Coverage-Verified Sparse Attention: Closed-Loop Quality Control for Long-Context LLM Decoding

Yan-Cheng Li
National Taiwan University
r15943142@ntu.edu.tw

Chun-Yi Lee
National Taiwan University
cylee@csie.ntu.edu.tw

Abstract

Sparse attention accelerates long-context decoding by reading a subset of the KV cache, but existing methods are open-loop: the kernel commits to its selection with no signal indicating whether enough attention mass was preserved and no recovery path when it was not. The needed signal already exists inside the kernel. The fraction of softmax mass on the selected tokens, which we call *coverage*, satisfies an exact algebraic equality with per-step output error and serves a dual role as the acceptance criterion and the interpolation weight for recovery. *Coverage-Verified Sparse Attention* (CVSA) turns coverage into a training-free, closed-loop verify-then-correct decode loop that accepts high-coverage drafts and recovers only the heads that fall short, with no sampling, no draft model, and no speculative-decoding infrastructure. Quality varies by only 0.14 points across the threshold sweep; budget and scheduling ablations characterize the deployed operating point. On LongBench [1] at 7B and 70B, CVSA comes within 1.13 points of dense attention, statistically matching it at 70B, and it loses 0.8 points on RULER; measured median decode-step speedup reaches $6.21\times$ at 128K context.

1 Introduction

Sparse attention accelerates long-context language model decoding by reading only a subset of cached KV tokens per step, since loading the KV cache becomes the main decoding bottleneck at long context [2]. Existing methods, however, commit to this subset without checking whether it captured enough attention mass: the kernel selects, attends, and emits, with no mechanism to detect or correct a bad choice. In autoregressive decoding, even a single unchecked error compounds across the generation, as each step conditions on all previous outputs. The compounding effect is amplified by an asymmetry in where failures occur: at tight budgets, a small minority of attention heads loses most of its softmax mass while the majority retains high coverage, yet a global budget that protects the fragile minority over-pays on the comfortable majority. Without a per-step quality readout, the kernel cannot distinguish these two regimes or respond differently to each. Current sparse attention is therefore open-loop: it lacks a verification signal that indicates whether the chosen subset was adequate, together with a recovery path for the cases where it was not.

The open-loop pattern persists because research has focused on improving the upstream selection rule rather than adding downstream verification. Fixed-budget methods [3–7] each propose a different scoring heuristic, and per-head budget adaptation further raises selection precision, yet every method follows the same protocol: select once per step, attend, emit, with no post-hoc quality check. A complementary line, exemplified by Tactic [8] and Twilight [9], replaces the fixed budget with a target attention mass and selects tokens until that mass is reached. The target-mass approach partially addresses the concentrated-failure problem but not output quality, yet the mass estimate must be obtained *before* the kernel runs through quantized auxiliary caches or distribution-fitting heuristics, and the resulting output is still emitted without verification. Concurrent methods that do close part of the loop, such as vAttention [10] and Vegas [11], achieve verification at the cost of a

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

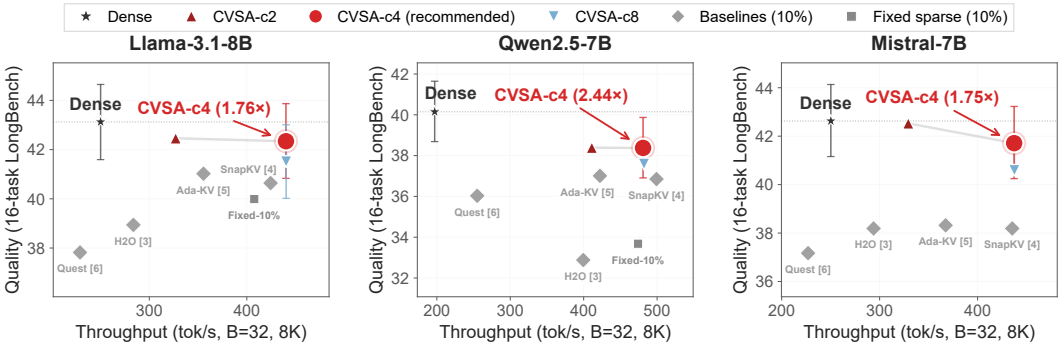


Figure 1: Quality–throughput frontier on three 7B models (LongBench 16-task average, 8K context, batch 32). Each marker represents one sparse-attention method at a fixed operating point; the x-axis is tokens/s derived from median decode-step latency and the y-axis is the benchmark score. Open-loop baselines are grey markers. CVSA combines higher quality with competitive speed; SnapKV is faster on Qwen. Coverage checks and recovery run every M steps.

sampling estimator or a full speculative-decoding pipeline. A quality signal that requires no auxiliary infrastructure and operates inside the standard decode loop has not yet appeared.

The signal that the preceding discussion calls for turns out to already exist inside the sparse-attention path, hidden in numerical by-products that the decode loop computes and then discards. The relevant quantity is the fraction of softmax mass that lands on the selected tokens, which we call *coverage*. High coverage means the sparse output closely approximates the dense one; low coverage pinpoints exactly which heads should receive additional compute. Recent adaptive-budget methods use coverage as a *pre-hoc target*: they estimate it before the kernel runs and select tokens until the estimate reaches a desired level [8, 9]. When read *after* the kernel has run rather than estimated beforehand, the same quantity costs nothing to evaluate: it reduces to a single log-sum-exp comparison between values the kernel already produces and adds under two percent to decode latency. Post-hoc coverage is also more than a heuristic proxy: at the token level, it satisfies an exact algebraic equality with output error, and in the paged implementation a lightweight proxy preserves this relationship with empirically validated fidelity across three architectures. Coverage therefore serves simultaneously as an acceptance criterion and as the interpolation weight for output recovery. The signal that existing sparse-attention kernels discard after every step is, in fact, a built-in quality readout.

Building on this signal, we introduce *Coverage-Verified Sparse Attention* (CVSA), a training-free framework that treats every sparse output as a draft and verifies it against coverage before committing. Unlike closed-loop attempts that require a sampling estimator or a speculative-decoding pipeline, verification in CVSA adds no dense pass and no auxiliary model: the verifier is the coverage scalar the kernel already computes. Only the minority of heads whose coverage falls short enter a lightweight recovery step, so amortized overhead stays close to plain sparse attention. Across three 7B models on LongBench [1], CVSA comes within 1.13 points of dense attention (Table 1), with median decode-step speedup reaching $6.21\times$ at 128K context. The framework scales to 70B parameters, where it statistically matches dense attention on LongBench. On the three 7B models, it narrows the RULER gap by nearly an order of magnitude relative to the strongest prior method. Quality is stable across a wide acceptance-threshold range, with only 0.14 points of variation on the three-model average. Long chain-of-thought reasoning is out of scope (Section P). The contributions are:

- Coverage as a free quality signal for sparse attention, grounded in an exact token-level algebraic equality between coverage and output error rather than a probabilistic bound or a sampling estimate.
- A verify-then-correct framework that wraps around off-the-shelf page selectors, accepts reliable sparse drafts, and applies lightweight recovery only on the heads that fail a per-head coverage check, all inside the standard decode loop with no auxiliary model.
- A stable operating point: quality varies by only 0.14 points across the threshold sweep. Fixed-5% trails dense by 5.48 points, and CVSA-c4 exceeds the Fixed-12% control by 3.57 points at similar effective read (Table 7); aligned controls show this margin also reflects refresh cadence and dense-layer placement (Table 19).
- LongBench quality within 1.13 points of dense at 7B and statistically matching dense at 70B, a 0.8-point RULER gap at 7B, and median decode-step speedup up to $6.21\times$ at 128K context.

2 Preliminaries

2.1 Attention Computation and the LSE By-Product

Autoregressive decoding in a decoder-only Transformer requires, at every step, a full read of the KV cache to compute one attention output. For a query $q \in \mathbb{R}^d$ and cached keys and values $K, V \in \mathbb{R}^{n \times d}$, the dense output is $O_{\text{dense}} = \sum_{i=1}^{n} \alpha_i v_i$ with weights $\alpha_i = \mathrm{softmax}(q^\top k_i/\sqrt{d})_i$. Fused-attention kernels [12] evaluate this sum in a single pass over the cache and, as a numerical side effect of the online softmax reduction, also produce the log-sum-exp $\mathrm{LSE} = \log \sum_{i=1}^{n} \exp(q^\top k_i/\sqrt{d})$ without another pass over the cache. The overall bandwidth cost of the cache read scales linearly with context length and is dominated by the KV component, an imbalance that grouped-query attention (GQA) [13] magnifies by sharing each KV head across multiple query heads (ratios of $4{:}1$, $7{:}1$, and $8{:}1$ in the models evaluated in this work). Per-head KV traffic is consequently the binding resource in decode-step latency, yet standard sparse-attention kernels allocate every head the same token budget.

2.2 Paged KV Cache and Sparse Attention

Production serving stacks such as vLLM [14] organize the KV cache into fixed-size pages of 16 tokens, and practical sparse-attention methods therefore operate at page granularity. A sparse-attention kernel reads a selected subset of pages S_{sel} and returns a draft output O_{draft} together with a partial log-sum-exp $\mathrm{LSE}_{S_{\text{sel}}}$ restricted to the selected tokens. Which pages enter S_{sel} is decided by *page scoring*: each page receives a score $\max_{j \in \text{page}}(q^\top k_j/\sqrt{d})$, and the top-$k$ pages by score form S_{sel}. Page scoring reads all cached keys, dominates each sparse-attention step, and grows linearly with context length. The pages excluded from S_{sel} form the complement set S_{comp}, and their contribution to the dense output is denoted O_{comp}. Together, O_{dense}, O_{draft}, and O_{comp} satisfy a partition identity over S_{sel} and S_{comp} whose role in approximation quality analysis is formalized in Section 3.2.

3 Methodology

Section 3.1 formulates sparse-attention quality control as a per-head decision problem and identifies coverage as the decision variable; Section 3.2 proves two exact token-level identities whose implications extend to the recovery mechanism. Section 3.3 translates these results into a closed-loop pipeline, and Sections 3.4 and 3.5 complete the system with scheduling and dense-layer policies.

3.1 Problem Definition and Coverage

Sparse attention produces a draft output O_{draft} that may differ from the dense output O_{dense}, and the magnitude of this difference varies across heads, layers, and decode steps in ways that depend on the input and are not known in advance. The quality-verification problem is to determine, at runtime and without ground-truth access, for each head independently, whether O_{draft} is an acceptable approximation of O_{dense}. The central constraint is that O_{dense} itself is unavailable, as computing it would eliminate the latency benefit that motivated sparse attention in the first place. A useful verifier thus must rely on quantities that the sparse-attention path already produces as part of its normal operation.

The partition identity reviewed in Section 2.2 points to exactly such a quantity. That identity decomposes O_{dense} into contributions from S_{sel} and S_{comp}, and the scalar governing this decomposition is the *coverage* $c = \sum_{i \in S_{\text{sel}}} \alpha_i$, the fraction of dense softmax mass captured by the sparse selection (α_i as defined in Section 2.1). High coverage means the draft is close to O_{dense}; low coverage flags heads where the selection missed important attention mass. This definition is token-level; the deployed system uses a page-level proxy validated in Section 3.2. Whether coverage is backed by a formal quality guarantee is the question that Section 3.2 addresses. For coverage to function as a practical quality signal, three conditions should hold: sparse-attention failures should be concentrated in a small minority of heads rather than spread uniformly, the coverage value should track the output error predicted by the token-level identity, and the low-coverage tail should be separable enough to target with recovery. Fig. 2 evaluates these conditions across three 7B model families and confirms that each is met, with the proxy limitations on Qwen quantified in Section C. The concentrated-failure pattern is particularly informative: it indicates that a verify-then-correct design can confine recovery to a small fraction of heads at each step and keep amortized cost close to that of plain sparse attention.

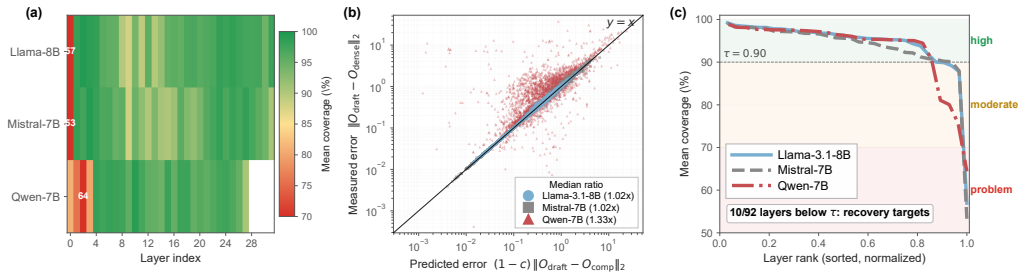


Figure 2: Empirical prerequisites for coverage-based verification, evaluated on Llama-3.1-8B, Mistral-7B-v0.3, and Qwen2.5-7B at a 5% page budget. **(a)** Per-layer attention mass retained under sparse selection (y-axis: fraction of dense softmax mass captured). Most layers retain high coverage; a small, model-dependent minority drops sharply, confirming that sparse failures are concentrated rather than uniform. **(b)** Measured sparse-output error (y-axis) vs. the error predicted by Proposition 2 (x-axis), evaluated per head-step. The page-level proxy over-accepts on all three models, most harmfully on Qwen through a known drift (Section C). **(c)** Sorted-rank view of per-layer coverage exposes a thresholdable long-tail structure: only the small low-coverage minority enters the correction path.

3.2 Theoretical Analysis

Coverage is not merely a useful heuristic: the two propositions below establish it as the exact interpolation weight between the sparse draft and the complement output, and show that it exactly determines per-step output error. These results hold at the token level under the softmax-attention setup in Section 2.1, with design implications that extend to the recovery mechanism in Section 3.3.

Proposition 1 (Exact Decomposition)**.** *For a selected subset S_{sel} with token-level coverage $0 < c < 1$ and complement S_{comp},*

$$O_{dense} \;=\; c \cdot O_{draft} \;+\; (1-c) \cdot O_{comp}\,, \tag{1}$$

where $O_{draft} = \frac{1}{c}\sum_{i\in S_{sel}} \alpha_i v_i$ and $O_{comp} = \frac{1}{1-c}\sum_{i\in S_{comp}} \alpha_i v_i$.

The proof partitions the softmax sum into contributions from S_{sel} and S_{comp} and renormalizes each branch (see Section B). Eq. (1) reveals that the dense output is a convex combination of the draft and the complement, with coverage as the mixing coefficient. Recovery from an insufficient draft therefore does not require recomputing dense attention from scratch. It suffices to compute the complement output O_{comp} and blend with the known weight $(1-c)$. This converts a binary accept-or-recompute decision into an interpolation whose cost scales with how much mass the draft missed.

Proposition 2 (Exact Error Identity)**.** *Under the same conditions as Proposition 1,*

$$\|O_{draft} - O_{dense}\|_2 \;=\; (1-c) \cdot \|O_{draft} - O_{comp}\|_2 \tag{2}$$

holds as an exact equality.

Proposition 2 follows from rearranging Eq. (1) (see Section B). The per-step approximation error of the sparse draft is the product of two factors: $(1-c)$, which measures how much attention mass the selection missed, and $\|O_{\text{draft}} - O_{\text{comp}}\|_2$, which measures how different the selected and omitted content are. When coverage is high, the prefactor $(1-c)$ is small and suppresses the error regardless of the content discrepancy, which justifies accepting the draft without further computation. When coverage is low, the large prefactor amplifies the discrepancy, and the identity prescribes exactly how much correction is needed through the decomposition in Eq. (1). Coverage thus plays a dual role that no prior sparse-attention signal provides: it is simultaneously the acceptance criterion (is c above τ?) and the recovery weight (blend with coefficient $1-c$). A direct corollary tightens the operational rule: if $c \geq \tau$ for some threshold $\tau \in (0,1)$, then $\|O_{\text{draft}} - O_{\text{dense}}\|_2 \leq (1-\tau)\|O_{\text{draft}} - O_{\text{comp}}\|_2$, which provides multiplicative per-step error control whose tightness is set by the threshold alone.

The dual role established above holds exactly at the token level. The deployed system, however, operates at page granularity and uses the page-level proxy introduced in Section 3.1, which raises the question of whether the theoretical guarantees transfer to the practical verifier. Section C provides the full validation: under token-level coverage, predicted and measured errors match with median ratio 1.0 on all three models; the page-level proxy over-accepts on all three models, most harmfully on Qwen through a known drift, quantified through false-accept and false-reject rates. The distinction is

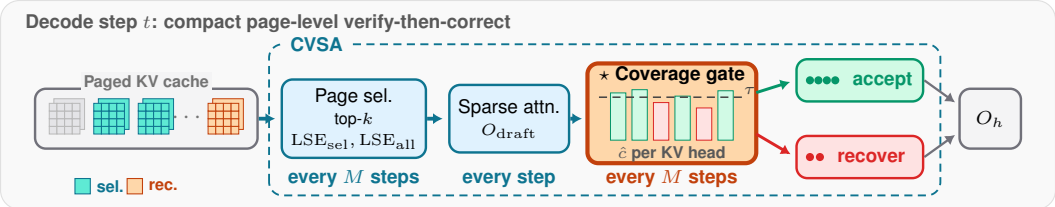


Figure 3: Closed-loop decode step in CVSA. Unlike open-loop sparse attention, which commits every sparse output without feedback, CVSA interposes a per-head coverage gate between the sparse kernel and the final output. **Draft**: an off-the-shelf page selector produces O_{draft} and the LSE statistics from which coverage is computed. **Verify**: the coverage gate compares each KV head's coverage against τ; most heads pass (green) while a small minority fails (red). **Correct**: accepted heads commit the draft at no additional cost; rejected heads expand the page budget and recompute attention. Page scoring and the coverage check run together every M steps (Section 3.4); sparse attention runs on every step, reusing the selected pages in between.

accordingly precise: the theoretical claim is token-level and exact, while the practical verifier is a page-level approximation whose accuracy is measured and reported rather than assumed.

