell,h}, \mathcal{C}[\mathcal{S}^{*\ell}_{b,h}])$
      end for
      $\mathbf{h}_\ell \leftarrow \mathrm{FFN}_\ell(\mathbf{o}_\ell)$
    end for
    Unmask high-confidence positions
  end for
  $\mathcal{C} \leftarrow \mathcal{C} \,\|\, \mathbf{KV}(\mathbf{x})$; **return** $\mathbf{x}$

This is precisely the oracle per-block KV subset of Eq. 2, computed without any estimator approximation. The dual-stream design hides the SELECT stream's cost behind FFN_ℓ: FFN_ℓ is compute-intensive while $\mathbf{QK}^\top$ is bound by HBM bandwidth, so the two kernels target different hardware resources and run without contention. The streams synchronize once at the end of Phase 1, leaving the main forward pass as MAGE's sole non-sparse cost—amortized to $1/T$ per step over the T-step trajectory.

Table 1. LongBench per-task and average accuracy for all (model, method, budget) configurations.

		Fast-dLLM v2 7B							SDAR-8B-Chat							LLaDA 2.0 mini						
k	Method	Hot.	Nrtv.	Qas.	QM.	Repo.	Triv.	Avg	Hot.	Nrtv.	Qas.	QM.	Repo.	Triv.	Avg	Hot.	Nrtv.	Qas.	QM.	Repo.	Triv.	Avg
–	Exact	42.38	23.63	36.66	24.16	51.84	75.49	42.36	59.59	28.94	31.02	24.41	50.04	88.83	47.14	60.29	27.91	46.19	22.86	57.11	86.43	50.13
256	MAGE	**37.98**	**22.56**	**31.43**	**23.92**	49.81	**75.17**	**40.15**	**55.87**	**28.86**	**30.89**	**23.57**	**46.56**	**89.63**	**45.90**	**56.75**	**26.19**	**45.23**	**22.82**	55.52	**87.49**	**49.00**
	Quest	32.13	14.98	25.82	21.68	41.20	57.72	32.26	45.81	20.11	22.70	18.90	33.17	78.32	36.50	48.56	13.74	32.89	19.42	46.86	80.71	40.36
	SparseD	38.72	21.83	25.05	20.95	**51.05**	69.66	37.88	50.74	23.01	17.98	20.30	46.66	78.75	39.57	54.20	21.95	32.87	20.21	**55.88**	85.15	45.04
512	MAGE	39.33	**24.23**	**33.41**	**23.94**	51.71	**75.65**	**41.38**	**57.17**	**28.74**	**31.11**	**23.97**	**48.41**	**88.21**	**46.27**	58.48	**28.48**	**46.41**	**22.86**	56.18	**86.70**	**49.85**
	Quest	34.74	15.31	31.60	22.85	46.69	61.86	35.51	49.93	24.19	25.68	21.48	41.29	86.23	41.47	55.37	18.40	40.50	21.43	53.83	84.26	45.63
	SparseD	**40.93**	22.93	31.65	22.90	**51.96**	70.20	40.09	56.54	26.22	26.47	21.75	48.38	86.70	44.34	**59.65**	24.90	44.25	22.05	**56.74**	86.43	49.00
1024	MAGE	41.29	22.42	34.05	**24.79**	**52.36**	**74.17**	**41.51**	57.00	**29.31**	**31.51**	**23.59**	48.66	88.16	**46.37**	59.61	**28.16**	**46.38**	**23.06**	56.71	86.42	**50.06**
	Quest	38.06	19.06	**34.54**	23.10	49.57	66.24	38.43	55.66	26.80	28.19	23.27	46.78	**89.36**	45.01	57.39	24.30	45.65	22.02	**57.59**	**86.93**	48.98
	SparseD	**42.32**	**23.42**	33.50	23.34	51.88	72.51	41.16	**60.03**	28.80	29.47	23.00	**48.95**	87.39	46.27	**60.15**	26.22	44.72	22.32	57.33	86.13	49.48

