MAC-Attention: a Match–Amend–Complete Scheme for Fast and Accurate Attention Computation

Jinghan Yao [1 2] **Sam Adé Jacobs** [2] **Walid Krichene** [2] **Masahiro Tanaka** [3] **Dhabaleswar K Panda** [1]

Abstract

Long-context decoding in LLMs is *IO-bound*: each token re-reads an ever-growing KV cache. Prior accelerations cut bytes via *compression* (lowering fidelity) or *selection/eviction* (restricting what remains accessible), which can degrade delayed recall and long-form generation. We introduce **MAC-Attention**, a *fidelity and access-preserving* alternative that accelerates decode by *reusing prior attention computations* for *semantically similar* recent queries. It starts with a *match* stage that performs pre-RoPE L2 matching over a short local window; an *amend* stage rectifies the reused attention by recomputing a small band near the match boundary; and a *complete* stage fuses the rectified results with a fresh attention computed on the KV tail, via a numerically stable merge. On a match hit, the compute and bandwidth complexity is *constant regardless of the context length*. The method is model-agnostic, and composes with IO-aware kernels, paged-KV managers, and MQA/GQA. Across LongBench v2 (120K), RULER (120K), and LongGenBench (16K continuous generation), compared to latest FlashInfer library, MAC-Attention reduces KV accesses by up to **99%**, cuts token generation latency by over **60%** at 128K, and achieves over **14.3**× attention-phase speedups (up to **2.6**× end-to-end), while maintaining full-attention quality. By reusing computation, MAC-Attention delivers long-context inference that is both *fast* and *faithful*. Code is available here.

1 Introduction

Large Language Models (LLMs) now operate over tens to hundreds of thousands of tokens, enabling long-document understanding, multi-turn dialogue, codebase analysis, and scientific literature processing at unprecedented scales. Despite impressive kernel and runtime engineering, long-context *decoding* remains dominated by two factors: repeatedly streaming an ever-growing Key–Value (KV) cache from high-bandwidth memory (HBM) and reducing attention over long prefixes at every generated step. IO-aware attention kernels (Dao, 2023; Dao et al., 2022; Ye et al., 2025) and memory managers such as vLLM's PagedAttention (Kwon et al., 2023) reduce waste and improve locality; nevertheless, the cost of re-reading and processing large KV regions continues to be a primary latency bottleneck at long lengths.

Two families of approaches mitigate this IO bottleneck. The first compresses KV states (e.g., low-rank projection or quantization) to reduce footprint and bandwidth (Chang et al., 2024; Zhang et al., 2024). The second selects or evicts tokens/pages (often adaptively) to shrink the effective context (Feng et al., 2024; Ge et al., 2023; Li et al., 2024; Liu et al., 2023; Tang et al., 2024; Xiao et al., 2024; Zhang et al., 2023). Both are effective at lowering bytes moved, but they introduce approximation errors either by

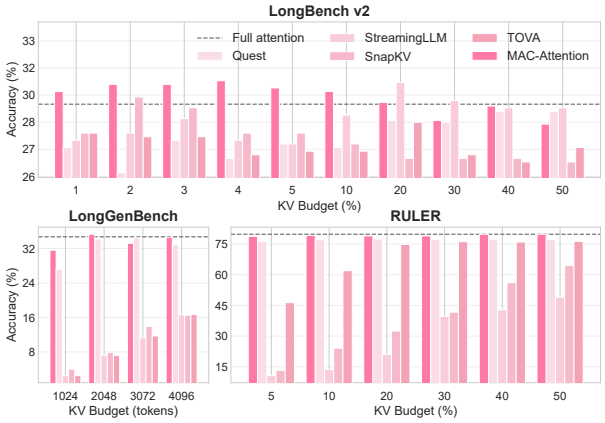

Figure 1. **Accuracy vs. KV budget (the fraction of KV cache used in each decoding step) across long-context benchmarks.** Top: *LongBench v2* (up to 120K context) (Bai et al., 2024). Bottom left: *LongGenBench* (up to 16K continuous generation) (Wu et al., 2024). Bottom right: *RULER* (120K context) (Hsieh et al., 2024). *MAC-Attention* is highlighted (dark pink); *Full attention* shown as a gray dashed line.

[1]The Ohio State University [2]Microsoft, WA, USA [3]Anyscale, CA, USA. Correspondence to: Jinghan Yao <yjhmitweb@gmail.com>.

Proceedings of the 9th MLSys Conference, Bellevue, WA, USA, 2026.

constraining the range of tokens the model can attend to (selection/eviction), or by reducing fidelity (compression). As a result, performance on tasks with delayed recall, cross-document linking, long generation/reasoning, or distributed evidence can degrade (Hsieh et al., 2024), as shown in Fig. 1.

This paper introduces **MAC-Attention (MAC)**—a training-free, model-agnostic mechanism that accelerates long-context decoding by *reusing prior attention computation* for *semantically similar* recent queries while preserving access to the full sequence, and achieving high fidelity. MAC maintains two short-horizon ring buffers per request: a pre-positional (pre-RoPE) *query ring* and a corresponding *rectified attention-summary ring*. When a new query matches one in the window, MAC reuses the cached attention-summary of the prefix, computes the contribution of the tail, then *merges* them via an *associative, numerically stable log-domain merge* (Dao et al., 2022; Milakov & Gimelshein, 2018). Whenever there is a match, the computation (and memory bandwidth) are *constant* in the sequence length. When there is no match, it falls back to full attention computation (linear).

How MAC Attention maintains high fidelity. There are two algorithmic improvements that are crucial to maintaining a high fidelity (i.e., a low approximation error of the attention output). The first improvement is how query matching is done. Most modern LLMs use positional encodings that express relative position via rotations or distance-dependent biases, e.g., RoPE and ALiBi (Press et al., 2021; Su et al., 2024). While it may be simpler to match in post-position (post-RoPE) space, we find that due to the effect of position encodings, distances between otherwise similar queries can become large in post-RoPE space, which significantly reduces close matches. Instead, MAC *matches in pre-RoPE space*, which increases the matching rate and results in matches that are semantically close (regardless of position). The second improvement is the observation that tokens that are near the match position account for a significant fraction of the approximation error introduced by reusing the cached output. This is because the softmax probability mass typically concentrates near the decoding cursor (Press et al., 2021; Su et al., 2024). Thus MAC *rectifies the cached output by recomputing a short band* (of width r) around the match, significantly reducing approximation errors, while only paying a constant $O(r)$ cost per-head.

Relation to other reuse paradigms. Beyond compression/selection, prior work reuses structure across requests (e.g., prefix caching or tree-structured decoding) (Gim et al., 2024; Yao et al., 2025) or alternates full and partial steps by recycling top-k attention from previous steps (Xu et al., 2024). MAC is orthogonal: it performs *in-stream semantic reuse with local rectification*, independent of literal prefix overlap or alternating step schedules, and preserves access to the full context.

Results at a glance. Across *LongBench v2* (120K) (Bai et al., 2024), *RULER* (120K) (Hsieh et al., 2024), and *LongGenBench* (16K continuous generation) (Wu et al., 2024), MAC *reduces KV accesses by up to 99%*, cuts *per-token latency by over 60% at 128K*, achieves $\geq 14.3\times$ attention-phase speedups (up to $\sim 46\times$ under 256K context settings), and delivers *up to* $2.6\times$ end-to-end generation speedups on LLaMA—*while maintaining full-attention quality*.

Contributions are as follows:

- We propose *Match–Amend–Complete (MAC) Attention*, a fidelity and access-preserving reuse scheme that *matches* pre-RoPE queries using an L2, dimension-aware threshold, *amends* a short high-mass band near the reuse boundary, and *completes* with an associative, numerically stable *log-domain merge*.
- MAC is *training-free*, *model-agnostic*, and *drop-in* for high-performance inference stacks, composing with IO-aware attention (Dao, 2023; Dao et al., 2022; Ye et al., 2025), optimized decode paths (Hong et al., 2024), paged KV (Kwon et al., 2023), and MQA/GQA (Ainslie et al., 2023; Shazeer, 2019).
- Across LongBench v2, RULER, and LongGenBench (Bai et al., 2024; Hsieh et al., 2024; Wu et al., 2024), MAC *reduces KV accesses by up to 99%*, achieves $\geq 14.3\times$ (up to $\sim 46\times$) attention-phase speedups and *up to* $2.6\times$ end-to-end speedups on LLaMA, while maintaining full-attention quality.

2 Background

Long-context decode Even with I/O-aware kernels, the dominant cost at long context decoding remains memory traffic: streaming large KV regions from high-bandwidth memory (HBM) (Dao et al., 2022; 2023; Hong et al., 2024). To lower bytes moved, existing work either (i) *compresses* KV states (e.g., by projection or quantization), or (ii) *selects/evicts* tokens or pages to shrink the effective context (Ge et al., 2024; Hooper et al., 2024; Liu et al., 2024; 2023; Xiao et al., 2024; Zhang et al., 2023). Both strategies can substantially cut I/O, but they also change either how information is represented (compression) or which tokens remain accessible (selection/eviction).

Exploiting temporal redundancy This paper explores a third path, rooted in the observation that computation within a single decoding stream often exhibits high **temporal**

redundancy (Chen et al., 2024; Liu et al., 2023; Xu et al., 2024). In many generative tasks, such as multi-turn dialogue, code generation, or long reasoning, the model's internal state-represented by the query vector at each step-is not drawn from a uniform distribution. Instead, queries often perform repetitive information-retrieving patterns. This redundancy presents a clear opportunity for reuse. The attention distribution of two similar queries induced over the shared prefix is likely to be highly correlated. Rather than re-streaming attention over the shared prefix, reusing it becomes a clear and intuitive path.