3.3 Framework Overview

The propositions establish that coverage plays a dual role in sparse attention: it decides whether to accept a draft and prescribes the interpolation weight for correcting one that falls short. CVSA translates this dual role into a closed-loop decode procedure with three phases: **Draft**, **Verify**, and **Correct**, as illustrated in Fig. 3. In the **Draft** phase, an off-the-shelf page selector produces a sparse output and the LSE values from which coverage is estimated. This phase is identical to the standard sparse-attention path, and the entire CVSA framework is training-free: it wraps around any existing paged selector without modifying the model, the weights, or the serving stack. In the **Verify** phase, the coverage gate evaluates each head's coverage against a threshold τ: heads above τ commit the draft directly, and heads below τ are routed to recovery. In the **Correct** phase, rejected heads recompute attention over an expanded page set; Section F compares this with the exact $(1-c)$ correction of Proposition 1. The per-head granularity of this decision is central to efficiency: sparse failures are localized across heads and layers, and confining recovery to the minority that actually fails keeps amortized cost close to that of plain sparse attention. Two additional design choices address concerns that span multiple steps or affect specific layers: the hybrid schedule (Section 3.4) runs page scoring and the coverage check every M steps to bound how long a stale selection is reused, and the dense-layer policy (Section 3.5) removes a structural outlier whose coverage is persistently low regardless of input.

3.4 Hybrid Scheduling

Each coverage check compares the log-sum-exp of the selected pages' scores with that over all pages, and both come from the page-scoring step, which reads all cached keys and whose cost grows linearly with context length. CVSA therefore runs page scoring and the coverage check together once every M decode steps: the refreshed scores check the coverage of the cached selection, which is kept if every head still passes; otherwise the pages are re-selected and the heads that still fall short enter recovery. On the intervening steps, sparse attention reuses the selected pages without a check. Reuse introduces temporal staleness, which the schedule bounds: a head whose coverage drifts below τ within the reuse window is caught at the next check rather than continuing on a stale selection. The scoring interval M controls the trade-off between throughput and selection freshness: a smaller M reduces drift at the cost of more frequent key reads, while a larger M amortizes scoring more aggressively but lets a stale selection run longer before the next check. CVSA uses $M{=}4$ as the 7B default, as this setting captures essentially all measured throughput gains while incurring only half the quality cost of $M{=}8$ (Section 4.3). This design directly addresses the error-compounding concern from the introduction: a page selection that was adequate when chosen can go stale as decoding progresses, and the coverage check catches this drift by the next scoring step.

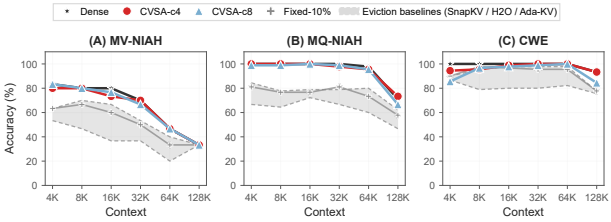


Figure 4: RULER retrieval stress test: accuracy on MV-NIAH, MQ-NIAH and CWE from 4K to 128K context, averaged over three 7B models. CVSA-c4 tracks dense; the grey band spans the open-loop baselines at 10% budget (Section J).

Table 1: LongBench 16-task mean (three 7B models) by design family. Baselines use a 10% nominal budget; CVSA-c4 reads 11.7% (measured).

Family	Method	Avg	Gap
Dense		41.83	—
Query-dep.	**CVSA-c4**	**40.70**	−1.13
	Quest	37.01	−4.82
Eviction	AdaKV	38.78	−3.05
	SnapKV	38.56	−3.27
	H2O	36.67	−5.16

3.5 Dense Layers for Low-Sparsity Attention

The hybrid schedule handles temporal drift in page selections across decode steps. A distinct failure mode, however, is structural rather than temporal: certain layers exhibit persistently low coverage regardless of the input, the decode step, or the page budget. Layer 0 is the clearest example. Early layers attend far less sparsely than the rest of the network: Tang et al. [6] measure below 10% sparsity in the first two layers and exclude them from sparse attention. Paged sparse attention consistently fails to capture layer 0's mass at tight budgets, and the per-layer coverage landscape across all three model families confirms that layer 0 is a universal outlier (Section H). Leaving layer 0 to recovery would trigger avoidable rejections at every check, and CVSA therefore keeps it dense and applies verify-then-correct to all remaining layers. The dense mask is limited to this single layer, as other low-coverage regions are less consistent across model families and tasks, and expanding the mask would create an architecture-specific policy that spends dense compute unnecessarily. The coverage gate is designed for these variable cases: layers that sometimes fail are caught by the per-head check, while the one layer that always fails is handled by the fixed policy. Section 4.3 validates both choices empirically.

4 Experimental Results

4.1 Experimental Setup

We evaluate three 7B instruction-tuned models (GQA 4:1 for Llama-3.1-8B and Mistral-7B-v0.3, 7:1 for Qwen2.5-7B) and Llama-3.1-70B (8:1) on LongBench [1] (16 tasks, 50 samples; seven tasks at 70B, Section K) and RULER [15] (4K–128K). The primary baseline is Quest [6], a query-dependent selector that, like CVSA, retains the full cache; a reproduction of Tactic's attention-mass-target principle [8] and Twilight [9] are compared in Section O. H2O [3], SnapKV [4], and AdaKV [5] discard tokens and therefore also save memory, which CVSA does not. All baselines run at a 10% nominal budget. CVSA-c4 uses a 5% base budget, τ=0.90, M=4, 4× recovery budget, and dense layer 0, reading 11.7% of the KV cache (rounded: base 5.0, recovery 3.7, layer 0 2.9). Speedups are median decode-step latency ratios on H100 (7B), H200 (long-context Qwen), and 4×H200 (70B); implementation and statistical details are in Sections C and Q.

4.2 Comparison with Sparse-Attention Baselines

We compare CVSA-c4 with the open-loop baselines on end-task quality (LongBench), long-range retrieval (RULER), and decode speed, and then examine how the default schedule transfers to 70B.

LongBench quality. Coverage-based verification closes the quality gap that upstream selection improvements alone cannot close. Table 1 reports the three-model LongBench average: CVSA-c4 is 1.13 points below dense, whereas the strongest prior baseline, AdaKV, is 3.05 points below. Unlike AdaKV, which adapts per-head budgets during selection, CVSA checks coverage after sparse attention and recovers the heads that fall short. Paired bootstrap intervals detect small losses on all three models, largest on Qwen (Section C). Raising the baseline budgets does not close the gap (Fig. 7): sweeping them from 10% to 20% closes at most 1.0 point, and CVSA-c8 at a strictly matched 9.9% effective budget still exceeds the strongest paged baseline by 2.78 points (Section N).

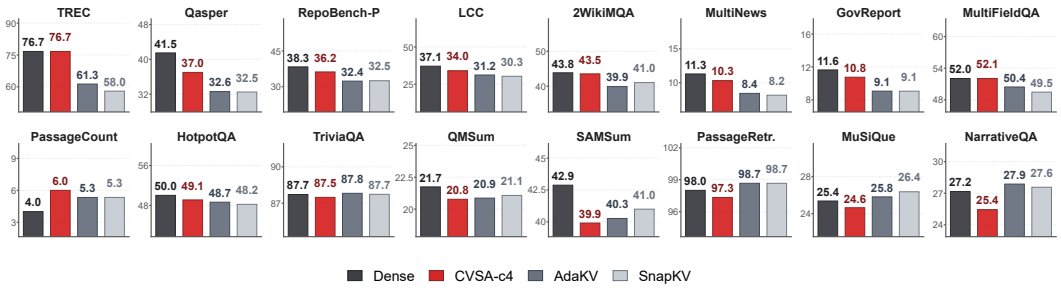


Figure 5: Per-task LongBench scores (three-model mean), sorted by CVSA-c4's gain over the better of AdaKV and SnapKV. Extractive and code tasks lose most; QA and retrieval stay near dense.

Table 2: Median decode-step speedup at 8K, batch 32. Eff. budget is measured on this benchmark (11.7% on LongBench).

Method	Eff. budget	Llama	Mistral	Qwen
CVSA-c4	**12.0%**	**1.76×**	**1.77×**	**2.50×**
SnapKV	10%	1.72×	1.74×	2.53×
H2O	10%	1.72×	1.17×	2.03×
AdaKV	10%	1.63×	1.47×	2.14×

Table 3: Median decode-step speedup on Qwen2.5-7B (H200, batch 8); c4 reads 11.7% on LongBench.

Context	Batch	CVSA-c4
32K	8	3.07×
65K	8	4.44×
128K	8	**6.21×**

Table 4: Llama-3.1-70B, LongBench 7-task subset (samsum excluded, Section K). CVSA-c2 statistically matches paged dense (CI overlaps zero). CVSA-c2/c4 refresh page scores every 2/4 steps. AdaKV and SnapKV Δ are vs. KVPress dense (54.50).

Method	Dense	CVSA-c2	CVSA-c4	AdaKV	SnapKV
Avg	52.02	**52.63**	49.76	54.08	53.33
Δ	—	**+0.60**	−2.27	−0.41	−1.17

Table 5: Scoring-interval sweep on the three 7B models. M=4 is the operating point, capturing essentially all of the speedup at half the quality cost of M=8. Eff. budget as in Table 2 (11.7% on LongBench).

Setting	Eff. budget	Speedup	Gap
c2	16.1%	1.57×	−0.43
c4 default	12.0%	1.98×	−1.13
c8	9.9%	1.99×	−2.04

Residual losses are concentrated (Fig. 5): extractive and code-heavy tasks (Qasper −4.48, LCC −3.04, SAMSum −2.97) lose more because a tight base budget can miss a critical span before recovery has evidence to correct the selection, whereas QA and retrieval tasks remain close to dense, indicating that verification prevents broad quality collapse.

Long-range retrieval. Fig. 4 reports RULER accuracy on the three sub-tasks that differentiate methods (MV-NIAH, MQ-NIAH, CWE) from 4K to 128K context. CVSA-c4 tracks dense across context lengths; averaged over the six RULER tasks at 4K–32K and three models, it loses 0.8 points relative to dense, compared with 8.7 points for the strongest baseline (Section J).

Decode speed. These quality gains do not come at the cost of speed. At 8K context with batch size 32 (Table 2), a conservative regime in which weight reads and other decode overheads still limit the attainable speedup, CVSA-c4 is the fastest method on Llama and Mistral and within 0.03× of SnapKV on Qwen. As context grows and KV reads dominate decode-step latency, the median decode-step speedup grows monotonically, reaching 6.21× at 128K on Qwen2.5-7B (Table 3). This trend follows the bandwidth-roofline model (Section L): as the KV-traffic fraction f_{kv} approaches unity, reducing the number of pages read translates into a proportionally larger end-to-end gain. Fig. 1 summarizes the trade-off: CVSA-c4 attains higher quality than every evaluated baseline, while SnapKV is faster on Qwen.

Scaling to 70B. On the seven-task LongBench subset (Table 4), CVSA-c2 statistically matches paged dense attention (Δ=+0.60, CI $[-0.79, +1.95]$), whereas CVSA-c4 loses 2.27 points. This reflects the reuse window rather than the gate, which still detects unreliable heads: with 80 layers, per-layer drift compounds more between scoring refreshes, so deeper models need a shorter window,

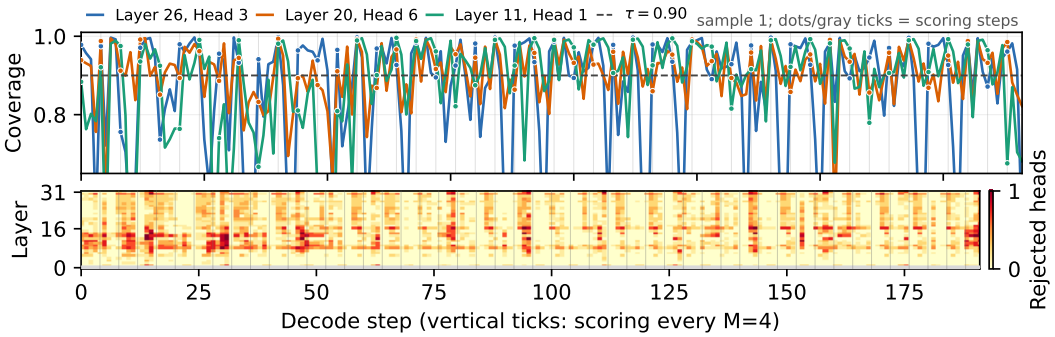


Figure 6: Runtime coverage trace under hybrid scheduling (M=4, τ=0.90, 5% budget) on Llama-3.1-8B, HotpotQA, 8K context. **Top**: per-head coverage (y-axis) across 200 decode steps (x-axis) for selected KV heads; vertical marks indicate scoring steps where page selections are refreshed. Coverage jumps at scoring steps and can drift between them; heads found below τ (dashed line) at a check enter recovery. **Bottom**: per-layer rejection fraction by decode step. Rejection is concentrated in a few layer-step regions, which is consistent with the static concentrated-failure pattern in Fig. 2 (a).

Table 6: Compact verification ablation on LongBench (three 7B models; CVSA-c4 reads 11.7% of the KV cache). **(a)** Quality and gap to dense. **(b)** Spread (max−min) of CVSA-c4 quality over $\tau \in [0.80, 0.97]$; Avg is the spread of the three-model mean.

(a) Quality versus dense

Configuration	Llama	Mistral	Qwen	Avg	Gap
Dense	42.57	42.73	40.18	41.83	—
CVSA-c4, τ=0.90	41.87	41.71	38.54	40.70	−1.13
Fixed-5%, no gate	39.00	37.57	32.49	36.35	−5.48

(b) Sensitivity to $\tau \in [0.80, 0.97]$

Statistic	Llama	Mistral	Qwen	Avg
CVSA-c4 spread (max−min)	0.38	0.25	0.32	0.14

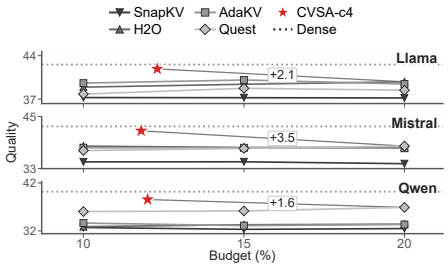


Figure 7: Budget fairness on LongBench, one panel per 7B model. Curves: baselines at 10–20% nominal budget; dotted: dense; star: CVSA-c4 at its 11.7% effective budget; boxes: its margin over the best baseline at 20%.

and we use M=2 at 70B. The 70B baselines run in the KVPress [16] pipeline, whose dense reference is 2.48 points above paged dense (Section K); we therefore compare methods within each pipeline.

4.3 Ablation Studies

Fig. 6 provides the runtime view: coverage is refreshed at scoring steps and can drift during the reuse window. At the next check, heads below τ are re-selected and, if still short, recovered, while the rest stay on the sparse fast path. The bottom panel confirms that rejection is concentrated in a few layer-step regions, consistent with the static pattern in Fig. 2, panel a, and the multi-model recovery heatmaps in Fig. 11 confirm this across all three architectures (mean recovered-head fraction: 21% Llama, 23% Mistral, 14% Qwen). Recovery is therefore not a hidden dense fallback: the verification loop spends extra compute selectively, keeping amortized throughput close to plain sparse attention.

Table 6 characterizes threshold sensitivity. Moving τ across $[0.80, 0.97]$ changes the average by only 0.14 points. Fixed-5% trails dense by 5.48 points; its gap to CVSA-c4 is 4.35 points across the reported configurations. The framework is selector-agnostic: replacing the uniform selector with an adaptive allocator changes quality by only 0.06 points when verification is active (Section O).

Verification configurations at similar effective read. Table 7 compares configurations in the same paged pipeline (11.67% effective read for CVSA-c4 versus 12.31% for Fixed-12% on Llama); CVSA-c4 leads the uniform and offline controls by +3.57 and +2.21. These controls also differ in refresh cadence (16 versus 4 steps); the uniform control runs layer 0 sparse. The margins therefore do not isolate verification. With four-step refresh and dense layer 0, additional ungated controls reach

Table 7: LongBench at similar effective read: CVSA-c4 gains $+3.57$ over Fixed-12% and $+2.21$ over the offline control. Controls use 16-step refresh versus four for CVSA-c4; Fixed-12% also lacks dense layer 0. Aligned controls are in Section F. CVSA-c4 reads 11.7% of the KV cache.