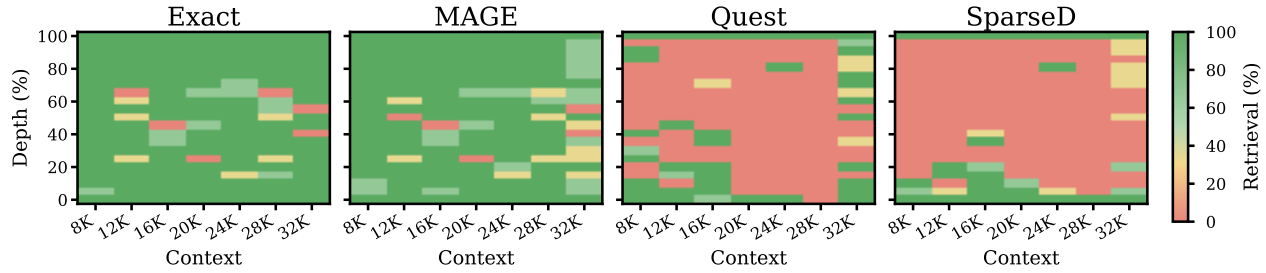

Figure 4. NIAH retrieval on LLaDA 2.0 mini at k=256. x: context (8K–32K); y: needle depth (0–100%). MAGE ≈ Exact; Quest fails beyond 8K; SparseD collapses on most cells. B=32.

Phase 2: Sparse Denoising for Step $s = 2, ..., T$. At every subsequent denoising step, attention at each layer/-head retrieves only the k KV pairs at indices $\mathcal{S}^{*\ell}_{b,h}$ and ignores the remaining prefix entries. The model then unmasks high-confidence positions in the block as in standard block-diffusion decoding. Because the selection is fixed once at Step 1 and reused unchanged across all heads, layers, and steps within the block, no re-estimation is performed at any $s \geq 2$.

5. Evaluation

Setup. We evaluate MAGE on three block-diffusion LLM families (Fast-dLLM v2 7B (Wu et al., 2026a), SDAR-8B-Chat (Cheng et al., 2025), LLaDA 2.0 mini (Bie et al., 2025)) against *Exact Attention* (FlashInfer (Ye et al., 2025b)) and two sparse baselines—Quest (Tang et al., 2024) and SparseD (Wang et al., 2026)—both adapted to per-block KV selection. Quest re-estimates per-block top-k at every step (p=16); SparseD runs exact for the first 20% of steps and reuses the captured pattern. All sparse methods share $k \in \{256, 512, 1024\}$ and keep layers 1–2 exact. We use B=32 and p_{conf}=0.95 (Bie et al., 2025); all kernels use FlashInfer on a single NVIDIA H100.

5.1. Accuracy

LongBench. We evaluate on six LongBench (Bai et al., 2024) long-context tasks spanning single-document QA (NarrativeQA, Qasper), multi-document QA (HotpotQA), few-shot learning (TriviaQA), summarization (QMSum), and code completion (RepoBench-P). Table 1 reports per-task scores (F1 / ROUGE-L / EditSim depending on task) and averages across the six tasks for each (model, method, budget) configuration.

At k=512, MAGE reaches near-lossless accuracy with average drops of only 0.98/0.87/0.28 on Fast-dLLM v2 7B/SDAR-8B-Chat/LLaDA 2.0 mini, far below Quest (6.85/5.67/4.50) and SparseD (2.27/2.80/1.13); even at k=256, MAGE's drops (2.21/1.24/1.13) stay well under Quest's (10.10/10.64/9.77) and SparseD's (4.48/7.57/5.09). MAGE's margin is largest on retrieval-heavy tasks (NarrativeQA, HotpotQA), because MAGE accurately extracts the KV subset each block needs whereas Quest's page-anchoring incurs approximation error and SparseD never refreshes its captured pattern, propagating errors across later blocks.

Needle-in-a-Haystack. Fig. 4 shows NIAH heatmaps for LLaDA 2.0 mini at k=256 over context lengths 8K–32K. MAGE closely tracks Exact with only a few scattered failures, whereas SparseD shows widespread red/yellow cells across most (context, depth) positions, and Quest collapses into near-total failure beyond 12K. At 8K, retrieval scores are MAGE 97, SparseD 17, Quest 41 (Exact 98); per-context scores for all three models at $k \in \{256, 512, 1024\}$ are in Table 3 (Appendix B).

