ART: ATTENTION RUN-TIME TERMINATION FOR EFFICIENT LARGE LANGUAGE MODEL DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context decoding in Large Language Models (LLMs) is constrained by the cost of accessing and processing the Key-Value (KV) cache. Despite evidence that attention outputs depend jointly on keys and values, most existing KV management methods rely on key-only pruning, since incorporating values incurs prohibitive overhead. In this paper, we propose **Attention Run-time Termination (ART)**, a lightweight run-time mechanism that tracks accumulated attention outputs during kernel execution and terminates subsequent KV block accesses when successive observed output updates remain small. Rather than replacing KV selection, ART dynamically terminates redundant KV traversal on top of existing dense or sparse attention policies. We introduce a stability-based criterion that monitors both magnitude and directional changes of intermediate attention outputs and provides a theoretical characterization of the resulting truncation error. Experiments on LongBench show that ART increases the generation throughput of existing KV-cache methods by up to 20% with marginal quality loss. Even under the demanding distant-retrieval conditions of RULER NIAH, ART provides additional efficiency gains to importance-based KV-cache methods.

1 INTRODUCTION

Large language models (LLMs) (Touvron et al., 2023; Yang et al., 2025) rely on a growing key–value (KV) cache to store the intermediate representations of previously generated tokens during autoregressive decoding. As the sequence length increases, each new query Q attends to an expanding set of cached keys K and values V, causing both latency and memory usage to grow linearly. Efficiently managing the KV cache, often referred to as KV pruning, has therefore become a central challenge in accelerating long-context inference.

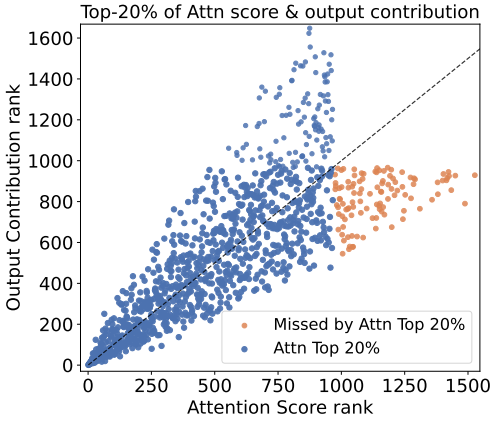


Figure 1: Attention score versus output contribution. Orange points denote high-contribution tokens missed by attention-score ranking.

Most existing approaches adopt key-centric KV cache pruning, retaining only tokens selected by heuristic or learned importance estimators. Representative methods (Xiao et al., 2024; Zhang et al., 2023; Tang et al., 2024; Liu et al., 2025; 2023) typically base these estimators on query–key similarity or attention scores. However, these proxies do not always reflect a token's true influence on the model output. As illustrated in Figure 1, this misalignment reveals a fundamental limitation of attention-based pruning: tokens with low attention weights can still have substantial influence through their value representations $\|A_{i,j}V_j\|_2$.

Several recent studies (Gu et al., 2025; Guo et al., 2024; Akhauri et al., 2025) have attempted to incorporate value information into KV cache management, demonstrating the potential benefits of value-aware modeling. However, these approaches typically rely on additional predictors, precomputation, or offline analysis to estimate value contributions, introducing non-trivial overhead

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

during inference. This raises a natural question: **Can we capture the joint effect of keys and values at run-time with negligible additional cost?**

Our key observation is that this joint effect is already reflected in the attention output itself. Rather than estimating token importance before decoding, we can monitor how the accumulated attention output evolves as additional KV blocks are processed. Modern FlashAttention-style kernels (Dao et al., 2022; Dao, 2024) naturally expose a sequence of intermediate attention outputs during block-wise execution. As KV blocks are progressively loaded and processed (Figure 2), each step incrementally updates the attention output, providing a direct signal of how much additional information the newly processed blocks contribute. When these incremental updates remain small, they provide a run-time signal for reducing further computation along the current traversal. This enables progressive evaluation of the joint influence of keys and values at run-time, without relying on expensive pre-decoding estimation.

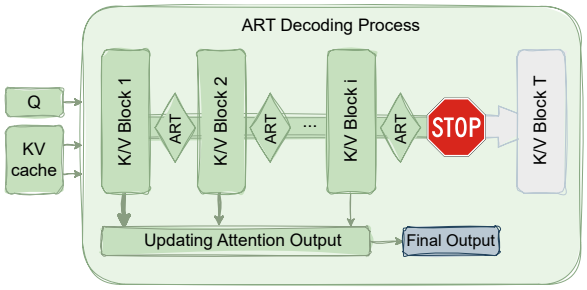


Figure 2: Overview of the ART. As KV blocks are processed sequentially, ART monitors the intermediate attention output and triggers an early termination once the output stabilizes, skipping the remaining KV blocks during decoding.

Building on this insight, we propose **Attention Run-time Termination (ART)**, a lightweight output-aware mechanism for terminating attention computation at run time. ART operates directly within block-wise attention kernels, incrementally monitoring the evolution of partial attention outputs as KV blocks are processed. Rather than replacing KV selection, ART dynamically terminates redundant KV traversal on top of existing dense or sparse attention policies. This makes ART a composable acceleration layer: a base KV policy determines which blocks are eligible to be processed, while ART uses the observed output evolution as an ex-post signal to reduce residual computation along that traversal.

We evaluate ART on LongBench (Bai et al., 2024a), the Needle-in-a-Haystack (NIAH) tasks from RULER (Hsieh et al., 2024), and the long-generation L0-Bench (Sun et al., 2025), using four models from 7B to 70B parameters. ART reduces decoding time per output token (TPOT) in all LongBench settings, by 10.5% on average on Mistral-7B and 9.7% on Llama-3.1-70B, and improves large-batch generation throughput by up to 20% when composed with existing KV cache management methods. Even in the distant-retrieval stress test of RULER NIAH, where recency-first traversal can miss critical evidence, ART provides further decoding-efficiency gains when combined with existing importance-based KV-cache methods, with substantially smaller accuracy losses. Integrated into the decode attention kernel of vLLM (Kwon et al., 2023), ART also improves end-to-end serving efficiency under the same engine configuration, without additional scheduler adaptations.

Our contributions are summarized as follows:

- We propose ART, an output-aware run-time termination mechanism that acts as a composable acceleration layer over dense or sparse KV traversal policies.
- We design a stability-based termination criterion that quantifies convergence in the output space by jointly monitoring scale and directional changes of intermediate attention outputs, thereby capturing the combined influence of keys and values.
- We provide a theoretical characterization of ART's approximation behavior, deriving a truncation-error bound that relates early termination to output stability and residual attention contributions.
- We demonstrate that ART improves generation throughput by up to 20% at large batch sizes, validate it in an end-to-end serving stack and on long-generation tasks, and characterize its traversal-dependent speed–retrieval trade-off on RULER NIAH across two model families.

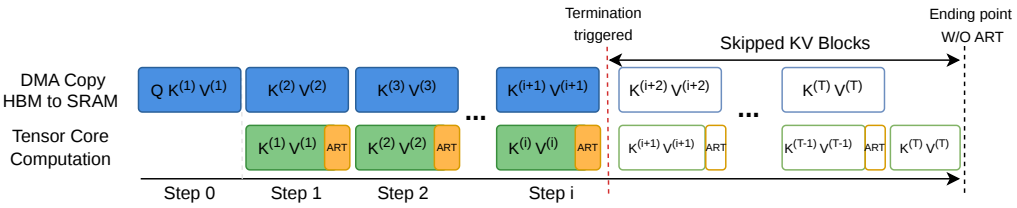


Figure 3: ART integrated into the FlashAttention execution pipeline. FlashAttention overlaps DMA-based KV block prefetch from HBM with Tensor Core computation in a block-wise manner. ART performs a lightweight run-time check on the evolving attention output during computation. Once the observed output updates satisfy the stability criterion (*e.g.*, at Step i), ART terminates the pipeline early by preventing further KV block prefetch and computation, reducing memory traffic and computation along the current traversal.

2 Related Work

KV cache management. Early approaches to efficient long-context inference adopt pattern-based pruning strategies, such as LM-Infinite (Han et al., 2024) and Attention Sink (Xiao et al., 2024), which maintain a fixed-size active context by discarding older tokens or concentrating attention on a small set of anchor positions. To move beyond static heuristics, subsequent methods estimate token importance using semantic, hierarchical, or dynamic criteria. ChunkKV (Liu et al., 2025), SnapKV (Li et al., 2024), Quest (Tang et al., 2024), and PyramidKV (Cai et al., 2024) preserve coarse-grained global information while filtering redundant tokens through structured grouping. H2O (Zhang et al., 2023), TOVA (Oren et al., 2024), and Scissorhands (Liu et al., 2023) further exploit the temporal persistence of importance scores to adaptively manage the KV cache in decoding steps.

Despite their effectiveness, these methods are fundamentally *key-centric*, implicitly assuming that the attention matrix alone determines the output and overlooking the role of value (V) magnitudes. Recent work (Guo et al., 2024) challenges this assumption by showing that values (V) also encode critical semantic signals and significantly influence attention outcomes. However, current value-aware approaches (Gu et al., 2025; Akhauri et al., 2025) typically rely on auxiliary predictors or offline analysis, incurring substantial overhead and limiting their practicality for online serving. This leaves an open gap to capture the joint influence of keys and values at *run time* without additional latency.

Efficient attention kernels and inference systems. At the kernel level, FlashAttention (Dao et al., 2022) and FlashAttention-2 (Dao, 2024) significantly improve attention efficiency through IO-aware tiling and online softmax. By computing attention in blocks, these kernels reduce memory traffic between HBM and SRAM while incrementally accumulating attention outputs. This block-wise execution model forms a key foundation for our method, as it exposes intermediate accumulation states that can be monitored at run time without modifying the attention formulation.

Building on such optimized kernels, serving frameworks like vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2024) further improve end-to-end efficiency via PagedAttention and advanced scheduling, addressing memory fragmentation and improving serving throughput. However, these systems treat attention kernels as atomic operators. Our work complements existing frameworks by opening this atomic operator and introducing a lightweight run-time early-termination mechanism directly within kernel execution, reducing unnecessary KV accesses at the source.

