QUANTIZE THE KEYS, AUDIT THE VALUES RETAINED-STATE CERTIFICATES FOR KV-CACHE COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

KV-cache compression aims to support long-context inference within limited memory while preserving output quality. How that memory is distributed matters because compression in different cache regions can affect the current attention output differently. Aggregate task scores assess overall quality but do not by themselves provide query-specific feedback for allocating memory within a compressed cache. We provide this feedback with a deterministic certificate that bounds attention error for each query, even after the original entries are discarded. It also gives the exact reduction in the bound that each increase in value precision would produce, which enables adaptive allocation while the cache stays compressed. This allocation lowers attention error by 30–67% under the same memory budget and reaches a calibrated value-error target with roughly two-thirds as much refinement memory as uniform allocation. In packed serving of two 7–8B models at 32K, adaptive allocation brings the next-token distribution 2.6–3.4 times closer to dense decoding in KL divergence than uniform allocation with the same memory. The design is also practical to serve. A fused kernel evaluates the certificate alongside attention with little additional arithmetic and reads the selected state directly from a packed cache. Its base representation uses about $0.26\times$ dense KV storage and serves a 70B model at 128K on one GPU.

1 INTRODUCTION

Transformer-based language models retain the keys and values of earlier tokens to avoid recomputing them during autoregressive generation. These stored states form the KV cache, which grows with context length and can consume a substantial share of GPU memory (Li et al., 2025a, Shi et al., 2024). Compression reduces this footprint in several ways. Eviction discards entries that attention statistics mark as unimportant (Zhang et al., 2023, Li et al., 2024), merging folds entries together (Zhang et al., 2024), low-rank projection shrinks every entry (Chang et al., 2025, Saxena et al., 2024), sharing across heads or layers changes the architecture (Ainslie et al., 2023, DeepSeek-AI, 2024, Brandon et al., 2024), and quantization stores every entry at reduced precision (Liu et al., 2024c, Hooper et al., 2024). Each can change attention outputs and degrade generation quality.

The system must therefore decide where to preserve precision within its memory budget. Two equally sized cache regions can have different compression errors and receive very different attention weights from the current query, so spending extra memory on one need not have the same effect as spending it on the other. The relative benefit can also change as decoding proceeds. Task benchmarks assess overall generation quality and provide none of the local feedback these choices need (Hsieh et al., 2024, Bai et al., 2024).

Obtaining this feedback poses a dilemma for the serving system. Computing the exact error requires a reference output from the original entries, but keeping a full-precision copy on the GPU would consume the memory that compression is meant to save. The challenge is therefore to retain enough information to guide allocation for each new query without keeping the original entries or imposing substantial storage and computation costs during decoding.

Our *retained-state certificate* addresses this dilemma with compact bounds on reconstruction error and value magnitude that are recorded during compression. These *receipts* let the system combine

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

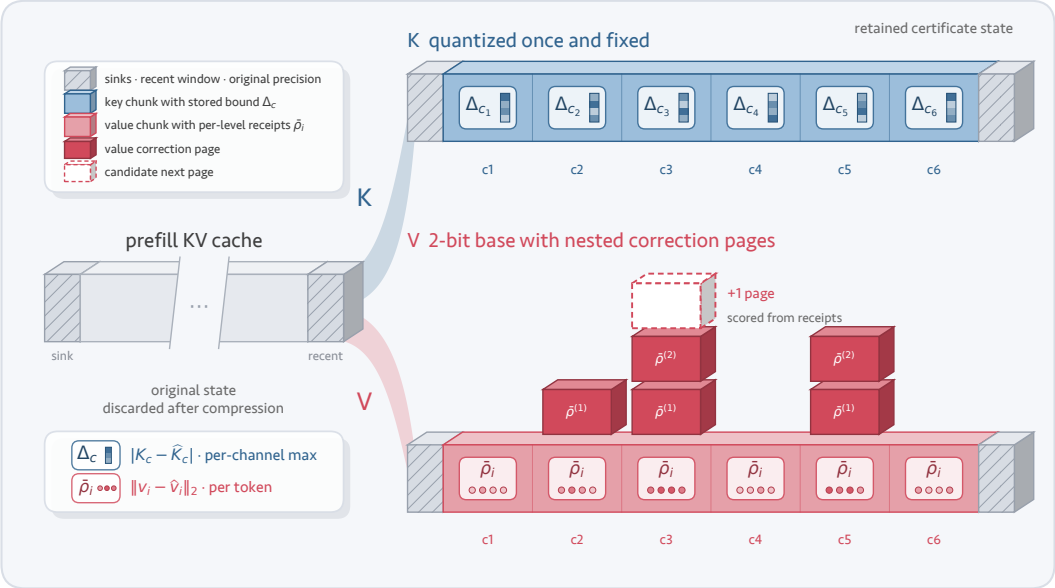


Figure 1. In the certificate-guided KV cache, sink and recent tokens stay exact, older keys are quantized once, and older values keep a 2-bit base with optional correction pages. Retained error summaries bound the attention error for each query and give the exact decrease in the bound that each candidate page would produce.

the retained cache with each new query and compute a deterministic upper bound on each head's attention error in exact arithmetic without accessing the discarded entries.

The certificate also tells the system where extra memory would help, because keys and values enter the bound differently. Changing a key can alter the weight on every value through the softmax, whereas values enter the output linearly once the query and keys are fixed. Guided by this asymmetry, our design quantizes both keys and values after training without calibration data, fixes the key precision once with per-channel error bounds, and spends any additional precision on values. Figure 1 shows the resulting layout, in which compressed values are grouped into chunks, each stored as a low-precision base that can be refined with small blocks of stored corrections called *correction pages*. Among the compression families, quantization keeps every token represented for later queries and records a reconstruction error bound for each entry at compression time, which is the information the certificate retains.

Each candidate page lowers the bound by an exactly computable amount for the current query, which lets a controller compare candidate pages and allocate a shared memory budget across chunks before installing any of them. Practical serving also requires this feedback to remain inexpensive in both memory and computation. We therefore use compact receipts and page metadata together with a fused Triton kernel (Tillet et al., 2019) that reads the selected state directly from the packed cache and accumulates certificate statistics alongside attention.

These considerations show that an inference-time certificate is useful only if it remains sound after the original entries in the compressed span are gone, assigns meaningful value to the decisions the system can take, and costs little enough to evaluate and store at every decode step.

Our contributions are as follows. 1) We show how to bound the attention error introduced by KV-cache compression at every decode step. The bound is validated by violation-free audits on models from 360M to 70B, and its structure explains the different roles of keys and values. 2) We make the bound actionable for memory allocation. Across six models and three corpora, adaptive allocation lowers attention-output error at the same byte budget and reaches a calibrated value-error target with roughly two-thirds of the refinement pages that uniform allocation needs. In packed serving at 32K it also brings the model output closer to dense decoding than uniform allocation. 3) We carry the design through to direct packed serving. The 2-bit base uses about $0.26\times$ dense KV storage, or $0.31\times$ with the selected pages, and the Triton kernel reaches near-dense latency at 32K and serves a 70B model at 128K on a single GPU.

2 A CERTIFICATE FROM THE COMPRESSED CACHE

Setting. At each decode step, a head of dimension d receives a query $q \in \mathbb{R}^d$ and attends over n cached entries (k_i, v_i), with positional rotation already applied. The attention sinks and a recent window stay exact, while the remaining entries form the compressed span T, which is partitioned into chunks c of C tokens. Exact attention forms scores $s_i = \langle q, k_i \rangle / \sqrt{d}$, weights $p = \mathrm{softmax}(s)$, and output $o = \sum_i p_i v_i$. The stored reconstructions $\hat{k}_i, \hat{v}_i$ equal the original entries outside T and define the served weights $\tilde{p}$ and output $\tilde{o} = \sum_i \tilde{p}_i \hat{v}_i$ in the same way. The exact weights and output are the reference against which error is measured. Computing the certificate uses only the compressed cache, its receipts, and the current query.

Receipts. Receipts are compact bounds recorded during compression and kept after the original entries in T are discarded. Keys are quantized per channel as in KIVI (Liu et al., 2024c), and each key group additionally retains one per-channel bound on its reconstruction error. Each compressed value i stores a residual bound $\bar{\rho}_i \geq \|v_i - \hat{v}_i\|_2$ on its reconstruction error together with a bound on its original norm $\|v_i\|_2$. The norm bounds describe the original values and therefore stay fixed across the value precision levels introduced in Section 3, while the residual bounds describe each level's reconstruction. The analysis uses only these bounds, so it applies to any representation that supplies them, while decisions that remove or re-encode keys, such as eviction, lie outside its scope.

Certificate anatomy. To bound attention error, we separate the effects of key and value compression. Every original cache consistent with the stored bounds satisfies

$$o - \tilde{o} = \underbrace{\sum_i (p_i - \tilde{p}_i) v_i}_{\text{changed attention weights}} + \underbrace{\sum_{i \in T} \tilde{p}_i (v_i - \hat{v}_i)}_{\text{value reconstruction}}. \tag{1}$$

Key errors change the attention weights, while value errors change the vectors being averaged. The second contribution separates over the chunks of T, so the certificate can also track where value refinement would help.

Theorem 1. *Fix the compressed cache with its receipts and a query q. For every original cache consistent with the stored receipts, the attention error in exact arithmetic satisfies*

$$\|o - \tilde{o}\|_2 \leq \underbrace{E_{\mathrm{key}}(q)}_{\textit{changed attention weights}} + \underbrace{\sum_c \sum_{i \in c} \tilde{p}_i \bar{\rho}_i}_{\textit{value reconstruction}} = E(q). \tag{2}$$

The two terms bound the two contributions in equation 1. For the key term, token i's group bound gives a score radius $\varepsilon_i \geq |s_i - \tilde{s}_i|$ from one inner product between $|q|$ and the group's bound vector, with $\varepsilon_i = 0$ on the exact entries. Writing $\bar{\nu}_i$ for the stored value-norm bound, equal to $\|v_i\|_2$ on the exact entries, and $A = \sum_i \tilde{p}_i (e^{\varepsilon_i} - 1)$ and $B = \sum_i \tilde{p}_i (1 - e^{-\varepsilon_i})$ for two attention-weighted sums of the radii, the key term is

$$E_{\mathrm{key}}(q) = \min\Big\{ \frac{\sum_i \tilde{p}_i \big(A + e^{\varepsilon_i} - 1\big) \bar{\nu}_i}{1 - B},\ 2 \max_i \bar{\nu}_i \Big\}. \tag{3}$$

The first branch charges each token in proportion to its served attention weight instead of through a uniform Lipschitz constant, which keeps the term informative on long spans where most tokens carry little attention. Appendix A.6 shows that it is sharp to first order for tokens that carry a small share of the attention. The second branch caps the error by the largest stored norm. The term is small when the key errors are small along the current query direction or when attention concentrates on the exact sinks and recent window (Xiao et al., 2024b). It stays fixed when only value precision changes, because $\tilde{p}$, every ε_i, and every $\bar{\nu}_i$ are then unchanged. The value term adds each token's attention weight times its residual bound, chunk by chunk. The guarantee covers one head and one decode step and uses only information available after compression. Appendix A.3 gives the proof.

Computational cost. In order to be deployable during inference, a certificate consulted at every decode step must satisfy two requirements. First, its time complexity must stay far below that of the

attention it certifies. Second, its space complexity must stay far below that of the cache it describes. The design meets both by reusing what serving already computes. Each of the G key groups stores one error-bound vector, so its radii cost one inner product with $|q|$, while A and B accumulate alongside the softmax. In total, the certificate adds $O(Gd + |T|)$ time and space per head and step, while attention and the cache each scale as $O(nd)$, so the relative overhead falls as the key group size and the head dimension grow. Evaluation needs no calibration data and no second pass over the cache, and the packed kernel accumulates these statistics in the same pass that serves attention, so its measured latencies include their accumulation. Scalar receipts and a nested value representation that shares bits across precision levels meet the space requirement.

3 Allocating value correction pages

Values in the compressed span are stored in chunks of C tokens. Each chunk keeps a 2-bit base and up to three correction pages, where each page quantizes to 2 bits the residual left by the previous level. The four levels are nested, so the base and its three pages together hold eight bits of codes per element, and every level records its own residual bound $\bar{\rho}_i$ together with its affine metadata. Installing a page replaces a chunk's reconstruction and residual bounds with those of the next level, while the keys and attention weights stay fixed. The certificate gives the resulting change in the bound for the current query.

Proposition 1. *Fix the query, the keys, and the value-norm receipts. If one correction page changes chunk c's residual bounds from $\bar{\rho}_i^{\mathrm{before}}$ to $\bar{\rho}_i^{\mathrm{after}}$, the certificate of Theorem 1 changes by exactly*

$$\Delta E_c(q) = E_{\mathrm{before}}(q) - E_{\mathrm{after}}(q) = \sum_{i \in c} \tilde{p}_i \big(\bar{\rho}_i^{\mathrm{before}} - \bar{\rho}_i^{\mathrm{after}}\big). \tag{4}$$

The key term and every unchanged chunk cancel in the difference. Each token of the refined chunk contributes its attention weight times the reduction in its stored residual bound, and the sum is the exact change in the certificate even when the total bound is loose. This result turns the value side of the certificate into a price list. Among candidate pages of equal cost, the controller installs the one with the largest positive $\Delta E_c(q)$ and repeats this choice until its target is met or no page lowers the bound. Keys admit no such shortcut, because a change to one key's score changes every softmax weight through the shared normalization. Whether these exact reductions in the bound also reduce the realized error is an empirical question, which Section 4 answers on held-out queries.

The price list costs little to keep. Receipts and page metadata together cost a few percent of the dense cache, and smaller chunks allow finer allocation at the cost of more metadata.

4 Experiments

The experiments proceed from the certificate itself to its use, testing first that the bound holds on served models, then that its page price improves memory allocation and preserves task quality at matched budgets, and finally that the packed representation serves efficiently. The allocation experiments compare two policies that start from the same 8-bit keys and 2-bit value base and install the same total page bytes. Uniform allocation gives every chunk the same number of correction pages, whereas adaptive allocation ranks candidate pages by their mean certificate decrease ΔE_c over a set of decision queries and installs the best ones from a pool shared across heads and layers. We measure p99 attention-output error against the original cache on held-out queries and report the ratio of uniform to adaptive error.

Both policies replay recorded attention streams from six models on WikiText-103, PG19, and C4, following the protocol in Figure 2. Each 2K window fixes its first 1,536 tokens as the cache, and the remaining 512 queries attend to the compressed cache without appending their own keys or values. The first 256 queries decide the placement and the last 256 evaluate it, which matches the single-query interface of Theorem 1. Original-cache outputs serve only as references for error measurement and audits and never enter the certificate or the page selection. In a second replay, shown in Figure 3, we calibrate a value-error target on the decision queries and count the pages each policy needs to meet it on the held-out queries.

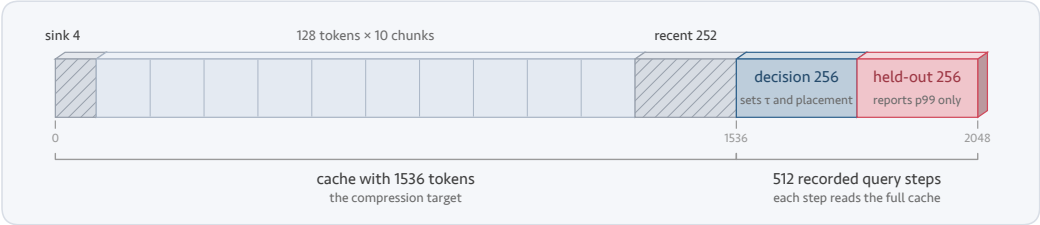


Figure 2. In each recorded window, the first 256 queries select an equal-budget allocation and the remaining 256 measure its error with the placement held fixed, while sink and recent tokens stay exact.