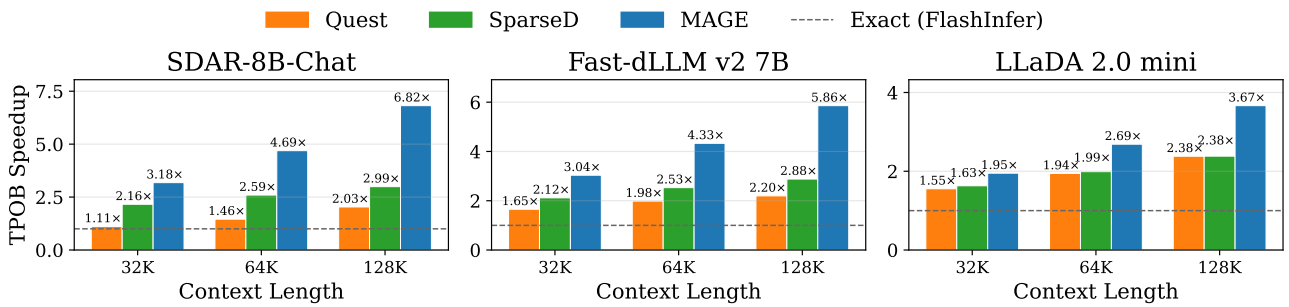

Figure 5. End-to-end TPOB (time-per-output-block) speedup over Exact Attention (FlashInfer) on three block-diffusion LLM families. Budget k=1024, 32 steps/block.

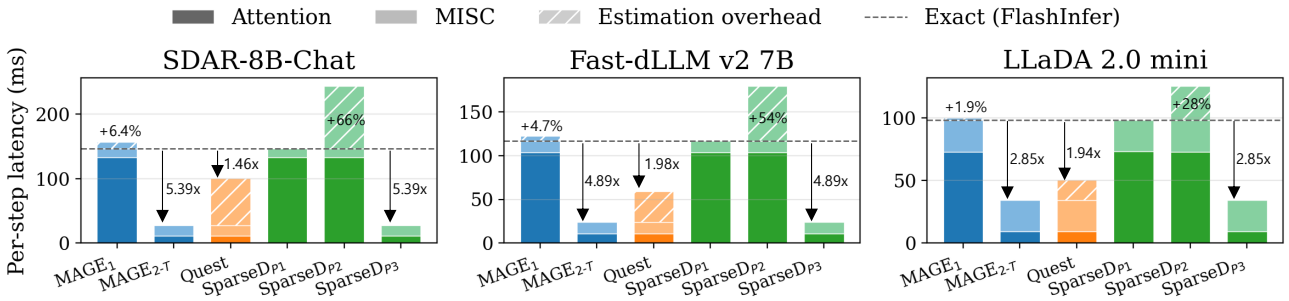

Figure 6. Per-step latency breakdown at 64K, k=1024. Bars decompose one denoising step into Attention, Estimation overhead (hatched), and MISC; dashed line: Exact Attention (FlashInfer).

5.2. Efficiency

End-to-End Speedup. Fig. 5 reports MAGE's end-to-end TPOB (time-per-output-block) speedup over Exact Attention (FlashInfer) on all three block-diffusion families at three context lengths (32K, 64K, 128K) with a fixed sparse budget k=1024 and n=32 block steps; the 64K and 128K settings use NTK-based RoPE extrapolation (Liu et al., 2025) to extend each model's context window. Across every (model, context) pair, the speedup grows with context length as the savings of sparse attention scale with N. At 128K context, MAGE's speedup over Exact reaches $\mathbf{6.82}\times$ on SDAR-8B-Chat, $\mathbf{5.86}\times$ on Fast-dLLM v2 7B, and $\mathbf{3.67}\times$ on LLaDA 2.0 mini — consistently ahead of both Quest and SparseD on every (model, context) cell, running up to $\mathbf{3.35}\times$ faster than Quest and $\mathbf{2.28}\times$ faster than SparseD. That MAGE dominates across three independent block-diffusion implementations indicates the gain is robust to the model rather than a property of any single one. The same ranking holds under shorter block-step budgets $n \in \{24, 16, 8\}$; see Appendix C.

Latency Breakdown. Fig. 6 decomposes latency of one denoising step at 64K, k=1024 into Attention, Estimation overhead, and MISC, with Exact as a dashed reference. MAGE's first step (MAGE $_1$) adds top-k selection on top of an exact pass, but the selection runs on the asynchronous SELECT stream in parallel with FFN (Algorithm 1), keeping critical-path overhead at $+\mathbf{1.9}\%$–$+\mathbf{6.4}\%$ over Exact. The remaining $n-1$ steps (MAGE $_{2\text{-}T}$) run pure sparse attention at $\mathbf{2.85}\times$–$\mathbf{5.39}\times$ over Exact, amortizing the single exact pass per block. Quest recomputes its page-anchor estimate on the critical path at every step, capping per-step speedup at $\mathbf{1.46}\times$–$\mathbf{1.98}\times$. SparseD runs exact attention for the first 20% of steps across the trajectory (P1), takes a capture step at the boundary (P2, $+\mathbf{28}\%$–$+\mathbf{66}\%$), and runs sparse attention for the remaining 80% (P3, matching MAGE $_{2\text{-}T}$); MAGE keeps only $1/n$ of steps exact while SparseD keeps 20%, so the gap accumulates end-to-end.