3 Methodology: ART

3.1 Execution Properties of FlashAttention

Although attention outputs are defined by a global softmax over all keys, modern FlashAttention-style kernels offer a crucial execution property: attention is computed in a streaming, tile-wise manner, where the output is incrementally accumulated while computation and memory transfers are overlapped (Figure 3).

To formalize this behavior, we consider the scaled dot-product attention for a query block $Q \in \mathbb{R}^{N\times d}$ attending to keys $K \in \mathbb{R}^{N\times d}$ and values $V \in \mathbb{R}^{N\times d}$:

$$O = \sigma\left(\frac{QK^\top}{\sqrt{d}}\right)V, \tag{1}$$

where σ is the softmax activation function acting row-wise; see definition in Appendix A. FlashAttention computes Equation (1) by partitioning keys and values into blocks $\{(K_t, V_t)\}_{t=1}^{T}$ and processing them sequentially. Using a numerically stable streaming softmax formulation, the kernel maintains an internal accumulator that is updated after each block. We denote by $O^{(t)}$ the attention output after processing the first t blocks.

As t increases, $O^{(t)}$ converges to the final output $O^{(T)}$. As shown in Figure 4, the attention output converges rapidly as KV blocks are processed. Only a subset of blocks contributes substantial error reduction, suggesting that the output may stabilize well before all KV blocks are traversed.

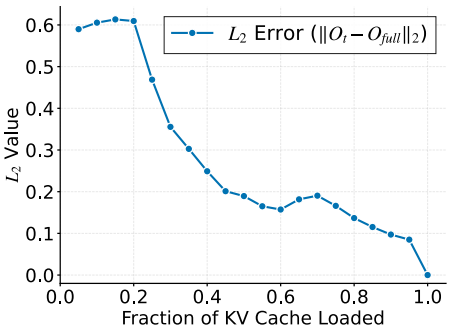


Figure 4: Convergence of attention output with recent KV retention. The plot illustrates the relative L_2 error between the attention output from the full KV cache and that from a truncated recent-KV window, as a function of the loaded fraction.

3.2 Stability-Based Run-time Termination

Given the early stabilization behavior, a natural question is how to use observed output changes to choose a stopping point within the current KV traversal at run time. A straightforward approach based on attention scores is insufficient, as high attention weights may be assigned to less informative or even zero-valued value vectors (Figure 1).

Motivated by this insight, our idea is to track the evolution of the intermediate attention output. ART uses sustained stability of the partial output $O^{(t)}$ as a stopping signal within the current traversal. The signal measures updates from visited blocks and does not evaluate the importance of unseen blocks or certify the quality of the traversal order. By measuring stability directly in the output space, ART naturally captures the joint contribution of keys and values.

Directly computing full-vector norms or performing cross-thread reductions on the global accumulator would introduce non-trivial overhead and undermine the benefits of early termination. This calls for a lightweight mechanism that can approximate output convergence without full-vector computation.

In optimized attention kernels utilizing NVIDIA Tensor Cores (*e.g.*, FlashAttention-2 (Dao, 2024)), the accumulator $O^{(t)}$ is distributed across threads via the swizzled Matrix Multiply-Accumulate (MMA) layout. We construct a probe vector $x^{(t)} \in \mathbb{R}^m$ by collecting from all 32 lanes of warp 0, the leading element of their respective register fragments. Due to the interleaved nature of the MMA layout, these 32 elements constitute a deterministic, scattered subsample of the full head dimension rather than a contiguous slice, providing dispersed coverage across dimensions at negligible overhead. We therefore monitor the stability of $x^{(t)}$ as a computationally efficient proxy signal for convergence. To improve decision reliability, the probe is integrated with the patience-based verification protocol (defined below) that dampens transient fluctuations before early termination is triggered. The effectiveness of the probe is studied in Appendix B.

As successive KV blocks are aggregated, the output state $O^{(t)}$ evolves from $O^{(t-1)}$ in two ways: it may grow or shrink in *magnitude*, and it may rotate in *direction* to align with information in the current block. To characterize convergence in both aspects, we decouple the stability evaluation into scale and direction components:

$$d^{(t)}_{\text{scale}} = \|x^{(t)} - x^{(t-1)}\|_2, \quad d^{(t)}_{\text{direction}} = 1 - \cos(x^{(t)}, x^{(t-1)}), \tag{2}$$

An attention update is considered *stable* when both the magnitude change and the directional deviation fall below predefined thresholds: $d^{(t)}_{\text{scale}} < \tau$ and $d^{(t)}_{\text{direction}} < \phi$.

Algorithm 1 ART run-time termination for FlashAttention

Require: Stability thresholds τ (scale) and ϕ (direction), patience p

$O^{(0)} \leftarrow \mathbf{0}$ {Initialize attention accumulator}
$c^{(0)} \leftarrow 0$ {Consecutive stable steps counter}
for $t = 1$ to T **do**
  Load next KV tile (K_t, V_t)
  Update streaming attention output $O^{(t)}$
  $x^{(t)} \leftarrow \textsc{Probe}(O^{(t)})$
  $d^{(t)}_{\text{scale}} \leftarrow \|x^{(t)} - x^{(t-1)}\|_2$; $d^{(t)}_{\text{direction}} \leftarrow 1 - \cos(x^{(t)}, x^{(t-1)})$
  $c^{(t)} \leftarrow c^{(t-1)} + 1$ **if** $d^{(t)}_{\text{scale}} < \tau$ **and** $d^{(t)}_{\text{direction}} < \phi$ **else** 0
  if $c^{(t)} \geq p$ **then break** {Termination triggered}
end for
Finalize normalization and **return** $O^{(t)}$

Although the ℓ_2 difference alone captures both effects, no single threshold balances them: thresholds sensitive enough to detect directional shifts are overly restrictive to benign scale variations, and vice versa. Decoupling the two yields a more robust criterion (Section 4.6).

To mitigate premature termination caused by transient fluctuations across neighboring KV blocks, we incorporate a patience mechanism. Let $c^{(t)}$ denote the number of consecutive stable steps up to tile t:

$$c^{(t)} = \begin{cases} c^{(t-1)} + 1, & \text{if } d^{(t)}_{\text{scale}} < \tau \text{ and } d^{(t)}_{\text{direction}} < \phi, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Early termination is triggered once $c^{(t)} \geq p$, where p is the patience parameter. As shown in Section 4.6, this substantially reduces false-positive terminations at negligible cost. Algorithm 1 summarizes the complete procedure.

Conditional truncation bound. When ART terminates at block $t^* < T$, it returns the intermediate output $o^{(t^*)}$ instead of the full-attention output $o^{(T)}$. We characterize this approximation under a mass-regulated traversal: After the final stable window begins, the remaining blocks have a decaying normalized attention contribution along the chosen traversal order. Under this condition and a non-degenerate probe, if ART terminates after p consecutive stable steps with scale tolerance τ, then:

$$\|o^{(T)} - o^{(t^*)}\| \leq \left(\frac{\tau b + \Delta\mu}{\nu}\right) \ln\left(\frac{T}{t^*}\right), \tag{4}$$

where $b = t^* - p + 1$ is the beginning of the patience window and ν measures how well the probe preserves the relevant output-update directions. Under the stated mass-regulated traversal and probe assumptions, this result characterizes the truncation error in terms of output stability and residual attention contributions (see Appendix A for details).

3.3 Composable Termination over KV Traversal

ART reduces computation within a given KV traversal by choosing its stopping point from observed attention-output updates. When an existing KV-cache method supplies a retained set and importance scores, ART preserves its selection and uses the order induced by those scores. Thus, existing importance-based policies supply what to traverse and in what order, while ART adds a run-time decision about how far to execute that traversal. ART does not require a separate importance estimator.

Without a base method, ART uses recency-first traversal over full KV, which is effective on LongBench. In distant-retrieval stress tests, critical evidence may occur after a stable prefix; ART can instead follow existing importance priorities and provide additional gains (Section 4.3).

- For full KV caching, the selected set contains all KV blocks. ART traverses them in reverse temporal order, using the recency prior without additional importance estimation.

- For pattern-based methods, such as StreamingLLM (Xiao et al., 2024), ART applies the same recency-first traversal over the retained KV blocks while explicitly preserving the sink blocks required by the base method.
- For methods that assign explicit importance scores to retained KV blocks, such as SnapKV (Li et al., 2024), ART utilizes an importance-first traversal by processing the selected blocks in descending order of their assigned priority. This is implemented through an auxiliary index mapping for indirect addressing and does not alter the base method's cache selection logic.

In paged-KV serving systems such as vLLM (Kwon et al., 2023), the traversal unit is a physical KV page, and reordering is realized by permuting the page-index table consumed by the attention kernel; Section 4.4 shows that this adds no measurable overhead.

If all selected blocks are processed, reordering preserves the attention output up to floating-point non-associativity; when ART terminates early, it adds a dynamic truncation step on top of the base method while retaining its selection policy.

Discussion. ART introduces an *ex-post*, output-aware convergence check that incorporates the joint effect of keys and values. It reduces execution along the current traversal, complementing existing KV selection and importance estimates when available.

ART treats the traversal policy as an input rather than an object to be learned or validated. Its role is to exploit output-space redundancy exposed during execution, while the base policy retains responsibility for candidate selection and prioritization. Patience improves the reliability of this run-time signal by filtering transient fluctuations among visited blocks.

4 Experimental Evaluation

4.1 Experimental Setup

Our experiments are conducted with Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct (AI@Meta, 2024), and Qwen3-8B (Yang et al., 2025), on NVIDIA A100-SXM4 (80GB) and H200 SXM (141GB) GPUs.

Benchmarked methods. We compare full KV with three representative KV-cache optimization methods: StreamingLLM (Xiao et al., 2024), SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2024). All methods use identical decoding configurations and random seed to ensure fair comparison; the detailed configurations for all comparative methods are listed in Appendix D. We combine each of the above methods with ART (shown as **+ART**), taking the traversal order from the base method and, for full KV, from the recency policy (Section 3.3). Unless stated otherwise, ART uses the default parameters: $\tau = 10^{-5}$, $\phi = 10^{-3}$, and $p = 5$. Refer to Appendix H for a sensitivity analysis of the parameters.