Configuration ($\sim$12% effective read)	Llama	Mistral	Qwen	Avg	Gap
Dense	42.57	42.73	40.18	41.83	—
Uniform Fixed-12%, no gate	39.67	38.08	33.65	37.13	-4.70
Offline per-(layer, head) budget, no gate	39.98	38.59	36.90	38.49	-3.34
CVSA-c4 (coverage gate)	41.87	41.71	38.54	40.70	-1.13

Table 8: Per-task gain of CVSA-c4 (11.7% read) over the Fixed-12% control of Table 7 (three-model average). The gain is positive on all sixteen tasks and averages $+3.57$; excluding the two largest contributors (samsum, repobench-p) it is still $+3.07$.

Task	Δ	Task	Δ	Task	Δ	Task	Δ
samsum	$+8.07$	multifieldqa_en	$+5.30$	lcc	$+2.23$	musique	$+1.59$
repobench-p	$+6.14$	hotpotqa	$+4.90$	triviaqa	$+2.17$	passage_count	$+1.03$
qasper	$+5.90$	2wikimqa	$+4.41$	multi_news	$+2.06$	gov_report	$+0.95$
passage_retrieval_en	$+5.50$	qmsum	$+4.22$	narrativeqa	$+2.01$	trec	$+0.67$

41.19 (uniform) and 41.03 (offline), versus 40.70 for CVSA-c4, at somewhat higher logged read (Section F). Table 8 reports the original per-task margins.

Design decisions. The comparisons above concern complete operating configurations. Three practical choices within the framework also require independent validation: the scoring interval M, the layer-0 dense policy, and approximate versus exact recovery. Table 5 reports the scoring-interval sweep: across $M \in \{2, 4, 8\}$, quality degrades gracefully while throughput saturates at $M \geq 4$, making CVSA-c4 the Pareto-optimal setting for 7B models. The 70B results confirm that deeper models benefit from $M=2$. Logged telemetry agrees: within a reuse window, mean coverage drops by 0.013 at $M=2$ and 0.047 at $M=8$ (two-sample Llama-3.1-8B HotpotQA trace), the input an adaptive schedule would use (Section P). The dense-layer policy contributes a further $+1.16$-point average gain, driven primarily by Qwen ($+3.00$), consistent with the layer-0 analysis in Section 3.5 (per-model breakdown in Section F). Note that approximate recovery (AR20) matches exact recovery to within 0.05 points on the three-model average (Section F), which confirms that the efficient AR path does not compromise the quality story established by the propositions.

5 Related Work

Open-loop sparse attention. Training-free sparse-attention methods reduce decode cost by choosing a fixed subset of the KV cache and committing to it without quality feedback. The methods differ in how they score and select: cumulative attention mass [3], prefill observation windows [4], per-head budget adaptation [5], query-aware page scoring [6], and semantic hashing [7]. Complementary systems address long-context inference through attention-sink windowing [17], external memory retrieval [18], head-type separation [19], prefill-specific sparse patterns [20], and trained sparse architectures [21]. All share a structural limitation: once a subset is selected, the model emits without a per-step signal that the retained attention mass was sufficient, and without a recovery path when it was not. CVSA targets this gap directly: the verification layer wraps any open-loop selector, checks coverage post-hoc, and recovers only the heads that fail. The ablations in Section 4.3 examine selector compatibility, budget, and scheduling within this framework.

Adaptive budgets and verification. Adaptive-budget methods move beyond fixed top-k selection. Tactic [8] and Twilight [9] estimate attention mass before the kernel runs and expand the selection until the estimate is reached, reducing some fixed-budget failures but leaving the emitted output unchecked. Concurrent verification-oriented work closes part of the loop: vAttention [10] employs a sampling estimator with probabilistic guarantees, and Vegas [11] couples sparse attention with a speculative-decoding pipeline. The pattern mirrors a broader trend: speculative decoding brought verify-then-accept to token generation [22, 23], and the same principle now extends to attention computation. The proposed CVSA instantiates this principle using log-sum-exp values over page scores the selector already computes, giving a deterministic post-hoc acceptance test and a direct

recovery weight without sampling, an auxiliary cache, or a second model. The result is a lighter verification mechanism: no additional KV reads for accepted heads, no stochastic estimator, and no speculative-decoding infrastructure. The exact algebraic equality underlying coverage (Proposition 2) provides a stronger theoretical basis than the probabilistic bounds of sampling-based approaches, while the dual role of coverage as both acceptance criterion and recovery weight is unique to CVSA.

KV compression and serving systems. KV-cache quantization [24, 25], grouped-query attention [13], and paged management in vLLM [14] reduce or reorganize KV-cache access. These techniques change the cost of each KV read, not the decision of whether a sparse subset captured enough attention mass on a particular step. They compose naturally with CVSA: the verification layer operates on top of whatever cache layout and precision the serving stack provides, and the coverage signal remains valid regardless of whether the underlying values are quantized or full-precision.

6 Conclusion

Coverage, the fraction of softmax mass on selected tokens, satisfies an exact token-level equality with output error and supplies the interpolation weight for exact recovery. CVSA uses a page-level proxy for closed-loop verification with existing paged selectors, without model modification or auxiliary infrastructure. Quality is stable across a wide τ range. Fixed-5% trails dense by 5.48 points, and CVSA-c4 exceeds the Fixed-12% control by 3.57 points at similar effective read (Table 7); aligned controls show this margin also reflects refresh cadence and dense-layer placement (Table 19). On LongBench at 7B scale, CVSA trails dense attention by 1.13 points; on RULER (six tasks, 4K–32K), it loses 0.8 points versus 8.7 for the strongest baseline. The same framework scales to 70B with a shorter scoring interval, where it statistically matches paged dense attention. Measured median decode-step speedup reaches $6.21\times$ at 128K context, Pareto-dominating every paged baseline except SnapKV on Qwen. Residual losses concentrate on extractive and code-heavy tasks where tight base budgets miss critical spans, which point to token-aware proxy calibration as the most promising direction for further improvement.

Limitations. The verifier relies on a page-level approximation of token-level coverage that tends to over-accept; the effect is most visible on Qwen, where false accepts concentrate in late layers; token-aware or GQA-aware calibration could address this (Section C). CVSA targets long-context decoding with short outputs; long chain-of-thought generation benefits from a shorter reuse window (Section P), and adaptive scheduling is a natural extension. The speedup grows with context length and batch size and is largest on Qwen ($6.21\times$ at 128K, $B{=}8$; Section M). Like Quest, CVSA reduces KV reads rather than KV memory, keeping the full cache available for recovery.

Acknowledgments and Disclosure of Funding

The authors gratefully acknowledge the support from the National Science and Technology Council (NSTC) in Taiwan under grant numbers NSTC 114-2221-E-002-069-MY3, NSTC 113-2221-E-002-212-MY3, NSTC 115-2218-E-A49-010, and NSTC 115-2640-E-002-014, as well as the support from the Academia Sinica Scholar Award (ASSA) under grant number AS-ASSA-115-02, NTU Artificial Intelligence Center of Research Excellence, and Taiwan Centers of Excellence in Artificial Intelligence. This research was also supported by the NVIDIA Academic Grant Program. The authors would also like to express their appreciation for the hardware grant donation of the GPUs from NVIDIA Corporation and NVIDIA AI Technology Center (NVAITC) used in this work. Furthermore, the authors extend their gratitude to the National Center for High-Performance Computing (NCHC) for providing computational and storage resources. The authors also thank the NVIDIA Taipei-1 supercomputer for providing essential computing resources.

References

[1] 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. *Annual Meeting of the Association for Computational Linguistics*, 2024.

[2] Piotr Nawrot, Robert Li, Renjie Huang, Sebastian Ruder, Kelly Marchisio, and Edoardo M. Ponti. The sparse frontier: Sparse attention trade-offs in transformer LLMs. *arXiv preprint arXiv:2504.17768*, 2025.

[3] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2023.

[4] 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, 2024.

[5] Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. *Advances in Neural Information Processing Systems*, 2025.

[6] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context LLM inference. *International Conference on Machine Learning*, 2024.

[7] Aditya Desai, Shuo Yang, Alejandro Cuadron, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. HashAttention: Semantic sparsity for faster inference. In *International Conference on Machine Learning (ICML)*, 2025.

[8] Kan Zhu, Tian Tang, Qinyu Xu, Yile Gu, Zhichen Zeng, Rohan Kadekodi, Liangyu Zhao, Ang Li, Arvind Krishnamurthy, and Baris Kasikci. Tactic: Adaptive sparse attention with clustering and distribution fitting for long-context LLMs. *arXiv preprint arXiv:2502.12216*, 2025.

[9] Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. Twilight: Adaptive attention sparsity with hierarchical top-p pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[10] Aditya Desai, Kumar Krishna Agrawal, Shuo Yang, Alejandro Cuadron, Luis Gaspar Schroeder, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. vAttention: Verified sparse attention. In *International Conference on Learning Representations (ICLR)*, 2026.

[11] Yikang Yue, Yuqi Xue, and Jian Huang. Vegas: Self-speculative decoding with verification-guided sparse attention. In *International Conference on Machine Learning (ICML)*, 2026.

[12] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 35, 2022.

[13] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. *Conference on Empirical Methods in Natural Language Processing*, 2023.

[14] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. *ACM Symposium on Operating Systems Principles*, 2023.

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

[16] Simon Jégou, Maximilian Jeblick, and David Austin. kvpress. `https://github.com/NVIDIA/kvpress`, 2024. NVIDIA software library.

[17] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *International Conference on Learning Representations*, 2024.

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

[19] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. *International Conference on Learning Representations*, 2025.

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

[21] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *Annual Meeting of the Association for Computational Linguistics*, 2025.

[22] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. *International Conference on Machine Learning*, 2023.

[23] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[24] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. *Advances in Neural Information Processing Systems*, 37, 2024.

[25] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. *International Conference on Machine Learning*, 2024.

Appendix

A CVSA Pseudocode

Algorithm 1 gives the deployed hybrid AR decode step for one layer. The gate uses page-level coverage. Reuse steps perform no check, and the base selection (not the expanded recovery set) is cached. When any head rejects, the batched implementation computes expanded attention and masks its outputs per KV head, including the query heads in that GQA group.

Algorithm 1 CVSA hybrid AR decode step for one layer

Require: query heads Q_t, paged KV cache (K, V), optional cached set $S_{\text{sel}t-1}$, step t, interval M, base budget B, threshold τ, recovery budget B_{AR}

if layer is in the fixed dense set **then**
  return dense attention(Q_t, K, V)
end if
if cache exists and $t \bmod M \neq 0$ **then**
  return sparse attention on $S_{\text{sel}t-1}$; keep the cached set
end if
score pages and compute the page-level all-context normalization
if cache exists and every head's cached-set page coverage is at least τ **then**
  return sparse attention on $S_{\text{sel}t-1}$; keep the cached set
end if
choose a fresh base set $S_{\text{sel}t}$ at budget B using the page scores
run sparse attention on $S_{\text{sel}t}$ to obtain O_{draft}
compute the page-level coverage $\hat{c}_h$ of $S_{\text{sel}t}$ for each KV head h
set $O_h \leftarrow O_{\text{draft},h}$ for all heads
if any head has $\hat{c}_h < \tau$ **then**
  choose an expanded set at budget B_{AR} and recompute attention
  replace O_h with its expanded-set output only where $\hat{c}_h < \tau$
end if
cache the base set $S_{\text{sel}t}$ and return concatenated head outputs

B Full Proofs of Propositions 1 and 2

We restate and prove the two propositions used in Section 3.1. The setup throughout is a single decode step with query $q \in \mathbb{R}^d$, cached keys and values $K, V \in \mathbb{R}^{n \times d}$, attention weights $\alpha_i = \text{softmax}(q^\top k_i/\sqrt{d}) \in (0,1)$ with $\sum_{i=1}^n \alpha_i = 1$, and a partition of token indices into a selected subset S_{sel} and its complement $S_{\text{comp}} = \{1, \ldots, n\} \setminus S_{\text{sel}}$. Coverage is defined as $c = \sum_{i \in S_{\text{sel}}} \alpha_i$, so $1 - c = \sum_{i \in S_{\text{comp}}} \alpha_i$.

The dense, draft, and complement attention outputs are

$$O_{\text{dense}} = \sum_{i=1}^{n} \alpha_i \, v_i, \tag{3}$$

$$O_{\text{draft}} = \frac{1}{c} \sum_{i \in S_{\text{sel}}} \alpha_i \, v_i, \qquad O_{\text{comp}} = \frac{1}{1-c} \sum_{i \in S_{\text{comp}}} \alpha_i \, v_i, \tag{4}$$

where O_{draft} and O_{comp} are the per-branch *normalized* attention outputs (so each is a softmax-weighted average over its respective index subset). We assume $0 < c < 1$ throughout; the boundary cases $c \in \{0, 1\}$ collapse trivially because one of the two branches is empty.

Proposition 1 (Exact Decomposition).

For any partition $S_{sel} \cup S_{comp} = \{1, \ldots, n\}$,

$$O_{\text{dense}} = c \cdot O_{\text{draft}} + (1 - c) \cdot O_{\text{comp}}.$$

Proof. Multiplying the definitions of O_{draft} and O_{comp} by their respective normalization constants returns the unnormalized partial sums:

$$c \cdot O_{\text{draft}} = \sum_{i \in S_{\text{sel}}} \alpha_i v_i, \qquad (1-c) \cdot O_{\text{comp}} = \sum_{i \in S_{\text{comp}}} \alpha_i v_i. \tag{5}$$

Summing the two equalities in (5) and using $S_{\text{sel}} \cup S_{\text{comp}} = \{1, \ldots, n\}$ with $S_{\text{sel}} \cap S_{\text{comp}} = \emptyset$,

$$c \cdot O_{\text{draft}} + (1-c) \cdot O_{\text{comp}} = \sum_{i \in S_{\text{sel}}} \alpha_i v_i + \sum_{i \in S_{\text{comp}}} \alpha_i v_i = \sum_{i=1}^{n} \alpha_i v_i = O_{\text{dense}}. \qquad \square$$

The proof uses no property of the value vectors v_i beyond linearity of the sum, and no property of the attention weights α_i beyond non-negativity and the partition $\sum_i \alpha_i = 1$. The decomposition therefore extends without modification to GQA (group-query attention), to multi-head attention, and to any selection rule for S_{sel} (top-k, sampled, learned, etc.); it is purely an algebraic identity on the softmax distribution.

Proposition 2 (Error Equality).

Under the same setup as Proposition 1,

$$\|O_{\text{draft}} - O_{\text{dense}}\|_2 = (1-c) \cdot \|O_{\text{draft}} - O_{\text{comp}}\|_2.$$

Proof. Subtract O_{draft} from both sides of Proposition 1:

$$\begin{aligned} O_{\text{dense}} - O_{\text{draft}} &= c \cdot O_{\text{draft}} + (1-c) \cdot O_{\text{comp}} - O_{\text{draft}} \\ &= (c-1) \cdot O_{\text{draft}} + (1-c) \cdot O_{\text{comp}} \\ &= (1-c) \cdot (O_{\text{comp}} - O_{\text{draft}}). \end{aligned} \tag{6}$$

Taking the Euclidean norm of both sides of (6) and using $1 - c \geq 0$ (so $|1-c| = 1-c$) together with $\|O_{\text{draft}} - O_{\text{dense}}\|_2 = \|O_{\text{dense}} - O_{\text{draft}}\|_2$ and $\|O_{\text{comp}} - O_{\text{draft}}\|_2 = \|O_{\text{draft}} - O_{\text{comp}}\|_2$,

$$\|O_{\text{draft}} - O_{\text{dense}}\|_2 = \|O_{\text{dense}} - O_{\text{draft}}\|_2 = \|(1-c) \cdot (O_{\text{comp}} - O_{\text{draft}})\|_2 = (1-c) \cdot \|O_{\text{draft}} - O_{\text{comp}}\|_2. \ \square$$

The proof yields an *equality*, not an inequality: $1 - c$ is the exact multiplicative factor relating the sparse–dense disagreement $\|O_{\text{draft}} - O_{\text{dense}}\|_2$ to the sparse–complement disagreement $\|O_{\text{draft}} - O_{\text{comp}}\|_2$, with no slack. Coverage is therefore a sufficient algebraic statistic for this identity: once c and the branch disagreement $\|O_{\text{draft}} - O_{\text{comp}}\|_2$ are known, the sparse–dense error is determined exactly.

Corollary (Acceptance Bound).

If $c \geq \tau$ for some threshold $\tau \in (0, 1)$, then

$$\|O_{\text{draft}} - O_{\text{dense}}\|_2 \leq (1-\tau) \cdot \|O_{\text{draft}} - O_{\text{comp}}\|_2.$$

Proof. By Proposition 2 and $c \geq \tau \Rightarrow 1 - c \leq 1 - \tau$,

$$\|O_{\text{draft}} - O_{\text{dense}}\|_2 = (1-c) \cdot \|O_{\text{draft}} - O_{\text{comp}}\|_2 \leq (1-\tau) \cdot \|O_{\text{draft}} - O_{\text{comp}}\|_2. \qquad \square$$

The corollary is the formal version of the token-level per-step acceptance guarantee: whenever an exact coverage verifier accepts ($c \geq \tau$), the sparse output is within $(1-\tau) \cdot \|O_{\text{draft}} - O_{\text{comp}}\|_2$ of dense. A simpler D_V-based form replaces $\|O_{\text{draft}} - O_{\text{comp}}\|_2$ with the value diameter $D_V = \max_{i,j} \|v_i - v_j\|_2$, yielding $\|O_{\text{draft}} - O_{\text{dense}}\|_2 \leq (1-\tau) \cdot D_V$. The D_V form is looser by roughly $14\times$ in our measurements; the equality of Proposition 2 is the primary theoretical and empirical result.