6. Conclusion

We present MAGE, a training-free sparse-attention method for block-diffusion LLMs. By exploiting the All-[MASK] anchoring property induced by block-diffusion training, MAGE runs one exact attention pass at Step 1 and reuses the resulting per-block top-k indices across the remaining $T-1$ steps, with selection cost hidden asynchronously behind the FFN. Across three block-diffusion families, MAGE matches Exact Attention at k=512 with near-lossless accuracy while achieving up to $6.82\times$ end-to-end speedup at 128K context.

References

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. LongBench: A bilingual, multitask benchmark for long context understanding. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Bie, T., Cao, M., Chen, K., Du, L., Gong, M., Gong, Z., Gu, Y., Hu, J., Huang, Z., Lan, Z., Li, C., Li, C., Li, J., Li, Z., Liu, H., Liu, L., Lu, G., Lu, X., Ma, Y., Tan, J., Wei, L., Wen, J.-R., Xing, Y., Zhang, X., Zhao, J., Zheng, D., Zhou, J., Zhou, J., Zhou, Z., Zhu, L., and Zhuang, Y. LLaDA2.0: Scaling up diffusion language models to 100b, 2025. URL https://arxiv.org/abs/2512.15745.

Cai, Z., Zhang, Y., Gao, B., Liu, Y., Li, Y., Liu, T., Lu, K., Xiong, W., Dong, Y., Hu, J., and Xiao, W. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ayi7qezU87.

Cheng, S., Bian, Y., Liu, D., Zhang, L., Yao, Q., Tian, Z., Wang, W., Guo, Q., Chen, K., Qi, B., and Zhou, B. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation, 2025. URL https://arxiv.org/abs/2510.06303.

Huang, J., Zhang, Y., Yang, Y., Huang, B., Qi, B., Liu, D., and Zhang, L. Mask tokens as prophet: Fine-grained cache eviction for efficient dllm inference, 2025. URL https://arxiv.org/abs/2510.09309.

Kim, M., Hooper, C., Tomar, A., Xu, C., Farajtabar, M., Mahoney, M. W., Keutzer, K., and Gholami, A. Beyond next-token prediction: A performance characterization of diffusion versus autoregressive language models. *arXiv preprint arXiv:2510.04146*, 2025. URL https://arxiv.org/abs/2510.04146.

Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. SnapKV: LLM knows what you are looking for before generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=poE54GOq2l.

Lin, C., Tang, J., Yang, S., Wang, H., Tang, T., Tian, B., Stoica, I., Han, S., and Gao, M. Twilight: Adaptive attention sparsity with hierarchical top-p pruning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=Ve693NkzcU.

Liu, X., Song, Y., Liu, Z., Huang, Z., Guo, Q., He, Z., and Qiu, X. LongLLaDA: Unlocking long context capabilities in diffusion llms, 2025. URL https://arxiv.org/abs/2506.14429.

Ma, X., Yu, R., Fang, G., and Wang, X. dKV-cache: The cache for diffusion language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=Gppo2JImHs.

Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., ZHOU, J., Lin, Y., Wen, J.-R., and Li, C. Large language diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=KnqiC0znVF.

Ribar, L., Chelombiev, I., Hudlass-Galley, L., Blake, C., Luschi, C., and Orr, D. Sparq attention: Bandwidth-efficient LLM inference. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024. URL https://openreview.net/forum?id=Ue8EHzaFI4.

Singhania, P., Singh, S., He, S., Feizi, S., and Bhatele, A. Loki: Low-rank keys for efficient sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=raABeiV71j.

Tang, J., Zhao, Y., Zhu, K., Xiao, G., Kasikci, B., and Han, S. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=KzACYw0MTV.

Wang, Z., Fang, G., Ma, X., Yang, X., and Wang, X. Sparsed: Sparse attention for diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dwbrZtYP04.