Metrics and measurement. To assess generation quality, we report the LongBench score (Bai et al., 2024a) and accuracy on the NIAH tasks from RULER (Hsieh et al., 2024), where premature termination could skip distant critical information. For latency, we report TPOT, measured as decode wall-clock time per generated token in a controlled HuggingFace Transformers stack with our custom FlashAttention-2 kernel; it includes the cost of traversal reordering and indirect addressing. We additionally report generation throughput (output tokens per second) and the runtime of the FlashAttention kernel itself (Appendix E). Serving-level metrics are reported in Section 4.4.

4.2 Overall Performance on LongBench

Table 1 reports the average LongBench scores and decoding TPOT. ART introduces only a marginal drop in quality score and consistently decreases TPOT across dense and sparse KV-cache settings, by approximately 10.5% on average for Mistral-7B and 9.7% for Llama-3.1-70B. Appendix J gives the full category-level breakdown.

Figure 5 reports generation throughput across batch sizes $B \in \{1, 2, 4, 8\}$, using SnapKV as a representative base method. Integrating ART consistently improves throughput, and the gains grow with batch size because long-context decoding is memory-bound: concurrent sequences amplify

Table 1: Average LongBench score and decoding TPOT speedup (TPOT×) on Mistral-7B-Instruct-v0.3 and Llama-3.1-70B-Instruct. For each base method, we report the original result and the result after integrating ART. Full KV with ART uses recency-first traversal as described in Section 3.3.

	Mistral-7B-Instruct-v0.3				**Llama-3.1-70B-Instruct**			
	LB Score		TPOT×		LB Score		TPOT×	
Method	Base	+ART	Base	+ART	Base	+ART	Base	+ART
Full KV	46.29	45.74	1.00	**1.16**	48.25	47.48	1.00	**1.15**
StrLLM(0.8)	40.29	40.20	1.08	**1.24**	42.31	41.80	1.13	**1.27**
StrLLM(0.2)	31.11	31.09	1.18	**1.30**	33.88	33.51	1.52	**1.66**
SnapKV(0.8)	43.92	43.12	1.07	**1.17**	46.49	45.54	1.08	**1.18**
SnapKV(0.2)	31.17	30.80	1.17	**1.27**	34.24	33.59	1.47	**1.58**
PyramidKV(0.8)	42.57	42.00	1.06	**1.19**	44.10	43.31	1.12	**1.25**
PyramidKV(0.2)	30.07	29.97	1.17	**1.29**	35.93	35.47	1.47	**1.60**

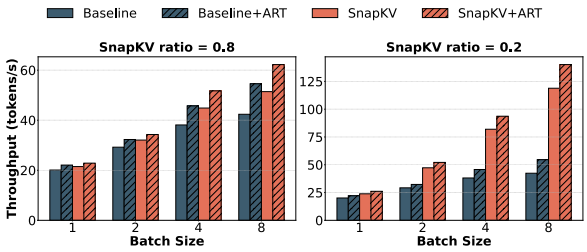


Figure 5: Generation throughput (tokens/s) versus batch size for full KV and SnapKV, combined with ART. The left and right panels correspond to KV cache retention ratios of 0.8 and 0.2, respectively. SnapKV+ART performs consistently better, especially for large batch size.

contention for HBM bandwidth, while ART reduces the KV blocks accessed per sequence and its detector cost stays constant per block. ART's speedup also grows with context length (Appendix I).

4.3 Retrieval Stress Test on RULER Needle-in-a-Haystack

We use RULER NIAH (Hsieh et al., 2024) as a distant-retrieval stress test: success depends on sparse evidence in a long context, which may be visited late under recency-first traversal. We use this setting specifically to expose the boundary of ART's observed-output signal when the traversal prior is mismatched to the retrieval pattern, rather than to assume that prefix stability certifies arbitrary unseen evidence. We evaluate Qwen3-8B with all base methods and Llama-3.1-8B with full KV and SnapKV, examining whether ART can further accelerate importance-based KV methods under these demanding conditions.

Table 2 shows that recency-first full-KV ART improves TPOT but reduces 32K accuracy from 99.34% to 78.65% on Qwen3-8B and from 98.88% to 79.04% on Llama-3.1-8B. Unlike the small score changes on LongBench, this stress test exposes cases where the output stabilizes over a visited prefix before distant evidence is reached. Recency alone is insufficient here, and the stopping signal cannot detect the unseen evidence.

Even here, ART further accelerates existing importance-based KV-cache methods with substantially smaller accuracy losses. At 32K on Qwen3-8B, SnapKV's TPOT speedup relative to full KV increases from $1.18\times$ to $1.36\times$ with ART, with accuracy changing from 99.37% to 95.68%; PyramidKV improves from $1.19\times$ to $1.37\times$, with accuracy changing from 99.24% to 96.00%. On Llama-3.1-8B, SnapKV improves from $1.18\times$ to $1.34\times$, with accuracy changing from 99.26% to 97.09%. Thus, composition provides additional efficiency even in this stress test, although a quality cost remains.

Importance-based methods supply selection and priority information; ART uses observed output updates to reduce execution within that traversal.

Table 2: Accuracy (%) and decoding TPOT speedup (TPOT×) on RULER NIAH. For each base method, we report the original result and the result after integrating ART. Recency + ART denotes full KV traversed in recency order, while importance-based methods use the order induced by their existing priorities.

Method	4K		8K		16K		32K	
	Acc.	TPOT×	Acc.	TPOT×	Acc.	TPOT×	Acc.	TPOT×
Qwen3-8B								
Full KV	99.95	1.00	99.88	1.00	99.73	1.00	99.34	1.00
Recency + ART	89.11	**1.10**	95.24	**1.15**	85.77	**1.17**	78.65	**1.27**
SnapKV	92.65	1.11	98.63	1.13	99.54	1.16	99.37	1.18
+ ART	92.73	1.11	98.73	**1.15**	98.93	**1.21**	95.68	**1.36**
StreamingLLM	80.13	1.12	82.72	1.15	81.42	1.20	77.78	1.21
+ ART	80.25	1.12	82.45	**1.18**	81.30	**1.26**	75.81	**1.42**
PyramidKV	90.09	1.10	97.34	1.12	99.36	1.18	99.24	1.19
+ ART	90.72	**1.11**	96.75	**1.15**	97.78	**1.21**	96.00	**1.37**
Llama-3.1-8B-Instruct								
Full KV	100.00	1.00	99.56	1.00	99.58	1.00	98.88	1.00
Recency + ART	90.12	**1.09**	90.63	**1.18**	85.26	**1.21**	79.04	**1.29**
SnapKV	93.08	1.14	98.57	1.13	98.44	1.15	99.26	1.18
+ ART	93.06	**1.15**	98.54	**1.14**	98.01	**1.19**	97.09	**1.34**

Table 3: End-to-end serving with vLLM on one H200 (Qwen3-8B, LongBench-v2, continuous batching, concurrency 64). Relative changes are with respect to the identical baseline configuration.

Method	Throughput (tok/s) ↑	p95 ITL (ms) ↓	p95 TTFT (s) ↓	p95 E2E (s) ↓	GPU util. (%)	p95 DRAM act. (%) ↓
Baseline	118.72	126.27	61.43	74.43	97.40	20.10
+ ART	**123.12**	**123.17**	**59.15**	**71.59**	97.34	**16.90**
Δ	+3.70%	−2.46%	−3.71%	−3.81%	−0.06 pp	−15.92%

For workloads with stringent retrieval requirements, ART can be deployed conservatively with a minimum KV-block coverage budget, calibrated on representative workloads, so that termination is enabled only after a prescribed fraction of candidate blocks has been processed. This does not detect unseen evidence, but provides an explicit control over the maximum amount of traversal that ART may skip.

4.4 End-to-End Serving Performance

To test whether the kernel-level gains persist in a realistic serving configuration, we integrate ART into vLLM 0.7.3 (Kwon et al., 2023), with paged KV, continuous batching and mixed prefill/decode traffic, serving Qwen3-8B on LongBench-v2 (Bai et al., 2024b) (Appendix F). As shown in Table 3, ART improves output throughput by 3.70% and reduces p95 inter-token latency (ITL) by 2.46% and p95 end-to-end latency by 3.81%, while reducing DRAM activity by 15.92% at essentially unchanged GPU utilization. The end-to-end gain is smaller than the kernel-level gain because ART only accelerates decode attention and, under continuous batching, requests terminate at different positions, so early-finishing thread blocks wait for slower ones.

Reordering overhead under paged KV and tensor parallelism. We isolate the cost of importance-based traversal by serving Llama-3.1-70B-Instruct with vLLM at tensor-parallel degree 4 on four H200 GPUs, with paged KV (256-token pages), full traversal forced and the probe disabled, so no early-termination gain is included. Relative to an identity-copy control that creates and consumes a page-index table of the same form, importance reordering changes throughput by $+0.016\% \pm 0.182\%$ and p95 ITL by $+0.006\% \pm 0.302\%$ (three runs; Appendix F), *i.e.*, no measurable overhead.

Table 4: Ablation study of ART on LongBench. We report the average score of Single-Document QA and Multi-Document QA (QA), summarization (Sum), few-shot learning (Fewshot) and overall LongBench score; the average FlashAttention kernel time per call; and their changes relative to the full ART configuration.

Method	QA	Sum	Fewshot	**LB Score**	**Δ Score**	**Time (ms)**	**Δ Time** (%)
ART (full KV)	35.58	26.40	63.04	**45.74**	–	0.633	–
w/o d_{scale}	14.78	21.18	27.74	19.34	−26.40	0.135	−78.7
w/o $d_{\text{direction}}$	35.32	26.25	62.64	43.87	−1.87	0.628	−0.8
w/o patience	33.58	25.96	58.11	41.54	−4.20	0.506	−20.1

4.5 Error Accumulation and Probe Reliability

Long generation. LongBench and RULER produce short answers, so they do not test whether approximation errors accumulate over long trajectories. We therefore evaluate Qwen3-8B with full KV on L0-Bench (Sun et al., 2025), which requires generating deterministic, mechanically verifiable program-execution traces (200 examples; the longest split averages ∼2,750 generated tokens and 250 steps). Error accumulation would appear as larger degradation at later positions or on longer splits; we observe neither. ART preserves final-state accuracy (45.0% vs. 45.0%), changes whole-trace accuracy from 22.5% to 21.5% and line-wise accuracy from 82.78% to 82.80%, and keeps the early/middle/late line-wise differences within ±0.05 points, with 180 of 200 outputs identical (Appendix G).