Certificate soundness and page pricing. We observe no violations in more than 2.5 billion float64 checks across packed sweeps, recorded-stream replays, the page-price ablation, and serving audits, which Appendix C itemizes. Keeping four sink tokens exact costs 0.02 percentage points of dense cache storage and more than halves the frequency with which the key term reaches its cap, the second branch of equation 3. The bound itself is conservative. On the representative 16K configurations reported in Appendix D.1, with 8-bit keys and four exact sink tokens, it exceeds the realized error by a median factor of about 14–16, and with the 5-bit keys of the serving configuration by about 69–72 (Table 7).

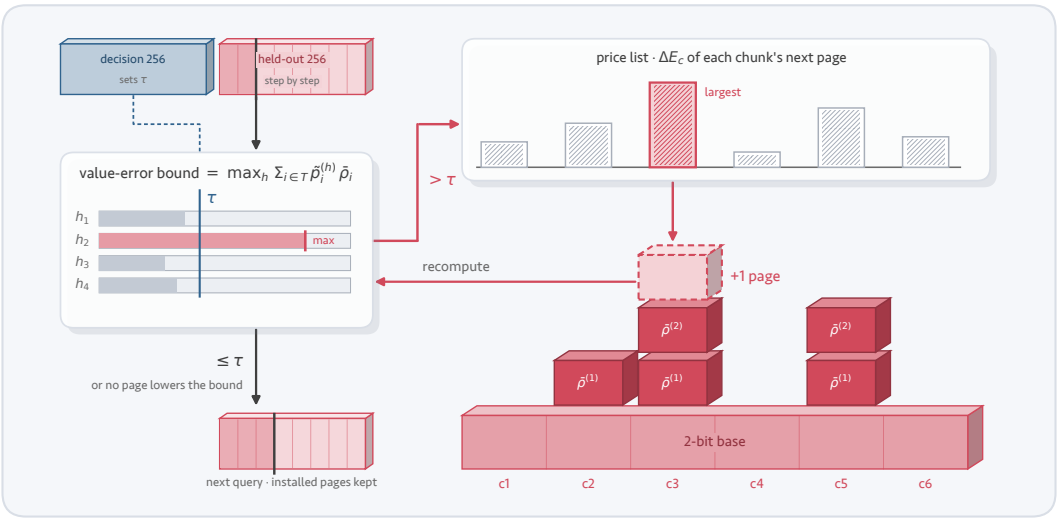


Figure 3. The target-driven controller holds the value-error bound below a target τ, which the decision queries set to the p99 bound under uniform one-page allocation. For each held-out query it compares the bound with τ head by head, and while the largest per-head bound exceeds τ it installs the page with the largest exact decrease ΔE_c in the price list and recomputes the bound. Once the target is met or no page lowers the bound, the query advances and the installed pages remain.

Memory allocation efficiency. Figure 4 illustrates the two policies at an equal page budget. Table 1 compares them at a budget of one page per chunk, where adaptive allocation reduces held-out p99 attention-output error by 30–67% relative to uniform allocation. The gain is also consistent window by window. Adaptive allocation lowers both the output error and the value error, the realized value-reconstruction term in equation 1, at both page budgets in 1,463 of the 1,465 windows drawn from six models, three corpora, and two context lengths. Table 2 turns to the memory side of the same allocation. When both policies must reach the same calibrated bound on value error, adaptive allocation needs about two-thirds of the refinement memory that uniform allocation installs. Moreover, drawing pages from one pool shared across heads and layers lowers error well below what an equal budget per head achieves.

Variation across heads. Figure 5 shows that under uniform allocation, per-head p99 value errors within one model differ by more than an order of magnitude. A shared pool exploits this spread by

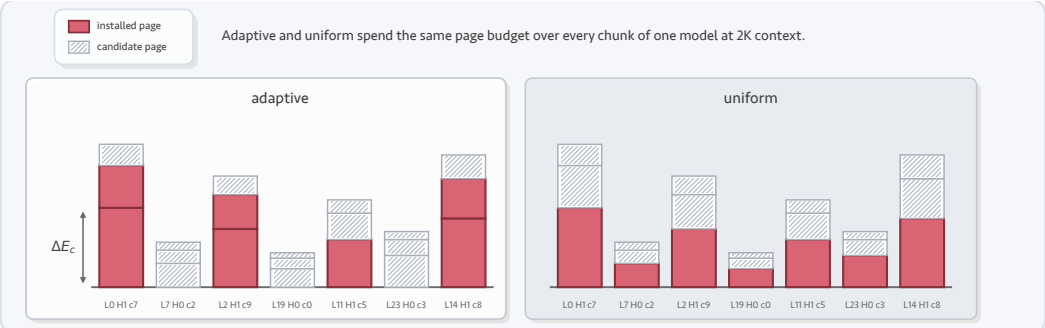


Figure 4. Adaptive allocation puts pages where they reduce the error bound most, while uniform allocation gives every chunk one page. Both policies install seven equal-size pages, so they use the same memory budget. Each stack represents a chunk, and each block's height shows how much adding that page lowers the bound.

Table 1. Adaptive allocation lowers attention error under the same memory budget.

Model	WikiText-103 ↑	PG19 ↑	C4 ↑	WikiText 8K ↑
Qwen2.5-0.5B	1.51× [1.47, 1.55]	1.43× [1.35, 1.51]	1.71× [1.61, 1.82]	—
Qwen2.5-7B	1.72× [1.66, 1.78]	1.71× [1.57, 1.85]	1.73× [1.62, 1.84]	1.90× [1.77, 2.07]
Qwen2.5-32B	1.91× [1.85, 1.98]	1.66× [1.57, 1.74]	1.61× [1.45, 1.79]	—
Qwen3-8B	2.99× [2.88, 3.09]	2.69× [2.55, 2.84]	2.94× [2.76, 3.13]	—
Llama-3.2-1B	1.63× [1.59, 1.68]	1.49× [1.42, 1.56]	1.88× [1.73, 2.03]	—
Llama-3.1-8B	1.94× [1.88, 2.00]	1.66× [1.60, 1.73]	1.81× [1.68, 1.95]	2.05× [1.84, 2.34]

Each entry divides the held-out p99 attention-output error of uniform allocation by that of adaptive allocation at one page per chunk, so 2× means that adaptive allocation halves the error. Arrows mark the better direction, and brackets give 95% confidence intervals throughout.

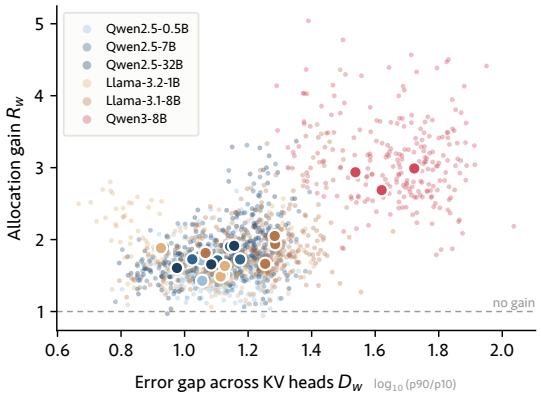
Table 2. Adaptive allocation uses less refinement memory, and sharing memory across heads lowers error.

	Refinement bytes at the same target Adaptive / uniform ↓			Error at the same total budget Separate head budgets / shared budget ↑		
Model	WikiText-103	PG19	C4	WikiText-103	PG19	C4
Qwen2.5-0.5B	0.669	0.672	0.679	1.57× [1.54, 1.60]	1.53× [1.46, 1.61]	1.79× [1.66, 1.94]
Qwen2.5-7B	0.662	0.665	0.674	1.67× [1.64, 1.70]	1.71× [1.61, 1.81]	1.66× [1.58, 1.74]
Qwen2.5-32B	0.660	0.662	0.665	1.75× [1.73, 1.77]	1.59× [1.54, 1.64]	1.46× [1.42, 1.51]
Qwen3-8B	0.658	0.657	0.670	2.68× [2.64, 2.72]	2.35× [2.26, 2.43]	2.34× [2.25, 2.43]
Llama-3.2-1B	0.654	0.641	0.639	1.60× [1.57, 1.62]	1.46× [1.42, 1.49]	1.47× [1.40, 1.56]
Llama-3.1-8B	0.651	0.646	0.641	1.75× [1.73, 1.77]	1.56× [1.53, 1.60]	1.57× [1.49, 1.64]

The left block divides the refinement bytes that adaptive allocation installs to reach the p99 value-error target, which uniform one-page allocation sets on the decision queries, by the bytes uniform allocation installs. The right block divides the held-out p99 attention-output error with an equal page budget per head by the error with one pool shared across heads and layers.

moving pages to the heads where they lower error most, and models with a wider spread gain more from adaptive allocation.

The contribution of the page price. Nonuniformity alone could account for part of the adaptive gain regardless of how the pages are ranked, so we replace the certificate price with two fixed heuristics that spend the same page budget on a fixed subset of 614 windows. The residual rule ranks pages by how much they reduce a chunk's stored reconstruction error, and the attention rule ranks chunks by their mean attention mass with a discount for each page already installed, a heuristic inspired by the importance signal of mixed-precision schemes (Yang et al., 2024, He et al., 2024). The certificate price combines both signals, since ΔE_c weights each token's residual reduction by the attention it receives. Table 3 shows that the price yields lower geometric-mean error than either heuristic in every setting and budget. Neither signal suffices on its own, since the attention rule trails



Model	Spread	Gain ↑
Llama-3.2-1B	11×	1.49–1.88×
Qwen2.5-32B	12×	1.61–1.91×
Qwen2.5-0.5B	12×	1.43–1.71×
Qwen2.5-7B	13×	1.71–1.90×
Llama-3.1-8B	17×	1.66–2.05×
Qwen3-8B	42×	2.69–2.99×

Figure 5. Models with a wider spread of head errors gain more from adaptive allocation. Each small point is a window, placed by the spread $\log_{10}(p90/p10)$ of its per-head p99 value errors under uniform allocation and by its uniform-to-adaptive p99 output-error ratio at equal bytes. Outlined points aggregate each setting, and the table lists each model's geometric-mean $p90/p10$ and range of pooled gains.

most on Qwen3-8B and the residual rule on Llama-3.2-1B. The margins narrow at two pages per chunk.

Table 3. The certificate price lowers error relative to fixed allocation heuristics.

	1 page/chunk		2 pages/chunk	
Model	Residual only ↑	Attention only ↑	Residual only ↑	Attention only ↑
Qwen2.5-0.5B	1.30× (1.15–1.61)	1.09× (1.08–1.11)	1.18× (1.10–1.35)	1.06× (1.01–1.10)
Qwen2.5-7B	1.32× (1.27–1.44)	1.41× (1.36–1.46)	1.06× (1.04–1.07)	1.17× (1.13–1.21)
Qwen2.5-32B	1.29× (1.22–1.32)	1.40× (1.34–1.45)	1.21× (1.17–1.26)	1.26× (1.20–1.30)
Qwen3-8B	2.11× (1.87–2.54)	3.19× (3.13–3.22)	1.30× (1.20–1.45)	2.12× (2.00–2.30)
Llama-3.2-1B	2.36× (1.83–3.12)	1.22× (1.19–1.27)	1.71× (1.51–1.95)	1.12× (1.08–1.15)
Llama-3.1-8B	1.86× (1.48–2.21)	1.28× (1.24–1.33)	1.78× (1.62–2.07)	1.17× (1.12–1.21)

Each ratio divides a heuristic's held-out p99 attention-output error by the certificate's at equal page bytes, so values above one favor the certificate. Entries are geometric means over a model's corpora and contexts, parentheses give the range of the setting-level ratios, and Table 11 lists them with 95% intervals.

Task quality and output fidelity at matched budgets. We measure end-to-end task scores to test whether the compression degrades downstream quality. For each request we compare adaptive and uniform allocation with a KIVI-style 4-bit baseline at matched KV bytes across the five settings of Table 4. On the primary RULER-32K study, adaptive allocation is noninferior to the baseline within 0.5 points, while the four other settings show no difference that their intervals can resolve. Figure 6 instead compares each arm's next-token distributions with the dense model's under a common token history. At the full page budget, adaptive allocation lowers the mean KL to dense by factors of 3.4 and 2.6 relative to uniform allocation on Qwen2.5-7B and Llama-3.1-8B, with paired intervals that exclude zero (Table 16). At half the page budget it still lies below the KIVI-style comparator in mean KL with about 9% fewer KV bytes. A separate 650-request task study on Qwen2.5-7B at this budget establishes neither superiority over uniform allocation nor noninferiority within 0.5 points, so the task benefit remains unresolved (Appendix D).

Serving speed and capacity. Table 17 in Appendix D.7 reports memory and latency relative to dense 16-bit serving. The packed 2-bit base stores the KV cache at about a quarter of its dense size, and the selected pages add a few percent. Relative latency improves with context length, because packing saves cache traffic in proportion to the context while its per-step overhead stays fixed. The 8B model moves from parity at 32K to a speedup at 64K and 128K. The 7B model at 16K and 32K remains slower than dense, as does the 70B model at 64K, whose cache is a smaller share of its memory traffic. At 128K, packing reduces the 70B cache from 42.9 GB to 11.1 GB, so one 180-GB B200 serves the model where the dense stack runs out of memory.

Table 4. Task-score differences at matched KV-cache budgets are small, and every interval covers zero.

Model	Benchmark	Requests	Adaptive − uniform ↑	Adaptive − KIVI-style ↑
Qwen2.5-7B†	RULER-32K	2,652	+0.41 [−0.14, +0.96]	+0.17 [−0.30, +0.65]
Llama-3.1-8B	RULER-32K	650	−0.35 [−1.29, +0.57]	−0.41 [−1.23, +0.34]
Qwen2.5-7B	LongBench	799	+0.11 [−0.25, +0.50]	+0.04 [−0.48, +0.59]
Llama-3.1-8B	LongBench	799	+0.03 [−0.46, +0.49]	−0.03 [−0.66, +0.61]
Qwen3-8B	LongBench	799	−0.09 [−0.58, +0.41]	−0.06 [−0.48, +0.36]

Each entry is a paired difference in score points with an interval from resampling requests. Adaptive and uniform allocation read packed caches at identical KV bytes per request within the budget of the KIVI-style baseline, which simulates quantization in dense tensors. Only the primary study (†), at $0.316\times$ the dense 16-bit KV size, establishes noninferiority to that baseline within 0.5 points.

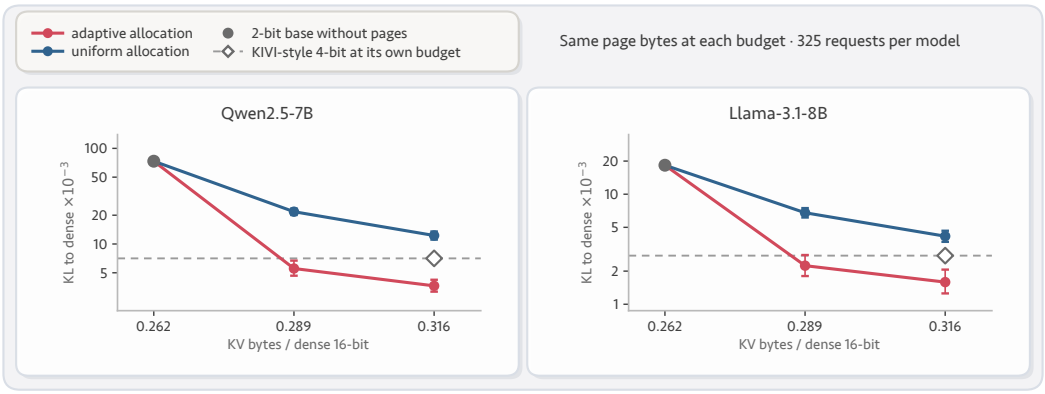


Figure 6. Certificate-guided allocation brings the model output closer to dense decoding than uniform allocation at identical bytes. Each point averages the KL divergence from dense over the continuation and over RULER-32K requests, with a bootstrap interval, and Table 16 lists these values with the top-1 disagreement rates.