Table 9: Proxy calibration at $\tau = 0.90$, 5% budget. "Proxy accept" uses the page-level $\hat{c}$ used in code; "Token accept" uses token-level $c^\star$. False accept/reject rates are fractions of all cells.

Model	Cells	Proxy accept	Token accept	False accept	False reject	Agreement
Llama-3.1-8B	3840	0.0%	0.0%	0.0%	0.0%	100.0%
Mistral-7B-v0.3	3840	0.0%	0.0%	0.0%	0.0%	100.0%
Qwen2.5-7B	3360	12.3%	4.3%	9.4%	1.4%	89.2%

Implementation note: $c^\star$ vs. $\hat{c}$.

The proofs above use the algebraic coverage $c = \sum_{i \in S_{\text{sel}}} \alpha_i$ defined over the full token softmax distribution. In our paged-decode implementation we never materialize the full token-level attention scores; instead, we compute a page-max proxy $\hat{c} = \exp(\text{LSE}^{\text{page}}_{S_{\text{sel}}} - \text{LSE}^{\text{page}}_{\text{all}})$ that is a free by-product of the page-scoring kernel. Proposition 1 and Proposition 2 are stated for the true coverage c (or equivalently, the token-level $c^\star = \exp(\text{LSE}^{\text{tok}}_{S_{\text{sel}}} - \text{LSE}^{\text{tok}}_{\text{all}})$); the equality is validated under $c^\star$ on the three 7B models (4:1 and 7:1 GQA), while $\hat{c}$ is evaluated as the runtime proxy used by the paged system (Fig. 8 and Section 3.2).

C Bootstrap Confidence Intervals and Theory Validation

Proposition 2 validation under $c^\star$. Fig. 8 validates Proposition 2 on $132{,}480$ logged head-steps across three 7B architectures (Llama-3.1-8B, 4:1 GQA; Mistral-7B-v0.3, 4:1 GQA; Qwen2.5-7B, 7:1 GQA), using the token-level coverage $c^\star = \exp(\text{LSE}^{\text{tok}}_{S_{\text{sel}}} - \text{LSE}^{\text{tok}}_{\text{all}})$ rather than the page-max proxy $\hat{c}$ used in code. The median ratio of measured error to predicted error is exactly 1.0000 on all three models, confirming the equality holds empirically across GQA factors. The visible scatter on the Qwen panel is concentrated at small predicted error where $(1-c^\star) \to 0$ amplifies bfloat16 numerical noise; the median is unaffected.

D_V corollary. The D_V corollary $\|O_{\text{draft}} - O_{\text{dense}}\| \leq (1-c) \cdot D_V$ holds with zero violations on Llama and Mistral under both $\hat{c}$ and $c^\star$ ($46{,}080$ cells per model). On Qwen ($40{,}320$ cells), $\hat{c}$ gives 3.30% violations concentrated in late layers (L17 7.2%, L27 21.7%) where $\hat{c}$ saturates near 1 and the proxy bound $(1-\hat{c})D_V$ becomes spuriously tight; switching to $c^\star$ reduces violations to 0.60%. The residual is attributable to the sampled-D_V approximation (the implementation samples up to 200 value vectors per head to estimate the diameter, which can underestimate the true geometric diameter for heavy-tailed value distributions), not to corollary failure.

Proxy accept/reject calibration. Table 9 compares the runtime page-level accept decision against the token-level accept decision on the same theorem-validation grid, restricted to the paper's main threshold $\tau = 0.90$ and 5% base budget. This is a stress diagnostic rather than the production accept-rate distribution: it uses 12 HotpotQA prompts, next-token queries, and 10 selected layers per model. Within this grid, Llama and Mistral have no accept cells under either proxy or token coverage at $\tau = 0.90$, so the proxy makes no false accepts. Qwen is the important caveat: the page-max proxy accepts 413/3360 cells, while token coverage accepts 146/3360; 315 cells are false accepts (9.4% of all cells, 76.3% of proxy accepts) and 48 are false rejects (1.4% of all cells). The false accepts concentrate in late layers (especially L17/L23/L27), consistent with the Qwen proxy drift discussed in Section 3.2.

Production-distribution confusion matrix. The grid above is a controlled stress diagnostic; Table 10 reports the same accept/reject confusion matrix at the operating point (τ=0.90, 5% base) over the production distribution, a probe of $161{,}280$–$280{,}640$ head-steps per model, 48–$73\times$ the size of the stress grid. The proxy is optimistic on every model: false accepts are 31%, 33%, and 39% of proxy accepts on Llama, Mistral, and Qwen. What differs across models is the consequence rather than the rate: the median output error of a false accept is 0.19, 0.27, and 0.61 (absolute), and the catastrophic fraction (true coverage $c^\star < 0.5$) is 4%, 4%, and 14%. On Qwen these false accepts carry $6.4\times$ the output error of a true accept and concentrate late in the network, with layers ≥ 17 holding 51% of them, which is the mechanism behind the -1.64 Qwen gap in Table 6. False rejects are the benign direction: 8.4%, 10.2%, and 15.8% of proxy rejects (so 84–92% of rejects are

Table 10: Accept/reject confusion matrix over the production distribution (161,280–280,640 head-steps per model, 48–73× the stress grid of Table 9). The proxy over-accepts on all three models, but the harmful tail (high-error and catastrophic false accepts) is confined to Qwen and to late layers (≥ 17 hold 51%), consistent with 7:1 GQA pooling concealing the worst group member. False rejects are quality-harmless and their cost is already counted in the 11.7% effective read.

Production probe (τ=0.90, 5% base)	Llama-3.1-8B	Mistral-7B-v0.3	Qwen2.5-7B
False accepts (% of proxy accepts)	31%	33%	39%
FA output-error median (absolute)	0.19	0.27	0.61
Catastrophic FA ($c^\star < 0.5$)	4%	4%	14%
False rejects (% of proxy rejects)	8.4%	10.2%	15.8%

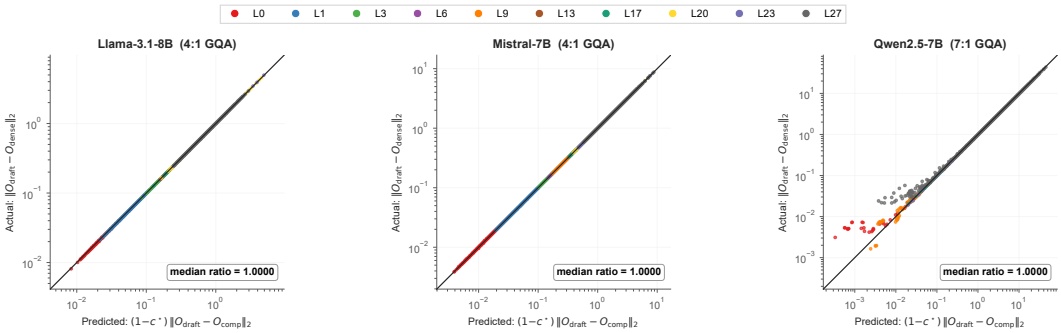


Figure 8: Empirical validation of Proposition 2 under token-level coverage $c^\star$, on three 7B architectures (12 prompts × 10 layers × 4 budgets × 3 thresholds). Predicted error $(1-c^\star)\|O_{\text{draft}} - O_{\text{comp}}\|_2$ matches measured error $\|O_{\text{draft}} - O_{\text{dense}}\|_2$ on the $y{=}x$ line per head-step; median ratio is 1.0000 on all three models, color-coded by layer. Qwen's tail scatter at small predicted error reflects bfloat16 numerical noise where $(1-c^\star)$ approaches zero; the equality holds exactly in the median.

Table 11: LongBench point estimates and bootstrap intervals for CVSA-c4 vs. dense (16 tasks, greedy decoding; n samples per task). Means are task-averaged sample scores. For the $n = 100$ rows, 10,000 resamples (NumPy default RNG, seed 42) draw paired example indices within each task, with tasks fixed. The 95% percentile intervals for Δ use the same draw for both methods; all three exclude zero. The $n = 50$ aggregate retains the main-table point estimates, without attaching an interval from the $n = 100$ experiment. CVSA-c4 reads 11.7% of the KV cache.

Model	Dense mean	CVSA-c4 mean	95% CI(CVSA)	95% CI(Δ)
3-model avg ($n = 50$)	41.83	40.70	—	—
Llama-3.1-8B ($n = 100$)	43.13	42.34	[40.83, 43.87]	[−1.27, −0.32]
Mistral-7B-v0.3 ($n = 100$)	42.63	41.71	[40.25, 43.23]	[−1.44, −0.38]
Qwen2.5-7B ($n = 100$)	40.15	38.38	[36.91, 39.87]	[−2.32, −1.24]

justified), with a median true coverage of 0.94–0.95; their only cost is recovery pages that the 11.7% effective-read accounting already includes.

Aggregate and per-model bootstrap intervals. Table 11 reports the bootstrap intervals used in the main text. The three-model main-table scores use 50 samples per task; the per-model intervals use the separate 100-sample evaluation. These bases are labeled explicitly. Marginal score intervals overlap, while paired Δ intervals exclude zero for all three models. The mean losses remain below two points.

Paired comparison against KVPress baselines. Table 12 reports paired bootstrap CIs for CVSA-c4 against the canonical KVPress SnapKV and AdaKV implementations on the shared 50 samples/task prefix. Five of six comparisons exclude zero; the remaining comparison (Llama vs. AdaKV) is positive but borderline.

Cross-pipeline dense matching. SnapKV and AdaKV quality numbers use KVPress in HuggingFace SDPA, while CVSA uses the paged-decode pipeline. Table 13 reports the dense reference in both pipelines. Llama matches within 0.2 pt; Mistral and Qwen show larger but still sub-2-pt drift,

Table 12: Paired bootstrap CIs for CVSA-c4 minus KVPress baselines on LongBench. Positive values favor CVSA; CVSA-c4 reads 11.7% of the KV cache.

Model	CVSA-c4 − SnapKV	CVSA-c4 − AdaKV
Llama-3.1-8B	+1.23 [+0.25, +2.26]	+0.85 [−0.15, +1.88]
Mistral-7B-v0.3	+3.52 [+2.35, +4.71]	+3.39 [+2.27, +4.54]
Qwen2.5-7B	+1.69 [+0.56, +2.86]	+1.53 [+0.44, +2.65]

Table 13: Dense-reference alignment between the paged-decode pipeline and KVPress/HuggingFace SDPA. CVSA-c4 reads 11.7% of the KV cache.

Model	Paged dense	KVPress dense	KVPress − paged
Llama-3.1-8B	42.57	42.70	+0.13
Mistral-7B-v0.3	42.73	41.03	−1.70
Qwen2.5-7B	40.18	39.27	−0.91

Table 14: Unified single-pipeline LongBench comparison (16 tasks, 50 samples per task, greedy decoding). All six methods run in the same paged-decode pipeline against one dense reference, removing the mixed-pipeline confound of Table 13. SnapKV and AdaKV are paged-decode adaptations of their released implementations and lose accuracy relative to their native token-level setting, which is why their margins widen here; the main-text comparison, where each baseline runs in its intended framework, is the conservative one for CVSA. CVSA-c4 reads 11.7% of the KV cache.

Method (all paged-decode)	Llama	Mistral	Qwen	Avg	Gap
Dense	42.57	42.73	40.18	41.83	—
CVSA-c4 (5% base)	41.87	41.71	38.54	**40.70**	**−1.13**
AdaKV (10%)	39.59	37.82	33.63	37.01	−4.82
Quest (10%)	37.82	37.17	36.03	37.01	−4.82
H2O (10%)	38.94	38.19	32.88	36.67	−5.16
SnapKV (10%)	35.96	33.09	31.99	33.68	−8.15

so we use paired within-pipeline comparisons where possible and avoid ranking 70B methods by absolute cross-pipeline score.

Single-pipeline rerun of all baselines. To remove the mixed-pipeline confound entirely, we re-ran SnapKV and AdaKV inside the same paged-decode pipeline as every other method, so that all six configurations share one dense reference (Table 14). Under this single yardstick CVSA-c4's margin over the strongest baseline grows from +1.92 to +3.70 points, which indicates that the mixed-pipeline table in the main text is conservative for CVSA rather than favourable to it. The margins for SnapKV and AdaKV widen because their token-level and prefill-observation mechanisms lose precision when adapted to 16-token pages, whereas coverage is page-native: it is read from the page scores that page selection already computes. We therefore keep the mixed-pipeline numbers in the main text, where each baseline runs in its authors' intended setting, and report this rerun as the stricter check.

Stability over long decode. The coverage traces in the main text cover a few hundred decode steps. To check whether the selection drifts over much longer generations, we ran a 32,768-step trace on Llama-3.1-8B with the KV cache growing from 8K to 41K tokens, in which the model decodes densely and the c4 schedule runs alongside it, logging coverage and recovery decisions. Across the entire second half of the trace (decode depths 16K to 32K), per-head coverage holds between 0.935 and 0.937 per 4,096-step bin and the recovery rate holds between 18.2% and 18.9%, with the within-window verification signature intact at the final step. Selection quality therefore settles at a stable operating point rather than degrading with decode depth, because the selection and the certificate are recomputed against the current query every M steps. Since the trace decodes densely, it measures selection quality, not whether sparse-output errors compound.

Stochastic decoding. All LongBench and RULER numbers in the paper use greedy decoding for reproducibility, which is standard in this family and applied identically to every method. The verification loop itself does not depend on that choice: coverage is computed from the current query's page scores and log-sum-exp before any token is drawn, so it is a property of the attention distribution

Table 15: LongBench per-task disclosure for the three 7B models. Dense and CVSA columns are three-model mean scores; Δ is CVSA-c4 minus dense. The last three columns show the per-model Δ, exposing where the aggregate loss comes from. CVSA-c4 reads 11.7% of the KV cache.

Task	Dense	CVSA	Mean Δ	Llama Δ	Mistral Δ	Qwen Δ
Qasper	41.47	36.99	-4.48	-4.69	-5.01	-3.73
LCC	37.08	34.04	-3.04	-3.44	-4.82	-0.86
SAMSum	42.87	39.90	-2.97	-1.67	-2.24	-5.00
RepoBench-P	38.33	36.21	-2.12	-1.58	-3.28	-1.50
NarrativeQA	27.17	25.44	-1.72	-1.65	-2.52	-1.00
MultiNews	11.29	10.31	-0.98	-0.30	-1.55	-1.08
QMSum	21.74	20.77	-0.97	$+0.01$	-0.03	-2.89
HotpotQA	50.02	49.11	-0.91	-0.37	-0.94	-1.42
GovReport	11.64	10.75	-0.89	$+0.00$	-0.79	-1.87
MuSiQue	25.37	24.64	-0.73	-0.55	$+0.09$	-1.72
Passage Retr.	98.00	97.33	-0.67	$+0.00$	$+2.00$	-4.00
2WikiMQA	43.85	43.54	-0.31	-2.14	$+1.42$	-0.20
TriviaQA	87.71	87.48	-0.23	-1.24	$+1.20$	-0.67
TREC	76.67	76.67	$+0.00$	$+0.00$	$+0.00$	$+0.00$
MultiFieldQA	52.05	52.05	$+0.01$	$+2.29$	-1.87	-0.39
Passage Count	4.00	6.00	$+2.00$	$+4.00$	$+2.00$	$+0.00$

rather than of the sampling rule. A direct check confirms this. Under seed-paired nucleus sampling on Llama-3.1-8B (temperature 0.6, top-p 0.9) over eight tasks that include the two on which CVSA-c4 performs weakest, its gap to dense is -0.63, against -1.31 for greedy decoding on the same tasks: the gap does not widen under stochastic decoding.

D LongBench 16-task Per-Task Breakdown

Table 15 expands the aggregate LongBench result reported in the main text into per-task scores, sorted by three-model mean gap so that the failure modes appear first. The pattern is concentrated rather than uniform: the largest residual losses occur on Qasper (-4.48), LCC (-3.04), SAMSum (-2.97), and RepoBench-P (-2.12), while retrieval and classification tasks are within noise of dense. These four tasks share a structural property: the information the model must reproduce is a short, position-sensitive span (a specific answer passage, a code continuation, or a dialogue turn) embedded in a long context. A 5% base budget may exclude the page containing that span, and the coverage verifier can only flag the miss after the pages are selected; if the critical span happens to contribute little to the overall softmax mass, coverage may not drop below τ even though the output is degraded. This failure mode points to a concrete future direction: token-aware or query-conditioned proxy calibration that would tighten the coverage signal on position-sensitive content. By contrast, broader QA and retrieval tasks tolerate the same sparse draft because the answer signal is distributed across multiple pages, making it unlikely that all relevant mass is excluded simultaneously.

E Check-Interval M Sweep (Quality, Latency, Effective Budget)

Table 5 (Section 3.4) summarizes the three-model average M-sweep that motivates CVSA-c4 as the 7B default. This appendix expands the per-model quality breakdown and explains why $M{=}4$ is the Pareto choice.