Probe reliability. We compare the 32-element probe with a detector that applies the same rule to the full 128-dimensional head output, over all layers and heads of Qwen3-8B and Mistral-7B at 4K–32K context (Appendix B). Agreement is 97.5%–98.3% and the premature-stop rate (FP) is 1.3%–2.0%, with no degradation as context length grows, and a mean offset of 1.3–1.5 blocks when the stopping positions differ. Per layer–head pair, the median FP is at most 1.0%, but 4.0%–7.4% of pairs exceed 5% FP; the LongBench results indicate that this localized mismatch has little task-level impact.

4.6 Ablation Study

Table 4 shows that all components are needed. Without d_{scale}, termination is excessively early, since the output may stabilize in direction while still changing in magnitude. $d_{\text{direction}}$ adds 1.87 points at negligible kernel-time cost, and patience adds 4.20 points by filtering out transient fluctuations that would otherwise cause false-positive termination.

5 Conclusion

We propose Attention Run-time Termination (ART), a lightweight run-time mechanism that accelerates long-context LLM decoding by terminating redundant KV traversal during attention execution. Unlike prior approaches that rely primarily on key-based importance estimates, ART monitors the convergence of intermediate attention outputs, capturing the joint influence of keys and values at run time. It introduces output-space convergence as a new optimization axis that works at the kernel level, complements existing KV cache management methods, and adds negligible overhead.

Our experiments show consistent decoding-efficiency gains, in TPOT and throughput, on top of dense and sparse KV traversal policies, and these gains persist in an end-to-end vLLM serving stack. On LongBench and the long-generation L0-Bench, ART has small effects on quality, with no observed systematic error accumulation on L0-Bench. Even in the demanding distant-retrieval conditions of RULER NIAH, ART provides additional decoding gains to existing importance-based KV-cache methods, with substantially smaller accuracy losses than recency-first full-KV traversal.

Limitations. ART reduces KV accesses rather than cache capacity. Its observed-update criterion neither assesses traversal quality nor certifies unseen blocks, and its truncation bound assumes mass regulation without verifying it at run time. End-to-end gains are limited by the decode-attention scope and load imbalance from heterogeneous stopping positions. Scheduler-aware batching, adaptive thresholds, traversal co-design and prefill are future directions.

ETHICS STATEMENT

ART aims to improve the efficiency of long-context LLM inference by reducing redundant attention computation and KV-cache accesses. Potential positive impacts include lower serving cost, reduced energy consumption, and improved accessibility of long-context applications under limited hardware resources. At the same time, more efficient inference can lower the cost of deploying large-scale generative systems, which may indirectly amplify misuse risks already associated with LLMs, such as spam, misinformation, or automated content generation. This work does not introduce a new generative model, training dataset, human-subject study, or user-facing deployment system; all benchmarks used are publicly available. Because ART is an approximation, practitioners should validate quality on their own workloads, including retrieval-sensitive cases such as those examined in Section 4.3.

REPRODUCIBILITY STATEMENT

Algorithm 1 specifies the complete termination procedure, and Section 4.1 lists the models, baselines, and default hyperparameters. The configurations of all baseline KV-cache methods are given in Appendix D, the benchmark descriptions in Appendix C, and the serving and long-generation protocols in Appendices F and G. The assumptions and complete proofs of the truncation bound are given in Appendix A. We will release the modified FlashAttention-2 kernel, the vLLM integration, and the evaluation scripts; an anonymized version is provided in the supplementary material.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish the writing of the manuscript. We did not use generative AI tools for research ideation, method design, experiments, or data analysis. We have reviewed all AI-assisted text, and we take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

AI@Meta. Llama 3.1: Model cards and prompt formats, 2024. URL https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_1/.

Yash Akhauri, Ahmed F. AbouElhamayed, Yifei Gao, Chi-Chih Chang, Nilesh Jain, and Mohamed S. Abdelfattah. TokenButler: Token importance is predictable. *arXiv preprint arXiv:2503.07518*, 2025. doi: 10.48550/arXiv.2503.07518. URL http://arxiv.org/abs/2503.07518.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024b.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024. doi: 10.48550/arXiv.2406.02069. URL http://arxiv.org/abs/2406.02069.

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

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

Yuzhe Gu, Xiyu Liang, Jiaojiao Zhao, and Enmao Diao. OBCache: Optimal brain KV cache pruning for efficient long-context LLM inference. *arXiv preprint arXiv:2510.07651*, 2025. doi: 10.48550/arXiv.2510.07651. URL http://arxiv.org/abs/2510.07651.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention score is not all you need for token importance indicator in KV cache reduction: Value also matters. *arXiv preprint arXiv:2406.12335*, 2024. doi: 10.48550/arXiv.2406.12335. URL http://arxiv.org/abs/2406.12335.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3991–4008, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023. doi: 10.48550/arXiv.2310.06825. URL http://arxiv.org/abs/2310.06825.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. ChunkKV: Semantic-preserving KV cache compression for efficient long-context LLM inference. *arXiv preprint arXiv:2502.00299*, 2025. doi: 10.48550/arXiv.2502.00299. URL http://arxiv.org/abs/2502.00299.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1043. URL https://aclanthology.org/2024.emnlp-main.1043/.

Simeng Sun, Cheng-Ping Hsieh, Faisal Ladhak, Erik Arakelyan, Santiago Akle Serano, and Boris Ginsburg. L0-Reasoning Bench: Evaluating procedural correctness in language models via simple program execution. *arXiv preprint arXiv:2503.22832*, 2025.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. In *International Conference on Machine Learning*, pp. 47901–47911. PMLR, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. doi: 10.48550/arXiv.2302.13971. URL http://arxiv.org/abs/2302.13971.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL http://arxiv.org/abs/2505.09388.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37: 62557–62583, 2024.

A THEORETICAL ERROR BOUND ANALYSIS

Below, we prove the conditional truncation bound of Equation 4.

Let K, V be partitioned into T blocks $\{(K_t, V_t)\}_{t=1}^{T}$, with $K_t, V_t \in \mathbb{R}^{n\times D}$ and $nT = N$. Given a query block $Q \in \mathbb{R}^{N\times d}$, the current attention output at t is given by

$$U^{(t)} = \sigma\left(\frac{QK_t^\top}{\sqrt{d}}\right)V_t,$$

where σ is the row-wise softmax operator defined element-wise on $S \in \mathbb{R}^{m\times l}$ by

$$\sigma(S)_{i,j} = \frac{\exp(S_{i,j})}{\sum_{k=1}^{l}\exp(S_{i,k})}.$$

Given the t-th block and the i-th query row $q_i \in \mathbb{R}^{1\times d}$, the vector's current output attention scale is given by $\ell_{t,i} = \sum_{k=1}^{n}\exp(\frac{q_i(K_{t,k}^T)}{\sqrt{d}})$, and the accumulating output attention scale is given by $A_{t,i} = \sum_{j=1}^{t}\ell_{j,i}$.

Then, the accumulating output for q_i during block t is obtained by the FlashAttention normalization procedure

$$o_i^{(t)} = \frac{A_{t-1,i}o_i^{(t-1)} + \ell_{t,i}u_i^{(t)}}{A_{t,i}},$$

where $u_i^{(t)}$ is the i-th row of $U^{(t)}$ and $o_i^{(t)}$ is the i-th row of the accumulating attention output $O^{(t)}$. By initializing $o_i^{(1)} = u_i^{(1)}$, the normalization procedure defined above returns $O^{(T)} = O$, where O is the entire output block as defined in Expression 1.

During autoregressive decoding, the most relevant query at each step is the most recently generated token q_N. The termination decision for q_N is therefore the most critical; hence, our analysis focuses on a single query vector.

For the remainder of this analysis we drop the subscript i and let $o^{(t)}, u^{(t)}, A_t, \ell_t$ represent $o_i^{(t)}, u_i^{(t)}, A_{t,i}, \ell_{t,i}$ respectively. Given the FlashAttention normalization, the following recurrence relationship holds:

$$o^{(t)} - o^{(t-1)} = \frac{\ell_t}{A_t}(u^{(t)} - o^{(t-1)}).$$

This recurrence is the central theme of our analysis. The size of each increment is governed by the mass ratio $\frac{\ell_t}{A_t}$ and the direction $u^{(t)} - o^{(t-1)}$. We consider a telescoping argument to represent the changes between the output obtained during ART's termination and the complete output.

Let $P \in \mathbb{R}^{m\times d}$ be a linear map. Let p denote the patience, τ denote the tolerance scale, and $Po^{(t)} = x^{(t)}$ denote the probe vector as defined in §3.1. Let t^* denote the stopping index for ART, and let $b = t^* - p + 1$ represent the first iteration that satisfies the scale and direction conditions under which ART terminates.

Definition 1. ***(Probe Update Dominance)*** *We define the probe update dominance as,* $\Delta := \max_{b\le j\le T}(\|P(u^{(j)} - o^{(j-1)})\| - \|P(u^{(b)} - o^{(b-1)})\|)$.

Remark 1. *Above definition implies that for all* $j \ge b$ *the expression* $\|P(u^{(j)} - o^{(j-1)})\| \le \|P(u^{(b)} - o^{(b-1)})\| + \Delta$ *holds.*

Definition 2. ***(ART Termination)*** *ART terminates on block* t^* *after* p *consecutive stable steps of*

$$\|x^{(j)} - x^{(j-1)}\| \le \tau, \quad \text{for all } j = b, b+1, \cdots, t^*.$$

Assumption 1. ***(Mass-Regulated Traversal)*** *Let* $\mu = \frac{b\ell_b}{A_b}$. *There exists constants* $\hat{\ell} > 0$ *and* $h > 0$ *such that for all* $j \ge b$*: (i)* $\ell_j \le \hat{\ell}$*,(ii)* $A_j \ge hj$*, with* $\hat{\ell}/h = \mu$.