Notation Let Q_t, K_t, V_t denote, respectively, the query, key and value encodings for token position t (our discussion applies to any attention block, so we omit indexing of layers or heads for notational simplicity). We also denote by $\tilde{Q}_t$ the query encodings prior to applying any positional encodings (in short, the per-RoPE query). We will denote by m the position of the current token, for which we seek to compute the attention output. Define the logits vector $\ell_t^{(m)} = \frac{1}{\sqrt{d}} Q_m^\top K_t$. The attention output is then defined as follows: $o_m = S^{(m)}/Z^{(m)}$ where $S^{(m)}$ is the weighted sum $S^{(m)} = \sum_{t\le m} e^{\ell_t^{(m)}} V_t$, and $Z^{(m)}$ is the normalizing constant $Z^{(m)} = \sum_{t\le m} e^{\ell_t^{(m)}}$.

3 MAC-Attention: Design

3.1 Design overview

Our design starts with the following simple observation: that the attention computation can be decomposed, at any position p, into a prefix and suffix computations. For a set $\mathcal{I}$, Define the *attention summary*

$$AS_{\mathcal{I}}^{(m)} = \left(S_{\mathcal{I}}^{(m)},\ Z_{\mathcal{I}}^{(m)}\right) = \left(\sum_{t\in\mathcal{I}} e^{\ell_t^{(m)}} v_t,\ \sum_{t\in\mathcal{I}} e^{\ell_t^{(m)}}\right).$$

The exact output is $o_m = S_{1:m}^{(m)}/Z_{1:m}^{(m)}$. Summaries over disjoint sets merge by addition. For any position p, attention computation can be decomposed into $o_m = \frac{S_{1:p}^{(m)} + S_{p+1:m}^{(m)}}{Z_{1:p}^{(m)} + Z_{p+1:m}^{(m)}}$. This decomposition is numerically stable and associative, and the special case $p = m-1$ is used in popular methods such as Flash Attention (Dao et al., 2022; Milakov & Gimelshein, 2018).

The starting point of the design is to attempt to *cache* the results of similar past computations and use them in the decomposition above. Specifically, we will attempt to reuse the *prefixes* $S_{1:p}^{(m)}$ and $Z_{1:p}^{(m)}$, and approximate them by $S_{1:p}^{(p)}, Z_{1:p}^{(p)}$. These quantities are similar but not identical: the first is computed based on the logits $\ell_t^{(m)} = K_t^\top Q_m, t \le p$ while the second uses $\ell_t^{(p)} = K_t^\top Q_p, t \le p$ (they share the keys but differ in the queries). However, notice that if the current query Q_m is close to Q_p, their logits are also close, and $S_{1:p}^{(p)}$ can be a good approximation of the true $S_{1:p}^{(m)}$ (and similarly for Z). This forms the basis of our method.

The next crucial observation is that in order to further reduce approximation errors, we can discard and *recompute* a very small part of the prefix (the interval near the match position $[p-r, p]$, which in practice accounts for most of approximation error for reasons we shall discuss). More precisely, we use the *truncated* prefix $S_{1:p-r}^{(p)}$ as an approximation of $S_{1:p-r}^{(m)}$, and freshly compute $S_{p-r+1:m}^{(m)}$. This only slightly increases the KV access (by a constant r) while significantly reducing approximation errors.

Figure 2 summarizes the method: MAC-Attention keeps two additional caches: a past query cache (for Q_t) and a past attention result cache (for the truncated/rectified attention summaries S, Z). In decoding step m, we first perform a past-query *match*; if a matching query Q_p is found ($p < m$), we reuse the corresponding summaries, which carry context information up to $p-r$. Then, we *complete* the result by computing the suffixes $S_{p-r+1:m}^{(m)}, Z_{p-r+1:m}^{(m)}$ and online-merge to obtain the final output.

In the remainder of this section, we discuss design choices and additional details for each step.

3.2 Match: where and how to compare

In this part, we discuss details of the match strategy.

3.2.1 *Pre vs. post position match*

A first question is at what stage should the query match be performed. Indeed, in most recent models, positional encodings are applied to the query after QKV projection, and a natural question is whether the match should occur before or after the positional encodings. Let $\tilde{Q}_t \in \mathbb{R}^d$ be the *pre-positional* query at position t, and let $Q_t = R(t)\tilde{Q}_t$ be the RoPE-transformed query with $R(\cdot)$ orthogonal (Su et al., 2024).

From the perspective of the operation order (*QKV projection* $\to$ RoPE $\to$ *attention*), post-RoPE matching seems more natural in principle because attention consumes the post-position vectors. Writing the logits at position m as $\ell_t^{(m)} = \frac{1}{\sqrt{d}} Q_m^\top K_t$ with $Q_m = R(m)\tilde{Q}_m$ and $K_t = R(t)\tilde{K}_t$ (Vaswani et al., 2017), Cauchy–Schwarz yields, for any key K_t, $\left|\ell_t^{(m)} - \ell_t^{(p)}\right| = \frac{\left|(Q_m - Q_p)^\top K_t\right|}{\sqrt{d}} \le \frac{\|Q_m - Q_p\|\,\|K_t\|}{\sqrt{d}}$. Thus a smaller post-RoPE distance directly bounds the *worst-case* logit error across all keys.

In practice, however, due to the relative rotation $R(m-p)$, as $|m-p|$ grows, the post-RoPE distance between similar queries tends to increase, *reducing match success*. Indeed, matching in post-RoPE space injects a *relative* phase

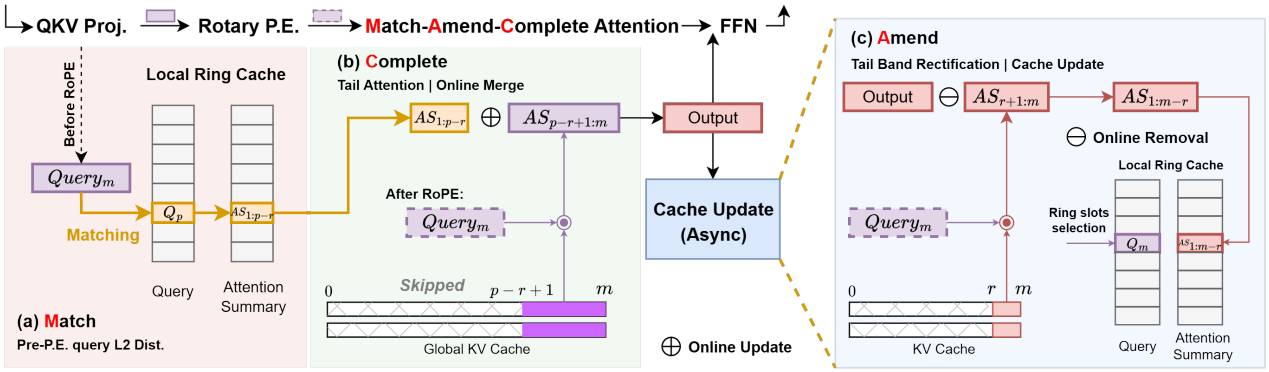

Figure 2. **MAC-Attention: Match–Amend–Complete with async cache update. (a) Match:** at position m, compare the *pre-RoPE* query $\tilde{Q}_m$ against a small ring of recent queries; if the nearest match p passes an L2 threshold, fetch its attention summary $\mathrm{AS}_{1:p-r}$ (otherwise run full attention). **(b) Complete (critical path):** with the *post-RoPE* query q_m, compute attention only on the band+tail $[\,p{-}r{+}1,\ m\,]$ and log-domain merge $\mathrm{AS}_{1:p-r} \oplus \mathrm{AS}_{p-r+1:m}$; the KV $[1, p{-}r]$ is not accessed. **(c) Amend (async):** compute a rectification term $\mathrm{AS}_{r+1:m}$ and update the cache via online subtraction to obtain AS_{m-r}; insert $\tilde{q}_m, \mathrm{AS}_{m-r}$ into the rings. Symbols: p—match position, r—band width; $\oplus$—log-domain merge, $\ominus$—log-domain removal. Shaded regions denote KV segments not re-read under reuse.

$R(m{-}p)$ into the comparison. For $m > p$,

$$\|Q_m - Q_p\|^2 \;=\; \left\|\tilde{Q}_m - R(m-p)\tilde{Q}_p\right\|^2. \qquad (1)$$

Even if $\tilde{Q}_m \approx \tilde{Q}_p$, the cross-term $-2\,\tilde{Q}_m^\top R(m{-}p)\tilde{Q}_p$ diminishes as $\Delta = |m{-}p|$ grows because RoPE is block-diagonal in $2{\times}2$ rotations:

$$R(\Delta) = \mathrm{diag}\big(R_{\omega_1}(\Delta), \ldots, R_{\omega_{d/2}}(\Delta)\big),$$
$$x^\top R(\Delta)x \;=\; \sum_{j=1}^{d/2} \cos(\omega_j \Delta)\, \|x_{(j)}\|^2, \qquad (2)$$

with $x_{(j)} \in \mathbb{R}^2$ the j-th frequency pair. When energy is distributed across frequencies, the weighted average

$$\mathrm{avg\_cos}(\Delta) \;\equiv\; \frac{\sum_j \cos(\omega_j \Delta)\, \|x_{(j)}\|^2}{\|x\|^2}$$

becomes small for moderate Δ due to destructive phase mixing, and $\|x - R(\Delta)x\|^2 = 2\|x\|^2\big(1 - \mathrm{avg\_cos}(\Delta)\big)$ approaches $2\|x\|^2$. Consequently, $\|Q_m{-}Q_p\|$ tendsd to increase with the gap $|m{-}p|$, potentially reducing matches even for semantically similar pairs.