Why Pareto-optimal at $M{=}4$. Three forces interact as M grows: **(i) Effective budget falls** from 16.1% at $M{=}2$ (frequent re-scoring + AR) to 9.9% at $M{=}8$ (rare re-scoring, accept-cached selection), **(ii) Page-scoring throughput cost amortizes**: scoring is the expensive non-attention step in any paged sparse-attention method, and beyond $M{=}4$ the per-step amortized cost is already negligible; the speedup curve flattens between $M{=}4$ ($1.98\times$) and $M{=}8$ ($1.99\times$), **(iii) Quality degrades super-linearly** once the cached selection drifts: Δ moves from -0.43 at $M{=}2$ to -1.13 at $M{=}4$ (a 0.70 pt jump) to -2.04 at $M{=}8$ (a 0.91 pt jump). $M{=}4$ is the elbow: it captures essentially all of the speedup ($1.98\times$ vs. $1.99\times$ at $M{=}8$) while paying only half the quality cost of $M{=}8$.

Per-model quality breakdown. Table 16 reports M-sweep quality on each of the three 7B models (LongBench 16-task average, 5% budget, AR$=20\%$, dL0). CVSA-c2 stays close to dense on Llama and Mistral and within 1 pt on Qwen; CVSA-c4 incurs a small additional drop on each model; CVSA-c8 drops more on Qwen (the 7:1 GQA model) than on the others, mirroring the page-max

Table 16: M-sweep per-model quality at 5% budget, AR$=20\%$, dL0. The c8 row uses the completed LongBench-16 runs for all three models. The three-model average (-2.04) matches Table 5. On LongBench c4 reads 11.7% of the KV cache.

Configuration	Llama	Mistral	Qwen	3-model avg	Δ vs. dense
Dense	42.57	42.73	40.18	41.83	—
CVSA-c1	42.67	42.47	39.45	41.53	-0.30
CVSA-c2	42.49	42.53	39.19	41.40	-0.43
CVSA-c4 (default)	**41.87**	**41.71**	**38.54**	**40.70**	$\mathbf{-1.13}$
CVSA-c8	41.18	40.59	37.59	39.79	-2.04

Table 17: Dense-layer ablation. The verified AR configuration improves on Fixed-5%; its refresh cadence also differs. Within that configuration, dL0 provides an additional safeguard, primarily on Qwen. The deployed configuration, CVSA-c4 with AR20 and dL0, reads 11.7% of the KV cache.

Configuration	Llama	Mistral	Qwen	Avg	Δ vs. dense
Dense	42.57	42.73	40.18	41.83	—
Fixed-5% (no verification)	39.00	37.57	32.49	36.35	-5.48
CVSA-c4 + AR20, no dL0	41.98	41.11	35.53	39.54	-2.29
CVSA-c4 + AR20 + dL0	41.87	41.71	38.54	40.70	-1.13

proxy degradation discussed in Section 3.2. On every model, c4 stays close to c2 quality at $\sim$25% lower effective budget, so the three-model Pareto observation also holds per model.

Latency and effective budget per M. The latency and effective-budget figures in Table 5 (Section 3.4) are three-model averages on H100 80GB at 8K context, batch size 32, measured in our paged-decode pipeline. At 32K–128K, CVSA-c8 stays within $0.03\times$ of CVSA-c4's median decode-step speedup on all three models, so the longer interval buys almost no speed at long context.

The $M{=}1$ endpoint. Refreshing the page selection at every step ($M{=}1$) is the quality-premium rather than the throughput configuration. On the LongBench 16-task average it recovers most of the remaining gap to dense, reaching $\Delta = -0.30$ across the three models against -1.13 for the deployed $M{=}4$, with Llama at $+0.10$, within noise of dense (Table 16). Its cost is speed: at 8K context and $B{=}32$ it runs at $0.93\times$ dense decode, so per-step scoring cancels the sparse-attention speedup; memory is unchanged from dense since the full KV cache is retained regardless of M. $M{=}1$ is thus the setting to select when exact per-step coverage is required at a known throughput cost, and it is the configuration whose reasoning-length behavior is analyzed in Section P.

Recommendation. CVSA-c4 ($M{=}4$) is the recommended 7B default; CVSA-c2 ($M{=}2$) is the recommended setting at $\geq$ 70B, where additional layers amplify error accumulation across the longer reuse window (Section K).

F Dense-Layer and Recovery Ablations

This appendix expands the ablation analysis in Section 4.3, separating two possible confounds in the main CVSA configuration: the fixed layer-0 dense policy (dL0) and approximate recovery (AR20). All results use LongBench 16-task averages with 50 samples per task on the three 7B models.

Dense-layer policy. Table 17 compares fixed 5% sparsity, CVSA without dL0, and the default CVSA configuration with dL0. The fixed-5% no-verification baseline trails dense by 5.48 pt on the three-model average. The verified AR20 configuration without dL0 improves the average from 36.35 to 39.54, recovering 3.19 pt. Adding dL0 improves the average further to 40.70, a $+1.16$ pt gain over no-dL0. The dL0 effect is not uniform: Llama changes by -0.11 pt, Mistral gains $+0.60$ pt, and Qwen gains $+3.00$ pt. This supports the framing used in Section 3.5: dL0 is a simple cross-model safeguard for the universal low-coverage layer, not a model-specific dense mask, but it is especially important for Qwen.

Extending the dense mask on Qwen. Because Qwen carries the largest residual gap, a natural question (raised in review) is whether densifying more early layers helps. Extending Qwen's dense

Table 18: Exact Recovery versus AR20. AR20 matches Exact Recovery on average, with per-model differences within 0.24 pt. CVSA-c4 reads 11.7% of the KV cache with AR20.

Recovery policy	Llama	Mistral	Qwen	Avg	Δ vs. dense
Dense	42.57	42.73	40.18	41.83	—
CVSA-c4 + Exact + dL0	42.11	41.53	38.31	40.65	-1.18
CVSA-c4 + AR20 + dL0	41.87	41.71	38.54	40.70	-1.13

Table 19: Additional ungated controls with four-step refresh and dense layer 0 (LongBench, 16 tasks, 50 samples/task). Logged attention-read fractions for the 9% uniform control are 12.04/12.01/12.48%, for the 10% uniform control 13.02/13.08/13.40%, and for the offline control 12.15/12.09/12.54% (Llama/Mistral/Qwen). This is a scheduling and dense-layer diagnostic, not an exact total-compute match.

Configuration	Llama	Mistral	Qwen	Avg
CVSA-c4 (reference)	41.87	41.71	38.54	40.70
Uniform 9%, refresh 4, dL0	42.39	42.06	39.13	41.19
Uniform 10%, refresh 4, dL0	42.39	41.92	39.17	41.16
Offline budget, refresh 4, dL0	42.58	41.78	38.72	41.03

mask from layer 0 to layers 0–3 recovers only $+0.42$ points (from -1.77 to -1.35 on the probe basis, paired-bootstrap CI $[-1.85, -0.88]$, still excluding zero) at a $+9.6$-point increase in effective read ($11.3\% \to 20.9\%$), an unfavorable trade. Early-layer recovery is detection cost the loop has already repaired: those rejections are caught and recovered, so quality there is already preserved. The residual leak instead originates in late-layer false accepts (layers ≥ 17 hold 51% of them; Section C), which early dense layers cannot reach. Layer-0-only dense therefore remains the deployed policy.

Approximate versus exact recovery. Table 18 compares the production AR20 recovery policy with Exact Recovery, which computes the complement attention and merges it with the draft using the exact decomposition. Exact Recovery is the clean algebraic reference path, but it is too expensive to use as the default production policy. Empirically, AR20 matches Exact Recovery on the three-model average: 40.70 versus 40.65. Per-model differences are small (Llama -0.24 pt, Mistral $+0.18$ pt, Qwen $+0.22$ pt), consistent with benchmark noise rather than a systematic quality gap. Thus, the reported system can use the efficient AR20 path without undermining the theory-to-system story: exact recovery is the reference implied by the proposition, while AR20 is the practical approximation validated here.

Offline per-(layer, head) budget. The static control in Table 7 gives each (layer, KV head) cell the budget CVSA spends there in expectation, $b_{l,h} = 5\% + r_{l,h} \cdot 15\%$, where $r_{l,h}$ is the cell's offline recovery rate, the fraction of decode steps whose coverage falls below $\tau = 0.90$ at the 5% base budget (four LongBench tasks, five prompts each, 128 decode steps). Budgets are clamped to $[5\%, 20\%]$ and rescaled to a matched effective read (12.1–12.5% across the three models), with the same selector, kernels, samples, and dense layer 0 as CVSA-c4. This original static control refreshes every 16 steps, while CVSA-c4 checks every four steps; the comparison also changes refresh cadence. The calibration prompts are drawn from the evaluation distribution, which favors the static policy. The offline configuration scores $+1.36$ points above the uniform Fixed-12% control ($+0.31$ / $+0.51$ / $+3.25$ on Llama / Mistral / Qwen), and CVSA-c4 scores a further $+2.21$ ($+1.89$ / $+3.12$ / $+1.63$); these are configuration-level differences. A per-step trace characterizes how coverage failures vary over time. On a two-sample Llama-3.1-8B HotpotQA trace with a fresh selection at every step (M=1), chronic cells (one per sample, layer, and head) that fail on at least 80% of steps are 1.6% of cells, none is chronic in both samples, and they account for only 11.1% of failure events, while intermittent cells failing on 5–80% of steps account for 86.4%, and the set of failing heads overlaps by a Jaccard index of only 0.33 between consecutive steps. The failing heads vary substantially between consecutive steps.

Controls with refresh cadence and dense layer aligned. To examine the refresh and dense-layer differences in Table 7, we also evaluate ungated controls with four-step refresh and dense layer 0 on the same first 50 examples per task. Table 19 retains the original CVSA-c4 reference and reports these additional configurations separately. The uniform 9% control reaches 41.19 and the offline

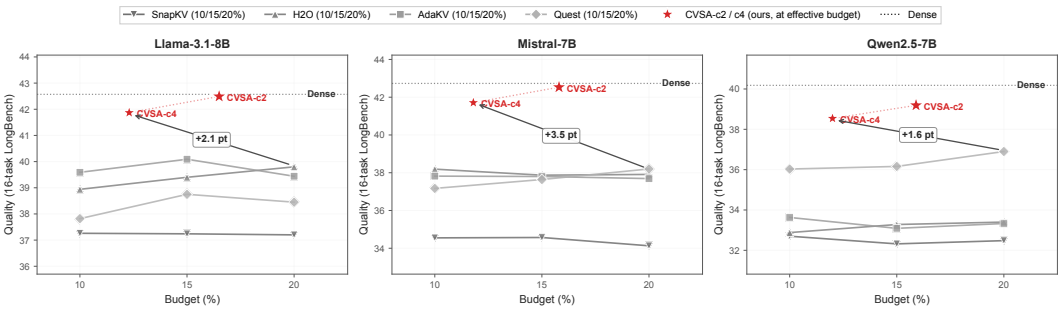


Figure 9: Budget-fairness within a single paged-decode implementation: baseline budget swept over $\{10, 15, 20\}\%$ (lines) versus CVSA-c2 / CVSA-c4 at their measured effective budgets ($\sim$16% / $\sim$12%). The baseline curves plateau across the swept range; the diagonal arrows show the remaining configuration-level gaps to CVSA-c4. Three 7B models, 16-task LongBench average.

control 41.03, versus 40.70 for CVSA-c4. Their logged attention-read fractions are somewhat higher, approximately 12.0–12.5% versus 11.7%; page-scoring traffic and wall-clock cost are not matched. Read fractions are pages-weighted over the last-sample decode trace retained for each task, rather than over all evaluation examples. These results show that the original $+3.57$ and $+2.21$ margins cannot be attributed to the coverage gate alone: refresh cadence and dense-layer placement materially affect quality.

G Budget-Fairness: Baselines at 10% / 15% / 20%

We sweep each baseline's nominal budget over $\{10, 15, 20\}\%$ within the same paged-decode implementation, on all three 7B models (16-task LongBench). The defensive question is whether CVSA-c4's quality lead disappears once baselines are given comparable or higher budget. Fig. 9 shows it does not: every baseline curve is essentially flat across the 10–20% range (max single-curve gain is ≤ 1.0 pt), so doubling the nominal budget does not close the gap to CVSA-c4 at $\sim$12% effective budget. The diagonal arrow on each panel measures CVSA-c4's residual lead over the strongest baseline at 20% budget: $+2.1$ pt on Llama, $+3.5$ pt on Mistral, $+1.6$ pt on Qwen. Together with Section N (CVSA-c8 at 9.9% effective read exceeds the best paged baseline by $+2.78$ pt), this shows that a larger nominal budget alone does not close the gap for these baselines. Refresh cadence, dense-layer placement, and scoring traffic also differ; these comparisons do not isolate the gate's causal contribution (Section F).

H Per-Layer Per-Model Coverage Landscape

This appendix provides the per-layer verifier diagnostics that underlie the dense-layer policy (§3.5) and the coverage-landscape discussion in the ablation studies (§4.3). Fig. 10 visualises the landscape on three 7B models from three angles: layer-by-model verifier acceptance, budget sweep on Llama, and per-layer acceptance distribution.

Three operational regimes. The acceptance heatmap (panel a) and the acceptance histogram (panel c) jointly classify the 92 layer-instances ($32 + 32 + 28$ across the three models) into three regimes:

- **Problem layers** are low-acceptance regions in panel (a) that enter recovery frequently and, in the lowest-coverage cases, cannot be repaired by a modest budget increase alone.
- **Moderate layers** are neither consistently accepted nor hopeless: they produce a substantial reject rate at the 5% base budget but improve under AR or a larger page budget.
- **High-coverage layers** are accepted almost always, so spending recovery on them would waste bandwidth without improving quality.

Budget alone does not fix problem layers. Panel (b) sweeps budget over $\{5, 10, 15, 20\}\%$ on Llama and tracks acceptance rate at $\tau = 0.90$ separately for each regime. The problem-layer curve (red squares, $n = 1$) reaches only 21% acceptance even at 20% budget (a $4\times$ increase over the

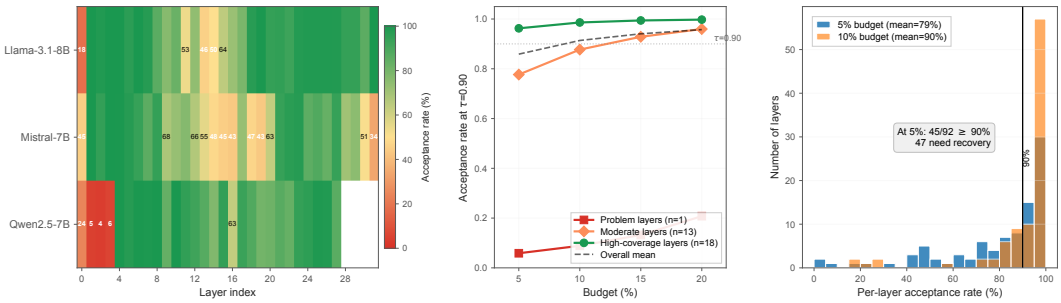


Figure 10: Per-layer verifier and coverage landscape on Llama-3.1-8B (32 layers, 4:1 GQA), Mistral-7B-v0.3 (32 layers, 4:1 GQA), and Qwen2.5-7B (28 layers, 7:1 GQA). *(a)* Verifier acceptance rate at 5% budget, $\tau = 0.90$: green cells are layer/model pairs that are usually accepted, while red cells identify concentrated reject regions that enter recovery frequently. *(b)* Acceptance rate vs. budget on Llama, layers grouped by category: problem layers (n=1) reach only 21% acceptance even at $4\times$ budget, ruling out budget alone as a fix and motivating the dL0 dense-layer policy; moderate (n=13) and high-coverage (n=18) groups saturate by 10–15%. *(c)* Per-layer acceptance distribution across all 92 layer-instances: at 5% budget $45/92$ layers accept $\geq 90\%$ of checks; doubling the budget to 10% shifts the distribution rightward but still leaves a low-acceptance tail.

default), confirming that no realistic sparse budget will recover these layers, and motivating the dense-layer policy: force these layers to run dense and accept the small fixed cost ($1/32$ of attention compute on a 32-layer model). Moderate and high-coverage layers in contrast saturate well before 20% budget.

Layer 0 universality. Layer 0 is the only layer whose acceptance falls below 50% on all three models (panel a), consistent with the low attention sparsity that Quest [6] reports for the first two layers. Beyond layer 0, the landscape is model-specific: acceptance also falls below 50% at L1–L3 on Qwen, L13–L14 on Llama, and L14–L16, L18–L19, and L31 on Mistral. These layers are not forced dense. Every deployed configuration runs only layer 0 dense (Section 3.5) and leaves the other flagged layers to the gate and approximate recovery; extending Qwen's dense mask to L0–L3 recovers only $+0.42$ points while raising effective read from 11.3% to 20.9% (Section F). The diagnostics run automatically on a held-out prompt set, and the dense set is layer 0 on every model, so no per-model tuning is involved.

Implication for verification. Of the 92 layer-instances, 47 accept fewer than 90% of checks at 5% budget (panel c). On Llama's moderate layers, a 20% budget raises acceptance from 78% to 96% (panel b); on problem layers it is insufficient, which is why layer 0, the one problem layer shared by all three models, runs dense. The combined recipe (high-coverage $\rightarrow$ accept, moderate $\rightarrow$ AR, layer 0 $\rightarrow$ dense) is the recipe used throughout the experiments; dense layer 0 raises the three-model average by 1.16 points, mostly on Qwen (Table 17).