The mass-regulated traversal assumption specifies the regime analyzed by the bound. ART uses recency-first traversal or priorities supplied by an existing KV policy; neither choice by itself establishes this assumption. The bound assumes bounded block mass and sufficient accumulated mass

growth along the chosen traversal, including its unvisited portion. ART does not check these conditions at run time. Even when all blocks have comparable mass, $\ell_j \approx \ell$ and $A_j \approx j\ell$, yielding $\ell_j/A_j \approx 1/j$. Thus, the assumption captures the natural dilution of each newly visited block as the accumulated attention mass grows.

Assumption 2. ***(Probe Constraints)*** *For some $\nu > 0$, the linear probe, $P\colon \mathbb{R}^d \to \mathbb{R}^m$ satisfies*

$$\nu\|z\| \le \|Pz\| \le \|P\|_2\|z\|, \quad \text{for all } z \in \textit{Span}\{o^{(j)} - o^{(j-1)} : 1 \le j \le T\}.$$

The probe is introduced as an efficiency-motivated surrogate for evaluating output stability without explicitly materializing the full attention output during kernel execution. In Appendix B, we provide empirical evidence that stability measured in the probe space is well aligned with the oracle stability criterion computed from the full attention output. Now, we are set to quote our intermediate results.

Lemma 1 follows directly from Assumption 1, yielding a decaying upper bound on normalized block mass. This is a conditional statement about the chosen traversal, rather than a certificate obtained from ART's stopping signal. Figure 4 and Table 1 provide empirical evidence for useful computation–quality trade-offs, but do not verify the mass assumption for every query. Table 2 further shows how these trade-offs change when ART follows priorities from importance-based KV methods in a distant-retrieval stress test.

Lemma 1. ***(Harmonic Accumulation Decay)*** *Let Assumption 1 hold. Then for all $j \ge b$ we have $\frac{\ell_j}{A_j} \le \frac{\mu}{j}$.*

Proof. Assumption 1 gives: $\frac{\ell_j}{A_j} \le \frac{\hat{\ell}}{jh} = \frac{\mu}{j}$, and completes proof. □

Lemma 1 tells us that the mass factor of $(o^{(j)} - o^{(j-1)})$ grows by the order $\mathcal{O}(1/j)$. Our next Lemma, Lemma 2, provides an adaptive bound on the probe vectors based on the threshold, τ, which is user-inferred and can be chosen as small as the user wants. Since ART terminates at iteration t^*, then it is obvious that the previous p steps satisfy the difference condition.

Lemma 2. ***(Post-Termination Probe Decay)*** *Let Assumption 1 hold, then for all $k \ge 1$*

$$\|x^{(t^*+k)} - x^{(t^*+k-1)}\| \le \frac{\tau b}{t^* + k} < \tau.$$

Proof. The linearity of P gives $x^{(j)} - x^{(j-1)} = \frac{\ell_j}{A_j} P(u^{(j)} - o^{(j-1)})$.

By Remark 1, for every $j \ge b$,

$$\|x^{(j)} - x^{(j-1)}\| = \frac{\ell_j}{A_j}\|P(u^{(j)} - o^{(j-1)})\| \le \frac{\ell_j}{A_j}(\|P(u^{(b)} - o^{(b-1)})\| + \Delta).$$

Assumption 1 and Definition 2 give $\|x^{(b)} - x^{(b-1)}\| < \tau$, hence $\|P(u^{(b)} - o^{(b-1)})\| < \frac{A_b}{\ell_b}\tau = \frac{b}{\mu}\tau$. Then,

$$\|x^{(j)} - x^{(j-1)}\| \le \frac{\ell_j}{A_j}(\|P(u^{(b)} - o^{(b-1)})\| + \Delta) \le \frac{b}{j}\tau + \frac{\mu}{j}\Delta.$$

Setting $j = t^* + k$ gives the result $\|x^{(t^*+k)} - x^{(t^*+k-1)}\| \le \frac{b\tau + \mu\Delta}{t^*+k} < \tau$. □

Taken together with Assumption 1, Lemma 2 shows that the difference continually shrinks by a factor of $\mathcal{O}(b/j)$. Our next Lemma quantifies the difference between the current output vector, $u^{(t)}$, and the previous accumulating attention output vector, $o^{(t-1)}$.

Lemma 3. ***(Uniform Difference Bound)*** *For $t = 2, \cdots, T$, $\|u^{(t)} - o^{(t-1)}\| \le 2C_v$, where C_v is the largest row norm in the value matrix V, i.e., $C_v \coloneqq \max_{1\le k\le N}\|v_k\|$.*

Proof. Since each $u^{(t)}$ represents the matrix product of a softmax output vector and the block V_t, then each $u^{(t)}$ is a convex combination of the rows $\{(v_t)_r\}_{1\le r\le n}$. Hence,

$$\|u^{(t)}\| \le \max_r \|(v_t)_r\| \le C_v.$$

Each $o^{(t)}$ can be written as $\sum_{j=1}^{t} \frac{\ell_j}{A_t} u^{(j)}$, where each $\frac{\ell_j}{A_t} > 0$ and sum to 1. Hence, $o^{(t)}$ is a convex combination of $u^{(t)}$. Hence,

$$\|o^{(t)}\| = \|\sum_{j=1}^{t} \frac{\ell_j}{A_t} u^{(j)}\| \leq \max_j \|u^{(j)}\| \leq C_v.$$

By the triangle-inequality, $\|u^{(t)} - o^{(t-1)}\| \leq \|u^{(t)}\| + \|o^{(t-1)}\| \leq 2C_v$, hence the result. □

Theorem 1 is the central result that bounds the truncation error using the probe-based termination condition. The bound depends on the tolerance τ, the probe constant ν, and the iteration ratio T/t^*.

Theorem 1. *Let Assumptions 1-2 hold, then*

$$\|o^{(T)} - o^{(t^*)}\| \leq \frac{\tau b + \Delta\mu}{\nu} \ln\left(\frac{T}{t^*}\right).$$

Proof. The triangle-inequality and Assumption 2 gives

$$\|o^{(T)} - o^{(t^*)}\| \leq \sum_{j=t^*+1}^{T} \|o^{(j)} - o^{(j-1)}\| \leq \frac{1}{\nu} \sum_{j=t^*+1}^{T} \|x^{(j)} - x^{(j-1)}\|.$$

Using Lemma 2, the following result can be obtained

$$\frac{1}{\nu} \sum_{j=t^*+1}^{T} \|x^{(j)} - x^{(j-1)}\| \leq \frac{\tau b + \Delta\mu}{\nu} \sum_{j=t^*+1}^{T} \frac{1}{j} \leq \frac{\tau b + \Delta\mu}{\nu} \int_{t^*}^{T} \frac{1}{x} dx = \frac{\tau b + \Delta\mu}{\nu} \ln\left(\frac{T}{t^*}\right).$$

□

Theorem 2 provides a complementary bound through Lemma 3, expressed in terms of value matrix norm C_v and the ratio T/t^*.

Theorem 2. *Let Assumption 1 hold, then*

$$\|o^{(T)} - o^{(t^*)}\| \leq 2\mu C_v \ln\left(\frac{T}{t^*}\right).$$

Proof. The triangle-inequality and Lemma 3 gives

$$\|o^{(T)} - o^{(t^*)}\| \leq \sum_{j=t^*+1}^{T} \|o^{(j)} - o^{(j-1)}\| = \sum_{j=t^*+1}^{T} \frac{\ell_j}{A_j} \|o^{(j)} - o^{(j-1)}\| \leq 2C_v \sum_{j=t^*+1}^{T} \frac{\mu}{j}.$$

We finish the proof by using the same upper bound on the sum $\sum_{j=t^*+1}^{T} \frac{1}{j} \leq \ln(\frac{T}{t^*})$:

$$\|o^{(T)} - o^{(t^*)}\| \leq 2\mu C_v \sum_{j=t^*+1}^{T} \frac{1}{j} \leq 2\mu C_v \int_{t}^{T} \frac{dx}{x} = 2\mu C_v \ln\left(\frac{T}{t^*}\right).$$

□

Theorems 1 and 2 can be put together under a single statement, as given in Remark 2.

Remark 2. *Let Assumptions 1- 2 hold, then*

$$\|o^{(T)} - o^{(t^*)}\| \leq \min\{\frac{\tau b + \mu\Delta}{\nu}, 2\mu C_v\} \ln\left(\frac{T}{t^*}\right).$$

Table 5: Consistency between the probe and the oracle on Qwen3-8B (36 layers, head_dim = 128). **FP**: probe stops before oracle, or probe stops while oracle never does (premature termination risk). **FN**: oracle stops before probe, or oracle stops while probe never does (conservative; no accuracy risk). **Δblk**: mean ± std block offset among mismatched stopping cases (i.e., both probe and oracle stop, but at different blocks).

Condition	n	**Agree** ↑	**FP** ↓	**FN** ↓	**Δblk**
(a) Sequence length	(layer 35, needle @ 50%)				
1 024 tokens	800	**100.00%**	**0.00%**	0.00%	—
2 048 tokens	800	**100.00%**	**0.00%**	0.00%	—
4 096 tokens	800	**100.00%**	**0.00%**	0.00%	—
8 192 tokens	800	99.88%	0.13%	0.00%	—
(b) Layer depth	(4 096 tokens, needle @ 50%)				
Layer 7	1 600	99.50%	0.50%	0.00%	1.00±0.00
Layer 17	1 600	94.06%	5.56%	0.38%	1.34±0.47
Layer 35	1 600	**100.00%**	**0.00%**	0.00%	—

B PROBE–ORACLE CONSISTENCY

ART applies a lightweight probe extracted from the distributed attention accumulator rather than reconstructing the full attention output at every KV block. While this design is essential for maintaining low kernel overhead, it raises a natural question: whether the probe observes the same convergence behavior as the full attention output. To validate this, we compare the probe-based stopping decision against an oracle detector that applies the same stability rule to the fully materialized attention output.