Our design, therefore, matches *pre*-RoPE to measure semantic similarity without positional phase (Fig. 2 a), then performs a short, local *rectification* near the reuse boundary to recompute the logits that carry most softmax mass under the true Q_m (see Section 3.4). This combination recovers the practical fidelity one would expect from post-RoPE matching while maintaining robust match frequency. If no match is found, fall back to full attention; outputs are identical to baseline.

3.2.2 *L2 vs. cosine match*

A second design choice is the similarity measure. Choosing the right measure is important to reduce approximation errors. Two natural choices are cosine and dot-product. Cosine similarity captures only *direction*; two colinear queries with different norms have the same cosine yet induce different logit scales and therefore different softmax sharpness. In contrast, the Euclidean distance controls both angle and magnitude. This matters for fidelity as shown above, hence minimizing $\|Q_m - Q_p\|$ uniformly bounds the worst-case logit approximation error.

For isotropic pre-RoPE queries, a simple Gaussian model $\tilde{Q}_m - \tilde{Q}_p \sim \mathcal{N}(0, 2I_d)$ implies $\|\tilde{Q}_m - \tilde{Q}_p\|^2 \overset{d}{=} 2\,\chi^2_d$, so $\|\tilde{Q}_m - \tilde{Q}_p\|$ concentrates near $\sqrt{2d}$. We therefore accept a match when the L2 distance falls *proportionally* below this baseline:

$$\|\tilde{Q}_m - \tilde{Q}_p\| \;<\; \sqrt{2d}\,(1-\tau), \qquad \tau \in [0, 1), \qquad (3)$$

The parameter τ defines a threshold relative to random coincidence, and is set to a fixed value across heads.

3.2.3 *Local window and search cost*

The candidate set is restricted to a short, recent window of size K per (grouped) head. There are several reasons for this: First, reuse is most valuable when the matched prefix is long (i.e. when the match position is near the current token). Second, this allows us to cache only a constant number of attention results, and match against a constant number of past queries (making the memory and compute cost of matching *constant, rather than linear in the sequence length*). Last, similar queries are more likely to occur within a short horizon, so restricting to the last K tokens does not significantly impact the match rate.

The matcher is a single streaming pass over the window: for each candidate $\tilde{Q}_p$ it accumulates two inner products to form the squared distance, $\|\tilde{Q}_m - \tilde{Q}_p\|^2 = \|\tilde{Q}_m\|^2 + \|\tilde{Q}_p\|^2 - 2\,\tilde{Q}_m^\top \tilde{Q}_p$, performing bf16 reads with fp32 accumulation. Arithmetic intensity is low, so the kernel is memory-bound but lightweight; tiling over the ring and a warp-only reducer ensures sufficient waves to hide latency (see §4).

3.3 Complete: attention summaries and online merge

As shown in Fig 2 b, suppose the matcher selects index $p < m$ and we have cached $AS^{(p)}_{1:p-r}$. If we could replace $AS^{(m)}_{1:p-r}$ by $AS^{(p)}_{1:p-r}$ without loss, then

$$o_m \;=\; \frac{S^{(m)}_{1:p-r} + S^{(m)}_{p-r+1:m}}{Z^{(m)}_{1:p-r} + Z^{(m)}_{p-r+1:m}} \;\approx\; \frac{S^{(p)}_{1:p-r} + S^{(m)}_{p-r+1:m}}{Z^{(p)}_{1:p-r} + Z^{(m)}_{p-r+1:m}}. \tag{4}$$

Equation (4) is the essence of *complete*: once a prefix is summarized, fusing it with a freshly computed tail is constant-time in the size of the tail.

3.4 Amend: why rectification is necessary, and how to do it

3.4.1 Where naive reuse errs

To understand why rectification is important, consider the naive reuse strategy, that directly replaces the full prefix $AS^{(m)}_{1:p}$ with $AS^{(p)}_{1:p}$. Let $\delta_t \equiv \ell^{(m)}_t - \ell^{(p)}_t$ for $t \le p$. Then $S^{(m)}_{1:p} = \sum_{t\le p} e^{\ell^{(p)}_t + \delta_t} v_t$ and $Z^{(m)}_{1:p} = \sum_{t\le p} e^{\ell^{(p)}_t + \delta_t}$. Using the (full) cached prefix corresponds to setting $\delta_t = 0$. A first-order expansion of softmax gives

$$\Delta\alpha_t \;\approx\; \alpha^{(p)}_t \Big(\delta_t - \mathbb{E}_{u\sim\alpha^{(p)}}[\delta_u]\Big), \tag{5}$$

so the approximation error at position k increases with $\alpha^{(p)}_k$. In practice, softmax weights are typically larger for *recent* tokens near p due to semantic similarity and recency bias from position encodings (Press et al., 2021; Su et al., 2024). Thus, the approximation error is typically dominated by local tokens.

3.4.2 Rectification as removing a short high-mass band

Let $\mathcal{R} = \{p - r + 1, \dots, p\}$ be a short band capturing most prefix mass, with $\rho = \sum_{t\in\mathcal{R}} \alpha^{(p)}_t$. Define the *rectified* cached prefix by *removing* this band at position p: $AS^{(p)}_{1:p-r}$.

At position m, we recompute only $[p - r + 1, m]$ *under the new query*, and merge with the cached summary:

$$S^{(m)}_{p-r+1:m} = \sum_{t=p-r+1}^{m} e^{\ell^{(m)}_t} v_t, \quad Z^{(m)}_{p-r+1:m} = \sum_{t=p-r+1}^{m} e^{\ell^{(m)}_t}. \tag{6}$$

Then the final output is

$$o_m \;=\; \frac{S^{(p)}_{1:p-r} + S^{(m)}_{p-r+1:m}}{Z^{(p)}_{1:p-r} + Z^{(m)}_{p-r+1:m}}. \tag{7}$$

Removing and recomputing only the short high-mass band significantly reduces approximation errors, while keeping a constant $O(r)$ cost per head (independent of context length).

3.4.3 Error decays with residual mass outside the band

Using a Lipschitz bound for exponentials and (5), one can show that the prefix error after rectification scales as

$$\big\|o^{\text{reuse}}_{1:p} - o^{(m)}_{1:p}\big\| \;\lesssim\; (e^{\Delta} - 1)\,(1 - \rho)\cdot \mathbb{E}_{t\sim\alpha^{(p)}_{\bar{\mathcal{R}}}}\big[\|v_t\|\big], \tag{8}$$

so choosing r to capture most mass ($\rho \uparrow$) drives the error down quickly. In practice, a fixed small r per head works well across lengths, as shown in Fig. 2 c, we *cache rectified summaries* at creation time (position p), making reuse a constant-time merge at position m via (7).

3.5 Putting it together: match + amend + complete

Per query, per head, and per layer:

1. **Match.** Search the recent window of K pre-RoPE queries using squared L2 and accept if (3) holds.
2. **Amend.** Load the *rectified* prefix summary $(S^{(p)}_{1:p-r}, Z^{(p)}_{1:p-r})$ and recompute only the band $[p-r+1, p]$ and tail $(p, m]$ under the current query.
3. **Complete.** Merge via the online identity (7).

We cache (i) the last K pre-RoPE queries and (ii) the corresponding K *rectified* prefix summary; Alone with the existing KV cache. In practice, $K \le 1024$, introducing a negligible memory overhead in long context scenarios.

3.6 Economics: when does MAC reduce memory bandwidth?

Let B_{KV} be bytes read per cached token in attention and B_q bytes read per candidate query during matching. If the match is at position p with rectification width r and we search K candidates, a coarse break-even condition (in terms of memory traffic) is

$$\underbrace{p\,B_{\text{KV}}}_{\text{prefix avoided}} \;\gtrsim\; \underbrace{K\,B_q}_{\text{match cost}} \;+\; \underbrace{r\,B_{\text{KV}}}_{\text{band+tail recompute}} \tag{9}$$

In practice, we will keep $K \le 1024$ and $r \le 512$ so that MAC-Attention can skip a substantial amount of KV access while also maintain a full attention fidelity. We provide a thorough analysis in §A.2.

3.7 Practical notes and compatibility

Short-horizon rings. We maintain per-request size K ring buffers for queries and rectified summaries. The ring capacity bounds memory and ensures $O(1)$ insertion.

KV sharing. MQA/GQA (Ainslie et al., 2023; Shazeer, 2019) reduce the number of KV heads. MAC-Attention matches and reuses at the query granularity as attention is performed per-query head, thus the matching kernel is more memory-bound then the attention kernel (See Figure 8(a)).

4 System Design

MAC-Attention targets that decoding IO-bound challenge directly by (i) matching to reuse a long prefix, (ii) amending only a short high-mass band, and (iii) completing with an online merge. The system must therefore (1) keep the matching and amendment overhead below the saved KV traffic, (2) preserve numerical precision during multiple merges, and (3) integrate cleanly with IO-aware attention and paged KV managers without perturbing batching or memory layout.