Scope. Our certificate bounds the attention output of one head at one decode step in exact arithmetic. Beyond a single head, the fidelity study measures how compression changes the model's output distribution, since a bound on logits or sequences would require composing head bounds across layers and decoding steps (Appendix A.9). The serving measurements time decoding with the selected pages already resident and compare it only with dense 16-bit serving, so the cost of storing and transferring pages in a live controller remains to be measured.

5 Related Work

Runtime certificates and refinement. Calver (2026) computes deterministic key and value bounds for each head and step, uses block contributions to promote selected values to FP16, and retains FP16 originals for dense fallback. WitCert combines stored residual norms with the current query to certify quantized attention. Its deterministic bound needs no exact copy, while its gate restores exact blocks from a backing store (Wei et al., 2026). Our certificate instead prices refinements between compressed states only, with no exact copy retained. Each value page has an exact effect on the bound for fixed queries and keys. Appendix D compares the two certificates on identical states.

Token eviction and budget allocation. StreamingLLM preserves sink and recent tokens, while H2O favors entries with high accumulated attention scores (Xiao et al., 2024b, Zhang et al., 2023). Later methods refine the retention signal with prefill attention windows (Li et al., 2024), per-head policies (Ge et al., 2024, Xiao et al., 2025), key norms (Devoto et al., 2024), or layer-wise budgets (Cai et al., 2025), merge entries in place of discarding them (Zhang et al., 2024, Wan et al., 2025), or load only the blocks relevant to each query, from GPU or host memory (Tang et al., 2024, Xiao et al., 2024a, Chen et al., 2025, Sun et al., 2025, Zhang et al., 2025). Ada-KV uses prefill attention statistics in a safeguarded allocation rule derived from an analytic L_1 bound (Feng et al., 2025),

and CriticalKV selects entries by an output-perturbation criterion that accounts for value states and the output projection (Feng et al., 2026). Error after eviction can be certified statistically (Xie & Alanwar, 2026) or reduced with MomentKV's first-order correction from retained moments (Li et al., 2026). Our controller keeps all entries represented and refines their values.

Compressed cache representations. KIVI and KVQuant motivate different quantization granularities for keys and values from their different numerical distributions (Liu et al., 2024c, Hooper et al., 2024). Subsequent work protects pivot or recent tokens (Liu et al., 2024a, Duanmu et al., 2024), assigns precision by token importance (Yang et al., 2024, He et al., 2024), adds low-rank residuals (Kang et al., 2024), applies vector quantization (Li et al., 2025b), or allocates bits from the spectra of keys and values (Hariri et al., 2026). Our analysis uses the different roles of keys and values in attention to hold keys fixed and refine values.

Progressive value precision. A coarse reconstruction can be refined by retaining more information, as studied in residual quantization and successive refinement (Juang & Gray, 1982, Equitz & Cover, 1991). Related representations have been used for media, weights, and KV caches (Zeghidour et al., 2022, Lu et al., 2021, Kusupati et al., 2022, Park et al., 2024, Wang et al., 2025, Kumar, 2024, Liu et al., 2024b, Xi et al., 2026, Luo et al., 2026). Our value pages differ by retaining an error bound at each precision, so the certificate prices each correction exactly for the current query.

Streaming-attention theory. Lower bounds and randomized discrepancy or sketching algorithms characterize worst-case error–space tradeoffs as context grows (Haris & Onak, 2025, Kochetkova et al., 2025, Chen et al., 2026), while coreset results establish the existence of nearly optimal summaries (Liberty et al., 2026). Haverbeck et al. (2026) analyze minimax expected risk under a query distribution for weighted summaries of at most K entries. Our certificate instead bounds one attention computation for the current query and the served cache.

6 Discussion

KV-cache compression raises a question beyond how faithfully a smaller representation preserves the original keys and values. Once the original entries are discarded, a serving system can no longer recover the exact output they would produce for the current query, yet it still needs to decide whether the compressed state is adequate and where additional memory would matter most. Our answer carries the uncertainty forward in error and norm summaries and lets each query reveal which parts of the cache matter, while the key–value asymmetry decides what can be priced. KV compression keeps compact approximations of keys and values, whereas the certificate keeps a compact account of what those approximations lost. In this sense, certification is itself a form of compression.

The audits also clarify which property of an allocation certificate matters. Although its absolute value exceeds realized error by an order of magnitude, the controller that consults it lowers realized error in nearly every window, since allocation depends only on the ordering of marginal gains. Each candidate page changes only its own chunk's residual term, so the conservative key term cancels from candidate comparisons and each page's price gives its exact reduction in the bound. A certificate can therefore be loose where it describes and exact where it prices, while the connection between these prices and realized error reductions remains an empirical question that the page-price ablation answers. Decisions that alter the keys change the key term itself, so pricing them requires a key-side summary with similarly local updates, a direction we leave for future work.

Beyond attention error, adaptive allocation also keeps the output closer to the uncompressed model than uniform allocation at equal memory, yet both reach statistically indistinguishable task scores. The two results are consistent, since a task score changes only when the answer changes, and small shifts in the output distribution usually leave it intact. A closer match also inherits the uncompressed model's mistakes, so it can gain at most the margin by which that model outscores the compressed one. Our experiments leave open which of these factors explains the unchanged scores, and whether larger studies would detect a difference. Tighter budgets or harder tasks, where compression measurably lowers the score, could show when a closer match improves task performance.

AI USE STATEMENT

Large language models assisted under author direction with proof review, reproduction-code refactoring, replication experiments, literature review, prose editing, figure design, feedback on experimental design, and the interpretation of results. Their proof feedback suggested the attention-weighted control used in the softmax perturbation lemma (Lemma 1) and Theorem 1, which we re-derived and checked through float64 audits. Contract tests and archived-result replays validated the code changes. Each replication pipeline was also checked bitwise against reference artifacts before new settings were run. We verified every cited source and reviewed all AI-assisted work against the established claims and results. We take responsibility for the final text, claims, and artifacts. No generative AI was used to create synthetic data.

REPRODUCIBILITY STATEMENT

Appendix A states the assumptions and gives complete proofs of Theorem 1 and Proposition 1, together with the finite-precision, sharpness, and allocation results that the implementation relies on. Appendix B derives the storage and arithmetic costs of the representation in closed form. Appendix C specifies the models, corpora, recorded-stream protocol, allocation rules, resampling procedure, task-quality studies, and serving measurements, and lists every audit family behind the reported checks, while Appendix D gives the protocol of the page-price ablation together with the per-setting, per-task, and per-configuration results behind every main-text table. The supplementary code package contains the implementation of the certificate, the packed representation, the allocation controller, and the Triton kernel, together with the recorded evidence behind every table and replay scripts that recompute the reported statistics from that evidence on a CPU in under a minute. Its README maps each reported number to the file and command that reproduces it, lists the pinned model revisions and dataset identifiers, and records the audit violation count of every run.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4895–4901, 2023.

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

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Conference on Language Modeling (COLM)*, 2025.

Dean Calver. Runtime-certified bounded-error quantized attention, 2026. arXiv:2605.20868.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: KV-cache compression with low-rank projection. In *International Conference on Learning Representations (ICLR)*, 2025.

Justin Y. Chen, Ying Feng, Piotr Indyk, Michael Kapralov, Ekaterina Kochetkova, and Boris Prokhorov. Towards tight bounds for streaming attention, 2026. arXiv:2606.07205.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In *International Conference on Learning Representations (ICLR)*, 2025.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model, 2024. arXiv:2405.04434.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective L_2 norm-based strategy for KV cache compression. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Haojie Duanmu, Zhihang Yuan, Xiuhong Li, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. SKVQ: Sliding-window key and value cache quantization for large language models. In *Conference on Language Modeling (COLM)*, 2024.

William H. R. Equitz and Thomas M. Cover. Successive refinement of information. *IEEE Transactions on Information Theory*, 37(2):269–275, 1991.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yuan Feng, Junlin Lv, Haoyu Guo, Yukun Cao, S. Kevin Zhou, and Xike Xie. CriticalKV: Optimizing KV cache eviction from an output perturbation perspective. In *International Conference on Machine Learning (ICML)*, 2026.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *International Conference on Learning Representations (ICLR)*, 2024.

Mohsen Hariri, Alan Luo, Weicong Chen, Tianyi Zhang, Qifan Wang, Xiaotian Han, and Vipin Chaudhary. Quantize what counts: More for keys, less for values. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26393–26420, 2026.

Themistoklis Haris and Krzysztof Onak. Compression barriers in autoregressive transformers. In *Conference on Learning Theory (COLT)*, pp. 2757–2785, 2025.

Lukas Haverbeck, Carmen Amo Alonso, Andres Felipe Posada-Moreno, Sebastian Trimpe, and Marco Pavone. The risk of KV cache compression, 2026. arXiv:2607.01520.

Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. ZipCache: Accurate and efficient KV cache quantization with salient token identification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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

Biing-Hwang Juang and Augustine H. Gray, Jr. Multiple stage vector quantization for speech coding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 7, pp. 597–600, 1982.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An efficient error reduction framework for KV cache compression in LLM inference. In *Proceedings of the 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pp. 305–321. PMLR, 2024.

Ekaterina Kochetkova, Kshiteej Jitesh Sheth, Insu Han, Amir Zandieh, and Michael Kapralov. Streaming attention approximation via discrepancy theory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Ankur Kumar. Residual vector quantization for KV cache compression in large language model. In *Proceedings of the 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pp. 485–490. PMLR, 2024.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. A survey on large language model acceleration based on KV cache management. *Transactions on Machine Learning Research (TMLR)*, 2025a.

Junyan Li, Yang Zhang, Muhammad Yusuf Hassan, Talha Chafekar, Tianle Cai, Zhile Ren, Pengsheng Guo, Foroozan Karimzadeh, Colorado Reed, Chong Wang, and Chuang Gan. CommVQ: Commutative vector quantization for KV cache compression. In *International Conference on Machine Learning (ICML)*, 2025b.

Yu Li, Binxu Li, and Tian Lan. MomentKV: Closing the directional gap in KV cache eviction for long-context inference, 2026. arXiv:2606.01563.

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

Edo Liberty, Alexandr Andoni, and Eldar Kleiner. Nearly optimal attention coresets, 2026. arXiv:2605.05602.

Ruikang Liu, Haoli Bai, Haokun Lin, Yuening Li, Han Gao, Zhengzhuo Xu, Lu Hou, Jun Yao, and Chun Yuan. IntactKV: Improving large language model quantization by keeping pivot tokens intact. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7716–7741, 2024a.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. CacheGen: KV cache compression and streaming for fast large language model serving. In *ACM SIGCOMM Conference*, pp. 38–56, 2024b.

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. In *International Conference on Machine Learning (ICML)*, 2024c.

Yadong Lu, Yinhao Zhu, Yang Yang, Amir Said, and Taco S. Cohen. Progressive neural image compression with nested quantization and latent ordering. In *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 539–543, 2021. doi: 10.1109/ICIP42928.2021.9506026.

Yu Luo, Bo Dong, Wenhua Cheng, and Haihao Shen. Recurrent residual quantization: A progressive multi-precision representation for LLMs, 2026. arXiv:2608.04048.

Yeonhong Park, Jake Hyun, SangLyul Cho, Bonggeun Sim, and Jae W. Lee. Any-precision LLM: Low-cost deployment of multiple, different-sized LLMs. In *International Conference on Machine Learning (ICML)*, pp. 39682–39701, 2024.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for KV cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15332–15344, 2024.

Luohe Shi, Hongyi Zhang, Yao Yao, Zuchao Li, and Hai Zhao. Keep the cost down: A review on methods to optimize LLM's KV-cache consumption. In *Conference on Language Modeling (COLM)*, 2024.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *International Conference on Machine Learning (ICML)*, pp. 57355–57373, 2025.

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

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*, pp. 10–19, 2019.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, Longyue Wang, and Mi Zhang. D2O: Dynamic discriminative operations for efficient long-context inference of large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Xinghao Wang, Pengyu Wang, Bo Wang, Dong Zhang, Yunhua Zhou, and Xipeng Qiu. BitStack: Any-size compression of large language models in variable memory environments. In *International Conference on Learning Representations (ICLR)*, 2025.

Fanzhe Wei, Li Liu, Ziyang Wang, and Chenyu Wang. WitCert: Sound runtime risk observability and gating for KV-cache quantization, 2026. arXiv:2607.28699.

Haocheng Xi, Shuo Yang, Yilong Zhao, Muyang Li, Han Cai, Xingyang Li, Yujun Lin, Zhuoyang Zhang, Jintao Zhang, Xiuyu Li, Zhiying Xu, Jun Wu, Chenfeng Xu, Ion Stoica, Song Han, and Kurt Keutzer. Quant VideoGen: Auto-regressive long video generation via 2-bit KV-cache quantization. In *International Conference on Machine Learning (ICML)*, 2026.

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

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

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. In *International Conference on Learning Representations (ICLR)*, 2025.

Peng Xie and Amr Alanwar. Error certificates for KV-cache eviction via randomized design, 2026. arXiv:2607.21475.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable KV cache compression via importance-aware mixed precision quantization, 2024. arXiv:2402.18096.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. SoundStream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2022.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. PQCache: Product quantization-based KVCache for long context LLM inference. *Proceedings of the ACM on Management of Data*, 3(3):201:1–201:30, 2025. doi: 10.1145/3725338.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. CaM: Cache merging for memory-efficient LLMs inference. In *International Conference on Machine Learning (ICML)*, 2024.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A PROOFS

This appendix collects the full expression of the key term, the proofs behind Sections 2 and 3, and four results that the implementation relies on. Appendices A.5 and A.6 show that the certificate stays valid when its receipts are stored in fp16 and how close the softmax lemma comes to the best possible bound. Appendices A.7 and A.8 then explain why a value page has a local price and a key refinement has none, and why the nested greedy over all candidate pages has a closed form. Appendix A.9 records the layer-composition bound that delimits the scope of the guarantee.

Notation. The dense set D holds the attention sinks and the recent window, which are stored exactly, and the compressed span T holds the remaining tokens. Each compressed key belongs to a group $g(i)$ whose stored vector $\Delta_g \in \mathbb{R}^d$ bounds, coordinate by coordinate, the key reconstruction error of every token in the group. Each compressed value stores the residual bound $\bar{\rho}_i \geq \|v_i - \hat{v}_i\|_2$ and the norm bound $\bar{\nu}_i \geq \|v_i\|_2$, and exact dense values satisfy $\bar{\nu}_i = \|v_i\|_2$. The served scores are $\tilde{s}_i = \langle q, \hat{k}_i \rangle / \sqrt{d}$, so that $\tilde{p} = \mathrm{softmax}(\tilde{s})$. We write $L_c(q) = \sum_{i \in c} \tilde{p}_i \bar{\rho}_i$ for the value term of chunk c, so the certificate reads $E_{\mathrm{rec}}(S, q) = E_{\mathrm{key}}(S, q) + \sum_c L_c(q)$, which the main text abbreviates to $E(q)$. The retained state S contains the stored dense entries, compressed reconstructions, layout, and error and norm receipts. Each receipt bounds the real-valued reconstruction and is rounded upward. All bounds below hold for this reconstruction in exact arithmetic, and the precision casts inside the serving kernel fall outside their scope. The original caches consistent with S form the set

$$\mathcal{U}_{\mathrm{receipt}}(S) = \Big\{ (K, V) \;\Big|\; (K_D, V_D) \text{ equal the stored dense entries},\; |k_{ij} - \hat{k}_{ij}| \leq \Delta_{g(i),j} \quad \text{for } i \in T,\ 1 \leq j \leq d,\; \|v_i - \hat{v}_i\|_2 \leq \bar{\rho}_i, \quad \|v_i\|_2 \leq \bar{\nu}_i \quad \text{for } i \in T \Big\}.$$

The cache that produced S lies in this set because the receipts bound its reconstruction errors and value norms.