Wu, C., Zhang, H., Xue, S., Diao, S., Fu, Y., Liu, Z., Molchanov, P., Luo, P., Han, S., and Xie, E. Fast-dLLM v2: Efficient block-diffusion LLM. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=1NZ3DHF9nT.

Wu, C., Zhang, H., Xue, S., Liu, Z., Diao, S., Zhu, L., Luo, P., Han, S., and Xie, E. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=3Z3Is6hnOT.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

Ye, J., Xie, Z., Zheng, L., Gao, J., Wu, Z., Jiang, X., Li, Z., and Kong, L. Dream 7b: Diffusion large language models, 2025a. URL https://arxiv.org/abs/2508.15487.

Ye, Z., Chen, L., Lai, R., Lin, W., Zhang, Y., Wang, S., Chen, T., Kasikci, B., Grover, V., Krishnamurthy, A., and Ceze, L. Flashinfer: Efficient and customizable attention engine for LLM inference serving. In *Eighth Conference on Machine Learning and Systems*, 2025b. URL https://openreview.net/forum?id=RXPofAsL8F.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Re, C., Barrett, C., Wang, Z., and Chen, B. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=RkRrPp7GKO.

Zhu, K., Tang, T., Xu, Q., Jin, Z., Gu, Y., Zeng, Z., Kadekodi, R., Zhao, L., Li, A., Krishnamurthy, A., and Kasikci, B. Tactic: Adaptive sparse attention with clustering and distribution fitting for long-context LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tJod11fK1A.

A. How Block-Diffusion Training Stabilizes the Per-Block Top-k

In §3.2 we sketched the five-stage chain by which the block-diffusion training objective induces the All-[MASK] anchoring property. Here we provide the full derivation and per-stage measurements, including the loss decomposition into per-token KL divergence and the Jensen bound on the block-average query gap.

We trace the All-[MASK] anchoring stability to the block-diffusion training procedure itself: the prefix is kept as clean tokens while the current block is corrupted at a random noise level, yielding a mask pattern $\mathbf{m}$, and the model is trained for every $\mathbf{m}$ to predict the ground-truth token at each masked position. Since the prefix is identical, the only difference seen by a [MASK] token under two mask patterns $\mathbf{m}^*$ and $\mathbf{m}$ is whether each surrounding position is itself a [MASK] or a revealed token. Yet training pushes the predictions under both patterns toward the same ground-truth target, forcing the activations of [MASK] and revealed positions to align across noise levels. This pressure propagates from hidden states to queries and ultimately to the per-block top-k subset; we verify the stages (**S1**: prediction → **S2**: hidden state → **S3**–**S4**: query → **S5**: top-k) against AR pre-training backbones in Table 2.

(Stage 1) Cross-noise-level prediction agreement. The block-diffusion loss averages cross-entropy over the Bernoulli mask schedule:

$$\mathcal{L}_{\text{block}}(\theta) \;=\; \mathbb{E}_{\substack{t\sim\mathcal{U}(0,1)\\ \mathbf{m}\sim\text{Bern}(t)^B}}\Big[-\sum_{j:\,m_j=1}\log p_\theta\big(x_j \mid \mathbf{m},\,\text{prefix}\big)\Big]. \tag{4}$$

At each [MASK] token j, this is $D_{\text{KL}}(\delta_{x_j}\,\|\,p_\theta(\cdot|\mathbf{m},\text{prefix}))$, minimized for every $\mathbf{m}$ independently—so predictions at different noise levels collapse onto δ_{x_j} and thus onto each other ($D_{\text{KL}}(p^{(\mathbf{m}^*)}\|p^{(\mathbf{m})})$ drops 43–63%, col. 1).

(Stage 2) Hidden state alignment. With the prefix KV fixed, this pressure aligns the block's hidden states across noise levels ($\|\tilde{h}^{(\mathbf{m}^*)}-\tilde{h}^{(\mathbf{m})}\|$ drops 8–19%, col. 2).

(Stage 3) Per-position query gap. Hidden-state alignment propagates linearly to queries via $q=W_Q h$. The per-position query gap $A:=\frac{1}{B}\sum_i\|\bar{\delta}_i\|$ with $\bar{\delta}_i:=\mathbb{E}_{\mathbf{m}}[q_i^{(\mathbf{m}^*)}-q_i^{(\mathbf{m})}]$ shrinks 13–27% (col. 3).