Concretely, for each decoding, we record the stopping block predicted by the probe and compare it with the stopping block obtained by the oracle. We classify the outcomes into three categories. An agreement means that the probe and oracle make the same stopping decision, either stopping at the same block or both deciding not to stop. A false positive (FP) means that the probe stops earlier than the oracle, or that the probe stops while the oracle never stops. This case corresponds to premature termination and is the only category that may introduce accuracy risk. A false negative (FN) means that the oracle stops earlier than the probe, or that the oracle stops while the probe never stops. This case is conservative: it may reduce the achievable speedup but does not increase approximation risk. For samples where both the probe and oracle stop but at different blocks, we also report the block offset Δ_{blk}.

Table 5 reports the consistency results on Qwen3-8B, which contains 36 layers with head dimension 128. We evaluate both the effect of sequence length and the effect of layer depth. In the sequence length study, we fix the last layer and place the needle at the middle of the context. Across sequence lengths from 1K to 8K tokens, the probe almost exactly matches the oracle decision. In particular, the agreement remains 100.00% up to 4K tokens and 99.88% at 8K tokens, with only 0.13% false positives. This indicates that increasing the number of KV blocks does not noticeably degrade the reliability of the probe.

We further evaluate different layers under a fixed 4K-token context. The probe remains highly consistent with the oracle, although the agreement varies across layers. The last layer again shows perfect agreement, while intermediate layers exhibit slightly higher mismatch rates. In Layer 17, the false-positive rate is 5.56%, and the mean block offset among mismatched stopping cases is only 1.34 ± 0.47 blocks. This suggests that even when the probe stops earlier than the oracle, the discrepancy is typically small. Since ART uses a patience-based criterion, such small offsets can be further controlled by increasing the patience parameter or using more conservative thresholds.

Overall, these results provide empirical support for the probe approximation used by ART. The probe is not assumed to be an exact reconstruction of the full attention output; instead, it is required to preserve the convergence behavior relevant to the stopping decision. The high agreement and low false-positive rates in Table 5 indicate that the probe satisfies this requirement in practice, especially in the late layers where decoding decisions are most directly reflected in the final representation.

B.1 Full-Model Probe Consistency Across Layers, Heads, and Context Lengths

The study above covers selected layers of a single model. We therefore extend the comparison to *all* layers and all query heads of Qwen3-8B and Mistral-7B-Instruct-v0.3 on RULER NIAH contexts from 4K to 32K tokens.

Normalization. The deployed probe has $m = 32$ elements, whereas the reference detector uses the full $d = 128$-dimensional head output. To compare them under the same rule, both detectors use the dimension-normalized RMS change, *i.e.*, the scale statistic is divided by the square root of the vector dimension:

$$\tilde{d}^{(t)}_{\text{scale}}(x) = \frac{\|x^{(t)} - x^{(t-1)}\|_2}{\sqrt{m}}, \qquad \tilde{d}^{(t)}_{\text{scale}}(o) = \frac{\|o^{(t)} - o^{(t-1)}\|_2}{\sqrt{d}}, \tag{5}$$

so that a probe threshold τ corresponds to a threshold of $\tau\sqrt{d/m} = 2\tau$ on the raw full-vector change. This scaling is fixed a priori and not tuned. The direction statistic and the patience rule are identical for both detectors.

FP denotes the probe stopping earlier than the full-vector detector and FN the probe stopping later. These statistics measure only the *additional* mismatch introduced by the 32-element probe; they do not measure ART's full task-level premature-termination risk, which is reflected by the task accuracies in Section 4.3.

Table 6: Probe versus full-vector detector, aggregated over all layers and heads. Δ_{blk}: mean $\pm$ std block offset when both detectors stop but at different blocks.

Model	Context	Agree ↑	FP ↓	FN ↓	Δ_{blk}
Qwen3-8B	4K	98.1%	1.5%	0.4%	1.31 ± 0.58
Qwen3-8B	8K	98.3%	1.3%	0.4%	1.27 ± 0.54
Qwen3-8B	16K	97.8%	1.7%	0.5%	1.44 ± 0.71
Qwen3-8B	32K	98.0%	1.6%	0.4%	1.39 ± 0.66
Mistral-7B	4K	97.7%	1.8%	0.5%	1.35 ± 0.62
Mistral-7B	8K	97.9%	1.6%	0.5%	1.30 ± 0.59
Mistral-7B	16K	97.5%	2.0%	0.5%	1.49 ± 0.76
Mistral-7B	32K	97.6%	1.8%	0.6%	1.43 ± 0.70

Table 7: Distribution of the false-positive rate computed independently for every layer–head pair.

Model	Context	Median FP	p95 FP	Max FP	Pairs with FP $> 5\%$
Qwen3-8B	4K	0.0%	4.7%	10.8%	4.4%
Qwen3-8B	8K	0.5%	4.4%	9.9%	4.0%
Qwen3-8B	16K	0.5%	5.6%	12.4%	6.3%
Qwen3-8B	32K	0.5%	5.2%	11.7%	5.8%
Mistral-7B	4K	0.5%	5.1%	11.4%	5.2%
Mistral-7B	8K	0.5%	4.8%	10.7%	4.7%
Mistral-7B	16K	1.0%	6.0%	13.5%	7.4%
Mistral-7B	32K	0.5%	5.7%	12.9%	6.9%

Table 6 shows that agreement remains between 97.5% and 98.3% and FP between 1.3% and 2.0%, with no monotonic degradation from 4K to 32K and no qualitative difference between the two models. When the stopping positions differ, the mean offset is 1.3–1.5 blocks. Table 7 shows that the median layer–head pair closely follows the full-vector detector, but a small upper tail has higher mismatch: 4.0%–7.4% of pairs exceed 5% FP, with a maximum of 13.5%. The probe is a deliberate efficiency-oriented approximation, and this tail should be read as its approximation cost. The LongBench results in Table 1 indicate small aggregate quality changes. Agreement with the full-vector detector measures the probe approximation; neither detector verifies traversal quality or certifies the contribution of unseen blocks.

C BENCHMARK DATASET DESCRIPTION

In this study, we utilize LongBench (Bai et al., 2024a) and RULER (Hsieh et al., 2024) as our evaluation benchmarks. LongBench is the first bilingual, multi-task benchmark specifically designed for long-context Large Language Models (LLMs). This benchmark comprises 21 distinct datasets across 6 key task categories, aiming to provide a comprehensive assessment of a model's capabilities in understanding, generation, and reasoning over long texts.

Most tasks in LongBench are derived from existing public datasets but have been rigorously cleaned and filtered to ensure suitability for long-context evaluation (e.g., filtering out short text samples). The benchmark covers both English and Chinese languages. The average length of the English datasets is 6,711 words, while the Chinese datasets average 13,386 characters.

LongBench consists of the following six major task categories:

- **Single-document QA**: Includes NarraQA, Qasper, MultiFieldQA-en, and MultiFieldQA-zh. These tasks require the model to retrieve and integrate information from a long document to answer specific questions.
- **Multi-document QA**: Includes HotpotQA, 2WikiMultihopQA, Musique and DuReader. These tasks involve complex reasoning and information synthesis across multiple documents.
- **Summarization**: Includes GovReport, QMSum, MultiNews and VCSUM. The goal is to generate high-quality summaries for long meeting transcripts, news collections, or government reports.
- **Few-shot Learning**: Includes TREC, LSHT, TriviaQA, and SAMSum. These tasks provide multiple long-context examples to evaluate the model's in-context learning capabilities.
- **Synthetic Tasks**: Includes Passage Retrieval (en and cn) and Number String. These are specifically designed pressure tests to detect the model's ability to accurately locate specific information within an extremely long context.
- **Code Completion**: Includes LCC and RepoBench-P. These tasks evaluate the model's understanding and completion abilities within long code repository files.

We additionally evaluate on the Needle-in-a-Haystack (NIAH) tasks from RULER (Hsieh et al., 2024), a synthetic benchmark designed to stress-test a model's ability to retrieve specific information embedded within long, distracting contexts. RULER's NIAH suite comprises eight tasks of increasing complexity, covering four retrieval scenarios:

- **Single-needle retrieval** (`niah_single_1/2/3`): A single key–value pair is hidden within a long distractor text. Variants differ in needle complexity: short word, long phrase, and UUID string, respectively.
- **Multi-key retrieval** (`niah_multikey_1/2/3`): Multiple needles with distinct keys are inserted; the model must retrieve the value for a specific queried key. Variants scale the number of distracting needles from 2 to 4.
- **Multi-value retrieval** (`niah_multivalue`): A single key is associated with multiple values spread across the context; the model must retrieve all of them.
- **Multi-query retrieval** (`niah_multiquery`): Multiple key–value pairs are hidden, and the model is asked to answer queries about several of them simultaneously.

We evaluate all eight NIAH tasks at context lengths of 4K, 8K, 16K, and 32K tokens. Each sample is scored as exact match (0 or 1) and results are averaged across samples and tasks.

D CONFIGURATION DETAIL

We evaluate the performance of KV cache compression methods on **LongBench** (Bai et al., 2024a) using the **Mistral-7B-Instruct-v0.3** model. To ensure a faithful implementation, we align our hyperparameter settings shown in Table 8 with the notations and definitions proposed in their respective original papers.

Table 8: Hyperparameter configurations mapped to original paper notations.

Method	Paper Notation / Term	Symbol	Value
StreamingLLM	Initial Tokens (Attention Sinks)	-	4
SnapKV	Observation Window	L_{obs}	32
	Pooling Kernel Size	-	5
PyramidKV	Instruction Tokens (Local Window)	α	128
	Pooling Kernel Size	β	5

StreamingLLM. Following the findings of Xiao et al. (2024), we rely on the *Attention Sink* phenomenon to maintain streaming stability. We retain the Key and Value states of the initial tokens as anchors. Specifically, we set the number of sink tokens to 4, combined with a rolling cache of recent tokens. This configuration addresses the "perplexity surge" issue observed when initial tokens are evicted from the window attention (Xiao et al., 2024).