4.1 State and data layout

We maintain two short-horizon, per-request ring buffers, each with a capacity of K tokens. The *query ring* stores pre-RoPE queries from the most recent K tokens for each request. The *attention-summary ring* preserves the rectified prefix summaries $(S_p^{\text{rect}}, Z_p^{\text{rect}})$. Both rings are indexed by the request's global length in modulo K order, so the recent window is contiguous and insertion is $O(1)$. The scheduler maps multiple query heads to the corresponding KV head without changing the underlying layout.

4.2 Per-step micro-pipeline

In each decode step, MAC-Attention executes three kernels with minimal synchronization:

K1. Match. A tiled, L2 nearest-neighbor scan compares each active query against the local window in its request's query ring (pre-RoPE) as shown in Fig. 3 (a). We emit per-tile minima and run a warp-only final reducer to choose *at most* one reuse point p per (request, head), then apply the dimension-aware threshold from Eq. (3). Implementation choices are geared to IO efficiency. The reducer is shuffle-based to eliminate barrier stalls. Compared to standard attention kernels with MQA/GQA, the match kernel is intrinsically more memory-bound as it needs to stream all query heads.

K2. Amend & complete. *(1) Workload shaping and load balance:* Because each head may reuse a different prefix, the band+tail span varies per head. We therefore flatten the query–head axis and allocate CTAs proportionally to workloads (Fig. 3 b), giving longer spans more thread blocks. *(2) Attention computation:* Given a hit, compute attention only over the band+tail under the current query with the effective KV. *(3) Merge:* Load the rectified prefix summary from the attention-summary ring and merge in place.

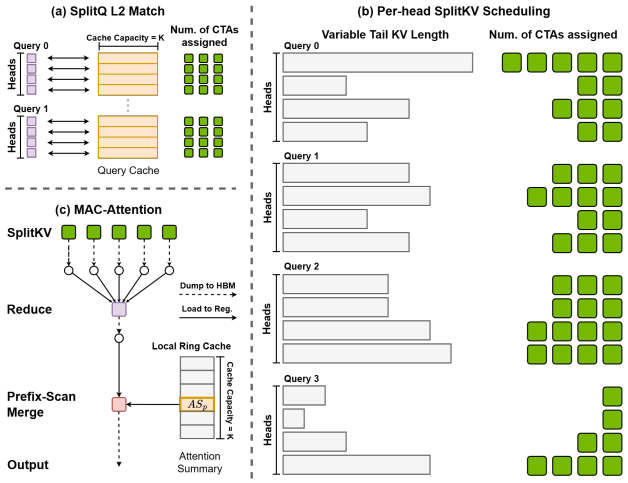

Figure 3. **Decode micro-pipeline for MAC-Attention.** *(a):* per-request rings L2 match with SplitQ design; *(b):* per-head work spans differ because the reuse point p and band size r vary, the schedule assigns more CTAs to longer band+tail spans as shown in green blocks; many heads do a cheap merge when reuse is strong; *(c):* Overview of MAC-Attention workflow.

K3. Rectify–append (ultra-fast, off-critical-path). To prepare future reuse, we *construct the next rectified summary* for the just-produced token and append it, together with the pre-RoPE query, to the rings. K3 writes three rows: the query, the rectified attention row, and the ln-LSE scalar. Crucially, K3 *runs on an auxiliary stream* with a single event dependency—the result is not needed until the next step visits the same layer. Crucially, as we implemented K3 with fully in-place operations, it also can run within the main stream as its latency is negligible.

The overall kernel execution order and HBM traffic within the attention operation is shown in Fig. 3 (c). MAC-Attention composes with IO-aware kernels (Dao, 2023; Dao et al., 2022; Ye et al., 2025), optimized decode paths (Hong et al., 2024), and KV virtualization (Kwon et al., 2023).

5 Evaluation

We evaluate whether MAC-Attention (i) preserves task quality relative to *full attention*, (ii) delivers end-to-end efficiency gains *in decode* with real serving frameworks, and (iii) derives its gains from its intended mechanisms (pre-RoPE matching, L2 metric, local-band rectification).

Table 1. **Main accuracy results across models and benchmarks.** Accuracy (%) on LongBench v2 (up to 120K context), LongGenBench (up to 16K continuous generation), and RULER (120k context). Rows compare *Full attention*, *Quest* variants (config / shows the budget settings for each benchmark; *Quest* only supports LLaMA models.), and our *MAC-Attention*. The $KV^{\downarrow}_{\%}$ columns report average KV-budget reduction relative to full attention (higher is more savings); – denotes not applicable. LongGenBench is generation-heavy; most models cannot complete the required task; we therefore report *Finish%* (higher is better), and finish-weighted accuracy.

		LongBench v2					LongGenBench			RULER	
Model	Config.	Overall	Short	Medium	Long	$KV^{\downarrow}_{\%}$	Finish %	Acc.	$KV^{\downarrow}_{\%}$	Acc.	$KV^{\downarrow}_{\%}$
LLaMA-3.1-8B	Full attention	29.0	35.0	26.0	25.0	0	93.1	34.7	0	79.8	0
(Grattafiori et al., 2024)	Quest-8K/4K/8K	28.0	33.9	24.2	25.9	85	92.8	32.9	35	77.0	93
	Quest-4K/3K/4K	27.4	33.3	24.7	23.1	92	91.6	34.8	48	75.5	97
	Quest-2K	27.8	32.2	26.0	24.1	96	89.9	33.8	63	73.4	98
	Quest-1K	26.2	33.3	22.3	22.2	98	83.6	27.2	80	72.4	99
	MAC-Attention	30.2	34.4	27.4	28.7	99	90.0	38.2	80	78.8	95
Phi-4-Mini	Full attention	28.0	32.8	23.3	29.6	0	–	–	–	74.4	0
(Abouelenin et al., 2025)	**MAC-Attention**	29.8	33.9	25.1	32.4	91	–	–	–	73.1	77
GLM-4-32B	Full attention	–	–	–	–	–	69.8	29.7	0	–	–
(GLM et al., 2024)	**MAC-Attention**	–	–	–	–	–	66.1	30.1	70	–	–
LLaMA-3.1-70B	Full attention	31.4	41.7	26.5	24.1	0	98.1	41.9	0	80.1	0
(Grattafiori et al., 2024)	Quest-8K/4K/8K	31.2	40.6	27.4	23.1	85	97.5	45.8	29	79.8	93
	Quest-4K/3K/4K	31.8	40.6	27.9	25.0	92	98.6	45.2	40	79.8	97
	Quest-2K	31.2	39.4	27.9	22.2	96	98.3	46.2	54	79.5	98
	Quest-1K	30.8	39.4	27.9	22.2	98	93.8	41.9	72	78.3	99
	MAC-Attention	32.2	40.6	28.8	25.0	99	98.6	43.6	75	78.0	99

GLM-4-32B has 32K context limit, Phi-4-Mini baseline generates poor-quality responses on LongGenBench, hence their results are excluded from the table.

Unless otherwise noted, MAC-Attention is enabled for *decode only*. MAC-Attention also applies to prefill and is fully functioning, but may require additional tweaking, see results and analysis in §A.3.

5.1 Experimental Setup

Models. As shown in Table 1.

Runtime. SGLang 0.4.9 with FlashInfer 0.2.7 kernels, running on NVIDIA H100 SXM5 with CUDA 12.8.1.

Besides Quest, we also compare our method to other efficient attention strategies, e.g. SnapKV, TOVA, in Fig. 1. For all experiments, we fix the random seed, set the temperature to 0, and keep only the attention path different.

5.2 Benchmark details

LongBench v2 (Bai et al., 2024) is a diverse long-context suite (up to 120K tokens in our experiment) spanning QA, summarization, and retrieval tasks; results are partitioned into *Short/Medium/Long* to evaluate length sensitivity.

RULER (Hsieh et al., 2024) provides controlled synthetic probes (fixed at 120K in our experiment) that stress long-range retrieval, delayed recall, and robustness to positional biases and length extrapolation.

LongGenBench (Wu et al., 2024) targets *continuous long-form generation* with outputs up to 16K tokens, using an LLM as judge (default LLaMA-3.3-70B). It is particularly stringent for KV-efficient methods: small perturbations in attention can accumulate over thousands of decode steps, producing cascading errors. To our knowledge, prior KV-efficiency works have rarely reported evaluations on this benchmark; our study fills this gap.

Across benchmarks, we mostly use a fixed search window $K = 1024$ with a rectification band $r = 256$, and a threshold $\tau = 0.45$ for context-heavy tasks and $\tau = 0.75$ for generation-heavy tasks. More detailed on how to choose these parameters can be found in §A.1, §A.2, and §B.

5.3 Task Fidelity Relative to Full Attention

In Table 1, we report accuracy/finish rates on these three benchmarks. Under identical decode settings, MAC-Attention *matches* (and in some cases, slightly improves) full-attention quality (we suspect it is due to some potential noise in long context computation for the baseline full attention), while achieving significant KV-budget reductions. Gains concentrate in the Medium/Long buckets of LongBench v2 and carry over to LongGenBench with comparable finish rates; the few regressions (e.g., RULER at 70B) are small.