(Stage 4) Block-average query gap. Block averaging absorbs token-specific variance, keeping the gap on $\bar{q}^{(t)}:=\frac{1}{B}\sum_i q_i^{(t)}$ strictly below the per-position gap (col. 4 ¡ col. 3 in every family). By Jensen's inequality,

$$\big\|\mathbb{E}_{\mathbf{m}}[\bar{q}^{(\mathbf{m}^*)}-\bar{q}^{(\mathbf{m})}]\big\| \;=\; \Big\|\tfrac{1}{B}\sum_{i=1}^{B}\bar{\delta}_i\Big\| \;\le\; \tfrac{1}{B}\sum_{i=1}^{B}\|\bar{\delta}_i\| \;=\; A.$$

(Stage 5) Per-block top-k stability. The stabilized block-average query, against the fixed prefix keys, stabilizes the per-block top-k subset (recall at k=512 rises 5–11 points, col. 5)—establishing $\mathbf{m}^*$ as a valid oracle anchor for the entire trajectory.

Table 2. Empirical verification of the five-stage alignment on LongBench, comparing block-diffusion-trained models against their AR pre-training backbones: cross-noise-level KL divergence of predictions at masked positions (Stage 1), normalized L2 gap of hidden states (Stage 2), per-position and block-average query L2 gaps (Stages 3–4), and top-k recall (Stage 5). All L2 gaps use unit-normalized vectors. Percentages are relative change from backbone to trained.

		Logit (Stage 1)	Hidden state (Stage 2)	Query $\|\tilde{q}^{(\mathbf{m}^*)}-\tilde{q}^{(\mathbf{m})}\|$ (Stages 3–4)		Top-k Recall (Stage 5)
Family	Model	$D_{\text{KL}}(p^{(\mathbf{m}^*)}\,\|\,p^{(\mathbf{m})})\downarrow$	$\|\tilde{h}^{(\mathbf{m}^*)}-\tilde{h}^{(\mathbf{m})}\|\downarrow$	per-position ↓	block-avg ↓	k=512 ↑
SDAR	Qwen3-8B *(backbone)*	2.40	0.57	0.38	0.28	0.808
	SDAR-8B-Chat *(trained)*	1.38 (–43%)	0.53 (–8%)	0.34 (–13%)	0.22 (–23%)	0.852 (+5%)
Fast-dLLM v2	Qwen2.5-7B *(backbone)*	5.66	0.94	0.38	0.27	0.753
	Fast-dLLM v2 7B *(trained)*	2.08 (–63%)	0.78 (–17%)	0.28 (–27%)	0.18 (–35%)	0.826 (+10%)
LLaDA2	Ling-mini-base-2.0 *(backbone)*	2.61	0.70	0.45	0.30	0.769
	LLaDA2.0-mini *(trained)*	0.99 (–62%)	0.57 (–19%)	0.36 (–20%)	0.24 (–20%)	0.851 (+11%)

B. NIAH Scores Across Models

Table 3 reports per-context-length NIAH retrieval scores at 8K and 32K context for Exact Attention and the three sparse methods (MAGE, Quest, SparseD) at $k\in\{256,512,1024\}$ across all three block-diffusion model families. Each cell is the

mean retrieval score over the 21 needle-depth grid at that context length. Bold marks the best sparse method (MAGE/ Quest / SparseD) per column within each model. Fig. 4 in the main text visualizes the underlying (context, depth) cells for LLaDA 2.0 mini at k=256.

Table 3. Needle-in-a-Haystack accuracy (%) at 8K and 32K context, averaged over depth positions, for sparse budgets $k \in \{256, 512, 1024\}$ across three block-diffusion models. Bold = best sparse method (MAGE / Quest / SparseD) per column within each model.

	SDAR-8B-Chat						*LLaDA 2.0 mini*						*Fast-dLLM v2 7B*					
	8K			32K			8K			32K			8K			32K		
Method	256	512	1024	256	512	1024	256	512	1024	256	512	1024	256	512	1024	256	512	1024
Exact	100	100	100	57	57	57	98	98	98	90	90	90	100	100	100	46	46	46
MAGE	**100**	**100**	**100**	**76**	**71**	**67**	**97**	**98**	98	**62**	68	**76**	89	94	**100**	**33**	**37**	46
Quest	90	100	100	16	16	16	41	48	60	57	**70**	75	**97**	98	100	22	35	**52**
SparseD	68	100	100	10	38	49	17	68	**100**	25	65	70	56	**100**	100	13	27	48

C. End-to-end Speedup at Shorter Block-step Budgets

The headline speedups in Fig. 5 use n=32 steps/block; Fig. 7 extends this to shorter budgets (n=24, 16, 8) on the same three model families. MAGE remains the fastest method on every (model, context) cell, with the absolute speedup tightening slightly at n=8 as its first-step dense pass amortizes over fewer sparse steps.