A.1 SCORE RADII AND THE KEY TERM

The current query determines how the stored key bounds can perturb the scores. The channel bounds of token i's group give the score radius

$$|s_i - \tilde{s}_i| \leq \varepsilon_i = \frac{\langle |q|, \Delta_{g(i)} \rangle}{\sqrt{d}} \qquad (i \in T), \tag{A.1}$$

and $\varepsilon_i = 0$ on D. Two attention-weighted sums of these radii,

$$A = \sum_i \tilde{p}_i \big(e^{\varepsilon_i} - 1\big), \qquad B = \sum_i \tilde{p}_i \big(1 - e^{-\varepsilon_i}\big), \tag{A.2}$$

control the change in each softmax weight and in their shared normalization (Lemma 1). Finite receipts ensure $B < 1$, and the key term of Theorem 1, displayed as equation 3 in the main text, is

$$E_{\mathrm{key}}(S, q) = \min \left\{ \frac{\sum_i \tilde{p}_i \big(A + e^{\varepsilon_i} - 1\big) \bar{\nu}_i}{1 - B},\; 2V_{\max} \right\}, \qquad V_{\max} = \max_i \bar{\nu}_i. \tag{A.3}$$

The first branch weights each token's possible attention change by its stored value norm, while the second caps the error using the largest norm. Small key errors in the current query direction reduce A and B, as does attention concentrated on exact sinks or the recent window. Changing value precision leaves $\tilde{p}$, every ε_i, and every $\bar{\nu}_i$ unchanged, so E_{key} is fixed during value refinement.

A.2 SOFTMAX PERTURBATION

Lemma 1. *Let $p = \mathrm{softmax}(s)$ and $\tilde{p} = \mathrm{softmax}(\tilde{s})$, with $|s_i - \tilde{s}_i| \leq \varepsilon_i$, $\varepsilon_i \geq 0$, and $\varepsilon_i = 0$ on D, and let A and B be the sums in equation A.2. If $B < 1$, then for every i*

$$|p_i - \tilde{p}_i| \leq \tilde{p}_i \frac{A + e^{\varepsilon_i} - 1}{1 - B}, \qquad \|p - \tilde{p}\|_1 \leq \min \left(\frac{2A}{1 - B}, 2 \right).$$

Proof. Write $\delta_i = s_i - \tilde{s}_i$ and $R = \sum_i \tilde{p}_i e^{\delta_i}$, so $p_i = \tilde{p}_i e^{\delta_i}/R$. The change in normalization is controlled by

$$R \geq \sum_i \tilde{p}_i e^{-\varepsilon_i} = 1 - B, \qquad |R - 1| \leq \sum_i \tilde{p}_i |e^{\delta_i} - 1| \leq \sum_i \tilde{p}_i (e^{\varepsilon_i} - 1) = A.$$

Separating the change in each exponential from the change in normalization now gives

$$|p_i - \tilde{p}_i| = \tilde{p}_i \frac{|e^{\delta_i} - R|}{R} \leq \tilde{p}_i \frac{|e^{\delta_i} - 1| + |R - 1|}{R} \leq \tilde{p}_i \frac{e^{\varepsilon_i} - 1 + A}{1 - B}.$$

For dense entries, the exponential is unchanged and only normalization contributes. Summing the token bounds and using $\sum_i \tilde{p}_i = 1$ gives

$$\|p - \tilde{p}\|_1 \leq \frac{\sum_i \tilde{p}_i (e^{\varepsilon_i} - 1) + A \sum_i \tilde{p}_i}{1 - B} = \frac{2A}{1 - B}.$$

Finally, both weight vectors are probability distributions, so $\|p - \tilde{p}\|_1 \leq \sum_i (p_i + \tilde{p}_i) = 2$ regardless of B. ■

A.3 Proof of Theorem 1

Proof. Fix a retained state S, a query q, and any original cache $(K, V) \in \mathcal{U}_{\text{receipt}}(S)$. Let o be its attention output and $\tilde{o}$ the real-valued interface output defined by S.

Score errors. For every $i \in T$, the stored channel bounds give

$$|s_i - \tilde{s}_i| = \frac{|\langle q, k_i - \hat{k}_i \rangle|}{\sqrt{d}} \leq \frac{\sum_j |q_j| \, |k_{ij} - \hat{k}_{ij}|}{\sqrt{d}} \leq \frac{\langle |q|, \Delta_{g(i)} \rangle}{\sqrt{d}} = \varepsilon_i.$$

Dense keys are unchanged, so the same bound holds with $\varepsilon_i = 0$ on D. Finite score-error bounds ensure $B < 1$, and Lemma 1 therefore applies.

Output errors. Separating the effects of changed weights and changed values gives

$$o - \tilde{o} = \sum_i (p_i - \tilde{p}_i) v_i + \sum_{i \in T} \tilde{p}_i (v_i - \hat{v}_i).$$

The value contribution follows directly from the stored residual bounds and the triangle inequality,

$$\Big\| \sum_{i \in T} \tilde{p}_i (v_i - \hat{v}_i) \Big\|_2 \leq \sum_{i \in T} \tilde{p}_i \bar{\rho}_i = \sum_c L_c(q).$$

For the contribution from changed weights, the token bounds in Lemma 1 give

$$\Big\| \sum_i (p_i - \tilde{p}_i) v_i \Big\|_2 \leq \sum_i |p_i - \tilde{p}_i| \bar{\nu}_i \leq \frac{\sum_i \tilde{p}_i (A + e^{\varepsilon_i} - 1) \bar{\nu}_i}{1 - B}.$$

The same contribution is also bounded by

$$\Big\| \sum_i (p_i - \tilde{p}_i) v_i \Big\|_2 \leq \|p - \tilde{p}\|_1 V_{\max} \leq 2 V_{\max}.$$

Taking the smaller of these two estimates gives E_{key} in equation A.3, and adding the value contribution proves

$$\|o - \tilde{o}\|_2 \leq E_{\text{key}}(S, q) + \sum_c L_c(q) = E_{\text{rec}}(S, q).$$

Every step uses only the constraints defining $\mathcal{U}_{\text{receipt}}(S)$, so the bound holds for every compatible original cache, including the one that produced S. ■

Relation to the total-variation bound. The attention-weighted expression in equation A.3 satisfies

$$\frac{\sum_i \tilde{p}_i (A + e^{\varepsilon_i} - 1) \bar{\nu}_i}{1 - B} \leq \frac{2 A V_{\max}}{1 - B},$$

because $\bar{\nu}_i \leq V_{\max}$, $A \geq 0$, and $\sum_i \tilde{p}_i = 1$. Thus combining it with the full ℓ_1 bound $\min(2A/(1 - B), 2) V_{\max}$ gives exactly the same E_{key}. The $2A/(1 - B)$ branch cannot improve the attention-weighted bound, so only the cap $2V_{\max}$ is needed.

A.4 Proof of Proposition 1

Proof. Fix the query, the keys, and the value-norm receipts $\bar{\nu}_i$, which are retained across precision levels. Changing value precision leaves $\tilde{p}$, every ε_i, and hence A and B unchanged. The fixed norm bounds also leave $V_{\max}$ unchanged, so E_{key} is identical before and after the change. Only the residual bounds can change in the certificate. Subtracting its two values therefore gives

$$E_{\text{rec}}^{\text{before}} - E_{\text{rec}}^{\text{after}} = \sum_{c} \sum_{i \in c} \tilde{p}_i \big(\bar{\rho}_i^{\text{before}} - \bar{\rho}_i^{\text{after}}\big).$$

For a single-page change, all unchanged chunks cancel and the remaining term is the stated ΔE_c. Changes in several chunks add, and successive transitions telescope because intermediate residual bounds cancel. The difference is signed, so an added page need not lower the bound. The identity concerns the certificate and does not assert the same change in realized error. ■

A.5 Monotonicity and finite-precision receipts

The certificate is evaluated from stored receipts, which are rounded versions of the exact reconstruction errors they bound, and the packed representation stores its metadata and receipts in fp16. Two further statements are therefore needed for Theorem 1 to hold for the state that is actually served. The first says that enlarging any receipt can only enlarge the certificate, and the second says that directed rounding of the quantization grid keeps every stored bound valid.

Lemma 2. *Fix the served weights $\tilde{p}$. The certificate $E_{\text{rec}}(S, q)$ is nondecreasing in every score radius ε_i, every residual bound $\bar{\rho}_i$, and every norm bound $\bar{\nu}_i$.*

Proof. The sums A and B are nondecreasing in every ε_i because $e^{\varepsilon}-1$ and $1-e^{-\varepsilon}$ are increasing and the weights are nonnegative. The numerator $\sum_i \tilde{p}_i(A+e^{\varepsilon_i}-1)\bar{\nu}_i$ of the first branch of equation A.3 is therefore nondecreasing in every ε_i and in every $\bar{\nu}_i$, while its denominator $1-B$ is positive and nonincreasing in every ε_i as long as $B < 1$. The cap $2V_{\max}$ is nondecreasing in every $\bar{\nu}_i$. The minimum of two nondecreasing functions is nondecreasing. As B increases to one the first branch grows without bound, so E_{key} passes continuously to the cap value that the certificate assigns when $B \geq 1$. The value term $\sum_i \tilde{p}_i \bar{\rho}_i$ is linear with nonnegative coefficients. ■

Lemma 3. *Let $x_1, \ldots, x_m$ be the entries of one quantization group with minimum ℓ and maximum h, and let a b-bit code take values in $\{0, \ldots, Q\}$ with $Q = 2^b - 1$. Suppose the stored offset and step satisfy $\ell' \leq \ell$ and $\sigma' \geq (h - \ell')/Q$. Then for every entry the nearest code $z = \text{round}\big((x - \ell')/\sigma'\big)$ lies in $\{0, \ldots, Q\}$, and the reconstruction $\hat{x} = \ell' + z\,\sigma'$ satisfies $|x - \hat{x}| \leq \sigma'/2$.*

Proof. The scaled entry $(x - \ell')/\sigma'$ lies in $[0, (h - \ell')/\sigma'] \subseteq [0, Q]$, so its nearest integer lies in $\{0, \ldots, Q\}$ and differs from it by at most one half. Multiplying by σ' gives the bound. ■

Corollary 1. *The packed representation rounds each group offset down and each step up to fp16, decides codes on the rounded grid, stores $\Delta_{g,j} = \sigma'_{g,j}/2$ for every key channel, and rounds $\|v_i - \hat{v}_i\|_2$ and $\|v_i\|_2$ up to fp16 at every precision level. By Lemma 3 every stored channel bound holds for every key of its group, and the value receipts dominate the quantities they bound. By Lemma 2 the certificate evaluated from the stored receipts is at least the certificate evaluated from exact receipts, so Theorem 1 holds for the served state.*

The recorded-window replays use float64 metadata, where the grid is exact and $\sigma' = (h - \ell)/Q$. The rounding of the certificate's own arithmetic is outside this argument, which is why every audit evaluates the certificate in float64 from the stored receipts and records the kernel's fp32 evaluation only as a diagnostic.

Lemma 3 also bounds what each correction page achieves. A page quantizes the residual r left by the previous level with its own offset and step per channel group. With exact metadata the step is at most $2 \max_j |r_j|/(2^b - 1)$, so every 2-bit page divides the largest residual entry of every channel group by at least three. The ℓ_2 residual norm of a token combines entries that can differ widely in magnitude, however, so a page can leave $\bar{\rho}_i$ unchanged or larger and receive a nonpositive price in Proposition 1. The controller therefore installs only pages with a positive price.

A.6 First-order sharpness of the softmax lemma

Lemma 1 bounds the change of every attention weight from the score radii alone. The following proposition shows that its per-token bound is attained up to a factor close to one for tokens that carry a small share of the attention, which describes almost every token of a long compressed span.

Proposition 2. *Fix $\tilde{p}$ and radii $\varepsilon = t\hat{\varepsilon}$ with $\hat{\varepsilon} \geq 0$ fixed and $t \to 0$, and write $\bar{\varepsilon} = \sum_j \tilde{p}_j \varepsilon_j$.*

1. *The per-token bound of Lemma 1 satisfies $\tilde{p}_i(A + e^{\varepsilon_i} - 1)/(1 - B) = \tilde{p}_i(\varepsilon_i + \bar{\varepsilon}) + O(t^2)$.*
2. *For every perturbation with $|\delta_j| \leq \varepsilon_j$, $p_i - \tilde{p}_i = \tilde{p}_i\big(\delta_i - \sum_j \tilde{p}_j \delta_j\big) + O(t^2)$. The first-order term is largest for $\delta_i = \varepsilon_i$ and $\delta_j = -\varepsilon_j$ for $j \neq i$, where it equals $\tilde{p}_i(\varepsilon_i + \bar{\varepsilon} - 2\tilde{p}_i\varepsilon_i)$.*
3. *The ratio of the bound to the largest realizable change of p_i therefore tends to a limit of at most $(1 + \tilde{p}_i)/(1 - \tilde{p}_i)$, which tends to one as $\tilde{p}_i \to 0$.*

Proof. With $R = \sum_j \tilde{p}_j e^{\delta_j} = 1 + \sum_j \tilde{p}_j \delta_j + O(t^2)$ and $p_i = \tilde{p}_i e^{\delta_i}/R$, expanding both factors gives the first-order form in the second claim. That term is linear in δ with coefficient $\tilde{p}_i(1 - \tilde{p}_i) \geq 0$ on δ_i and $-\tilde{p}_i\tilde{p}_j \leq 0$ on every other δ_j, so its maximum over the box $|\delta_j| \leq \varepsilon_j$ is attained at the stated corner and equals $\tilde{p}_i\big((1-\tilde{p}_i)\varepsilon_i + \sum_{j\neq i} \tilde{p}_j\varepsilon_j\big) = \tilde{p}_i(\varepsilon_i + \bar{\varepsilon} - 2\tilde{p}_i\varepsilon_i)$. For the first claim, $A = \bar{\varepsilon} + O(t^2)$, $B = O(t)$, and $e^{\varepsilon_i} - 1 = \varepsilon_i + O(t^2)$. The ratio in the third claim is $1 + 2\tilde{p}_i\varepsilon_i/(\varepsilon_i + \bar{\varepsilon} - 2\tilde{p}_i\varepsilon_i)$, and $\bar{\varepsilon} \geq \tilde{p}_i\varepsilon_i$ bounds its denominator below by $(1 - \tilde{p}_i)\varepsilon_i$. ■

The conservatism of the full certificate reported in Appendix D.1 therefore has two other sources. Realized key errors are quantization residuals whose signs are far from the adversarial pattern of the proposition, and the vector sums in the proof of Theorem 1 cancel partially, whereas the triangle inequality replaces v_i and $v_i - \hat{v}_i$ by their norm bounds.

A.7 Coupling of the key term across groups

Proposition 1 gives every value page an exact and local price. The next proposition records how the key term couples the key groups, which formalizes the asymmetry that drives the design.

Proposition 3. *The decrease of the key term E_{key} obtained by refining one key group depends on the radii of the other groups, so E_{key} is not a sum of per-group terms. Moreover, any change of a stored key that moves its score changes the served weights $\tilde{p}$ and can therefore affect the chunk terms $L_c(q)$ and page prices ΔE_c across the head.*

Proof. One instance suffices for the first claim. Take two compressed tokens with weights 0.3 and 0.5 and norm bounds 1 and 2, one exact token with weight 0.2 and norm 1.5, and radius 0.2 for every compressed group that is not refined. Evaluating equation A.3 gives $E_{\mathrm{key}} = 0$ when both groups are exact, 0.183 when only the first token keeps its radius, 0.438 when only the second does, and 0.668 when both do. Refining the first group therefore lowers the key term by 0.183 when the second group is exact and by 0.230 when it is not. The coupling is structural, because A and B enter every token's coefficient and $1 - B$ divides the whole sum. The second claim follows because a change in $\tilde{s}_i = \langle q, \hat{k}_i\rangle/\sqrt{d}$ changes the softmax normalization and with it every weight, whereas a key change orthogonal to the query leaves the score and the weights unchanged. ■

Two consequences follow for a controller. The marginal value of a key refinement can be assessed only by recomputing the bound, and it can change when another group changes. A value page has a price that is exact for the current query and independent of every other chunk, which is what makes the greedy allocation of Section 3 well defined.