Table 2. Overall accuracy on LongBench v2

KV %	Full Attn.	Quest	RocketKV	Multipole	MAC-Attn.
1	29.0	27.6	29.4	27.6	**30.2**
5	29.0	27.8	29.2	27.8	**30.4**
10	29.0	27.6	29.2	**30.2**	**30.2**
20	29.0	28.2	29.4	28.0	**29.6**

Table 3. End-to-end Attention Latency (µs) on 120K context

KV %	Full Attn.	Quest	RocketKV	Multipole	MAC-Attn.
1	234.2	581.2	822.8	192.4	**62.9**
5	234.2	594.7	844.7	210.8	**64.0**
10	234.2	608.5	1042.5	265.4	**78.1**
20	234.2	640.5	1855.6	324.6	**103.8**

Compared to selection. Relative to token-selection baselines (Quest), MAC-Attention typically attains equal or higher accuracy at similar or stricter KV budgets—consistent with preserving access to all tokens rather than compressing/evicting them.

5.4 Comparison on accuracy vs. speedup

We evaluate more efficient attention methods on LongBench v2, and measure their end-to-end latencies at 120K context length. **Full Attn.** denotes the FlashInfer full attention baseline. For Quest (Tang et al., 2024), RocketKV (Behnam et al., 2025), and Multipole (Hooper et al., 2025), we followed the authors' official repositories to build kernels and used their official evaluation scripts for accuracy/latency. As shown in Table 2 and Table 3, most existing efficient attention methods have significant overheads in addition to attention, thereby failing to surpass even the full attention FlashInfer baseline.

5.5 Query Hit Rate Beyond Dense Models

Table 4. Evaluation on MoE model (Qwen3-30B-A3B-Instruct)

Model / Setting	τ	K	r	Overall Acc.	Hit (%)	Skip (%)
Full attention	–	–	–	37.0	–	–
MAC-Attention	0.45	512	256	37.0	99.5	98.9
MAC-Attention	0.45	1024	256	36.6	99.6	99.0
MAC-Attention	0.45	2048	256	37.6	99.6	98.8
MAC-Attention	0.45	4096	256	36.6	99.7	98.6

Mixture-of-Experts (MoE) models introduce dynamic expert routing, therefore, we evaluate our method on the MoE model **Qwen3-30B-A3B-Instruct** using the same threshold ($\tau = 0.45$). Table 4 shows that MAC-Attention consistently achieves a $\geq 99\%$ hit ratio while maintaining comparable accuracy to full attention. This indicates that MoE expert routing does not significantly hinder effective semantic query matching in practice.

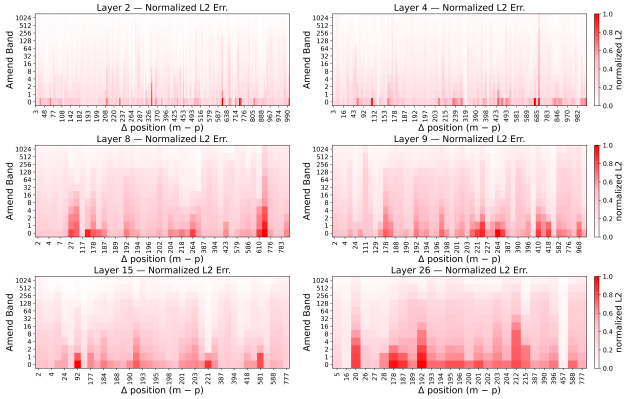

Figure 4. **Rectification error vs. reuse gap and band width.** Layerwise heatmaps show the normalized output error as a function of reuse gap (Δ) and rectification band width (r); representative layers are displayed.

Discussion. Across all window sizes, MAC-Attention maintains stable accuracy relative to full attention while achieving very high hit and skip ratios. This suggests that, despite the conditional computation introduced by MoE routing, the semantic structure exploited by our matching mechanism remains sufficiently consistent for effective reuse.

5.6 Rectification Error vs. Reuse Gap and Band Width

Setup and metric. We quantify how much *amend* reduces approximation error as a function of the rectification band width r and the reuse gap $\Delta = m - p$, where m is the current decode position and p is the matched query position selected by the pre-RoPE L2 matcher from §3.2. For each layer we compute the per-token output error

$$\mathrm{Err}(m) = \frac{\left\| o_m^{\mathrm{MAC}} - o_m^{\mathrm{full}} \right\|_2}{\left\| o_m^{\mathrm{full}} \right\|_2},$$

and average over LongGenBench sequences and over all query heads (*GQA*: 32 Q heads sharing 8 KV heads). Figure 4 visualizes the average error as a heatmap for a few representative layers of LLaMA-3.1-8B (32 layers): layers 2, 4, 8, 9, 15, and 20. The x-axis is the relative match position Δ, the y-axis is the rectification band width r, and lighter color indicates lower error.

Insights. Error decays rapidly as the band widens, $r \approx 8$ removes most discrepancy, and $r \geq 256$ is effectively indistinguishable from full attention, indicating that recomputing a short high-mass band captures the dominant discrepancy near the reuse boundary. At fixed r, larger reuse gaps are more sensitive, and deeper layers tend to require slightly wider bands—both consistent with position-induced logit drift and layerwise aggregation. Occasional outliers arise when residual attention mass extends beyond the

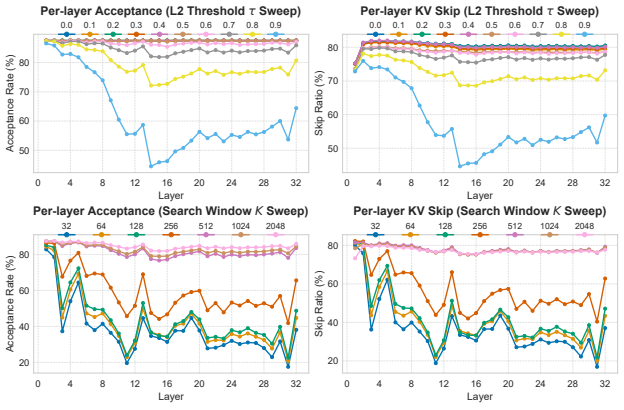

Figure 5. **Layerwise reuse patterns.** Acceptance rate (left) and skipped-prefix fraction (right) per layer. Top: threshold sweep at fixed window size. Bottom: window-size sweep at fixed threshold.

Table 5. Fine-grained attention-path latency profile (µs) at fixed match window $K = 1024$. Full attention reports FlashInfer attention latency. MAC reports the Match kernel, load balancer, attention kernel, and overall attention-path latency.

Context	KV %	Full Attn.	MAC-Attention			
		Attention	Match	Load bal.	Attention	Overall
32K	1	71.6	9.1	28.8	26.0	63.9
32K	5	71.6	9.1	28.9	25.2	63.2
32K	10	71.6	9.1	28.6	25.1	62.8
32K	20	71.6	9.1	29.5	27.3	65.9
60K	1	133.0	9.1	28.6	24.8	62.5
60K	5	133.0	9.1	29.1	25.7	63.9
60K	10	133.0	9.1	29.4	27.3	65.8
60K	20	133.0	9.1	30.2	41.2	80.5
120K	1	234.2	9.1	28.9	25.6	63.6
120K	5	234.2	9.1	30.1	25.5	64.7
120K	10	234.2	9.1	30.6	39.1	78.8
120K	20	234.2	9.1	30.2	65.2	104.5

amended band or when query differences behave like near-uniform logit rescaling. Overall, a small, fixed r per head suffices to make reuse faithful across layers and sequence lengths.

5.7 Layerwise Reuse Patterns: Acceptance and Skipped-Prefix Fraction

Setup and metrics. We profile how often MAC can reuse a prefix *per layer* using LongGenBench. For a current decode position m and a matched index p (see §3.2), Figure 5 reports: (i) the *acceptance rate*—the fraction of decoding steps for which at least one candidate in the cache window passes the pre-RoPE L2 test in Eq. (3); and (ii) the *skip ratio*—the expected fraction of the prefix that is not re-read when a reuse occurs, $\mathbb{E}\left[(p-r)_+/m\right]$ (we count misses as zero). All quantities are averaged over sequences and over the 32 query heads that share 8 KV heads (GQA) within each layer in the LLaMA-3.1-8B model.

Threshold sweep (fixed window K=2048) varies the L2 threshold $\tau \in [0, 0.9]$ from Eq. (3), revealing layer dependent query similarity. **Window sweep (fixed threshold τ=0.75)** varies the search window $K \in \{32, \ldots, 2048\}$, revealing layer dependent temporal locality.

Insights. Reuse is strongly layer-dependent. Early layers exhibit high self-similarity and thus frequent reuse; specific mid and upper layers are more variable and benefit from slightly larger windows. Increasing the search window stabilizes acceptance and increases the skipped-prefix fraction up to a point, beyond which returns diminish. These profiles suggest that lightweight, per-layer tuning of the match threshold and window size could further raise acceptance and skip ratios without meaningful overhead.

5.8 Fine-Grained Match/Attention Latency Profile

The match kernel runs only within the fixed search window K, never over the full context. Thus its overhead is a constant regardless of the context length. Table 5 makes this explicit by showing the fine-grained latency profile of the Match kernel, load balancer, and attention kernel under 32K, 60K, and 120K contexts.

First, the Match kernel stays fixed at 9.1 µs across all contexts and KV-budget settings, confirming that it is bounded by the search window rather than the total sequence length. Second, the load balancer is also nearly constant, staying in the 28.6 to 30.6 µs range; this indicates that the scheduling overhead also does not scale materially with context length. Third, the MAC attention kernel is the only component that grows meaningfully, because it still depends on the effective band+tail span after reuse. At aggressive budgets (1%–5%), that kernel remains around 25 µs even at 120K, while at the more relaxed 20% budget it grows from 27.3 µs at 32K to 65.2 µs at 120K.