SnapKV. Consistent with the algorithm described by Li et al. (2024), we implement the observation-based compression strategy. We define the *Observation Window* size, denoted as L_{obs} in the paper, to be 32. This window captures the attention patterns from the last segment of the prompt. To effectively cluster important features and avoid noise, we utilize the voting mechanism with a max-pooling layer, setting the pooling kernel size to 5.

PyramidKV. Building upon the *Pyramidal Information Funneling* hypothesis (Cai et al., 2024), we employ a layer-wise dynamic budget allocation. While PyramidKV shares the pooling kernel setting ($k = 5$) with SnapKV, we significantly increase the size of the local window, referred to as "instruction tokens" and denoted as α in Cai et al. (2024), to 128. This adjustment of α (where $\alpha > L_{obs}$) ensures that the model can accurately measure the importance of tokens even under high compression rates in upper layers (k^l), mitigating the risk of information loss during the pyramidal aggregation process.

E Computational Overhead

Table 9: ART incurs negligible overhead when early termination is disabled. With $p = \infty$, the ART detector executes at every FlashAttention call without triggering termination, increasing kernel time by only 1.3%.

Setting	Avg. FA kernel time (ms)
Baseline	0.78
patience=∞	0.79

This computational overhead isolates the intrinsic runtime overhead of ART's early-termination detector, independent of any actual early stopping. Our goal is to verify that the detector itself introduces negligible computational cost.

ART is evaluated after each tile completes the standard streaming softmax update and value aggregation, followed by a block-wide synchronization. This design ensures deterministic control flow and avoids warp divergence. Crucially, the detector incurs a constant amount of computation per tile and introduces no additional global memory accesses; therefore, the overall complexity remains proportional to the number of processed KV tiles.

To quantify this overhead, we disable early termination by setting the patience parameter to $p = \infty$, forcing the ART detector to execute at every FlashAttention invocation without ever triggering termination. This configuration allows us to measure the pure computational overhead of the detection mechanism. As reported in Table 9, enabling ART increases the average FlashAttention kernel runtime by only 1.3% compared to the baseline, while producing identical evaluation scores. These results confirm that ART introduces negligible computational overhead.

F End-to-End Serving Protocol and Reordering Overhead

vLLM integration. We integrate ART into the decode attention kernel of vLLM 0.7.3 (Kwon et al., 2023) and evaluate Qwen3-8B on LongBench-v2 (Bai et al., 2024b) on one H200 GPU. The baseline and ART runs use identical engine configurations and continuous batching with concurrency 64. ART uses a traversal granularity of 256 tokens (one physical KV page) and $\tau = 10^{-5}$, $\phi = 10^{-3}$, $p = 5$.

Metric definitions. *Output throughput* is the number of generated tokens divided by the wall-clock duration of the benchmark. *Inter-token latency* (ITL) is the time between consecutive output tokens of a request; *time to first token* (TTFT) is the time from request arrival to its first output token; *end-to-end latency* is the time from request arrival to its last output token. We report the 95th percentile over all requests (for ITL, over all token intervals). *GPU utilization* is the utilization reported by the driver, averaged over the run, and *DRAM activity* is the DCGM profiling metric `DCGM_FI_PROF_DRAM_ACTIVE`, the fraction of cycles in which the device memory interface is active.

Why the end-to-end gain is smaller than the kernel-level gain. First, ART only shortens decode attention; the linear layers, sampling, scheduling, and prefill are unchanged. Second, under continuous batching, requests in the same decode batch terminate at different KV positions, so thread blocks that finish early wait for slower ones. Scheduler-aware grouping of requests with similar stopping behavior is a natural extension that we leave to future work. TTFT improves because shorter decode occupancy reduces contention and queueing for chunked prefills.

Reordering overhead under paged KV and tensor parallelism. The TPOT measurements in the main text include traversal reordering and indirect addressing. The Llama-3.1-70B LongBench runs in Table 1 shard the model across four A100 GPUs by layers (`device_map="auto"`) and therefore do not use tensor parallelism. To isolate the cost of importance-based traversal under tensor parallelism, we serve Llama-3.1-70B-Instruct with vLLM at tensor-parallel degree 4 on four H200 GPUs, with FlashAttention-2 paged KV and the minimum supported physical page size of 256 tokens. We serve 16 LongBench-v2 requests at concurrency 16, each generating 64 tokens. All configurations force full traversal and disable the probe, so no early-termination gain is included, and each configuration is run three times. We compare three page tables: the original table, an identity-copy control that creates and consumes a new page-index table in the original order, and an importance-reordered table. The clean comparison is importance reordering versus identity copy, since both pay the cost of creating and consuming a new table.

Table 10: Reordering overhead under vLLM paged KV with tensor parallelism (Llama-3.1-70B-Instruct, TP=4, four H200 GPUs; mean $\pm$ std over three runs; full traversal forced, probe disabled).

Configuration	**Throughput (tok/s)**	**p50 ITL (ms)**	**p95 ITL (ms)**
Original page table	27.646 ± 0.082	204.004 ± 0.214	235.868 ± 0.809
Identity-copy control	27.747 ± 0.032	204.140 ± 0.232	235.867 ± 0.588
Importance reorder	27.751 ± 0.019	203.537 ± 0.158	235.880 ± 0.182
Relative difference (%), importance reorder − identity copy			
Throughput	$+0.016$		
Total runtime	-0.016		
p50 ITL	-0.295		
p95 ITL	$+0.006$		

As shown in Table 10, the differences are within run-to-run variation, *i.e.*, importance-based page-index reordering has no measurable overhead in this configuration.

G Long-Generation Evaluation on L0-Bench

L0-Bench (Sun et al., 2025) requires models to generate deterministic, mechanically verifiable program-execution traces, which allows every generated line to be checked. If ART-induced errors

accumulated, we would expect increasing degradation at later positions of a trace, earlier first errors, or larger gaps on longer trajectories. We evaluate Qwen3-8B with full KV, BF16, and greedy decoding, with ART parameters $\tau = 10^{-5}$, $\phi = 10^{-3}$, and $p = 5$, and a generation cap of 4,096 tokens. The evaluation contains four splits of increasing trace length with 50 examples each. Responses average 1,419 generated tokens overall; the longest split averages approximately 2,751 tokens and 250 execution steps.

Table 11: L0-Bench results for Qwen3-8B (Base / ART). Line-wise Δ: change in line-wise accuracy (ART − Base). Early/middle/late Δ: the same change restricted to the first, middle, and last third of each trace.

Split	Whole-trace	Final-state	Line-wise Δ	Early / middle / late Δ	Identical
1	30.0 / 30.0	48.0 / 48.0	0.00 pp	0.00 / 0.00 / 0.00 pp	50/50
3	18.0 / 18.0	38.0 / 38.0	0.00 pp	0.00 / 0.00 / 0.00 pp	49/50
5	24.0 / 22.0	50.0 / 50.0	−0.18 pp	+0.11 / −0.24 / −0.39 pp	40/50
7	18.0 / 16.0	44.0 / 44.0	+0.27 pp	+0.10 / +0.43 / +0.28 pp	41/50
Overall	22.5 / 21.5	45.0 / 45.0	+0.02 pp	+0.05 / +0.05 / −0.03 pp	180/200

ART preserves final-state accuracy at 45.0%, while line-wise accuracy changes only from 82.78% to 82.80% (Table 11). The early, middle, and late differences remain near zero, with neither widening degradation toward later positions nor monotonic deterioration across longer splits. Paired outcomes are also highly consistent: whole-trace and final-state correctness agree on 198/200 (99%) and 196/200 (98%) examples, respectively. The correct-prefix ratio changes from 44.80% to 44.45%, and the mean first-error position from 53.86 to 53.23 steps. In the longest split, both methods have the same 78% EOS completion rate, with the remaining examples reaching the 4,096-token cap. These results show no evidence of systematic accumulation of ART-induced errors over long deterministic trajectories. We do not claim universal robustness for all open-ended generation tasks, but this experiment tests error accumulation with outputs substantially longer than those of LongBench and RULER.

H Parameter Sensitivity

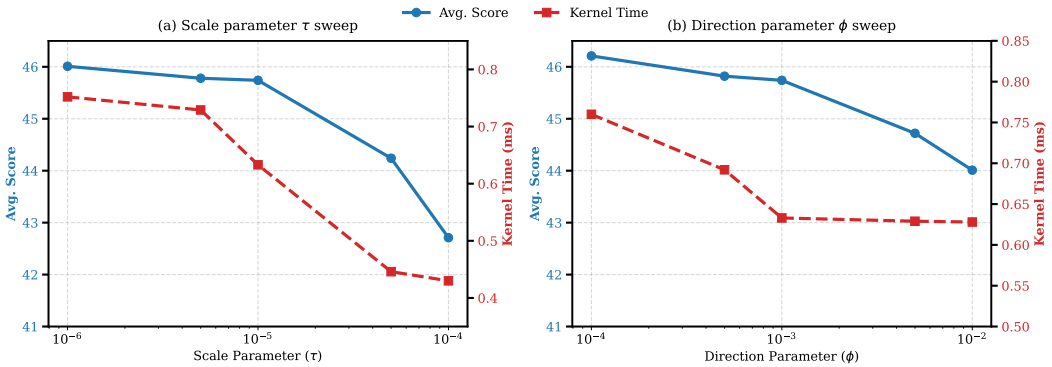


Figure 6: Sensitivity analysis of the scale parameter τ. The blue solid line (left axis) tracks the LongBench score, while the red dashed line (right axis) shows the kernel execution time. The results highlight a trade-off: $\tau = 10^{-5}$ and $\phi = 10^{-3}$ offers an optimal balance between stability and speed, while larger values provide maximum acceleration with a slight performance cost.