A.8 Prices over a query set and the nested greedy

The allocation experiments rank pages by their mean price over a set of decision queries and install them with a nested greedy over a pool shared by all layers and heads, where each layer–head pair forms one unit. Two statements make this procedure inexpensive. The first reduces the price over a query set to one weighted sum, and the second gives the greedy in closed form.

Lemma 4. *Fix the receipts. The price $\Delta E_c(q)$ of equation 4 is linear in the served weights $\tilde{p}(q)$, so its mean over a set $\mathcal{Q}$ of queries equals the price evaluated at the mean weights $\bar{p} = |\mathcal{Q}|^{-1} \sum_{q \in \mathcal{Q}} \tilde{p}(q)$. One accumulation of attention weights per token therefore prices every page of a head, and after installing any set of pages the certificate equals its initial value minus the sum of their prices.*

Proof. Linearity is immediate from equation 4, and the last claim is the telescoping noted in the proof of Proposition 1. ■

Proposition 4. *Let unit $u \in \{1, \dots, U\}$ offer pages in a fixed nested order with prices $g_u(1), \dots, g_u(m_u)$, so that page j becomes available once pages $1, \dots, j-1$ are installed. We call this sequence the chain of unit u and let $\hat{g}_u(j) = \min_{j' \le j} g_u(j')$ be its running minimum. The greedy that repeatedly installs the available page of largest price, with ties broken by unit index, installs after k steps exactly the k pages with the largest values of $\hat{g}$, ties at the threshold again resolved by unit index. The number of pages installed in unit u is therefore the number of j with $\hat{g}_u(j) > \theta$ plus its share of the ties at θ, where θ is the k-th largest value of $\hat{g}$.*

Proof. The running minimum is the effective priority of a page. A page waits for every predecessor in its chain, including those with a lower price, so it competes for installation at the smallest price among itself and its predecessors, which is $\hat{g}_u(j)$. We make this precise by showing, by induction over the steps, that whenever the greedy installs page (u, j) its running minimum $\hat{g}_u(j)$ is at least $\hat{g}_{u'}(j')$ for every page (u', j') not yet installed.

For $u' \neq u$ let h be the available page of u'. The greedy's choice gives $g_u(j) \ge g_{u'}(h)$, and $\hat{g}$ is nonincreasing along every chain, so $\hat{g}_{u'}(j') \le \hat{g}_{u'}(h) \le g_{u'}(h) \le g_u(j)$ for all $j' \ge h$. If $j > 1$ the page $(u, j-1)$ was installed earlier while (u', j') remained, so the induction hypothesis gives $\hat{g}_u(j-1) \ge \hat{g}_{u'}(j')$. Hence $\hat{g}_u(j) = \min\{\hat{g}_u(j-1), g_u(j)\} \ge \hat{g}_{u'}(j')$, and for $u' = u$ the claim is the monotonicity of $\hat{g}_u$. The installed values of $\hat{g}$ are thus nonincreasing in installation order and each dominates every value still available, which identifies the installed set with the top k values of $\hat{g}$. Equal values are installed in unit order by the tie rule. ■

Corollary 2. *If every chain is nonincreasing, then $\hat{g} = g$ and the first k pages of the greedy have the largest total price among all nested selections of k pages. By Lemma 4 this selection minimizes the certificate at every page budget.*

Proof. With nonincreasing chains the set of the k largest prices contains the predecessor of every page it contains, so it is itself a nested selection, and no nested selection of k pages can exceed the sum of the k largest prices. ■

In the recorded-window replays a unit is one layer–head pair, its chain is the page order that the local greedy over the head's chunks produces, and the global greedy runs over all units of a window as described in Appendix C. The closed form evaluates one allocation by selecting the k-th largest running minimum instead of simulating the greedy step by step, which makes the 10,000 paired refits of the within-window unit resampling of Appendix C practical. The running minimum describes the greedy exactly for every chain. When chunk prices decrease with depth, the local greedy produces a nonincreasing chain for every unit, and Corollary 2 adds optimality. The serving controller runs the same nested greedy with price per byte as its key, because the last chunk of a head can be shorter than C tokens.

A.9 Layer composition

Concatenating head errors bounded by E_h and applying the output projection W_O gives

$$\|\Delta \,\mathrm{layer}\|_2 \le \|W_O\|_2 \Big(\sum_h E_h^2\Big)^{1/2}.$$

Here $\|W_O\|_2$ is the spectral norm, which can be precomputed. Extending the bound to logits would also require controlling how later layers amplify attention errors. We do not establish tight bounds on this amplification, so the reported certificate remains restricted to the attention interface at one layer and one step.

B STORAGE AND COMPUTE ACCOUNTING

Storage. Table 5 itemizes the bytes that the packed representation stores per cached token and key–value head at head dimension $d = 128$ with 64-wide groups, which covers every model in Table 17. Dense fp16 storage costs 512 bytes. The 2-bit base costs 132 bytes, of which the 5-bit key codes take 80, the 2-bit value codes 32, the offset and scale metadata of keys and values 16, and the two receipts 4. Its ratio to dense storage is 0.258 before the exact window, and the 128 exact tokens raise it to 0.264 at 16K and 0.259 at 128K, which reproduces the base column of Table 17 to three decimals. Receipts therefore cost 0.8% of dense storage and all quantization metadata 3.1%. One correction page adds 42 bytes, so all three pages together double the base to 0.504 of dense storage, and a paged configuration that fills the KIVI-style budget reaches the comparator's 0.3125 plus the window share. The implementation counts bytes with the same integers, and the task studies check the resulting byte totals request by request.

Table 5. Storage per cached token and key–value head at $d = 128$ with 64-wide groups.

Component	Bytes	Share of dense
Dense fp16 keys and values	512	1.0000
5-bit key codes	80	0.1562
Key metadata (fp16 lo and scale per channel per 64-token group)	8	0.0156
2-bit value codes	32	0.0625
Value metadata (fp16 lo and scale per 64-channel group)	8	0.0156
Receipts $\bar{\rho}_i$ and $\bar{\nu}_i$ (fp16)	4	0.0078
Packed 2-bit base	132	0.2578
One correction page (2-bit codes, metadata, $\bar{\rho}_i$ of the new level)	42	0.0820
Base with all three pages resident	258	0.5039
KIVI-style comparator (4-bit keys and values, fp16 zero and scale per 32-group)	160	0.3125

Shares are ratios to the 512 bytes of dense fp16 storage and exclude the exact window of 128 tokens, whose share is $128/n$ at context length n. The packed base together with the window gives the base column of Table 17.

Certificate arithmetic in the kernel. Evaluating the certificate reuses the attention pass of the packed kernel. One kernel reads the packed keys and writes the span scores, and the softmax over span and window runs in PyTorch. A second kernel reads the packed values and, beside the attention output of every query head, accumulates the attention mass $M_g = \sum_{i \in g} \tilde{p}_i$ and the weighted norm $N_g = \sum_{i \in g} \tilde{p}_i \bar{\nu}_i$ of every 64-token key group together with the span's value term $\sum_i \tilde{p}_i \bar{\rho}_i$ of Theorem 1. A third kernel adds the weighted residual of every resident page. Because the score radius is shared within a key group, a reduction over the G groups then forms $A = \sum_g M_g (e^{\varepsilon_g} - 1)$, $B = \sum_g M_g (1 - e^{-\varepsilon_g})$, $\sum_i \tilde{p}_i (e^{\varepsilon_i} - 1)\bar{\nu}_i = \sum_g N_g (e^{\varepsilon_g} - 1)$, and $\sum_i \tilde{p}_i \bar{\nu}_i$ as $\sum_g N_g$ plus the window contribution. A closing evaluation of equation A.3 then adds the value term. The G radii $\varepsilon_g = \langle |q|, \Delta_g \rangle / \sqrt{d}$ cost one product of $|q|$ with the $G \times d$ matrix of channel bounds. Per head and decode step, the certificate therefore adds $O(Gd)$ arithmetic for the radii, $O(|T|)$ multiply-adds within the value pass, $O(G)$ for the reduction over groups, and $O(1)$ for the closing evaluation, and it holds five scalars until that evaluation. The kernel accumulates in fp32, and the serving audits of Appendix D.7 compare its certificate with the float64 evaluation from the same stored receipts.

C EVALUATION PROTOCOL

Models and data. Public models from the Qwen, Llama, and SmolLM families span 360M to 70B parameters. Allocation experiments draw attention streams from WikiText-103, PG19, and C4. Task quality is measured separately on RULER and LongBench. Exact model revisions and dataset identifiers are listed in the README of the supplementary code package. The recorded GPU runs used PyTorch 2.11, Transformers 5.5, and Datasets 5.0 under Python 3.12, with CUDA 12.8 on the RTX PRO 6000 and CUDA 13.0 on the B200.

Numerical audits. Every audit compares a float64 attention error against the float64 certificate of the same stored state and query. Table 6 breaks the 2,539,402,208 recorded comparisons down

by audit family together with the criterion of each, and no comparison in any family crosses its threshold. The largest realized error relative to its bound is 0.40 in the timed serving runs, 0.36 in the task-study audits, and 0.77 in the ablation replays, and the supplementary code package lists the audit count of every recorded run.

Table 6. Audit families behind the recorded float64 checks.

Audit family	Checks	Violations
Serving trajectories of the matched-budget RULER-32K studies (39 and 13 audited requests, both packed arms)	11,954,912	0
Timed serving runs of Table 17 (recorded queries of every round)	4,723,200	0
Packed-cache certificate sweep (eight precision settings, two sink arms)	137,536	0
WitCert comparison instances at 5-bit keys (the 8-bit rows are audited in the replay)	29,376	0
Reference certificate pairs of the two sub-1B models	117,504	0
Subtotal under the criterion $\mathrm{err} > \mathrm{cert}(1 + 10^{-9}) + 10^{-12}$	16,962,528	0
Final states of the target-driven controller on the 1,465 recorded windows (no allowance)	434,135,040	0
Held-out outputs of the page-price ablation on 614 windows (allowance $10^{-10}(1 + E)$)	2,088,304,640	0
Total	2,539,402,208	0

The first five families share the criterion of the subtotal row, and the last two use the allowances stated in their rows, which were fixed with their protocols.

Recorded streams and query split. Each 2K corpus window fixes its first 1,536 tokens as the cache and uses the next 512 as queries. The first 256 queries determine a placement that remains fixed while the other 256 measure p99 error. Both policies replay exactly the same queries without appending keys or values from those positions. At 8K, the fixed cache grows to 6,144 tokens, while the remaining queries are again divided equally between placement and evaluation. Windows tile each corpus stream from its start, and a stream whose length is not a multiple of the window adds one final window aligned with its end, which overlaps its predecessor. Within a window the four sink tokens and the 128 most recent cache tokens stay exact, keys are quantized symmetrically to 8 bits per channel within 128-token chunks, and the queries of all heads that share a key–value head form one unit.

Equal-byte allocation. All policies start from the same key state and value base. Each correction page quantizes the residual left by the preceding precision level. Uniform placement gives every value chunk the same depth of one or two pages. Adaptive placement spends the identical page budget on eligible pages ranked by their mean decrease in the value-error bound over the decision queries. Both the codes and their offset and scale metadata count toward the budget. Each unit first orders its own pages by the local greedy over its chunks, and the global allocation then merges the units' chains by the nested greedy of Proposition 4, with ties resolved by unit and then chunk order.

Paired resampling. Placement intervals use 10,000 paired whole-window bootstrap replicates, each of which recomputes the pooled p99 ratio. A separate within-window analysis resamples layer–head units. For each of its 10,000 replicates, the analysis refits adaptive placement on the decision queries, rebuilds the equal-byte uniform comparator, then recomputes errors on the held-out queries. Pairing exposes both policies to the same sampled windows or units. Because units within a window share a forward pass and a corpus's final window can overlap its predecessor, these intervals describe resampling sensitivity and should not be interpreted as uncertainty across independent samples.

Allocation to an error target. A separate replay defines each layer–head target as the p99 value-error bound reached by uniform one-page allocation on the first half of the queries. Static comparators select a single precision that meets this target either at p99 or on every decision query. The dynamic controller processes the second half, adding a page whenever the current bound exceeds the target. It stops once the target is reached or no remaining page would lower the bound, and a held-out unit–query pair at which the second condition ends the loop is counted as uncertifiable. The

refinement-byte ratios of Table 2 divide the pages the dynamic controller installs by the pages of the static comparator that meets the target on every decision query.

Packed serving and page selection. The packed representation keeps the most recent 128 tokens exact and packs the preceding span, whose length is the largest multiple of 64 that fits. Keys use 5-bit codes per channel within 64-token groups and values 2-bit codes within 64-channel groups, each with fp16 offset and scale rounded outward as in Corollary 1, and every token stores fp16 receipts $\bar{\rho}_i$ and $\bar{\nu}_i$ rounded up. Three nested 2-bit pages are prepared per 128-token chunk. The certificate arm prices every page from the last 32 prompt queries, installs pages in order of price per byte while respecting the nesting of levels, and stops at the first candidate that does not fit its budget. The uniform arm installs the same numbers of pages of each size, one level per unit per pass in a fixed order and independently of the prices. Both arms stay within the byte budget of the KIVI-style comparator computed for the same request from the byte counts of Table 5, and a page is admissible only if its price is finite and positive. Selected pages are copied into compact buffers that the kernel reads directly.

Task-quality studies. The RULER-32K requests are generated with the official recipe for 13 tasks, and a manifest pins every request by its content hash. The primary study scores request indices 0 to 203 of every task, 2,652 requests in total, while indices 204 to 255 served as a resource test and later as the fidelity pilot and are excluded from every task statistic. The Llama replication uses 50 requests per task. The RULER score of a request is the fraction of reference answers whose normalized form appears in the prediction. LongBench uses the English subset of 16 tasks with 50 requests each and the official scorers, and one MultiNews request whose KIVI-style budget falls below the packed base is skipped in every model's run, leaving 799 rows. Every request runs the KIVI-style comparator first and then the two packed arms. Intervals use a paired bootstrap that resamples requests within each task with 100,000 replicates and a fixed seed. The decision rule has two criteria. The primary criterion requires the lower interval limit of the adaptive minus uniform difference to exceed zero, and the secondary criterion requires the lower limit of the adaptive minus KIVI-style difference to exceed -0.5 points. The primary study met only the secondary criterion, which establishes noninferiority within 0.5 points and leaves the equal-byte placement difference unresolved. The Llama replication met neither. For three requests per task in the primary study and one per task in the replication, the serving trajectories of both packed arms were recorded and replayed offline in float64 against the original span, which is the first family of Table 6. A further study at half the page budget runs the same three arms and the same decision rule on request indices 0 to 49 of every task. Its rule was frozen before the run and its outcome is reported in Appendix D.5.

Output-fidelity study. Each request first decodes greedily from the dense cache, and the question together with that continuation, without a trailing end-of-sequence token, becomes the forced token sequence of every arm. Each arm consumes it one token at a time from the same absolute position through its own full forward pass, so its hidden states, queries and appended keys and values are its own while the token history is common. At every position the arm's next-token distribution over the full vocabulary is compared in float32 with the dense one by the KL divergence from dense, the top-1 disagreement indicator, and the negative log-probability of the forced token. The arms are the KIVI-style comparator, the packed base without pages, and adaptive and uniform allocation at identical page bytes. Their pages fill either the headroom between the packed base and the KIVI-style budget, as in Table 4, or half of it. Pages are priced from the last 32 prompt queries only. The request statistic is the mean over the continuation positions, from the last question token on, and intervals use the task-stratified paired bootstrap of the task studies. The primary criterion requires the lower interval limit of the uniform minus adaptive KL difference at the full page budget to exceed zero. The protocol was frozen before the pilot on the 204–255 indices, which used 6 requests per task and a permuted-order uniform arm. The confirmation then ran on request indices 0 to 24 of every task with the criterion unchanged. Three self-tests ran on three test requests of Qwen2.5-7B. Adaptive and uniform allocation with no pages reproduce the base arm exactly, a repeated replay is identical, and an fp32 reference attention over the same packed state reproduces the kernel's per-request KL to within 1.3%.