Overall, these numbers show that MAC converts the dominant context-sensitive cost into a small constant front-end plus a reduced attention computation. For example, at 1% KV budget the overall attention-path latency remains essentially flat, from 63.9 µs at 32K to 63.6 µs at 120K, whereas full attention increases from 71.6 to 234.2 µs. Even at 20% KV budget, MAC rises only to 104.5 µs at 120K, still well below the full-attention baseline. This is the intended operating behavior: once the match window is fixed, the remaining latency growth is driven mainly by the unreused suffix rather than by the full prefix length.

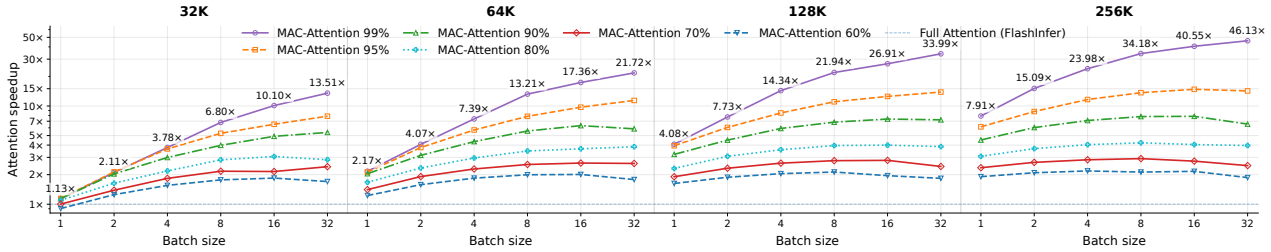

Figure 6. **Attention speedup vs. batch size and KV skip ratio across context lengths.** Speedup is computed as the *combined attention-phase latency* of our scheme (512-token matching window, load-balanced planning, rectified-prefix reuse, and tail attention). The wide shared axes are split into four subgroups for contexts {32K, 64K, 128K, 256K}; within each subgroup, the x-axis places batch sizes $\{1, 2, 4, 8, 16, 32\}$, the y-axis (shown once, left) is log-scale with a $1\times$ reference line. Each line denotes a KV skip ratio. For more thorough latency comparison with existing efficient attention methods, refer to §5.4.

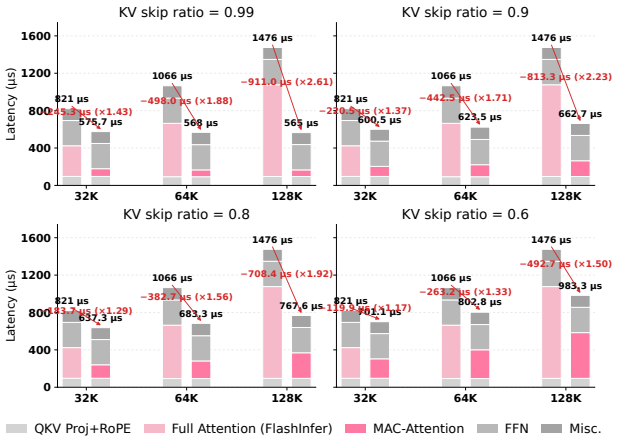

Figure 7. **End-to-end decode latency breakdown and speedup.** For a fixed batch size, stacked bars show phase-level latency (QKV+RoPE, attention, FFN, misc.) under full attention and MAC-Attention across contexts and skip ratios; speedup is computed end-to-end.

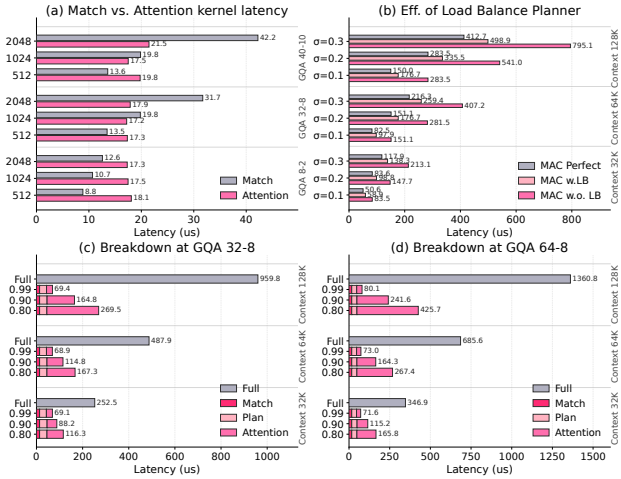

Figure 8. **Kernel-level costs and load balancing. (a)** L2 *Match* kernel latencies under different GQA settings and window lengths (using standard Attention for reference). **(b)** Effect of the load-balancing planner compared with a perfectly balanced oracle and an unbalanced baseline. **(c–d)** Per-step composition of MAC-Attention (Match, Plan, Attention) across contexts.

5.9 Attention-Phase Speedup vs. Batch Size, Context Length, and Skip Ratio

Figure 6 shows that MAC-Attention's gains grow systematically with (i) the KV skip ratio, (ii) batch size, and (iii) context length—exactly what we expect for an IO-bound workload. At fixed length and batch, higher skip (e.g., **MAC-99%**) yields the largest acceleration because only $\sim 1\%$ of the KV region is streamed and reduced per step; the remaining overheads—pre-RoPE matching, a small rectification band, and a constant-time log-domain merge—are largely length-independent. As batch size increases, the baseline saturates HBM bandwidth, so *reducing KV bytes* translates into direct wall-clock gains for MAC-Attention; for example at **32K**, the MAC-99% series scales from $\sim 1.1\times$ (batch 1) to $\sim 13.5\times$ (batch 32). The effect compounds with longer contexts: at batch 32 and MAC-99%, speedups climb from $\sim 13.5\times$ (32K) to $\sim 21\times$ (64K), $\sim 34\times$ (128K), and $\sim 46\times$ (256K).

At lower skip (e.g., **MAC-60%**) and very small batches, the fixed reuse overhead can outweigh KV savings—hence a small regression vs. baseline at 32K/batch 1, suggesting a regression point where we should use full attention instead.

5.10 End-to-End Decode Latency: Phase Breakdown and Speedup

Figure 7 shows that MAC's end-to-end gains on LLaMA3.1-8B-Instruct increase monotonically with both context length and the KV skip ratio at a fixed batch size. MAC-Attention only changes the attention path, with other operations untouched. This is the IO-bound regime we target: as sequences grow longer, attention (and thus KV traffic) occupies a larger fraction of the decode profile, so reducing KV bytes moved translates into proportionally larger wall-clock wins. Conversely, non-attention stages

(QKV+RoPE, FFN, and runtime overheads) cap the overall speedup, consistent with Amdahl's law. At high skip (e.g., $\gtrsim 0.9$), MAC amortizes its fixed overheads (short-band rectification and constant-time log-domain merge), yielding substantially larger gains at 64K–128K than at 32K.

5.11 Kernel-Level Costs and Load Balancing Effects

Kernel micro-benchmarks (Fig. 8a). The L2 *Match* kernel is inherently more memory-bound than FlashInfer *Attention* under GQA because it scans per *query* head while attention streams per (fewer) *KV* heads. Consequently, with GQA $8-2$ the matcher remains faster across lengths; at $32-8$ it is faster at 512 but becomes slower at 1,024–2,048 as DRAM traffic dominates; and at $40-10$ the trend amplifies, with match latency approaching $\sim 2\times$ attention at 2,048. Note, Figure 8(a) is designed to evaluate the L2 *Match* kernel and the reference *Attention* kernel at the same token counts to expose their raw kernel behavior, since matching is more memory-bound under GQA. Across all experiments, however, the match window is fixed at $K = 1024$, so its latency is effectively constant regardless of the context length. Table 5 makes this explicit by showing the fine-grained latency profile of the Match kernel, load balancer, and attention kernel under 32K, 60K, and 120K contexts.

Load balancing efficacy (Fig. 8b). Across contexts $\{32\text{K}, 64\text{K}, 128\text{K}\}$ and reuse skew $\sigma \in \{0.1, 0.2, 0.3\}$, the CTA load-balancing planner consistently narrows the gap to an oracle "perfect" schedule. Relative to a naïve (unbalanced) assignment, load balancing trims attention-phase latency by $\sim 29\%$–38%, and recovers $\approx 75\%$–80% of the *excess* over the perfect baseline. A residual $\sim 16\%$–21% overhead remains due to CTA-granularity limits—once a CTA is partially filled, intra-CTA imbalance cannot be further corrected under the GQA scheme due to the workload mapping strategy in the current implementation. We conducted a more in-depth study to investigate the effect of this reuse skew in §5.12.

Component breakdown (Fig. 8c–d). The per-step *Match* cost is flat with respect to context length (bounded by fixed K); the *Plan* stage is nearly constant and negligible. The *Attention* component scales with length and dominates variability, decreasing predictably with higher KV skip. Hence end-to-end gains are driven by avoided KV bytes, growing with both skip ratio and sequence length, in line with the budget in Eq. (9).

5.12 Performance fallback inside GQA KV group

MAC-Attentionperforms matching per query head, so different query heads within the same KV group may reuse different prefix points. Because our implementation maps a thread block to a KV group and loads KV once for all heads in the group, a single head with a low KV skip ratio directly increase work for that group.

Table 6. MAC Attention matching skew within each KV group at different model layers. Numbers are rounded to integers.