Practitioner rule for keeping a layer dense. The recipe above reduces to a two-signal test over quantities the system already logs offline: keep a layer dense if and only if *(a)* its recovery rate at the operating τ stays near one (the gate almost always rejects it) *and (b)* its coverage remains low even after recovery (recovery does not restore the missing mass). Condition (a) alone is insufficient, since a layer can reject often yet be fully repaired by recovery; it is the conjunction that identifies a layer no sparse budget can serve. Only layer 0 falls below 50% acceptance on all three 7B models; Qwen's layers 1–3 reject more often but are repaired by recovery, and the probe in Section F prices densifying them explicitly ($+0.42$ points for $+9.6$ points of read), which the rule correctly declines.

I Runtime Recovery Concentration

This appendix complements the static coverage landscape in Section H with a runtime view of where recovery is actually spent. The key point is that recovery is not a hidden dense fallback: most heads (77–86% per model) are accepted, while the rejected heads concentrate in a small set of model-specific layers.

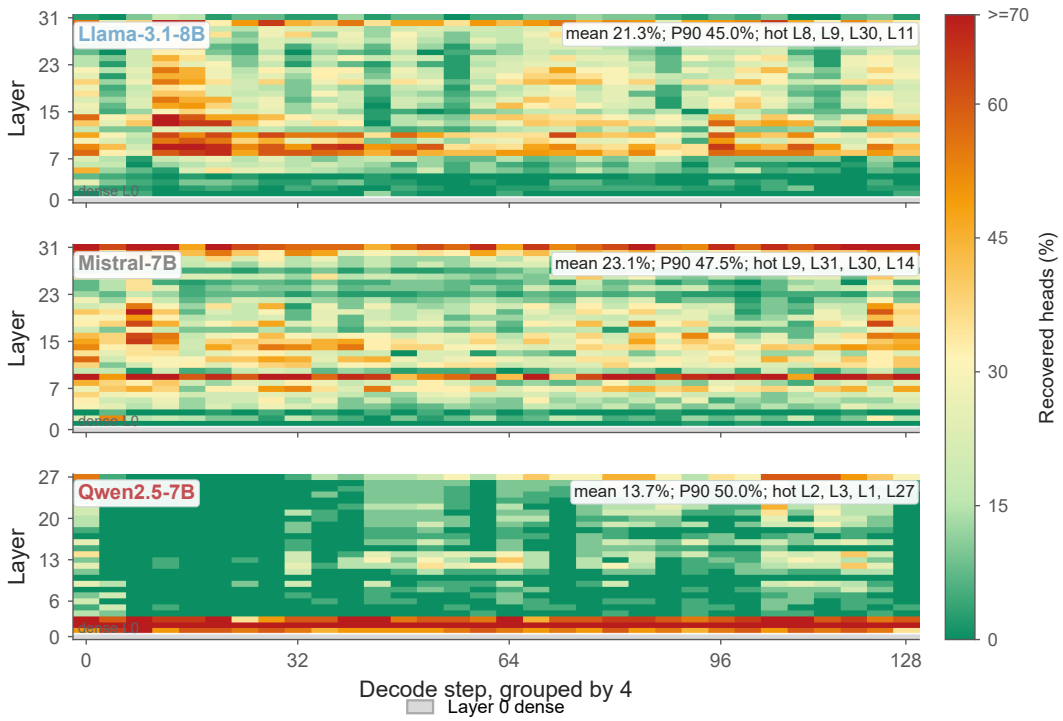


Figure 11: Where recovery is spent at runtime. Each panel is a CVSA-c4 decode trace on HotpotQA, averaged over five samples and grouped into $M{=}4$ decode-step bins. Color denotes the fraction of KV heads in a layer-step bin that fail the coverage check and enter recovery; layer 0 is gray because the default configuration runs it dense. Recovery is concentrated rather than universal: the non-dense mean recovered-head rates are 21.3% on Llama-3.1-8B, 23.1% on Mistral-7B, and 13.7% on Qwen2.5-7B, with 90th-percentile layer-step rates of 45.0%, 47.5%, and 50.0% respectively. The hotspots are model-specific (Llama L8/L9/L30/L11, Mistral L9/L31/L30/L14, Qwen L2/L3/L1/L27), supporting the claim that verification routes recovery to a small low-coverage tail while most heads (77–86% per model) remain on the accepted sparse path.

Fig. 11 should be read as a runtime complement to the static coverage landscape in Section H: the landscape identifies which layers have low coverage at a given budget, while the heatmap shows where recovery is actually spent during generation. Color saturation encodes the fraction of KV heads in each layer-step bin that fail the coverage check and enter recovery. The key finding is that recovery is selective rather than global. Llama and Mistral show late-layer and mid-layer hotspots (L8/L9/L30 on Llama, L9/L31/L30 on Mistral), while Qwen concentrates rejection in the first few non-dense layers (L2/L3), consistent with the low early-layer acceptance on Qwen in Section H. The non-dense mean recovered-head fractions are 21.3% (Llama), 23.1% (Mistral), and 13.7% (Qwen), with 90th-percentile layer-step rates of 45–50%. Because these hotspots occupy a small portion of the layer-step plane, the amortized bandwidth cost of recovery stays close to the base sparse budget, which explains why the throughput reported in the main paper is not erased by the verify-then-correct loop. This pattern supports the central design assumption in Section 3.1: sparse-attention failures are concentrated rather than uniform, making selective recovery more efficient than uniform budget inflation.

J Per-Task $\times$ Per-Model $\times$ Per-Length Numbers (RULER)

The full RULER grid behind Fig. 4 covers all six tasks at 4K–32K for every method, plus 64K and 128K runs for dense, CVSA, and Fixed-10%, and for the three eviction baselines on the Fig. 4 tasks (MV-NIAH, MQ-NIAH, CWE). Table 32 condenses the grid into the mean gap to dense, and Tables 33 to 35 give every cell. On the 72 cells every method covers (six tasks at 4K–32K on three models), CVSA-c4 is 0.83 points below dense and the strongest baseline, H2O, is 8.70 points below. The tables are generated by `eval/build_ruler_table.py` from the JSON files under `results/`.

Table 20: Llama-3.1-70B-Instruct per-task scores on the 7-task LongBench subset (samsum excluded; see text). KVPress baselines run in HuggingFace SDPA, while CVSA runs in our paged-decode loop; a 2.48-pt cross-pipeline drift on the dense reference (paged 52.02 vs. KVPress 54.50) sets a noise floor for cross-implementation comparisons; KVPress baselines therefore appear 2.48 points stronger than they would under the paged reference. Within each pipeline, CVSA-c2 statistically matches paged dense (CI $[-0.79, +1.95]$); CVSA-c4 underperforms because $M{=}4$ is too long for 80 layers.

Task	Paged dense	CVSA-c2	CVSA-c4	KVPress AdaKV	KVPress SnapKV
hotpotqa	43.10	**45.14**	40.84	48.37	48.37
triviaqa	90.37	**91.40**	86.70	92.00	90.00
qasper	58.85	58.04	55.31	56.90	55.46
multifieldqa_en	55.79	**58.20**	54.22	60.79	58.78
trec	66.67	**66.67**	66.67	66.67	66.67
lcc	23.93	23.67	20.77	27.90	28.50
narrativeqa	25.46	25.27	23.80	25.96	25.55
7-task average	52.02	**52.63**	49.76	54.08	53.33
Δ vs. paged dense	—	**+0.60**	−2.27	+2.06	+1.31

K Llama-3.1-70B Detailed Results

This appendix expands Table 4 with the full per-task breakdown on Llama-3.1-70B-Instruct (4×H200) and explains the methodology behind the 7-task subset.

Why 7 tasks and not 8. Our 8-task English subset of LongBench includes `samsum`, which is verbose dialogue summarization with multiple acceptable end-of-turn markers. Our custom multi-GPU paged-decode generation loop does not reproduce HuggingFace `generate`'s stopping behavior on this open-ended summarization task, even after matching the multiple EOS variants used by Llama-3.1-Instruct. On 70B at long context this divergence inflates samsum's verbose tail and produces score artifacts (paged: ∼19 ROUGE-L; HF: ∼45) that are not method-attributable. We exclude samsum on 70B for this reason and report 7-task averages throughout this appendix; the same task is included for the 7B models, where the divergence does not manifest.

Per-task breakdown. Table 20 reports per-task scores for paged dense, CVSA-c2, CVSA-c4, KVPress AdaKV, and KVPress SnapKV ($n = 30$ samples per task, greedy decoding, 10% nominal budget for baselines, 5% budget plus AR= 20% and dL0 for CVSA). CVSA-c2 matches or exceeds paged dense on 4 of 7 tasks (hotpotqa, triviaqa, multifieldqa_en, trec) and is within 1 pt on the remaining three; the bootstrap 95% CI on the 7-task average is $[-0.79, +1.95]$, overlapping zero. CVSA-c4 degrades on extractive tasks (lcc -3.16, hotpotqa -2.26, triviaqa -3.67) because error accumulates over 80 layers when the check interval is too long relative to model depth, which motivates the $M{=}2$ guideline for 70B-scale models.

Cross-pipeline reading. The $+2.06$ pt apparent gain of KVPress AdaKV over paged dense in Table 20 reflects the cross-pipeline drift, not method gain: the KVPress dense reference itself is 54.50, $+2.48$ pt above paged dense, larger than the AdaKV margin. The within-pipeline reading is: KVPress AdaKV is $\Delta = -0.41$ vs. KVPress dense, while CVSA-c2 is $\Delta = +0.60$ vs. paged dense; each method is evaluated against its own pipeline.

Recommendation for 70B-scale models. On 80-layer models, error accumulates faster than at 32-layer scale; we recommend $M{=}2$ (CVSA-c2) for ≥70B and keep $M{=}4$ as the 7B default. The M-vs.-depth trade-off is not yet captured by a closed-form policy and is a candidate for follow-up work.

L Bandwidth-Roofline Derivation

The decode-step throughput speedups reported in §4.2 are predicted by a simple bandwidth-roofline model. This appendix derives the model, instantiates the parameters per architecture, and reports measured-vs-predicted speedups across {model, batch} pairs.

Table 21: Architecture specs used in the roofline model. KV-cache bytes scale linearly with $\mathrm{kv_heads}$ and context length; MLP bytes (which dominate weight reads at small B) scale with $\mathrm{hidden} \times \mathrm{intermediate}$.

Model	Layers	Q/KV heads	Head dim	Hidden	MLP inter.
Llama-3.1-8B	32	32 / 8	128	4096	14336
Mistral-7B-v0.3	32	32 / 8	128	4096	14336
Qwen2.5-7B	28	28 / 4	128	3584	18944

Setup. At a single decode step, the GPU must read (i) all model weights W once and (ii) the KV cache for every sequence in the batch. Let

$$\begin{aligned} W_{\text{bytes}} &= \text{total model-weight bytes (bf16)}, \\ \mathrm{KV}_{\text{bytes}} &= \text{KV-cache bytes for one sequence at the current context length}, \\ B &= \text{batch size}, \\ \eta &= \text{effective KV-read fraction relative to dense}. \end{aligned}$$

For dense attention $\eta = 1$; for CVSA-c4, $\eta \approx 0.12$.

The dense decode-step is then bandwidth-bound by $W_{\text{bytes}} + B \cdot \mathrm{KV}_{\text{bytes}}$ HBM bytes per step. CVSA-c4 reads the same weights but only η of the KV cache, so its total HBM traffic per step is $W_{\text{bytes}} + B \cdot \eta \cdot \mathrm{KV}_{\text{bytes}}$.

Speedup formula. Define f_{kv} as the KV-cache fraction of total decode-step HBM traffic for dense:

$$f_{\text{kv}} = \frac{B \cdot \mathrm{KV}_{\text{bytes}}}{W_{\text{bytes}} + B \cdot \mathrm{KV}_{\text{bytes}}}. \tag{7}$$

Because dense decode is bandwidth-bound, the speedup of CVSA-c4 over dense is the inverse ratio of HBM traffic:

$$\begin{aligned} \text{speedup} &= \frac{W_{\text{bytes}} + B \cdot \mathrm{KV}_{\text{bytes}}}{W_{\text{bytes}} + B \cdot \eta \cdot \mathrm{KV}_{\text{bytes}}} \\ &= \frac{1}{1 - f_{\text{kv}} + f_{\text{kv}} \cdot \eta}, \end{aligned} \tag{8}$$

where the second line follows by dividing both numerator and denominator by $W_{\text{bytes}} + B \cdot \mathrm{KV}_{\text{bytes}}$ and substituting Eq. (7).

Parameter values. Table 21 lists the architecture parameters used to compute W_{bytes} (per-layer Q + K + V + O projections plus the three-matrix MLP, summed over layers, in bf16) and $\mathrm{KV}_{\text{bytes}}$ (16-token pages, $\mathrm{kv_heads} \cdot \mathrm{head_dim}$ bytes per page-side-K-or-V, summed over layers and the running 8K context). Qwen2.5-7B's 7:1 GQA ratio gives the smallest KV cache per layer relative to its larger MLP (intermediate dim 18944 vs. 14336 on Llama / Mistral), so its f_{kv} is the lowest of the three models at every batch size and the bandwidth-only model predicts it the smallest speedup; its measured speedup is nonetheless the largest, for the reasons given under Table 22.

Predicted vs. measured speedup. Table 22 reports the per-(model, batch) breakdown at 8K context with $\eta = 0.12$ (CVSA-c4's measured average effective budget). Fig. 12 plots all nine points against the theoretical speedup curve (Eq. (8)).

Implications for paper claims. The roofline model justifies the qualitative scaling claims in §4.2: as context grows, $\mathrm{KV}_{\text{bytes}}$ grows linearly in context while W_{bytes} stays fixed, so $f_{\text{kv}} \to 1$ and speedup approaches the asymptotic ceiling $1/\eta \approx 8.3\times$. At short context and small batch, weight reads dominate ($f_{\text{kv}} \leq 7\%$ at 8K, B=1), so the gain is small regardless of η (measured 1.09–$1.12\times$).

M Long-Context Throughput Tables

This appendix expands the headline numbers in Table 3 (§4.2) with measurements across three 7B models at long context. All values are median decode-step speedup vs. dense ($\mathrm{dense_ms}/\mathrm{c4_ms}$,

Table 22: Predicted speedup from Eq. (8) (using $\eta = 0.12$, CVSA-c4's 11.7% effective read on LongBench, rounded) versus measured speedup at 8K context, three batch sizes. Qwen consistently exceeds the bandwidth-only prediction at $B \geq 8$ because dL0 (force layer 0 to dense) reduces effective per-decode latency disproportionately on its 7:1 GQA architecture, and the Triton-kernel index-computation overhead is also smaller for the smaller KV per layer; Llama and Mistral at B=32 fall short of the bandwidth ceiling because non-attention overheads (LSE merge, page scoring) become a larger fraction of the shrunk attention time. The model is intended as a directional predictor, not an exact fit.

Model	B	f_{kv} (%)	Predicted (Eq. 8)	Measured
Llama-3.1-8B	1	7.1	1.07×	1.09×
	8	38.1	1.50×	1.44×
	32	71.1	2.67×	1.78×
Mistral-7B-v0.3	1	7.1	1.07×	1.09×
	8	38.1	1.50×	1.44×
	32	71.1	2.67×	1.77×
Qwen2.5-7B	1	3.5	1.03×	1.12×
	8	22.4	1.24×	1.87×
	32	53.5	1.89×	**2.50**×

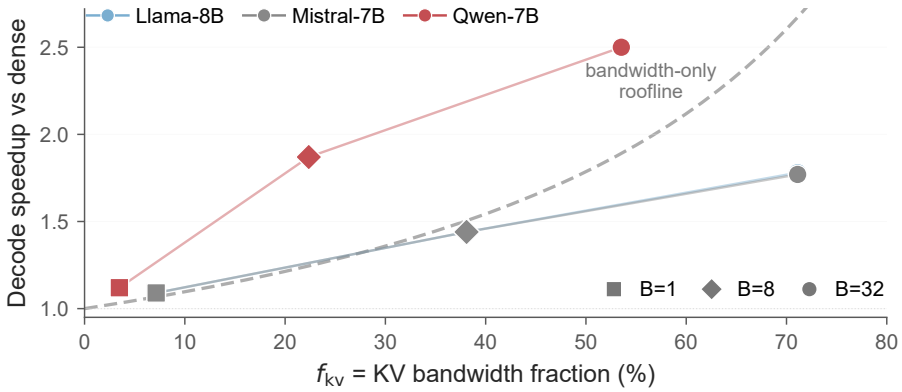


Figure 12: Decode-step speedup vs. KV bandwidth fraction f_{kv}. The dashed grey curve is the bandwidth-only prediction $1/(1 - f_{\text{kv}} + f_{\text{kv}} \cdot \eta)$ with $\eta = 0.12$; markers are measured CVSA-c4 speedups across three 7B models and three batch sizes ($B \in \{1, 8, 32\}$, 8K context). The model captures the qualitative trend (speedup grows with f_{kv}, i.e. with both context length and batch size); Qwen's elevated $B \geq 8$ measurements reflect dL0 overhead amortization that is not in the simple roofline.

median over timed decode steps) on the paged-decode pipeline. This statistic differs from aggregate tokens/s when scoring steps have higher latency: at Qwen 128K, $B = 8$, the median-step ratio is 6.21×, while the logged tokens/s ratio is $115.60/24.86 = 4.65\times$. Llama and Mistral are measured on an H100 (80 GB, §4.1); at B=8 and ≥65K their bf16 KV cache plus weights exceeds this memory (≈85 GB at 65K), so these cells are not reported.