Serving measurements. Latency measurement begins after the selected pages are resident, so it excludes page loading and transfer. Keeping every candidate page resident would raise the packed

16K cache from $0.265\times$ to about $0.509\times$ dense KV. Runs on the 7B and 8B models use one RTX PRO 6000 Blackwell GPU, whereas the 70B model uses one 180-GB B200. Each configuration runs as three processes. The serving process prefills a fixed token stream, packs every layer, optionally selects pages and copies them into their buffers, drops the dense span, records a trajectory in which the model consumes its own generated tokens, and then times three teacher-forced rounds of 200 decode steps through the Triton kernel after eight warm-up steps. The dense process times the same rounds on the dense cache in a fresh process, and the audit process recomputes the original span, replays the recorded queries in float64 on both sides, counts violations, and checks the kernel output against an fp32 reference from the same packed state, with a threshold of 10^{-2} on the relative ℓ_2 error. The reported latency is the median of the three round medians. The 70B model and the 128K contexts pack one layer at a time, and the 128K audit streams the original span from CPU memory. In the 70B run at 128K the packed process had 142.5 GiB of the B200's memory allocated before timing, while the dense comparator ran out of memory.

Packed-cache sweep. The sweep records the final-position query and the full key–value state of seven evenly spaced layers for each WikiText segment, two segments for Qwen2.5-0.5B on CPU in float32 and eight segments for the 7B and 32B models on CUDA in bfloat16. Every setting packs the cache with the serving codec at 8, 6, 5, or 4 key bits and 4 or 2 value bits and evaluates Theorem 1 in float64 from stored quantities only, against the exact output of the original cache. The deployment arm keeps four sink tokens at their original precision and the strict arm keeps none. Cap frequency is the share of instances at which $\min(2A/(1-B), 2)$ reaches 2, and the tightness statistics are quantiles of the certificate-to-error ratio over instances whose error exceeds 10^{-12}.

WitCert comparison. Dense greedy decoding of three 1K WikiText contexts for 12 steps records every layer's queries and new keys and values of the two sub-1B models. Each state is then compressed with four exact sinks, chunks of 128 tokens sealed once 128 further tokens follow, symmetric per-chunk per-channel keys at 5 or 8 bits, and 2-bit values in 64-channel groups. For every query head we record the true error and WitCert's band-norm witness radii over 16 rotary bands, stored in fp16 with outward rounding and with 8 bands as a sensitivity setting. We then evaluate four output bounds with the value term held fixed. These are our certificate, the witness radii through WitCert's total-variation assembly, our chunk envelope through the same assembly, and the witness radii through our receipt-weighted assembly.

D Additional evidence

This appendix follows the order of Section 4 and reports the per-setting, per-task, and per-configuration results behind its tables.

D.1 Certificate tightness across precision settings

No violations occur across the 137,536 instances of the packed sweep, and Table 7 reports every setting. Three regularities appear. At 8-bit keys the softmax cap is reached in at most 0.2% of instances in every setting, at 6 bits in 1–16%, and at 5 bits in 21–64% without sinks, so the key term stays informative down to 5 key bits and loses most of its information at 4. Keeping four sink tokens exact lowers the cap frequency by a factor of two to four at 5 and 6 key bits, while the certificate-to-error ratio rises because exact sinks reduce the realized error more than they reduce the bound. At fixed key precision the ratio roughly halves when values move from 4 to 2 bits, which indicates that the value term is the tighter of the two terms and that the key term contributes most of the conservatism of the total. Ratios change little from 16K to 32K and between the 7B and 32B models. Reaching the softmax cap does not force E_{key} to equal $2V_{\max}$ because its receipt-weighted bound may still be smaller. Cap frequency therefore need not track the certificate-to-error ratio, which also depends on realized error.

D.2 Allocation across models and corpora

Across 1,465 windows, Table 8 covers 80 conditions spanning six models and three corpora, with two extensions to 8K. Adaptive allocation lowers pooled error in every condition, with all 80 window-resampling intervals lying above one. Uniform placement wins only six of the 5,860

Table 7. The complete packed-cache certificate sweep.

		cap %		median ×		p90 ×	
Setting	KV / dense	strict	deploy.	strict	deploy.	strict	deploy.
Qwen2.5-0.5B at 2K · 196 instances per setting and arm							
k8v4	0.520	0.0	0.0	8	22	28	36
k8v2	0.462	0.0	0.0	4	10	15	18
k6v4	0.462	3.1	3.1	29	68	108	115
k6v2	0.403	3.1	3.1	10	21	32	35
k5v4	0.433	20.9	5.6	74	138	222	249
k5v2	0.374	20.9	5.6	23	41	70	72
k4v4	0.403	71.4	21.9	171	318	551	576
k4v2	0.345	71.4	21.9	69	117	214	217
Qwen2.5-7B at 2K · 1,568 instances per setting and arm							
k8v4	0.451	–	0.0	–	22	–	39
k8v2	0.392	–	0.0	–	10	–	18
k6v4	0.392	–	1.0	–	76	–	131
k6v2	0.334	–	1.0	–	22	–	38
k5v4	0.363	–	9.6	–	167	–	288
k5v2	0.304	–	9.6	–	46	–	81
k4v4	0.334	–	41.1	–	392	–	728
k4v2	0.275	–	41.1	–	133	–	245
Qwen2.5-7B at 16K · 1,568 instances per setting and arm							
k8v4	0.419	0.1	0.1	16	32	42	74
k8v2	0.357	0.1	0.1	8	14	18	33
k6v4	0.357	14.7	3.2	66	120	187	257
k6v2	0.295	14.7	3.2	21	32	52	71
k5v4	0.326	61.2	24.0	167	272	470	603
k5v2	0.264	61.2	24.0	53	72	135	157
k4v4	0.295	88.1	64.7	295	539	854	1313
k4v2	0.233	88.1	64.7	106	180	295	476
Qwen2.5-32B at 16K · 2,240 instances per setting and arm							
k8v4	0.419	0.0	0.0	19	38	61	81
k8v2	0.357	0.0	0.0	8	16	24	34
k6v4	0.357	9.9	1.0	72	132	243	286
k6v2	0.295	9.9	1.0	20	33	62	73
k5v4	0.326	64.1	27.6	198	297	637	675
k5v2	0.264	64.1	27.6	49	69	159	164
k4v4	0.295	93.0	66.1	440	681	1416	1539
k4v2	0.233	93.0	66.1	122	181	433	482
Qwen2.5-7B at 32K · 1,568 instances per setting and arm							
k8v4	0.416	0.2	0.2	18	30	45	75
k8v2	0.354	0.2	0.2	8	12	19	34
k6v4	0.354	15.9	3.6	74	111	190	267
k6v2	0.292	15.9	3.6	22	30	52	72
k5v4	0.323	62.8	28.5	190	264	483	612
k5v2	0.261	62.8	28.5	55	66	137	160
k4v4	0.292	87.8	68.9	329	538	922	1380
k4v2	0.230	87.8	68.9	119	169	324	469
Qwen2.5-32B at 32K · 2,240 instances per setting and arm							
k8v4	0.416	0.0	0.0	21	37	62	87
k8v2	0.354	0.0	0.0	9	15	24	36
k6v4	0.354	11.5	1.6	82	136	249	306
k6v2	0.292	11.5	1.6	22	32	62	76
k5v4	0.323	64.0	30.5	223	321	684	731
k5v2	0.261	64.0	30.5	53	70	160	171
k4v4	0.292	92.4	68.4	470	719	1531	1677
k4v2	0.230	92.4	68.4	130	191	436	515

A setting kavb stores keys at a bits and values at b bits, and KV / dense is the byte ratio of the packed state, including the exact window, to dense fp16 storage. The strict arm keeps no sink token exact, while the deployment arm keeps four and is the only arm of the 2K sweep on the 7B model. Cap % is the frequency with which $\min(2A/(1-B), 2)$ reaches 2, and the median and p90 are quantiles of the certificate-to-error ratio over instances whose error exceeds 10^{-12}. Every row records zero violations.

window-level comparisons. Those exceptions come from two C4 windows with ratios between 0.93 and 0.97. A global page pool also lowers error relative to equal layer–head pools in 5,858 of

the 5,860 comparisons, and its advantage narrows at two pages per chunk, where a larger share of the budget is committed regardless of the pool, while every interval stays above one.

Table 8. Error ratios for uniform versus adaptive allocation under the same memory budget.

		1 page/chunk		2 pages/chunk	
Model	Corpus	Value error ↑	Output error ↑	Value error ↑	Output error ↑
Qwen2.5-0.5B	WikiText-103	1.52× [1.47, 1.56]	1.51× [1.47, 1.55]	1.59× [1.56, 1.62]	1.47× [1.45, 1.49]
Qwen2.5-0.5B	PG19	1.42× [1.34, 1.51]	1.43× [1.35, 1.51]	1.43× [1.38, 1.48]	1.35× [1.31, 1.39]
Qwen2.5-0.5B	C4	1.75× [1.64, 1.86]†	1.71× [1.61, 1.82]†	1.59× [1.52, 1.65]	1.43× [1.38, 1.48]
Qwen2.5-7B	WikiText-103	2.45× [2.35, 2.55]	1.72× [1.66, 1.78]	1.96× [1.91, 2.01]	1.42× [1.39, 1.44]
Qwen2.5-7B	PG19	2.49× [2.34, 2.64]	1.71× [1.57, 1.85]	1.96× [1.86, 2.05]	1.40× [1.33, 1.46]
Qwen2.5-7B	C4	2.49× [2.35, 2.62]	1.73× [1.62, 1.84]	1.93× [1.84, 2.02]	1.39× [1.35, 1.44]
Qwen2.5-32B	WikiText-103	1.94× [1.87, 2.00]	1.91× [1.85, 1.98]	1.79× [1.75, 1.84]	1.62× [1.59, 1.65]
Qwen2.5-32B	PG19	1.67× [1.58, 1.76]	1.66× [1.57, 1.74]	1.59× [1.52, 1.66]	1.49× [1.43, 1.54]
Qwen2.5-32B	C4	1.62× [1.46, 1.81]†	1.61× [1.45, 1.79]†	1.53× [1.40, 1.67]†	1.43× [1.32, 1.54]†
Qwen3-8B	WikiText-103	3.09× [2.97, 3.20]	2.99× [2.88, 3.09]	2.42× [2.38, 2.45]	2.12× [2.09, 2.14]
Qwen3-8B	PG19	2.74× [2.60, 2.91]	2.69× [2.55, 2.84]	2.20× [2.14, 2.26]	2.00× [1.95, 2.04]
Qwen3-8B	C4	3.03× [2.84, 3.22]	2.94× [2.76, 3.13]	2.29× [2.22, 2.36]	2.05× [2.00, 2.10]
Llama-3.2-1B	WikiText-103	1.65× [1.60, 1.70]	1.63× [1.59, 1.68]	1.70× [1.66, 1.74]	1.53× [1.51, 1.55]
Llama-3.2-1B	PG19	1.50× [1.43, 1.57]	1.49× [1.42, 1.56]	1.55× [1.49, 1.61]	1.44× [1.40, 1.49]
Llama-3.2-1B	C4	1.91× [1.76, 2.06]	1.88× [1.73, 2.03]	1.64× [1.55, 1.78]	1.45× [1.40, 1.58]
Llama-3.1-8B	WikiText-103	1.97× [1.91, 2.04]	1.94× [1.88, 2.00]	1.93× [1.88, 1.97]	1.71× [1.68, 1.74]
Llama-3.1-8B	PG19	1.69× [1.63, 1.76]	1.66× [1.60, 1.73]	1.71× [1.65, 1.77]	1.57× [1.53, 1.61]
Llama-3.1-8B	C4	1.85× [1.71, 1.99]	1.81× [1.68, 1.95]	1.81× [1.70, 1.91]	1.61× [1.53, 1.68]
Qwen2.5-7B	WikiText (8K)	2.75× [2.53, 2.99]	1.90× [1.77, 2.07]	2.11× [2.02, 2.22]	1.37× [1.33, 1.41]
Llama-3.1-8B	WikiText (8K)	2.10× [1.87, 2.42]	2.05× [1.84, 2.34]	2.05× [1.92, 2.20]	1.72× [1.65, 1.81]

Each ratio divides the pooled p99 error of uniform allocation by that of adaptive allocation, as in Table 1, for the realized value-reconstruction term or for the full attention output. A † marks the conditions with at least one window in which uniform allocation has the lower error.

Variation across heads. Table 9 quantifies the head-error spread behind Figure 5 for every setting. Under uniform one-page allocation the per-unit p99 value errors of a window span a factor of 8 to 52 between their 10th and 90th percentiles. Layer identity explains 63–73% of the variance of the logarithmic unit errors in the Qwen models and 31–47% in the Llama models, so the nonuniformity is largely a layer effect in the former and more head-specific in the latter. Adaptive allocation responds by leaving between 0.1 and 17.5% of the units without any page and by giving the top tenth of the units 19–25% of all pages, against the 10% they receive under uniform allocation. Within a setting the Spearman correlation between a window's spread and its gain ranges from -0.47 to $+0.44$, so the spread separates models better than it separates the windows of one model. Across the 20 settings the correlation between mean spread and pooled gain is 0.61, or 0.37 without the three Qwen3-8B settings, and across all 1,465 windows it is 0.54.

Table 9. Head-error spread and how adaptive allocation concentrates pages.

Model	Corpus	Windows	Spread	Layer %	Idle %	Top-10%	Spearman	Gain
Qwen2.5-0.5B	WikiText-103	146	1.12	72	7.4	22.5	+0.04	1.51×
Qwen2.5-0.5B	PG19	29	1.06	72	7.4	21.5	-0.08	1.43×
Qwen2.5-0.5B	C4	64	1.05	73	7.3	22.2	+0.08	1.71×
Qwen2.5-7B	WikiText-103	146	1.17	68	3.9	23.6	+0.20	1.72×
Qwen2.5-7B	PG19	29	1.10	66	5.0	23.6	+0.04	1.71×
Qwen2.5-7B	C4	64	1.02	65	2.4	22.7	+0.28	1.73×
Qwen2.5-32B	WikiText-103	146	1.16	70	4.9	22.5	+0.35	1.91×
Qwen2.5-32B	PG19	29	1.08	66	5.3	21.9	+0.11	1.66×
Qwen2.5-32B	C4	32	0.98	63	2.8	21.0	+0.31	1.61×
Qwen3-8B	WikiText-103	146	1.72	72	17.5	25.2	+0.06	2.99×
Qwen3-8B	PG19	29	1.62	71	17.5	24.4	-0.42	2.69×
Qwen3-8B	C4	64	1.54	69	12.1	24.3	-0.31	2.94×
Llama-3.2-1B	WikiText-103	142	1.13	31	9.8	19.3	+0.07	1.63×
Llama-3.2-1B	PG19	28	1.11	31	11.3	18.9	+0.12	1.49×
Llama-3.2-1B	C4	64	0.93	35	6.6	18.8	-0.47	1.88×
Llama-3.1-8B	WikiText-103	142	1.29	45	11.3	20.9	+0.12	1.94×
Llama-3.1-8B	PG19	28	1.25	47	11.0	20.3	+0.34	1.66×
Llama-3.1-8B	C4	64	1.07	46	6.0	19.7	+0.42	1.81×
Qwen2.5-7B	WikiText (8K)	37	1.14	69	0.1	22.4	+0.44	1.90×
Llama-3.1-8B	WikiText (8K)	36	1.28	45	2.7	20.7	+0.17	2.05×

Apart from the window count and the pooled gain, entries are means over the windows of a setting, and a unit is one layer–head pair. Spread is $\log_{10}(p90/p10)$ of the per-unit p99 value errors under uniform one-page allocation, and Layer % is the share of the variance of the logarithmic unit errors that layer identity explains. Idle % is the share of units that receive no page under adaptive allocation at one page per chunk, and Top-10% is the share of all pages that goes to the tenth of units with the most pages. Spearman is the rank correlation across the windows of a setting between the spread and the output-error ratio of Table 1, whose pooled value is the gain.