Layer	Avg.	Std.	Layer	Avg.	Std.
0	-399	0	18	-278	17
6	-369	64	24	-483	121
12	-272	15	31	-346	40

The latency numbers of MAC-Attentionwe reported already take into account this potential fallback. As shown in Fig 8(b), the gap to the perfect balance is due to this non-uniformity. To quantify this, in Table 6 we measured within-KV-group match-position skew across layers on LongBench v2. The skew is typically small with some outliers. **Avg.** measures the averaged median of the relative position of successful matches, e.g., -399 denotes matching to a query that is 399 tokens prior to the current decoding token. **Std.** measures the median non-uniformity of these relative positions within each KV group. In the first layer, surprisingly, we observed a uniform matching, where all attention heads match to the same previous query. In other layers, we observed some fluctuations in matching positions.

5.13 Auxiliary HBM Overhead of Query and Summary Caches

An important property of MAC-Attention is that its extra state is bounded by the search window K, rather than by the full context length L. Besides the baseline KV cache, we only keep a ring of recent queries for matching and the associated cached summary bookkeeping within the same window. Therefore, the auxiliary HBM footprint grows as $O(K)$, whereas the baseline KV cache grows as $O(L)$. In practice, the query ring is the dominant term, while the cached summary scalars contribute only a small additional overhead.

Table 7. Approximate auxiliary HBM overhead at 128K context.

Search window K	Aux. HBM overhead %
256	1.2
512	2.3
1024	4.7
2048	9.4

In our default setting $K = 1024$, the measured auxiliary footprint is only about 5% of the KV-cache size at 120K context, while the same setting delivers a $7.9\times$ attention-phase speedup. This is the key systems tradeoff: MAC-Attention pays a small, constant-size cache overhead

in exchange for substantially reducing repeated KV traffic over the much longer active context.

Using the profiled $K = 1024$, $L = 120\text{K}$ point as a reference, the relative auxiliary footprint can be approximated as

$$\text{Overhead}(K, L) \approx 5\% \cdot \frac{K}{1024} \cdot \frac{120\text{K}}{L},$$

which reflects the linear dependence on K and the inverse dependence on the total context length L. Table 7 reports the resulting auxiliary HBM overhead at $L = 128\text{K}$. Even at $K = 2048$, the extra footprint remains below 10%, which supports the practical operating regime used throughout the paper.

6 Related Work

Compression and selection

KV compression cuts memory traffic via low-rank/quantized caches and two-stage pipelines (Behnam et al., 2025; Chang et al., 2024; Zhang et al., 2024), while selection/eviction prunes tokens or pages using statistics, heuristics, or learned signals—including position-persistent sparsity (Feng et al., 2024; Ge et al., 2023; Li et al., 2024; Liu et al., 2023; Tang et al., 2024; Xiao et al., 2024; Yang et al., 2025; Zhang et al., 2023); in contrast, MAC-Attention keeps access to KV as a fallback or a correction in a very low frequency, avoiding prefix reads by reusing prior attention summaries, mathematically approaching the original full attention's fidelity.

Structural/prefix reuse and stepwise reuse

Prefix reuse across requests and tree-structured decoding shares computation when textual prefixes overlap (Gim et al., 2024; Yao et al., 2025), and stepwise reuse alternates full and partial updates from prior attention patterns (Xu et al., 2024) which alternates full attention steps with partial steps that attend only to the top-k tokens from a previously computed attention pattern; MAC-Attention reuses within a single stream based on query similarity, independent of literal prefix overlap or decoding topology.

Systems and kernels

IO-aware kernels and serving systems improve locality, tiling, and paged-KV memory for high-throughput decode (Dao, 2023; Dao et al., 2022; Hong et al., 2023; Kwon et al., 2023; Ye et al., 2025); MAC-Attention composes with these stacks by reducing how much of the prefix must be used, without any training changes.

7 Discussion and Future Work

In long context scenarios, we often observe that MAC-Attention can achieve an acceptance/hit rate greater than 99% within the K search window, achieve a de-facto $O(1)$ complexity computation and memory access once hits. Although it still needs to visit all context on a match miss, falling back to $O(n)$ overall, we believe this scheme can be a promising direction for future research in efficient attention. More broadly, MAC-Attention complements linear and sub-quadratic attention by amortizing computation across time rather than altering representations or discarding tokens. As we suggested in §5.6 and §5.7, a more robust and adaptive configuration may further strengthen it.

8 Conclusion

We presented a training-free, model-agnostic scheme, MAC-Attention. It accelerates long context inference by reusing prior attention via a lightweight Match–Amend–Complete pipeline. The approach preserves full access to context, composes with IO-aware kernels and paged-KV managers. Across a wide range of long context benchmarks, MAC-Attention maintains full-attention quality while largely reducing KV reads, cutting end-to-end latency by over 60% at 128K, and yielding up to $2.6\times$ faster token generation on LLaMA models (and up to $46\times$ on the attention phase). In practice, high match rates make step cost largely prefix-insensitive, directly targeting the IO bottleneck that dominates long-context decode. By reusing computation rather than compressing or discarding tokens, MAC-Attention complements KV compression/selection and broadens the design space for fast, faithful attention.

References

Abouelenin, A. et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025. doi: 10.48550/arXiv.2503.01743.

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. GQA: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Bai, Y., Tu, S., et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024.

Behnam, P., Fu, Y., Zhao, R., Tsai, P.-A., Yu, Z., and Tumanov, A. Rocketkv: Accelerating long-context LLM inference via two-stage KV cache compression. *arXiv preprint arXiv:2502.14051*, 2025.

Chang, C.-C., Lin, W.-C., Lin, C.-Y., Chen, C.-Y., Hu,

Y.-F., Wang, P.-S., Huang, N.-C., Ceze, L., Abdelfattah, M. S., and Wu, K.-C. Palu: Compressing kv-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

Chen, R., Wang, Z., Cao, B., Wu, T., Zheng, S., Li, X., Wei, X., Yan, S., Li, M., and Liang, Y. Arkvale: Efficient generative llm inference with recallable key-value eviction. *Advances in Neural Information Processing Systems*, 37:113134–113155, 2024.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Dao, T., Haziza, D., Massa, F., and Sizov, G. Flash-decoding for long-context inference. Stanford CRFM Blog, 2023.

Feng, Y., Lv, J., Cao, Y., Xie, X., and Zhou, S. K. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *arXiv preprint arXiv:2407.11550*, 2024.

Ge, S., Zhang, Y., Liu, L., Zhang, M., Han, J., and Gao, J. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

Ge, S. et al. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Gim, I., Chen, G., Lee, S.-s., Sarda, N., Khandelwal, A., and Zhong, L. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.

GLM, T., Zeng, A., Xu, B., Wang, B., Zhang, C., Yin, D., Zhang, D., Rojas, D., Feng, G., Zhao, H., et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024.

Grattafiori, A. et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783.

Hong, K., Li, X., Huang, X., Ma, X., Hao, Y., Wu, W., Dai, G., Yang, H., et al. Faster large language model inference on gpus (flashdecoding++). *Proceedings of MLSys*, 2024.

Hong, K. et al. Flashdecoding++: Faster large language model inference on gpus. *arXiv preprint arXiv:2311.01282*, 2023.

Hooper, C., Shao, Y. S., et al. Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Hooper, C., Zhao, S., Manolache, L., Kim, S., Mahoney, M. W., Shao, Y. S., Keutzer, K., and Gholami, A. Multipole attention for efficient long context reasoning. *arXiv preprint arXiv:2506.13059*, 2025.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

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. *Advances in Neural Information Processing Systems*, 37: 22947–22970, 2024.

Liu, A. et al. Minicache: KV cache compression in depth dimension for efficient large language model inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36: 52342–52364, 2023.

Milakov, M. and Gimelshein, N. Online normalizer calculation for softmax. *arXiv preprint arXiv:1805.02867*, 2018.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Tang, J., Zhao, Y., Zhu, K., Xiao, G., Kasikci, B., and Han, S. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wu, Y. et al. Benchmarking long-form generation in long-context llms. *arXiv preprint arXiv:2409.02076*, 2024.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2024. ICLR 2024.

Xu, F., Goyal, T., and Choi, E. Recycled attention: Efficient inference for long-context language models. *arXiv preprint arXiv:2411.05787*, 2024.

Yang, L. et al. Tidaldecode: Fast and accurate LLM decoding with position-persistent sparse attention. In *International Conference on Learning Representations (ICLR)*, 2025.

Yao, J., Chen, K., Zhang, K., You, J., Yuan, B., Wang, Z., and Lin, T. Deft: Decoding with flash tree-attention for efficient tree-structured LLM inference. In *International Conference on Learning Representations (ICLR)*, 2025.

Ye, Z., Chen, L., Lai, R., Lin, W., Zhang, Y., Wang, S., Chen, T., Kasikci, B., Grover, V., Krishnamurthy, A., et al. Flashinfer: Efficient and customizable attention engine for llm inference serving. *arXiv preprint arXiv:2501.01005*, 2025.

Zhang, R., Wang, K., Liu, L., Wang, S., Cheng, H., Zhang, C., and Shen, Y. LoRC: Low-rank compression for llms kv cache with a progressive compression strategy. *arXiv preprint arXiv:2410.03111*, 2024.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

A Extended Experiments and Ablations

This appendix is reorganized to present *experimental results first*, followed by operator pseudocode and implementation analyses. We begin with (i) the quality–efficiency trade-off curves (§A.1), (ii) the band–window heatmap for configuration selection (§A.2), and then (iii) an exploratory study enabling MAC-Attention during both *prefill* and *decode* (§A.3). Subsequent sections provide the operator contract, numerics, data layout, pseudocode, tuning guidance, scheduling notes, diagnostics, and limitations.