Speedup grows with context. Within every (model, batch) pair, speedup is monotonic in context length, consistent with the bandwidth-roofline prediction (Section L): f_{kv} rises with context, so attention dominates more of the decode-step cost and the $\eta = 0.12$ KV-read reduction translates into a larger end-to-end win.

Speedup grows with batch. At every (model, context) pair, B=8 is faster than B=1 because batching amortizes the fixed weight reads across more queries, shifting f_{kv} upward (Eq. 7). For example on Qwen at 128K: B=1 gives 2.68×, B=8 gives 6.21×.

Qwen has the largest speedup, though not the largest KV share. Qwen's 7:1 GQA combined with its larger MLP intermediate (18944 vs. 14336 on Llama / Mistral) produces the smallest KV-cache-to-weights ratio and hence the lowest f_{kv} of the three models (Table 22). Its larger speedup

Table 23: CVSA-c4 median decode-step speedup vs. dense across three 7B models, three context lengths, two batch sizes; Llama and Mistral run on H100, Qwen on H200. "—": exceeds the H100's 80 GB (Llama / Mistral, B=8, $\geq$ 65K). All speedups are medians over the timed decode steps on the paged-decode pipeline. CVSA-c4 reads 11.7% of the KV cache on LongBench.

	B=1 (single-stream)			B=8 (batched)		
Model	32K	65K	128K	32K	65K	128K
Llama-3.1-8B	1.12×	1.15×	1.19×	1.55×	—	—
Mistral-7B-v0.3	1.13×	1.16×	1.19×	1.56×	—	—
Qwen2.5-7B	1.40×	1.86×	2.68×	3.07×	4.44×	**6.21**×

Table 24: Per-kernel decode-step latency on Llama-3.1-8B (batch 8, 65K context, one H200, 67 decode steps). The verifier (coverage check) is under 2% of attention-path time because it consumes quantities the kernels already compute; page scoring and conditional recovery occur within the timed decode steps, although their cost is not fully represented by a median-step speedup. Mistral-7B reproduces each figure within 8%. CVSA-c4 reads 11.7% of the KV cache on LongBench.

Attention-path kernel	ms / decode step	Share
Coverage check (verifier)	0.45	$<$ 2%
Page scoring	8.81	∼39%
Conditional recovery	9.61	∼42%
Sparse draft attention	3.80	∼17%

comes from what the bandwidth-only roofline omits: the dense layer 0 costs it disproportionately little per decode step, and the Triton kernels' index-computation overhead is smaller for its smaller per-layer KV. Llama and Mistral cluster within 0.02× of each other at every measured cell because their architectural specs are nearly identical (Table 21).

Per-kernel latency breakdown. To locate where decode-step time is spent, and in particular to measure the cost of the verifier itself, we profile the four attention-path kernels with per-kernel CUDA timing on Llama-3.1-8B at batch 8, 65K context on one H200, over 67 decode steps (Table 24). The coverage check, i.e. the verifier, accounts for under 2% of attention-path latency: it reuses the page scores that page scoring already produces and launches no additional attention pass. The material costs are page scoring (every M=4 steps) and conditional recovery. Mistral-7B reproduces all four figures within 8%.

CUDA Graph compatibility. The recovery decision is computed on the device: the coverage kernel emits a per-head accept mask from quantities already resident in GPU memory, and the merge of draft and recovery outputs is a per-head masked select, so the mechanism requires no host-side round trip. The current implementation takes one host-side branch per scoring step, an early exit that skips the recovery kernel when every head accepts, which is an implementation convenience rather than a requirement of the method. A graph-capturable variant launches the recovery kernel unconditionally with the accept mask applied as predication, at small added cost because recovery already runs on nearly every scoring step at the deployed operating point, or places the branch in a conditional graph node (available since CUDA 12.4). Reuse steps, three of every four at M=4, decode a cached fixed-shape selection and are directly capturable. All speedups in this paper were measured in a single eager pipeline with no graph capture for any method, a setting that if anything understates CVSA's relative speed because CVSA issues the most kernel launches per decode step.

N CVSA-c8 at a Matched Effective Budget

This appendix elaborates on the matched-budget claim in §4.2: CVSA-c8's $\Delta = -2.04$ at 9.9% effective budget beats the strongest paged baseline by +2.78 pt at a matched effective budget, showing a quality advantage at a comparable attention-read budget for these configurations.

Effective-budget definition (recap). Effective budget is the fraction of KV pages read by attention relative to dense, including AR expansion and dense layer 0; page-scoring reads are counted separately (Section O). For CVSA-c8: 5% base budget, $\tau = 0.90$, $M = 8$, AR= 20% on rejected heads. The

Table 25: CVSA-c8 at a matched effective budget: 9.9% effective budget vs. baselines at 10% nominal budget (SnapKV and AdaKV via KVPress; H2O and Quest paged). The $+2.78$ pt margin over the strongest paged baseline (Quest) holds at a marginally lower attention-read fraction. Comparing against canonical KVPress AdaKV and SnapKV at 10% (Table 1), CVSA-c8's $\Delta = -2.04$ is also better than KVPress AdaKV's $\Delta = -3.05$.

Method	Eff. budget	3-model avg	Δ vs. dense	Δ vs. Quest
Dense	100%	41.83	—	—
CVSA-c8 (ours)	$\mathbf{9.9\%}$	$\mathbf{39.79}$	$\mathbf{-2.04}$	$\mathbf{+2.78}$
SnapKV (KVPress)	10%	38.56	-3.27	$+1.55$
AdaKV (KVPress)	10%	38.78	-3.05	$+1.77$
H2O (paged)	10%	36.67	-5.16	-0.34
Quest (paged)	10%	37.01	-4.82	—

Table 26: Per-configuration effective KV-read (%) and LongBench quality across three 7B models. CVSA-c4 sets a coverage level rather than a per-head budget, so its traffic is an outcome ($\sim$11.7% on every model with no per-model tuning), not a tuned input. The pre-hoc selector consumes more traffic at lower quality; adding the gate improves it by $+3.67$ but does not reach CVSA-c4.

Configuration (read% / quality)	Llama	Mistral	Qwen	3-model
Pre-hoc top-p (alone)	14.42 / 35.12	13.55 / 32.79	10.21 / 27.16	12.73 / 31.69
Pre-hoc top-p + coverage gate	16.88 / 38.94	16.94 / 36.21	13.64 / 30.93	15.82 / 35.36
CVSA-c4 (verified top-k)	11.67 / 41.87	11.74 / 41.71	11.66 / 38.54	11.69 / 40.70

accept rate at $\tau = 0.90$ is $\sim$50%, so weighted across head-steps the effective budget lands at 9.9%, marginally below the baselines' 10% nominal budget, allowing a matched-budget comparison.

Quality at iso-bandwidth. Table 25 compares CVSA-c8 (effective budget 9.9%) against the four baselines (nominal budget 10%) on three 7B models, LongBench 16-task average. CVSA-c8 retains the verification advantage: at a lower attention-read fraction, it still beats every paged baseline on the three-model average and matches or beats every baseline on every individual model.

Implication. The attention-read comparison and the budget-fairness sweep (Section G) show that increasing the evaluated baselines' nominal budgets does not close their quality gap to CVSA. Attention-page counts do not match total compute or refresh cadence; Section F examines the latter together with dense-layer placement.

O Selector Compatibility and Traffic Accounting

This appendix measures compatibility with alternative page selectors and accounts for the traffic each configuration consumes. Unless noted otherwise, all numbers use the same LongBench 16-task protocol, samples, kernels, and scorer as the main results; effective KV-read is pages-weighted over the last-sample decode trace retained for each task in the execution logs.

Per-configuration traffic accounting. Table 26 reports effective read and quality for three configurations that share our pipeline: a pre-hoc top-p selector alone, the same selector with the unchanged coverage gate, and CVSA-c4. The pre-hoc selector reads more than CVSA-c4 (12.73% versus 11.69% on the three-model average) yet scores 9.0 points lower (31.69 vs. 40.70); composing the gate on that selector recovers $+3.67$ (to 35.36) while still reading more than CVSA-c4. Quality does not follow from traffic alone.

Selector-agnostic verification. Table 27 evaluates the coverage gate under three selector families beyond the deployed exact top-k, with the gate and recovery unchanged. The gate detects whatever each selector misses and routes recovery there, so a weaker selector degrades gracefully rather than failing silently, and exchanging the allocation rule leaves the outcome essentially unchanged in the gated configurations. Ungated comparisons also differ in scheduling; they are configuration-level comparisons.

Table 27: Coverage gate under alternative page selectors (LongBench 16-task average, three 7B models). The gated configurations improve on the corresponding ungated configurations on every model ($+2.19$ / $+1.74$ / $+1.37$ and $+3.82$ / $+3.42$ / $+3.77$). The AdaKV-style allocator is measured against CVSA-c4's 40.70 at the same effective read ($+0.10$ / -0.31 / $+0.01$ per model), so exchanging the allocation rule moves quality by 0.06 points. CVSA-c4 reads 11.7% of the KV cache.

Selector (gate and recovery unchanged)	Without gate	With gate	Change
Quest-style min/max indexer	37.01	38.77	$+1.76$
Pre-hoc top-p (adaptive budget)	31.69	35.36	$+3.67$
AdaKV-style cross-head allocation	—	40.64	-0.06

Table 28: The validated Tactic reproduction under this paper's protocol (Llama-3.1-8B, LongBench 16-task average, same samples, kernels and scorer as Table 1). KV read is the share of the cache each method names; pages touched is the share a paged serving stack must fetch to deliver it. The two coincide for CVSA, which reads pages directly. Both columns count attention traffic only; scoring traffic is larger for CVSA than for Tactic and is accounted for in the text.

Method	Quality	Δ	KV read	Pages touched
Dense	42.57	—	100%	100%
CVSA-c4	41.87	-0.71	11.67%	11.67%
Tactic (reproduced)	41.82	-0.76	22.98%	48.60%
+ coverage gate	42.49	-0.08	31.73%	59.21%

The top-p rows of Table 26 expand the selection until an estimated mass threshold is met at page granularity, using an fp16 estimator stronger than the 4-bit estimator of the released Twilight implementation, so that family is evaluated in the most favorable setting we could construct for it.

Reproducing Tactic. Tactic [8] has no public reference implementation, so we reimplemented it from the paper: k-means over keys at prefill, decode-time cluster ranking by centroid inner product, exact scores for the leading clusters, a $y = a/x + b$ fit for the remaining tail, and accumulation until the estimated attention mass reaches the target, unioned across each GQA group. We validated it against the two operating points the paper reports for Llama-3.1-8B, on the six tasks it evaluates, against a criterion fixed before the runs: the reproduction reaches -0.54 against a published -1.37 at a target mass of 0.70, and -0.59 against a published -0.34 at 0.90.

One departure from the text is necessary, and we state it plainly: we force the first four tokens into every selection. The paper does not mention attention sinks, and without them the reproduction misses the published low-target point by 3.35 while matching the high-target one, since at a low target too few clusters are selected for the sink to be included incidentally. Adding the sinks raises quality by 4.19 and *lowers* read by 1.81 points, and brings both operating points inside the criterion, so the departure strengthens and cheapens the method rather than weakening it. A recent-window variant was also tried and is not used: it did not improve on sinks alone, and inventing further machinery for a competing method is its own kind of infidelity.

Table 28 places the validated reproduction under this paper's protocol. Quality is matched: 0.05 points separate the two methods, well inside run-to-run variation, so neither is the more accurate. They differ in what they read to get there. Tactic selects individual tokens, and those tokens are scattered by construction, because its clusters group keys by similarity rather than by position. Charged at page granularity, the unit a paged serving stack fetches and the setting this paper targets, the 22.98% of tokens it names occupy 48.60% of pages; CVSA reads pages directly, so its two figures coincide at 11.67%. Even before page granularity is applied, the reproduction names roughly twice the cache share for the same quality.

These columns count attention traffic only, and the scoring traffic that each method needs to produce its selection runs the other way. CVSA scores pages from a full key scan amortized over the $M{=}4$ reuse window, about 12.5% of the cache per step, while Tactic's centroid ranking, leading-cluster scores and tail fit come to roughly 5%. Including both, the totals are 24.2% for CVSA against 28.0% for the reproduction where token gather is free, and 53.6% on a paged stack. The advantage in total traffic is therefore a factor of 1.2 in the first setting and 2.2 in the second, not the ratio the attention columns alone would suggest.

Table 29: Realized coverage of the released Twilight implementation at its default configuration, read from its own telemetry (Llama-3.1-8B-Instruct, 16 LongBench tasks, 100 samples each). Coverage is the per-generation mean exact-weight attention mass captured by the method's own selection, and Read is the mean number of selected tokens divided by the context's word count. The configured target is 0.85; the realized value is below it on 1600 of 1600 generations, in every task.

Task	Coverage	Read	Task	Coverage	Read
narrativeqa	0.634	2.1%	passage_retrieval_en	0.766	1.2%
musique	0.719	2.2%	triviaqa	0.767	1.8%
qmsum	0.726	2.4%	trec	0.786	5.4%
repobench-p	0.743	5.9%	multifieldqa_en	0.799	4.4%
hotpotqa	0.743	2.5%	2wikimqa	0.806	3.8%
passage_count	0.754	1.2%	lcc	0.809	9.3%
samsum	0.759	2.9%	qasper	0.812	4.8%
gov_report	0.759	2.8%	multi_news	0.815	5.2%

Composing the unchanged coverage gate on top of the reproduction is the result that bears on this paper's claim. It moves Tactic from -0.76 to -0.08, closing most of the remaining distance to dense, and the verifier rejects 64.85% of the checks it performs: over sixteen tasks the pre-hoc mass target is missed about two times in three, and repairing those steps is worth 0.68 points at the cost of the traffic the repair consumes. Verification is therefore complementary to this selector rather than an alternative to it, which is the same relationship the audit below finds for the deployed member of the family. The comparison is at this paper's protocol; the six validation tasks are Tactic's own selection and are weighted toward retrieval and extractive question answering, where token-granular selection is at its strongest.

Auditing a pre-hoc adaptive budget at its source. The top-p rows above instantiate the pre-hoc principle inside our own pipeline; it can also be audited where it is actually deployed. We ran the released Twilight implementation [9] unchanged, at its own default configuration (Quest selector, token budget 8192, hierarchical top-p pruning at p=0.85, 4-bit min/max weight estimator) on Llama-3.1-8B-Instruct, over all sixteen LongBench tasks at 100 samples each. That implementation logs, for each generation, the exact-weight attention mass captured by its own selection, which is the same quantity as our coverage c. The audit therefore runs entirely on the method's own code, configuration and instrumentation, and is independent of our kernels, our scorer and our page granularity.

Realized coverage falls below the configured 0.85 target on all 1600 generations, with per-task means from 0.634 to 0.815 and a single-generation minimum of 0.504, so the shortfall is the normal operating point rather than a tail event. Output quality meanwhile stays close to the method's own dense arm (mean -0.35 over the sixteen tasks under a common scorer), so nothing in the output indicates that the target was missed. That combination is the condition a certificate addresses. The threshold is applied to estimated weights before the step is executed, so the realized mass is never formed as a quantity the system could act on, and at run time a step that met its target is indistinguishable from one that did not. Two scope conditions apply. Absolute scores are not comparable across the two harnesses, so only each system's difference from its own dense reference within its own harness is quoted. And the audit is at token granularity and low read (0.9–4.1% of context tokens); the same pre-hoc principle at the page granularity that paged serving stacks expose is the top-p row of Table 27, where adding verification recovers $+3.67$ points.

Threshold-sweep telemetry. Table 30 extends the threshold sweep of Table 6 with the verifier's internal telemetry. A single global τ trades traffic for coverage monotonically: raising τ from 0.80 to 0.97 more than halves per-head acceptance ($49.6\% \to 19.1\%$) and grows re-scoring ($79.4\% \to 98.1\%$) and effective read ($10.3\% \to 13.0\%$), while quality stays within 0.14 points. This is why $\tau = 0.90$ is used everywhere without per-setting tuning: the loop converts the traffic that τ controls into nearly constant quality.

For practitioners the reading is simple: τ sets the coverage floor one insists on per head, its cost is paid in read traffic rather than in quality, and 0.90 is a sound default at every scale we measured, used unchanged across three 7B architectures and at 70B. Raising τ buys a stricter guarantee at a few points of extra traffic; lowering it saves traffic at the same quality until the floor becomes loose enough that recovery is no longer triggered where it is needed.

Table 30: Threshold sweep with verifier telemetry (three-model average; accept and re-score rates are task means). Quality moves by 0.14 points across the full range while acceptance, re-scoring, and effective read vary substantially. $\tau = 0.90$ is the deployed default at every scale.

τ	Quality (3-model)	Effective read	Per-head accept	Re-score rate
0.80	40.82	10.3%	49.6%	79.4%
0.85	40.72	10.8%	44.7%	83.9%
0.90 (deployed)	40.70	11.7%	37.8%	89.5%
0.95	40.84	12.6%	27.1%	95.4%
0.97	40.83	13.0%	19.1%	98.1%

Table 31: Fully lightweight CVSA on Llama-3.1-8B (16 LongBench tasks, identical samples). Replacing the exact normalizer with the sketch changes quality by $+0.06$ (CI $[-0.73, +0.84]$); the remaining gap to CVSA-c4 is -1.80 (CI $[-2.82, -0.81]$) and is attributable to the selector, not the certificate. The fully lightweight path has the lowest total read traffic of any configuration measured.