D.3 Target-driven controller

Table 10 reports the controller replay of Figure 3 for every setting. The static comparator that meets the calibrated target on every decision query defines the page budget of uniform allocation in Table 2. The dynamic controller installs 57–68% of those pages and meets the target on every held-out query except the uncertifiable pairs, at most 172 per setting out of 2.5 to 96 million. A static comparator that meets the target only at the p99 of the decision queries installs about half as many pages but misses the target on 1–3% of the held-out queries on average and on up to 77% of them in its worst unit. Between 0.3 and 0.7% of the held-out queries trigger an installation, and each trigger installs 1.6 to 3.5 pages before the bound returns below the target, with the larger counts at 8K.

D.4 Page-price ablation

We select up to 32 full, nonoverlapping windows at evenly spaced indices in each of the existing 20 model–corpus–context settings before replay. The resulting 614 windows use the same 8-bit keys, 2-bit value base, three nested 2-bit refinement pages, and decision/held-out query split as the original allocation experiments. All policies share one global page pool and spend exactly one or two refinement pages per chunk on average. Only the page ranking changes.

For a chunk at refinement depth r, the residual rule ranks the next-page decrease in $\sum_{i\in c}\bar{\rho}_i(r)^2$. The attention rule ranks $\bar{m}_c 4^{-r}$, where $\bar{m}_c$ is the chunk's mean attention mass over decision queries, and the factor 4^{-r} discounts slightly more steeply than the contraction by at least three that each 2-bit page guarantees (Appendix A.5). The certificate rule instead ranks the mean decrease in the attention-weighted residual norm. It therefore uses both the actual residual change and the attention that weights it. Each rule isolates one signal in a simplified form that reproduces no published allocator in full. Because the residual rule ranks squared norms, the comparison measures the usefulness of the full price and leaves the interaction between the two signals unestimated. A separate sensitivity check replaces the attention discount with the harmonic priority $\max(\bar{m}_c, 10^{-30})/(r+1)$.

Table 10. The target-driven controller in every setting.

			Pages / worst case		Static p99 attains %				
Model	Corpus	Pairs (M)	Dynamic	Static p99	Mean	Worst	Trig. %	Esc.	Uncert.
Qwen2.5-0.5B	WikiText-103	12.6	0.669	0.498	97.1	45	0.47	1.61	0
Qwen2.5-0.5B	PG19	2.5	0.672	0.497	97.3	54	0.47	1.61	0
Qwen2.5-0.5B	C4	5.5	0.679	0.491	96.9	52	0.47	1.64	0
Qwen2.5-7B	WikiText-103	29.3	0.662	0.498	98.0	63	0.45	1.66	4
Qwen2.5-7B	PG19	5.8	0.665	0.498	97.9	64	0.45	1.67	0
Qwen2.5-7B	C4	12.8	0.674	0.494	97.7	69	0.45	1.70	1
Qwen2.5-32B	WikiText-103	95.7	0.660	0.498	97.7	23	0.60	1.73	21
Qwen2.5-32B	PG19	19.0	0.662	0.498	97.7	24	0.62	1.69	5
Qwen2.5-32B	C4	21.0	0.665	0.493	97.4	24	0.60	1.77	33
Qwen3-8B	WikiText-103	43.1	0.658	0.498	98.0	69	0.74	1.74	5
Qwen3-8B	PG19	8.6	0.657	0.497	98.0	77	0.74	1.75	3
Qwen3-8B	C4	18.9	0.670	0.493	97.7	66	0.72	1.84	22
Llama-3.2-1B	WikiText-103	18.6	0.654	0.497	97.8	34	0.71	1.81	7
Llama-3.2-1B	PG19	3.7	0.641	0.495	98.3	55	0.70	1.81	0
Llama-3.2-1B	C4	8.4	0.638	0.492	98.0	36	0.70	1.82	130
Llama-3.1-8B	WikiText-103	37.2	0.651	0.495	98.5	67	0.68	1.91	20
Llama-3.1-8B	PG19	7.3	0.645	0.493	98.7	70	0.68	1.89	9
Llama-3.1-8B	C4	16.8	0.641	0.486	98.6	75	0.65	1.99	172
Qwen2.5-7B	WikiText (8K)	29.7	0.591	0.500	98.5	84	0.31	2.43	0
Llama-3.1-8B	WikiText (8K)	37.7	0.569	0.488	99.1	84	0.38	3.46	12

Each setting uses the windows of Table 8, and Pairs counts its held-out unit–query pairs in millions. Pages / worst case divides the pages of the dynamic controller, or of the static comparator calibrated at the p99 of the decision queries, by the pages of the static comparator that meets the target on every decision query, and the static p99 columns give the share of held-out queries at which that comparator meets the target, on average and in its worst unit. Trig. % is the share of held-out queries at which the dynamic controller installs at least one page, and Esc. is the mean number of pages per trigger. Uncert. counts the unit–query pairs at which the controller stopped above the target because no remaining page lowered the bound, and every final state passed the float64 audit.

We evaluate realized attention-output ℓ_2 error on identical held-out queries and compute p99 within each window. For each setting and budget, the reported estimate is the geometric mean of paired window error ratios. Unlike the original allocation tables, it is not a ratio of pooled p99s. Intervals use 10,000 paired whole-window resamples with seed 1. Like the allocation intervals of Appendix C, they describe resampling sensitivity within fixed corpus streams and carry no adjustment for multiple comparisons. Table 11 reports every setting, while the per-model summaries in Table 3 weight settings equally and carry no aggregate intervals. CUDA float64 replay audits find no violations under the fixed allowance $10^{-10}(1 + E)$ for a bound E, and every policy matches the page count and byte budget.

The complete comparisons show lower geometric-mean error for the certificate price against both primary heuristics in every setting and budget. The advantage does not hold in every window. Across the 1,228 window–budget pairs, the certificate rule wins 1,156 comparisons against the residual rule and 1,188 against the attention rule. These results concern attention error at fixed refinement bytes and do not establish a downstream task-quality or serving-speed advantage.

Either signal alone can also fall behind uniform allocation. Dividing a heuristic's ratio in Table 11 by the uniform ratio in Table 12 gives its error relative to uniform allocation on the same windows, and the supplementary code package reports the paired-window interval of each such ratio. The residual rule raises error above uniform allocation with an interval above one in 9 of the 40 setting–budget pairs, all on the Llama models, and the attention rule does so in 2 pairs on Qwen3-8B. The certificate price lowers error below uniform allocation in all 40 pairs.

Table 11. Setting-level page-price ablation with paired-window intervals.

		1 page/chunk		2 pages/chunk	
Model	Corpus	Residual only ↑	Attention only ↑	Residual only ↑	Attention only ↑
Qwen2.5-0.5B	WikiText-103	1.15× [1.07, 1.24]	1.11× [1.08, 1.14]	1.11× [1.06, 1.15]	1.10× [1.07, 1.14]
	PG19	1.17× [1.10, 1.26]	1.08× [1.05, 1.11]	1.10× [1.06, 1.14]	1.07× [1.04, 1.10]
	C4	1.61× [1.45, 1.77]	1.09× [1.05, 1.12]	1.35× [1.26, 1.44]	1.01× [0.97, 1.05]
Qwen2.5-7B	WikiText-103	1.27× [1.19, 1.35]	1.40× [1.37, 1.44]	1.04× [1.01, 1.07]	1.20× [1.17, 1.23]
	PG19	1.27× [1.19, 1.34]	1.46× [1.40, 1.52]	1.05× [1.01, 1.08]	1.21× [1.18, 1.24]
	C4	1.30× [1.22, 1.40]	1.36× [1.32, 1.40]	1.07× [1.03, 1.11]	1.15× [1.12, 1.18]
	WikiText (8K)	1.44× [1.35, 1.54]	1.41× [1.38, 1.44]	1.07× [1.05, 1.09]	1.13× [1.11, 1.15]
Qwen2.5-32B	WikiText-103	1.32× [1.24, 1.42]	1.45× [1.43, 1.48]	1.17× [1.13, 1.22]	1.27× [1.24, 1.30]
	PG19	1.22× [1.17, 1.28]	1.42× [1.39, 1.45]	1.20× [1.15, 1.24]	1.30× [1.27, 1.32]
	C4	1.32× [1.23, 1.42]	1.34× [1.31, 1.37]	1.26× [1.19, 1.33]	1.20× [1.18, 1.23]
Qwen3-8B	WikiText-103	1.87× [1.73, 2.03]	3.21× [3.13, 3.28]	1.20× [1.16, 1.25]	2.08× [1.99, 2.17]
	PG19	1.99× [1.84, 2.16]	3.22× [3.10, 3.34]	1.25× [1.20, 1.30]	2.30× [2.24, 2.35]
	C4	2.54× [2.24, 2.88]	3.13× [3.01, 3.26]	1.45× [1.37, 1.54]	2.00× [1.92, 2.10]
Llama-3.2-1B	WikiText-103	2.29× [2.09, 2.52]	1.27× [1.24, 1.30]	1.70× [1.62, 1.81]	1.15× [1.13, 1.17]
	PG19	1.83× [1.68, 1.99]	1.20× [1.17, 1.23]	1.51× [1.47, 1.57]	1.15× [1.13, 1.17]
	C4	3.12× [2.65, 3.67]	1.19× [1.16, 1.22]	1.95× [1.78, 2.14]	1.08× [1.06, 1.09]
Llama-3.1-8B	WikiText-103	1.84× [1.73, 1.95]	1.33× [1.31, 1.35]	1.62× [1.55, 1.70]	1.18× [1.17, 1.20]
	PG19	1.48× [1.44, 1.54]	1.28× [1.27, 1.30]	1.64× [1.57, 1.71]	1.21× [1.20, 1.23]
	C4	1.98× [1.80, 2.17]	1.28× [1.24, 1.32]	2.07× [1.91, 2.25]	1.17× [1.15, 1.18]
	WikiText (8K)	2.21× [1.97, 2.46]	1.24× [1.22, 1.26]	1.81× [1.70, 1.93]	1.12× [1.11, 1.14]

Each ratio divides a heuristic's held-out p99 attention-output error by the certificate's at equal page bytes, so values above one favor the certificate. Entries are geometric means of paired window ratios with 95% paired-window bootstrap intervals, whereas Tables 1 and 8 report ratios of pooled p99 errors. Windows span 2K tokens unless marked 8K, and Table 3 summarizes these ratios per model.

Table 12. Uniform allocation and an alternative attention discount on the same windows.

		1 page/chunk		2 pages/chunk	
Model	Corpus	Uniform ↑	Harmonic attention ↑	Uniform ↑	Harmonic attention ↑
Qwen2.5-0.5B	WikiText-103	1.56× [1.46, 1.68]	1.42× [1.39, 1.46]	1.46× [1.41, 1.51]	1.35× [1.28, 1.43]
Qwen2.5-0.5B	PG19	1.45× [1.37, 1.53]	1.38× [1.33, 1.42]	1.35× [1.31, 1.39]	1.31× [1.25, 1.38]
Qwen2.5-0.5B	C4	1.63× [1.52, 1.74]	1.37× [1.32, 1.43]	1.35× [1.30, 1.40]	1.28× [1.23, 1.35]
Qwen2.5-7B	WikiText-103	1.74× [1.63, 1.85]	1.64× [1.58, 1.70]	1.40× [1.36, 1.44]	1.43× [1.36, 1.49]
Qwen2.5-7B	PG19	1.76× [1.65, 1.88]	1.59× [1.54, 1.64]	1.42× [1.37, 1.48]	1.45× [1.40, 1.49]
Qwen2.5-7B	C4	1.62× [1.52, 1.73]	1.58× [1.53, 1.65]	1.33× [1.28, 1.38]	1.42× [1.34, 1.49]
Qwen2.5-32B	WikiText-103	1.91× [1.80, 2.03]	1.69× [1.65, 1.73]	1.60× [1.56, 1.66]	1.53× [1.49, 1.59]
Qwen2.5-32B	PG19	1.65× [1.57, 1.74]	1.68× [1.66, 1.71]	1.49× [1.43, 1.54]	1.61× [1.57, 1.65]
Qwen2.5-32B	C4	1.59× [1.46, 1.73]	1.64× [1.59, 1.68]	1.40× [1.32, 1.48]	1.62× [1.56, 1.69]
Qwen3-8B	WikiText-103	3.03× [2.85, 3.23]	4.53× [4.37, 4.69]	2.08× [2.04, 2.13]	2.99× [2.79, 3.19]
Qwen3-8B	PG19	2.73× [2.58, 2.89]	4.55× [4.37, 4.72]	2.00× [1.95, 2.04]	3.53× [3.38, 3.69]
Qwen3-8B	C4	3.04× [2.80, 3.30]	4.14× [4.00, 4.29]	2.01× [1.94, 2.07]	2.96× [2.75, 3.19]
Llama-3.2-1B	WikiText-103	1.70× [1.61, 1.80]	1.66× [1.63, 1.70]	1.51× [1.46, 1.56]	1.35× [1.31, 1.38]
Llama-3.2-1B	PG19	1.49× [1.42, 1.56]	1.62× [1.58, 1.67]	1.43× [1.38, 1.48]	1.34× [1.32, 1.37]
Llama-3.2-1B	C4	1.98× [1.80, 2.16]	1.55× [1.50, 1.60]	1.66× [1.56, 1.76]	1.42× [1.38, 1.46]
Llama-3.1-8B	WikiText-103	1.98× [1.89, 2.08]	1.54× [1.51, 1.56]	1.71× [1.67, 1.76]	1.33× [1.30, 1.36]
Llama-3.1-8B	PG19	1.65× [1.60, 1.71]	1.59× [1.56, 1.62]	1.56× [1.52, 1.60]	1.43× [1.40, 1.46]
Llama-3.1-8B	C4	1.68× [1.56, 1.84]	1.54× [1.50, 1.58]	1.50× [1.43, 1.58]	1.56× [1.50, 1.62]
Qwen2.5-7B	WikiText (8K)	1.90× [1.76, 2.06]	1.69× [1.65, 1.74]	1.37× [1.33, 1.41]	1.30× [1.25, 1.35]
Llama-3.1-8B	WikiText (8K)	2.13× [1.93, 2.36]	1.53× [1.49, 1.57]	1.71× [1.65, 1.78]	1.25× [1.22, 1.29]

Ratios and intervals are computed as in Table 11, with uniform allocation or a harmonic attention rule in place of the heuristic. Uniform allocation gives every chunk the same depth, while the harmonic rule uses $\max(\bar{m}_c, 10^{-30})/(r+1)$ as its next-page priority, and every interval favors the certificate price on these fixed windows.

D.5 TASK QUALITY

Tables 14 and 15 break the matched budget comparisons of Table 4 down by task. The single-needle tasks of RULER are saturated for all three arms, and the remaining task-level differences arise in the multikey, aggregation, and question-answering tasks, where the three arms lie within 3 points of one another in the primary study and within 8 points in the 50-request replication. The paired intervals of Table 4, which pool these tasks, cover zero in every comparison. On LongBench the macro averages of the three arms agree to within 0.1 points for every model. The largest task-level movements are the code completion task LCC, where both packed arms exceed the KIVI-style comparator by up to 5.9 points on Qwen2.5-7B and Llama-3.1-8B, and PassageCount on Llama-3.1-8B, where adaptive allocation falls 4.6 points below the comparator on a task that every arm scores below 7 points.

Table 13. RULER scores below the KIVI-style budget.

Comparison	Path	KV / dense 16-bit ↓	Δ score ↑
32K k5v2 / KIVI	unpacked/simulated	0.261/0.316	-1.79 [$-3.11, -0.51$]
32K k6v2 / KIVI	unpacked/simulated	0.293/0.316	-1.83 [$-3.37, -0.33$]
32K adaptive / uniform at half budget	packed	0.289/0.289	-0.09 [$-1.33, 1.14$]
32K adaptive / KIVI at half budget	packed/simulated	0.289/0.316	-0.29 [$-1.25, 0.67$]

Every row compares two arms of Qwen2.5-7B on 650 RULER-32K requests, and its Path and KV entries follow the order of the first column. A kXv2 configuration stores X-bit keys and 2-bit values, an unpacked run expands the compressed cache into dense tensors before decoding, and simulated denotes the KIVI-style comparator, which simulates quantization in dense tensors. The two packed rows come from the half-budget study of Appendix C.