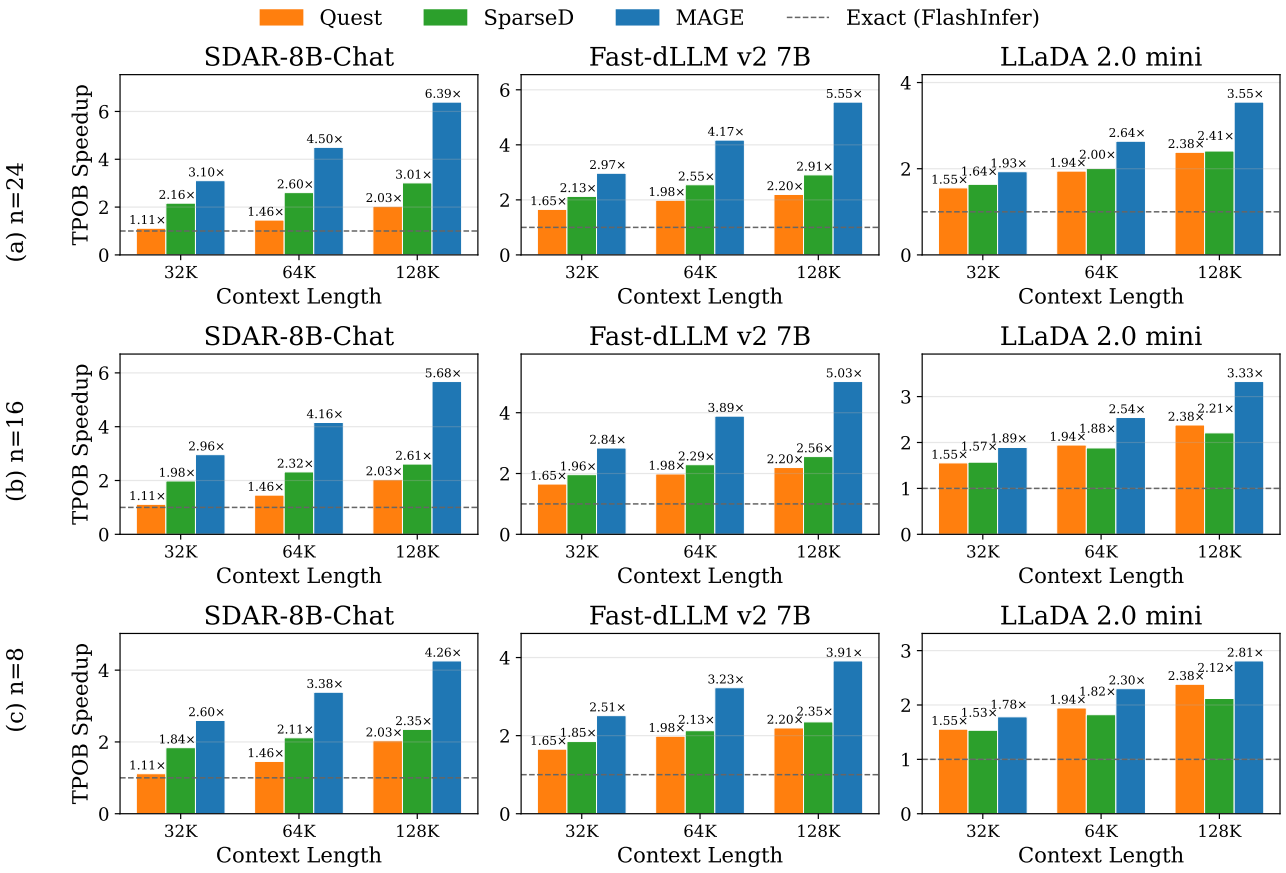

Figure 7. End-to-end TPOB speedup over Exact Attention (FlashInfer) at (a) n=24 steps/block, (b) n=16 steps/block, (c) n=8 steps/block. budget k=1024.