Configuration	Quality	Effective read	Total read incl. scoring
Dense	42.57	100%	1.00×
CVSA-c4 (exact selector + norm.)	41.87	11.67%	0.24×
Quest selector, exact norm.	40.01	12.36%	0.25×
Quest selector, sketch norm. (lightweight)	40.07	12.95%	0.15×

Fully lightweight end-to-end. The selector-agnostic experiments above keep the exact scorer for the gate's normalizer. Table 31 closes that gap: a Quest min/max sketch supplies both the page selection and the gate's all-context normalizer, so no full-key scan runs anywhere in the pipeline, including recovery expansion. On Llama the fully lightweight configuration scores 40.07 at 12.95% effective read, against 40.01 for the same selector with an exact normalizer (a $+0.06$ difference, paired-bootstrap 95% CI $[-0.73, +0.84]$): estimating the normalizer carries no discernible quality cost, and the gated configuration retains its margin ($+2.25$ over the ungated 37.82, versus $+2.19$ for the exact-normalizer gate). Exactness matters in the selector (exchanging it for the sketch costs 1.86 points) rather than in the certificate. The sketch gate also fails conservatively: it accepts 24.8% of head checks against 31.1% and re-scores at 99.5% against 94.8%, turning estimation noise into extra recovery traffic rather than silent quality loss. Total read traffic, including page scoring amortized over $M{=}4$, is $\sim$0.15× dense for the fully lightweight configuration versus $\sim$0.24× for CVSA-c4, since the sketch reads only per-page min/max metadata rather than full keys.

P Limitations and Future Directions

Scope of the median decode-step speedup. The median decode-step speedup that CVSA provides is moderate at short context ($1.76\times$ at 8K, $B{=}32$) because weight reads and non-attention work remain a large share of the decode step at that operating point. The advantage grows with context length and batch size and reaches $6.21\times$ on Qwen at 128K with $B{=}8$, where KV traffic dominates decode-step latency; Llama and Mistral reach $1.19\times$ at 128K with $B{=}1$ (Table 23). Applications that operate primarily at short context would see limited benefit from the current framework.

Theoretical coverage of reuse steps. The formal error guarantee (Proposition 2, corollary) applies to a single scoring step with a fixed selected set. On reuse steps ($t \bmod M \neq 0$), the page selection and the all-context normalizer are stale and no check runs; the check at the next scoring step serves as an empirical safeguard rather than a formal bound. A compositional error analysis that tracks how single-step staleness accumulates across the reuse window is an open theoretical question. The runtime coverage traces in Fig. 6, and the 32,768-step trace in Section C, indicate that selection drift is bounded in practice, but a formal treatment would strengthen the guarantee for deeper models where compounding has more layers to traverse.

Page-level proxy fidelity. The deployed verifier uses a page-max proxy for coverage rather than the exact token-level quantity. The proxy over-accepts on all three models: in the production probe of Section C, 31%, 33%, and 39% of proxy accepts on Llama, Mistral, and Qwen are false accepts. The harm concentrates on Qwen, where false accepts carry the largest output error and cluster in late layers where the page-max saturates under 7:1 GQA. Token-aware or GQA-aware proxy calibration

Table 32: Mean RULER accuracy gap to dense (points), averaged over the six tasks and three models. The 4K–32K column covers every method on the same 72 cells and is the basis of the RULER numbers in the main text; the second column adds the 64K and 128K cells each method was run on (cell counts in parentheses).

Method	4K–32K	All measured contexts
CVSA-c4	-0.83 (72)	-0.93 (108)
CVSA-c8	-1.02 (72)	-1.48 (108)
SnapKV	-16.16 (72)	-17.07 (90)
H2O	-8.70 (72)	-9.11 (90)
AdaKV	-9.44 (72)	-9.48 (90)
Fixed-10%	-9.35 (72)	-8.95 (108)

would address this drift and is the most direct route to improving quality on architectures with high GQA ratios.

Depth-dependent scheduling. At 70B scale the scoring interval must be reduced from $M{=}4$ to $M{=}2$ to contain the additional per-layer drift that 80 layers introduce. This is currently a manual guideline based on the ablation in Table 4. An adaptive policy that sets M based on observed coverage statistics during the first few decode steps would make the framework self-tuning, and is a natural extension of the coverage check.

Reasoning-length generation. CVSA targets long-context decoding, where the prompt is long but the generated answer is short. Long chain-of-thought reasoning, where the generated trace itself is long, is a different regime, which we probe here to map the method's boundary. Transferring the deployed $M{=}4$ schedule to it unchanged is suboptimal: on MATH-500 (R1-Distill-Llama-8B, 200 problems, seed-paired sampling) CVSA-c4 reaches 64.0% against dense 80.5%. The reuse window, rather than verification, accounts for the larger part of this loss: refreshing the selection every step ($M{=}1$) recovers two-thirds of the gap, to 75.0% ($+11.0$ points over $M{=}4$, paired-bootstrap CI $[+5.5, +16.5]$), and more than halves both signatures of non-convergence: cap-outs fall from 31 to 13 of 200 problems and generation-length inflation from $+61\%$ to $+20\%$ above dense. A residual 5.5-point gap remains and the per-step schedule runs at $0.93\times$ dense decode (8K, $B{=}32$), so we present reasoning-length decoding as future work, addressed by a drift-driven adaptive schedule that shortens the reuse window when the generation is long (consistent with the depth-dependent guideline above), rather than as a current capability.

Broader impacts. Reducing inference cost can improve access to long-context models. It can also make harmful generation cheaper; deployments inherit the underlying model's misuse, privacy, and bias risks. Attention efficiency does not itself mitigate these risks.

Q Reproducibility

The implementation of CVSA, including the paged-decode pipeline, coverage verification, hybrid scheduling, and evaluation scripts, is prepared for public release at `https://github.com/DJ-Melvin/coverage-verified-sparse-attention`. The repository contains: (1) the Triton kernels for page scoring, sparse attention, and coverage computation, (2) the verify-then-correct decode loop with configurable τ, M, and recovery budget, (3) evaluation harnesses for LongBench and RULER with the exact prompt templates and sampling parameters used in this paper, (4) launch scripts and a guide mapping every table and figure in the main paper and appendix to its command, and (5) pre-computed result JSON files that allow figure generation without GPU access. Model weights are publicly available from the respective model providers (Meta Llama, Mistral AI, Alibaba Qwen). LongBench and RULER use greedy decoding, which is deterministic given the same model weights and input; the MATH-500 and stochastic-decoding experiments use seed-paired sampling. LongBench evaluates the first N samples of each task and RULER instances are generated from a fixed seed, so all methods see identical inputs. Speedup is reported as the ratio of median decode-step latencies over the timed steps of each run; aggregate tokens/s is a different statistic (Section M).

Table 33: RULER accuracy (%) on Llama-3.1-8B. Through 32K each cell averages 10 trials, at 64K and 128K five; "–" marks a setting not run. Baselines use a 10% budget.

Task	Context	Dense	CVSA-c4	CVSA-c8	SnapKV	H2O	AdaKV	Fixed-10%
S-NIAH	4K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	8K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	16K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	32K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	64K	100.0	100.0	100.0	–	–	–	100.0
	128K	100.0	100.0	100.0	–	–	–	100.0
MK-NIAH	4K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	8K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	16K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	32K	90.0	90.0	90.0	90.0	90.0	90.0	90.0
	64K	80.0	100.0	100.0	–	–	–	80.0
	128K	100.0	80.0	80.0	–	–	–	80.0
MV-NIAH	4K	80.0	70.0	80.0	70.0	70.0	80.0	70.0
	8K	60.0	60.0	60.0	60.0	60.0	60.0	60.0
	16K	50.0	50.0	50.0	50.0	40.0	60.0	50.0
	32K	60.0	50.0	50.0	50.0	50.0	50.0	50.0
	64K	20.0	20.0	20.0	20.0	20.0	20.0	20.0
	128K	40.0	40.0	40.0	40.0	40.0	40.0	40.0
MQ-NIAH	4K	100.0	100.0	96.7	60.0	90.0	90.0	93.3
	8K	100.0	100.0	96.7	66.7	86.7	90.0	83.3
	16K	100.0	100.0	100.0	70.0	90.0	96.7	93.3
	32K	100.0	96.7	100.0	60.0	86.7	80.0	83.3
	64K	93.3	93.3	100.0	80.0	86.7	80.0	86.7
	128K	86.7	86.7	86.7	80.0	86.7	86.7	86.7
VT	4K	70.0	70.0	80.0	80.0	80.0	80.0	80.0
	8K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	16K	80.0	80.0	80.0	80.0	80.0	80.0	80.0
	32K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	64K	40.0	40.0	40.0	–	–	–	40.0
	128K	40.0	40.0	40.0	–	–	–	40.0
CWE	4K	100.0	93.3	76.7	63.3	76.7	73.3	73.3
	8K	100.0	90.0	93.3	40.0	76.7	90.0	96.7
	16K	100.0	100.0	96.7	56.7	96.7	90.0	93.3
	32K	100.0	100.0	100.0	43.3	93.3	86.7	86.7
	64K	100.0	100.0	100.0	53.3	93.3	100.0	93.3
	128K	100.0	93.3	73.3	60.0	60.0	73.3	60.0

Table 34: RULER accuracy (%) on Mistral-7B-v0.3. Through 32K each cell averages 10 trials, at 64K and 128K five; "–" marks a setting not run. Baselines use a 10% budget.

Task	Context	Dense	CVSA-c4	CVSA-c8	SnapKV	H2O	AdaKV	Fixed-10%
S-NIAH	4K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	8K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	16K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	32K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	64K	100.0	100.0	100.0	–	–	–	100.0
	128K	100.0	100.0	100.0	–	–	–	100.0
MK-NIAH	4K	100.0	100.0	100.0	90.0	90.0	90.0	90.0
	8K	100.0	100.0	100.0	90.0	90.0	90.0	90.0
	16K	100.0	100.0	100.0	90.0	90.0	90.0	90.0
	32K	100.0	100.0	100.0	100.0	100.0	90.0	90.0
	64K	80.0	60.0	60.0	–	–	–	40.0
	128K	20.0	0.0	0.0	–	–	–	20.0
MV-NIAH	4K	70.0	70.0	70.0	60.0	60.0	60.0	60.0
	8K	90.0	90.0	90.0	60.0	80.0	80.0	80.0
	16K	90.0	80.0	80.0	30.0	80.0	60.0	60.0
	32K	70.0	80.0	70.0	20.0	50.0	50.0	50.0
	64K	40.0	40.0	40.0	20.0	20.0	20.0	20.0
	128K	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MQ-NIAH	4K	100.0	100.0	100.0	83.3	100.0	100.0	100.0
	8K	100.0	100.0	100.0	80.0	96.7	96.7	96.7
	16K	100.0	100.0	100.0	86.7	96.7	96.7	96.7
	32K	100.0	100.0	100.0	80.0	96.7	96.7	96.7
	64K	100.0	100.0	93.3	66.7	100.0	86.7	86.7
	128K	60.0	73.3	60.0	20.0	53.3	53.3	53.3
VT	4K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	8K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	16K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	32K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	64K	60.0	60.0	60.0	–	–	–	60.0
	128K	20.0	20.0	20.0	–	–	–	40.0
CWE	4K	100.0	96.7	90.0	100.0	100.0	100.0	100.0
	8K	100.0	96.7	96.7	96.7	100.0	100.0	100.0
	16K	100.0	96.7	96.7	86.7	100.0	100.0	100.0
	32K	100.0	100.0	96.7	96.7	100.0	100.0	100.0
	64K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	128K	93.3	86.7	86.7	93.3	86.7	93.3	93.3

Table 35: RULER accuracy (%) on Qwen2.5-7B. Through 32K each cell averages 10 trials, at 64K and 128K five; "–" marks a setting not run. Baselines use a 10% budget.

Task	Context	Dense	CVSA-c4	CVSA-c8	SnapKV	H2O	AdaKV	Fixed-10%
S-NIAH	4K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	8K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	16K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	32K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	64K	100.0	100.0	100.0	–	–	–	100.0
	128K	20.0	20.0	20.0	–	–	–	20.0
MK-NIAH	4K	100.0	100.0	100.0	70.0	70.0	70.0	70.0
	8K	100.0	100.0	100.0	70.0	70.0	70.0	70.0
	16K	90.0	100.0	100.0	70.0	70.0	70.0	70.0
	32K	100.0	100.0	100.0	70.0	70.0	70.0	70.0
	64K	60.0	60.0	60.0	–	–	–	40.0
	128K	20.0	20.0	20.0	–	–	–	0.0
MV-NIAH	4K	100.0	100.0	100.0	30.0	60.0	40.0	60.0
	8K	90.0	90.0	90.0	20.0	70.0	60.0	60.0
	16K	100.0	90.0	100.0	30.0	80.0	70.0	70.0
	32K	80.0	80.0	80.0	40.0	60.0	60.0	50.0
	64K	80.0	80.0	80.0	20.0	60.0	80.0	60.0
	128K	60.0	60.0	60.0	60.0	60.0	60.0	60.0
MQ-NIAH	4K	100.0	100.0	100.0	56.7	63.3	53.3	50.0
	8K	100.0	100.0	100.0	46.7	50.0	43.3	50.0
	16K	100.0	100.0	100.0	60.0	40.0	43.3	40.0
	32K	100.0	96.7	96.7	60.0	53.3	53.3	63.3
	64K	100.0	93.3	93.3	33.3	53.3	53.3	46.7
	128K	66.7	60.0	53.3	40.0	40.0	46.7	33.3
VT	4K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	8K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	16K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	32K	70.0	70.0	70.0	70.0	70.0	70.0	70.0
	64K	40.0	40.0	40.0	–	–	–	40.0
	128K	20.0	20.0	20.0	–	–	–	20.0
CWE	4K	100.0	93.3	90.0	96.7	93.3	96.7	96.7
	8K	100.0	100.0	100.0	100.0	100.0	96.7	96.7
	16K	100.0	100.0	100.0	96.7	96.7	96.7	96.7
	32K	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	64K	100.0	100.0	100.0	93.3	93.3	93.3	93.3
	128K	86.7	100.0	93.3	73.3	80.0	66.7	80.0

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state that CVSA targets long-context decode efficiency while preserving dense-attention quality. The empirical scope is then supported in Sections 4–4.3 and bounded by Appendix P.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Appendix P discusses the long-context operating regime, the page-level proxy, reuse-window guarantees, fixed operating points, and the lack of a closed-form layer-wise error propagation model.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: Propositions 1 and 2 state the selected-set and token-level coverage setup, and Appendix B provides the complete algebraic proofs and clarifies the distinction between token-level coverage and the runtime page-level proxy.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: Sections 4–4.3 specify the models, benchmarks, baselines, budgets, scheduling interval, recovery policy, and hardware setup. The appendix reports per-task, per-model, bootstrap, RULER, throughput, and ablation details used to support the main claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: Code will be publicly released at `https://github.com/DJ-Melvin/coverage-verified-sparse-attention` upon publication. The repository includes the implementation, evaluation and plotting scripts, figure inputs, and REPRODUCIBILITY.md with reproduction instructions.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: This work performs inference-time evaluation rather than training. Section 4 specifies the benchmark suite, model families, baselines, sparse budgets, threshold, scheduling interval, recovery budget, dense-layer policy, and decoding setting; appendices provide the expanded tables.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

Justification: The main LongBench comparisons report bootstrap confidence intervals, and Appendix C distinguishes main-table point estimates from the per-model paired bootstrap intervals and specifies the resampling procedure. Speedups are ratios of median timed decode-step latencies.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The experiments report the GPU class and relevant memory regime, including H100 80GB, H200, and 4×H200 settings where applicable. Speedup tables specify context length, batch size, and the timed decode-step protocol; the paper does not rely on training runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work studies inference efficiency for existing publicly available language models and benchmarks and does not introduce human-subject data collection.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: Appendix P discusses the scope of the method as an inference-efficiency technique. The main societal effect is lower serving cost and memory traffic for long-context LLMs; the method can also make long-context generation cheaper, so deployment inherits the risks of the underlying model.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release a new pretrained model, dataset, scraped corpus, or generator. It proposes an inference-time sparse-attention mechanism evaluated on existing models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The paper cites the model families, benchmarks, and baseline methods used in the evaluation, including LongBench, RULER, SnapKV, H2O, AdaKV, Quest, vLLM/PagedAttention, and FlashAttention-style kernels. We use these assets for research evaluation under their respective public terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The new asset is the research code for the CVSA inference-time implementation and evaluation pipeline, prepared for release under the MIT license at `https://github.com/DJ-Melvin/coverage-verified-sparse-attention` with documentation and reproduction scripts; the paper does not introduce a new dataset or pretrained model.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The work does not involve crowdsourcing, user studies, or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The work does not involve crowdsourcing, user studies, or human-subject experiments, so IRB or equivalent approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: LLMs are the evaluation targets, not an original or non-standard component used to conduct the research beyond the inference experiments reported in the paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.