We further investigate the impact of the scale parameter τ and direction parameter ϕ . As shown in Figure 6, we vary τ from 10^{-6} to 10^{-4} and ϕ from 10^{-4} to 10^{-2}. Increasing τ significantly reduces the kernel execution time, reaching a peak speedup at 10^{-4} . Regarding the quality of the results, the method exhibits strong robustness for smaller thresholds where $\tau \leq 10^{-5}$. The average score remains stable around 46.0. At the most aggressive setting , we observe a moderate performance

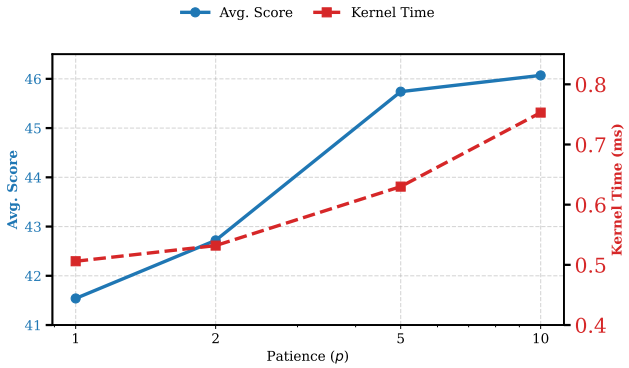


Figure 7: Sensitivity analysis of the parameter p. The blue solid line (left axis) tracks the LongBench score, while the red dashed line (right axis) shows the kernel execution time.

drop to 44.71. This creates a flexible trade-off space: users can select $\tau = 10^{-5}$ as a balanced operating point to enjoy reduced FA kernel time with negligible accuracy loss, or larger thresholds when speed is the primary priority.

As discussed in our ablation study, ϕ primarily serves to preserve correctness. The final value of ϕ is chosen to capture directional stability under a fixed scale parameter τ. When $\tau = 10^{-5}$, the corresponding best-performing value is $\phi = 10^{-3}$. A more relaxed, i.e., larger, value of ϕ can degrade accuracy, but does not yield a significant efficiency improvement.

We further study the sensitivity of ART to the patience parameter p, which controls how many consecutive KV blocks must satisfy the termination criterion before early stopping is triggered. As shown in Figure 7, a smaller patience value such as $p = 1$ or $p = 2$ leads to more aggressive termination and slightly lower kernel time, but it also increases the risk of premature stopping and results in a noticeable drop in average score. Increasing p improves robustness by requiring the attention output to remain stable for multiple consecutive blocks. In particular, $p = 5$ achieves the best trade-off between accuracy and efficiency: it recovers most of the performance while retaining a clear kernel-time reduction. Further increasing the patience to $p = 10$ only brings marginal accuracy improvement, but substantially weakens the efficiency gain due to later termination. Therefore, we use $p = 5$ as the default setting in our experiments, as it provides a balanced operating point between reliable output stability detection and practical acceleration.

I Impact of Context Length

To further verify the scalability of our approach, we break down the performance into three context length intervals: 0-4k, 4-8k, and 8k+. As illustrated in Figure 8, the efficiency advantage of ART becomes increasingly pronounced as the input length grows. In short-context scenarios, the runtime difference is marginal; however, in the long-context regime, ART significantly mitigates the computational overhead.

On LongBench, the efficiency gain remains accompanied by only small score changes across context-length buckets. As shown in Figure 8, the score trajectories of ART-enhanced methods closely track their original counterparts across the evaluated length buckets. These results indicate that the LongBench quality–efficiency trade-off does not deteriorate systematically with context length.

J Full LongBench Results

Table 12 provides the full category-level LongBench breakdown for Mistral-7B-Instruct-v0.3 and Llama-3.1-70B-Instruct. The results are consistent with the average-score summary in the main text: ART introduces only small category-level fluctuations while preserving the overall LongBench score across dense and sparse KV-cache settings.

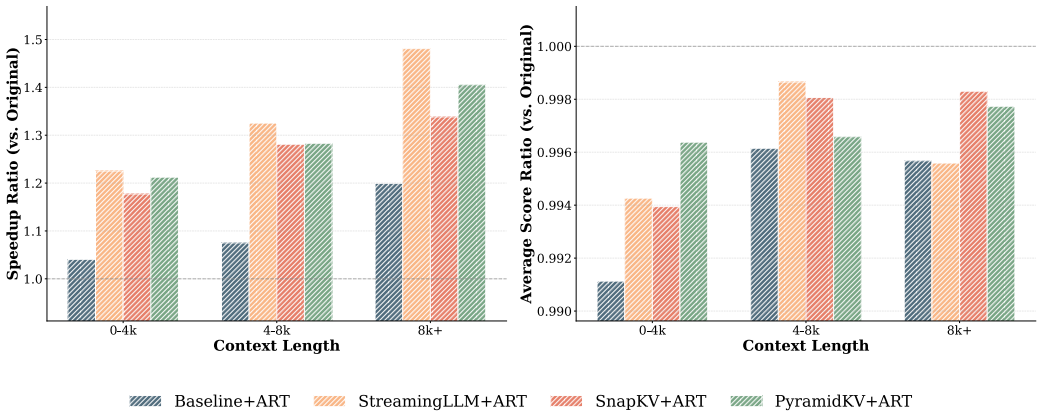


Figure 8: Impact of context length on inference efficiency and performance. The dataset is categorized into three length intervals: 0-4k, 4-8k, and 8k+. The integration of ART significantly reduces kernel running time on long context length with minimal effect on the quality of the results.

For the larger-scale evaluation, we deploy Llama-3.1-70B-Instruct on a computing node equipped with four NVIDIA A100 80GB GPUs, with the model sharded across GPUs by layers (`device_map="auto"`) rather than by tensor parallelism; Appendix F evaluates tensor-parallel serving separately. This setting evaluates whether ART remains effective under larger model scales and multi-GPU inference. As summarized in Table 1, ART consistently reduces decoding TPOT for Llama-3.1-70B across full KV caching and sparse KV-cache methods, including StreamingLLM, SnapKV, and PyramidKV. The relative speedup is more moderate than that observed on Mistral-7B, which is expected because the overall decoding time composition changes with model scale. As the parameter count increases, the cost of loading weights and computing linear projections, especially in FFN layers, grows substantially. Meanwhile, both Mistral-7B and Llama-3.1-70B use Grouped-Query Attention (GQA), so the KV-cache size does not grow proportionally with the total parameter count. Consequently, attention accounts for a smaller fraction of the total decoding time in the 70B model, limiting the end-to-end speedup from an attention-specific optimization. Nevertheless, ART still reduces the attention-side cost and remains a useful acceleration layer in large-scale long-context inference.

Table 12: Category-level LongBench score and decoding TPOT speedup comparison on Mistral-7B-Instruct-v0.3 and Llama-3.1-70B-Instruct. Each category score is averaged over its corresponding tasks. The **Avg.** column reports the original LongBench mean score over all 21 tasks, Δ denotes the marginal score change compared with the corresponding base method.

Method	Single QA	Multi QA	Summ.	Few-shot	Synth.	Code	Avg.	Δ	TPOT speedup
Mistral-7B-Instruct-v0.3									
Baseline	37.88	34.65	26.06	63.43	65.63	63.56	**46.29**	–	1.00 ×
+ ART	37.20	33.96	26.40	63.04	64.85	61.80	**45.74**	-0.55	**1.16**×
StrLLM(0.8)	28.34	27.61	25.81	62.27	47.21	64.16	**40.29**	–	1.08×
+ ART	28.05	27.34	25.83	62.42	47.51	63.52	**40.20**	-0.09	**1.24**×
StrLLM(0.2)	20.04	19.69	22.71	56.57	18.20	61.27	**31.11**	–	1.18×
+ ART	19.98	19.60	22.70	56.74	18.20	61.11	**31.09**	-0.02	**1.30**×
SnapKV(0.8)	33.13	35.56	26.33	61.36	64.67	51.33	**43.92**	–	1.07×
+ ART	34.05	31.76	25.39	59.62	65.11	53.45	**43.12**	-0.80	**1.17**×
SnapKV(0.2)	19.96	19.35	21.04	46.78	44.34	46.49	**31.17**	–	1.17×
+ ART	19.77	19.07	21.07	45.50	44.09	46.39	**30.80**	-0.37	**1.27**×
PyramidKV(0.8)	37.02	35.08	26.04	55.48	56.25	55.37	**42.57**	–	1.06×
+ ART	36.28	35.02	25.92	53.96	55.21	55.77	**42.00**	-0.57	**1.19**×
PyramidKV(0.2)	23.30	30.31	22.62	47.54	12.69	49.18	**30.07**	–	1.17×
+ ART	23.50	29.71	22.70	47.66	12.92	48.11	**29.97**	-0.10	**1.29**×
Llama-3.1-70B-Instruct									
Baseline	45.47	41.48	24.36	60.17	70.31	58.20	**48.25**	–	1.00×
+ ART	45.99	38.42	24.41	60.27	69.53	56.05	**47.48**	-0.77	**1.15**×
StrLLM(0.8)	33.97	35.81	25.16	59.53	50.49	59.61	**42.31**	–	1.13×
+ ART	33.68	34.77	24.69	59.44	50.23	58.38	**41.80**	-0.51	**1.27**×
StrLLM(0.2)	20.54	32.19	23.11	58.23	16.37	63.06	**33.88**	–	1.52×
+ ART	20.39	31.47	24.00	56.64	16.25	62.52	**33.51**	-0.37	**1.66**×
SnapKV(0.8)	38.83	40.48	24.41	57.10	70.41	60.93	**46.49**	–	1.08×
+ ART	36.36	39.77	25.72	55.83	69.19	59.09	**45.54**	-0.95	**1.18**×
SnapKV(0.2)	24.70	33.19	21.67	48.38	32.03	55.56	**34.24**	–	1.47×
+ ART	22.97	32.31	22.30	47.79	31.51	54.70	**33.59**	-0.65	**1.58**×
PyramidKV(0.8)	45.27	41.93	24.87	56.45	46.76	55.80	**44.10**	–	1.12×
+ ART	44.42	40.58	24.50	54.20	48.06	55.27	**43.31**	-0.79	**1.25**×
PyramidKV(0.2)	28.64	44.07	22.65	49.72	28.91	43.74	**35.93**	–	1.47×
+ ART	27.76	43.70	22.41	47.92	30.47	43.17	**35.47**	-0.46	**1.60**×