A.1 Quality–Efficiency trade-off curve

Figure 9 plots normalized model quality (solid) versus KV usage (dashed) as the matching threshold τ sweeps. The "knee" typically occurs where KV usage drops sharply while quality remains $\approx 100\%$. Choose τ at or just before the knee, then adjust K and r minimally.

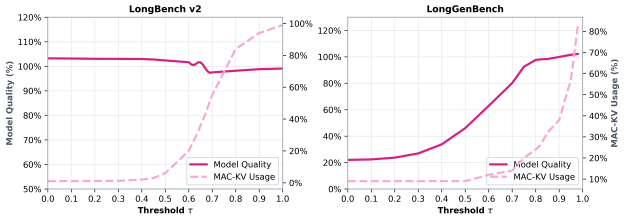

Figure 9. **Quality–efficiency trade-off vs. matching threshold τ.** Left: LongBench v2. Right: LongGenBench. Solid: quality (relative to baseline). Dashed: KV usage (lower is better).

A.2 Band–Window heatmap: normalized efficiency & accuracy gating

Figure 10 shows the normalized KV token efficiency $\mathrm{Eff}_{\mathrm{raw}}(r, K) = \mathrm{Acc}(r, K)/\mathrm{KVFrac}(r, K)$ over rectification band r and window K. Colors are normalized within the grid (white→pink: higher is better). A star marks configurations whose *accuracy* satisfies $\mathrm{Acc}(r, K) \geq \theta\, \mathrm{Acc}_{\mathrm{base}}$ (default $\theta = 0.95$).

Empirically: (i) very small windows ($K \leq 128$) look accurate but inefficient (few matches, little skipping); (ii) larger windows expose longer reuse gaps and require larger bands (typically $r \geq 256$) for fidelity; and (iii) practical knees cluster near $(K, r) \in \{(1024, 256), (1024, 512), (2048, 256)\}$.

A.3 MAC-Attention for both prefilling and decoding: exploratory study

Setting. Modern serving systems prefill in large chunks (e.g., 8K–16K tokens). We ablate enabling MAC-Attention *inside prefill* by letting each prefill token match against any earlier token (global window, denoted `inf.`), then amending a short band and completing with a merge—exactly as in decode. This explores feasibility; it is *not* a production configuration.

Observations (Table 8). Even with very high hit/skip rates, overall accuracy trails full attention when reuse spans very long gaps inside prefill. Length stratification shows that MAC-Attention can help in the *Long* bucket but may underperform in *Short/Medium* buckets if the rectification band is too small. Increasing band size r improves fidelity for long gaps but adds per-step cost.

Table 8. **Enabling MAC-Attention for both prefilling and decoding on LongBench v2.** We sweep τ and band r; the *window* is set to `inf.` to permit global search across the entire prefix (ablation-only; not production). Hit% and Skip% are reported as `prefill/decode`. Baseline is full attention.

Settings			LongBench v2					
Threshold τ	Window	Band r	Overall	Short	Medium	Long	Hit %	Skip %
Baseline (full attention)			29.0	35.0	26.0	25.0	–	–
0.30	`inf.`	512	26.6	30.6	25.1	23.1	99/99	55/86
0.35	`inf.`	512	26.8	34.4	22.8	22.2	99/99	55/86
0.45	`inf.`	128	25.6	29.4	21.4	27.8	96/95	56/85
0.45	`inf.`	256	27.2	33.3	22.3	26.9	96/95	55/85
0.45	`inf.`	512	27.2	30.0	21.9	33.3	96/95	54/84
0.55	`inf.`	512	26.8	30.6	21.4	31.5	86/83	49/74
0.60	`inf.`	512	27.4	31.7	23.7	27.8	77/71	45/62

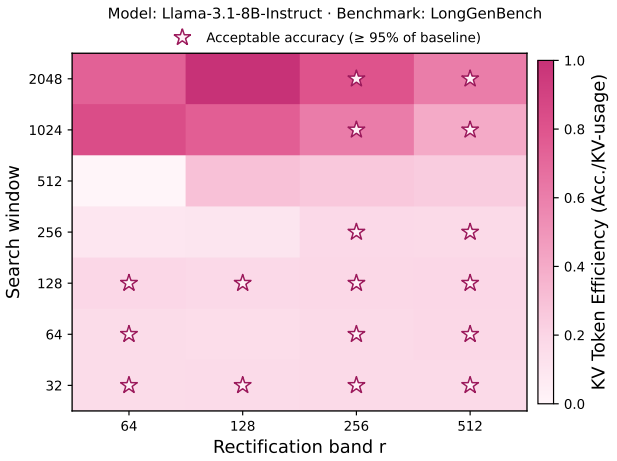

Figure 10. **Normalized KV token efficiency** $\big(\text{Acc.}/\text{KV-usage fraction}\big)$ across **rectification band** and **search window**. Stars denote accuracy $\geq \theta$ of baseline.

Why prefill is harder. During prefill, the reuse gap Δ for late tokens in a chunk often spans the entire chunk because no within-chunk summaries exist yet. Post-position (post-RoPE) logit drift accumulates across many positions; rectifying only a short band can leave residual error. By contrast, decode reuse typically happens over shorter horizons where a small fixed band suffices.

Recommendations. (i) **Distance cap**: accept a match only if the pre-RoPEL2 test passes and $\Delta \leq \Delta_{\max}$ (e.g., 1–2K). (ii) **Adaptive band**: scale r with Δ or until the local mass outside the band is below ε (e.g., $1-\rho \leq 0.01$–0.05). (iii) **Two-gate acceptance**: combine pre-RoPEL2 with a cheap post-RoPEdrift proxy (e.g., cached $\|Q_m - Q_p\|$). (iv) **Microprefill**: if supported, use smaller prefill chunks to reduce reuse gaps.

Bottom line. MAC-Attention is robust for *decode*. For *prefill*, naive global matching with a short fixed band underperforms full attention overall despite high hit rates. Prefill viability improves with distance caps and adaptive bands.

B Practical Guidance, Scheduling, Diagnostics, and Assumptions

B.1 Tuning guide

Window K. Start with $K \in \{1024, 2048\}$; increase only if mid/deep layers show low acceptance. Diminishing returns beyond $\approx$4096.

Threshold τ. Under an isotropic null, pick τ to target a false-positive rate α via $\alpha \approx F_{\chi^2_{d_q}}\big(d_q(1-\tau)^2\big)$, $\tau \approx 1 - \sqrt{F^{-1}_{\chi^2_{d_q}}(\alpha)/d_q}$. In practice, $\tau \in [0.4, 0.8]$ works well globally; minor per-layer nudges can help.

Band r. Choose the smallest r whose cumulative mass near the cursor exceeds $1-\varepsilon$ under baseline ($\varepsilon \in [0.01, 0.05]$). Rules of thumb: $r \in [128, 512]$; deeper layers may prefer slightly larger r.

Dtypes. Rings in `bf16`/`fp16` with `fp32` accumulators; keep (m, Z) in `fp32`.

B.2 Scheduling and concurrency

Micro-pipeline. Three kernels with minimal sync each step: K1 L2-match (memory-bound, warp-reduced), K2 band+tail attention + merge (dominant), K3 rectify–append (aux stream).

Auxiliary stream and graphs. Launch K3 on an auxiliary CUDA stream with a single event wait; its outputs are consumed when the same layer is revisited. Capture

steady-state into a CUDA Graph to amortize launches.

Load balancing. Heads have heterogeneous spans; flatten the (request, head) axis, assign CTAs proportional to span length, cap CTA size to preserve occupancy. This recovers most of the gap to an oracle perfectly balanced schedule.

B.3 Diagnostics, failure Modes, and safeguards

Log (per layer/head). (i) acceptance rate; (ii) skipped-prefix fraction $\mathbb{E}[(p-r)_+/m]$ (misses as 0); (iii) baseline band mass $\sum_{t=p-r+1}^{p} \alpha_t^{(p)}$ (periodic probe); (iv) norms $\|\tilde{q}\|_2$, $\|v\|_2$ percentiles.

Common issues and mitigations. (1) Low acceptance in specific layers: increase K or relax τ *for those layers only*. (2) Quality drift at very long gaps: enlarge r slightly or cap reuse by Δ. (3) Numerical glitches in downdate: enforce common baseline m, and fall back to full attention if $Z^{\mathrm{rect}} \approx 0$. (4) Memory pressure: reduce K; keep (m, Z) in `fp32`.

Determinism. With fixed seeds and decode settings, MAC-Attention is deterministic given fixed match decisions; misses execute the full attention baseline.

B.4 Deployment scope and assumptions

Decode-only augmentation. MAC-Attention modifies *decode* and leaves *prefill* and weights unchanged unless explicitly enabled (ablation in §A.3).

Serving stack. Assumes an IO-aware attention kernel (FlashAttention-style) and a paged-KV manager; composes with MQA/GQA (matching is per query head; attention streams shared KV heads).

Per-request state. Two rings of capacity K (we only cache what we search): a *query ring* with pre-RoPE queries and an *attention-summary ring* with rectified prefix summaries.

B.5 Limitations and when to back off

MAC-Attention relies on short-horizon semantic repetition and recency-biased attention. Heads with global, flat attention or idiosyncratic behaviors may see few matches and naturally fall back to baseline. For extremely large reuse gaps, a small fixed band can under-rectify; enlarge r or skip reuse beyond a distance cap.