Table 14. Per-task RULER-32K scores of the three arms at matched budgets.

	Qwen2.5-7B (204 per task)			Llama-3.1-8B (50 per task)		
Task	KIVI-style	Uniform	Adaptive	KIVI-style	Uniform	Adaptive
NIAH single 1	100.0	100.0	100.0	100.0	100.0	100.0
NIAH single 2	100.0	100.0	100.0	100.0	100.0	100.0
NIAH single 3	98.5	99.5	99.5	100.0	100.0	100.0
NIAH multikey 1	99.5	100.0	100.0	100.0	100.0	100.0
NIAH multikey 2	98.5	97.5	98.0	100.0	100.0	100.0
NIAH multikey 3	84.3	83.8	86.8	98.0	98.0	98.0
NIAH multiquery	99.4	99.4	99.6	100.0	100.0	100.0
NIAH multivalue	90.8	91.5	91.2	99.5	99.5	100.0
Variable tracking	95.8	94.6	96.0	99.2	99.2	99.6
Common words	68.1	68.7	65.9	73.0	74.2	68.6
Frequent words	77.1	76.6	76.5	81.3	82.7	84.7
QA 1 (SQuAD)	55.8	55.0	56.1	71.0	71.7	63.8
QA 2 (HotpotQA)	48.5	46.6	49.0	72.0	68.0	74.0
Mean over tasks	85.9	85.6	86.0	91.8	91.8	91.4

Entries are mean containment scores in points, the fraction of reference answers that appear in the prediction. Adaptive and uniform allocation read packed caches at identical KV bytes per request within the KIVI-style budget, and Table 4 gives the paired differences and intervals for the same requests.

Task quality at lower budgets. Lowering KV memory below the KIVI-style budget produces a measurable task cost in Table 13 when the correction pages are absent. The two unpacked 32K configurations lose about 1.8 RULER points, and both intervals lie entirely below zero. Against the dense reference of the same requests, the KIVI-style comparator stays within 0.7 points of dense decoding, whereas these two configurations lose 2.5 points, concentrated in the third multikey needle task, the common-words task, and the second question-answering task. The packed representation at half the page budget, $0.289\times$ dense KV and 9% below the KIVI-style bytes, behaves differently. On 650 requests both policies score within 0.3 points of the comparator, and the study meets neither the primary criterion of adaptive above uniform nor the 0.5-point noninferiority criterion, because the intervals span about ± 1.2 points at this size. The task scores thus leave the two policies unresolved

Table 15. Per-task LongBench scores of the three arms at matched budgets.

	Qwen2.5-7B			Llama-3.1-8B			Qwen3-8B		
Task	KIVI	Unif.	Adapt.	KIVI	Unif.	Adapt.	KIVI	Unif.	Adapt.
NarrativeQA	4.1	4.3	4.8	32.3	30.8	30.9	4.1	4.1	4.0
Qasper	10.7	10.4	9.3	13.2	14.0	15.0	11.9	12.1	11.6
MultiFieldQA	25.7	25.6	24.7	30.4	30.4	29.4	25.7	26.1	24.6
HotpotQA	10.8	11.2	11.0	19.6	18.9	19.5	14.1	13.2	13.0
2WikiMQA	10.9	11.2	11.0	17.9	17.1	17.5	13.5	12.4	13.8
MuSiQue	6.3	6.3	6.4	13.5	13.3	12.7	7.5	8.1	7.7
GovReport	35.5	35.6	35.0	34.8	36.2	36.0	31.0	31.0	30.8
QMSum	19.1	18.8	19.4	23.1	23.6	23.4	20.0	20.3	20.8
MultiNews	23.9	24.3	24.1	26.1	26.6	25.9	23.4	24.3	23.8
TREC	42.0	39.0	40.0	35.7	34.0	36.0	38.7	39.0	37.3
TriviaQA	18.8	18.2	17.9	17.5	17.8	17.8	21.7	22.1	22.4
SAMSum	27.1	27.1	26.8	23.0	23.8	23.1	25.8	24.9	25.3
PassageCount	5.3	5.0	6.0	6.5	4.1	1.9	6.5	4.2	5.2
PassageRetrieval	98.7	98.7	99.3	98.0	98.0	96.4	93.3	95.3	95.0
LCC	62.6	66.6	68.5	59.7	62.1	64.2	70.0	70.0	69.7
RepoBench-P	62.0	59.8	59.9	53.3	52.6	54.3	59.3	60.0	60.3
Macro average	29.0	28.9	29.0	31.5	31.5	31.5	29.1	29.2	29.1

Entries are the official LongBench metrics on a 0–100 scale over 50 requests per task, or 49 for MultiNews. KIVI denotes the KIVI-style comparator, and Unif. and Adapt. denote uniform and adaptive allocation at identical packed KV bytes.

at a budget where the fidelity study of Appendix D.6 separates them by a factor of about four in KL divergence.

D.6 Output fidelity at matched budgets

Table 16 lists the values behind Figure 6, the study of Appendix C that measures the model output directly under a common token history, on request indices 0 to 24 of every task, 325 per model. The packed base without pages lies an order of magnitude from dense on both models, and the pages recover most of that distance under either policy. At the page budget of Table 4, adaptive allocation lowers the KL divergence to dense by a factor of 3.4 on Qwen2.5-7B and 2.6 on Llama-3.1-8B relative to uniform allocation, with paired intervals that exclude zero. It has the lower divergence on 97% and 92% of the requests and in all 13 tasks, and it also lies below the KIVI-style comparator on both models. Top-1 disagreements with dense fall in the same order. At half the page budget, 9% below the KIVI-style bytes, adaptive allocation stays below the comparator while uniform allocation moves further from dense, so the ratio between the two policies widens to 3.9 and 3.0. The question positions, which every arm consumes before the continuation, show the same ordering. The pilot on the 204–255 indices gave ratios of 3.6 and 2.4 at the full budget, so the confirmation reproduces its magnitude. A sensitivity arm of the pilot spends the uniform page multiset in a seeded random unit order instead of the fixed chunk order. It lies above the fixed order on both models and budgets, so the fixed order of Appendix C gives uniform allocation its stronger form. A supplementary reading splits the same per-position traces into decoding positions 1–8, 9–32, and 33–128, counted from the last question token, with the request mean over all continuation positions unchanged as the endpoint. Adaptive allocation keeps the lower KL divergence in every band on both models and at both budgets, by 1.9 to 17.8×10^{-3} nats with paired intervals above zero, so the gain persists beyond the first predictions.

Table 16. Certificate-guided allocation brings the model output closer to dense decoding than uniform allocation at identical bytes.

			Qwen2.5-7B			Llama-3.1-8B		
Arm	Pages	KV / dense 16-bit ↓	KL ↓	Flips ↓	Unif. − adapt. ↑	KL ↓	Flips ↓	Unif. − adapt. ↑
KIVI-style	–	0.316	7.07	2.1	–	2.77	1.5	–
2-bit base	0	0.262	73.40	6.7	–	18.34	4.9	–
Uniform	1	0.316	12.27	2.8	–	4.15	1.7	–
Adaptive	1	0.316	3.65	1.4	+8.62 [+7.59, +9.71]	1.59	0.9	+2.56 [+2.10, +3.03]
Uniform	1/2	0.289	21.77	3.9	–	6.79	2.2	–
Adaptive	1/2	0.289	5.54	1.8	+16.23 [+14.63, +17.83]	2.24	1.0	+4.54 [+3.93, +5.16]

The dense model decodes each RULER-32K request greedily, and every arm then consumes the same tokens through its own forward pass. KL is the mean KL divergence from dense per position in units of 10^{-3} nats, and Flips is the percentage of positions whose top-1 token differs from dense, both over the continuation. Pages gives the share of the headroom between the packed base and the KIVI-style budget that the pages fill, and the last column of each block is the paired uniform minus adaptive KL difference with its task-stratified bootstrap interval.

D.7 SERVING

Table 18 lists the absolute measurements behind Table 17. Per-step latencies vary by less than 3% between the 10th and 90th percentile of steps in every run except the 70B model at 64K, and the receipts occupy 2.6–3.0% of the packed state in every configuration. The paged configurations fill the KIVI-style budget with between 8,128 and 186,880 pages, whose one-time selection took 1.6 to 53 seconds. Packing itself took under three seconds for the 7B and 8B models below 128K and up to 214 seconds for the 70B model at 128K, where the layers are packed one at a time.

Packed implementation checks. The float64 audit of every timed configuration compares the attention error with the certificate at 156,800 to 1,024,000 layer–head–step points of the recorded queries, and no point violates the bound. The worst ratio of realized error to certificate ranges from 0.14 to 0.40 across configurations. The kernel's fp32 output stays within a relative ℓ_2 distance of 9.5×10^{-4} of an fp32 reference from the same packed state, against the threshold of 10^{-2}, and its fp32 certificate within 4.5×10^{-4} of the float64 evaluation. Whenever a dense timing comparator is available, the packed path also reproduces all 64 tokens generated by the reference decoder from the unpacked compressed cache. For the 650 requests of the 32K study whose byte accounting was recorded, the packed arm's total bytes never exceed the KIVI-style budget of the same request, the unused slack ranges from 0 to 4,480 bytes and stays below one page of 5,376 bytes, and the arm installs between 12,813 and 16,384 pages per request.

Table 17. KV-cache memory and decoding latency at batch size one.

		2-bit base		Base + selected pages	
Model	Ctx.	KV / dense 16-bit ↓	Latency / dense ↓	KV / dense 16-bit ↓	Latency / dense ↓
Qwen2.5-7B	16K	0.264	1.195	0.318	1.305
Qwen2.5-7B	32K	0.261	1.097	0.315	1.198
Llama-3.1-8B	32K	0.261	**0.999**	0.315	1.063
Llama-3.1-8B	64K	0.259	**0.831**	0.314	**0.908**
Llama-3.1-8B	128K	0.259	**0.713**	n/r	n/r
Llama-3.1-70B	64K	0.259	1.791	0.314	1.952
Llama-3.1-70B	128K	0.259	**dense OOM**	n/r	n/r

Entries are ratios to dense 16-bit serving on identical inputs in PyTorch eager mode, timed with the selected pages already resident, so online selection, transfer, and migration are excluded. Bold marks latency at or below dense and the configuration where dense serving runs out of memory, n/r marks configurations that were not measured, and Appendix C lists the hardware.

Table 18. Absolute serving measurements behind Table 17.

			Latency per step (ms)		KV GB	Rec.		Sel.	Pack
Model	Ctx.	Cfg.	Packed	Dense	packed / dense	%	Pages	(s)	(s)
Qwen2.5-7B	16K	base	14.70 (14.66–14.75)	12.30	0.25 / 0.94	2.9	—	—	0.1
Qwen2.5-7B	16K	+pages	16.06 (16.01–16.11)	12.30	0.30 / 0.94	2.6	8,128	1.6	0.3
Qwen2.5-7B	32K	base	15.11 (15.06–15.15)	13.78	0.49 / 1.88	3.0	—	—	0.2
Qwen2.5-7B	32K	+pages	16.51 (16.45–16.56)	13.78	0.59 / 1.88	2.6	16,320	3.1	0.6
Llama-3.1-8B	32K	base	18.45 (18.32–18.53)	18.47	1.12 / 4.29	3.0	—	—	0.4
Llama-3.1-8B	32K	+pages	19.66 (19.54–19.75)	18.50	1.35 / 4.29	2.8	37,302	6.8	1.3
Llama-3.1-8B	64K	base	22.58 (22.39–22.73)	27.18	2.23 / 8.59	3.0	—	—	0.8
Llama-3.1-8B	64K	+pages	24.69 (24.50–24.80)	27.18	2.70 / 8.59	2.7	74,752	13.6	2.6
Llama-3.1-8B	128K	base	31.71 (31.45–31.82)	44.45	4.44 / 17.18	3.0	—	—	55.0
Llama-3.1-70B	64K	base	78.27 (76.68–82.24)	43.70	5.57 / 21.47	3.0	—	—	60.6
Llama-3.1-70B	64K	+pages	85.12 (84.22–85.71)	43.60	6.74 / 21.47	2.6	186,880	53.1	60.4
Llama-3.1-70B	128K	base	110.76 (108.88–111.44)	OOM	11.10 / 42.95	3.0	—	—	214.0

Packed latency is the median of three round medians over 200 teacher-forced steps, with the 10th to 90th percentile of steps in parentheses, and dense latency the same statistic for the dense fp16 cache in a separate process. KV GB gives the packed state including the exact window and the dense equivalent. Rec. % is the share of the packed state held by $\bar{\rho}_i$ and $\bar{\nu}_i$. Pages counts the resident correction pages, Sel. the one-time pricing and greedy over all layers before decoding, and Pack the one-time quantization of the prefilled cache. The base configuration of the 70B model at 128K has no dense comparator because the dense cache did not fit.

D.8 Comparison with WitCert

We evaluate WitCert's band-norm witness (Wei et al., 2026) on 58,752 recorded attention instances from two sub-1B models. Table 19 crosses two key-error summaries with two output-bound formulas while holding the value-error term fixed. This design isolates the information stored about key error from the formula that converts score uncertainty into output error.

Under its total-variation assembly, WitCert's tighter key radii yield a complete bound 1.09–1.34 times ours. Passing the same radii through our receipt-weighted assembly lowers the ratio to 0.63–0.91, which reverses the comparison. Storing those radii costs 32 bytes per token and head, whereas our key-error metadata uses roughly one byte.

Table 19. Certificate tightness under two key-error summaries and two output-bound formulas.

Key radius	Receipt-weighted assembly ↓	Total-variation assembly ↓
Receipt envelope	1.00	1.40–2.05
WitCert witness	0.63–0.91	1.09–1.34

Rows select the key-error summary and columns the output-bound formula. Each entry gives the range over four model–precision settings of the median ratio to our certificate, so values below one are tighter.

Table 20 reports the four settings separately with the absolute tightness of each bound. At 8-bit keys every bound is within 16 times the realized error at the median, our softmax bound reaches its cap in at most 2.3% of the instances, and WitCert's total-variation bound is vacuous in at most 1.3%. At 5-bit keys the receipt-weighted assembly reaches its cap in 27–37% of the instances, and WitCert's total-variation bound becomes vacuous, with a total-variation value of one, in 7–23%. The combination of WitCert's radii with our assembly is the tightest of the three complete bounds in 81–99.7% of the instances, which shows that the assembly and the stored key information contribute separately.

Table 20. The WitCert comparison by model and key precision.

			Bound / error, median / p90 ↓				Cap	Vac.	Hybrid
Model	Keys	n	Ours	WitCert	Hybrid	Env. via TV	%	%	best %
SmolLM2-360M	5-bit	17,280	54.7 / 105.4	79.4 / 184.5	34.6 / 67.0	121.5 / 312.5	27.3	6.8	96.1
SmolLM2-360M	8-bit	17,280	8.2 / 15.5	12.1 / 25.0	7.4 / 14.0	14.6 / 30.4	0.0	0.0	99.7
Qwen2.5-0.5B	5-bit	12,096	83.0 / 160.1	99.5 / 198.7	60.6 / 111.4	130.4 / 273.0	37.3	22.6	80.7
Qwen2.5-0.5B	8-bit	12,096	11.2 / 19.4	13.9 / 25.9	10.1 / 17.7	15.9 / 30.2	2.3	1.3	94.5

Every setting uses 2-bit values and has n instances. Ours is the certificate of Theorem 1, WitCert is the 16-band witness through its total-variation assembly, Hybrid passes the witness radii through our receipt-weighted assembly, and Env. via TV passes our chunk envelope through the total-variation assembly. Cap % is the share of instances at which our softmax bound reaches its cap, Vac. % the share at which WitCert's total-variation bound equals one, and Hybrid best % the share at which the hybrid is the tightest of the three complete bounds. None of the four bounds is violated on any instance.