On the Limits of KV Cache Merging

Anonymous authors
Paper under double-blind review

Abstract

Merging offers an appealing alternative to eviction from the key–value (KV) cache: an entry can be incorporated into a retained slot instead of discarded. We examine the limits of this approach in frozen long-context language models, both with and without corrections for the attention mass represented by each slot. For value-only merging, we prove that, when keys are in general position, exact recovery for every future query is possible only if eviction already incurs zero error. We then derive a calibrated criterion for approximate merging to improve on eviction. Across four rotary models from three families, at most a few percent of evicted tokens meet this criterion even under oracle routing and verbatim repetition: contextualisation and rotary embeddings keep their keys distinct. Attention-mass compensation escapes the theorem, but none of the published operators we port yields a statistically resolved LongBench gain over its own eviction baseline under a matched protocol. Ported operators that rewrite retained keys reduce performance; in the clearest case, fixing the keys eliminates the loss. Our fixed-key construction jointly fits the value mixture and mass correction to minimise attention-output error and accepts only merges that reduce it. Its statistically resolved gain of +0.70 on the 400-sample development split falls to an unresolved +0.18 on the 800-sample held-out split and reverses to −0.74 on Qwen2.5-7B-Instruct. These results identify key geometry and calibration transfer as constraints on merging. The eviction-error decomposition instead motivates value-weighted selection, which weights attention by value norm. Under our shortened-generation protocol, this selector has no statistically detectable LongBench difference from the strongest eviction baselines, while improving needle retrieval. Code and per-sample results: `https://anonymous.4open.science/r/kv-cache-merging-limits-4202`.

1 Introduction

The KV cache is the dominant memory cost of long-context LLM inference. Many compression methods reduce this cost by *evicting* tokens that attention scores suggest are unlikely to be needed again (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024; Cai et al., 2024). Merging offers a natural alternative. Long inputs contain substantial repetition: contracts repeat defined terms, and source code reuses identifiers. An entry selected for eviction may therefore resemble a retained entry and could be incorporated into it. Existing methods merge entries with similar keys (Wan et al., 2025; Wang et al., 2024; Wan et al., 2024) or combine neighbouring positions using attention-derived weights (Zhang et al., 2024; Yuan et al., 2025). More recent methods also compensate for the attention mass represented by merged slots (Tian et al., 2026; Bouyahiaoui et al., 2026; Liu et al., 2026). Their reported gains are generally small and often measured against weaker eviction baselines. We ask *how much KV-cache merging can improve on eviction in a frozen model, which mechanisms enable those gains, and what limits them.*

Contributions.

1. **Limits of value-coefficient merging** (§4). For keys in general position, a query-independent value edit recovers dense attention for every query only if eviction is already exact. Under a Gaussian logit-gap model supported by our measurements, the best constant route coefficient removes a fraction $e^{-\sigma^2}$ of squared coefficient

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.

error, where σ is the key gap under the query covariance. A calibrated criterion places break-even at $\sigma \leq 0.285$. An unrestricted class-A oracle jointly fitting all retained values removes 66% of squared eviction error in sample and 21%–28% on held-out queries.

2. **The mechanism across model families** (§5). Under oracle routing, median σ is 0.94–1.38 across four rotary models, with ≤5.1% of evicted tokens below break-even, including for verbatim repetition. Contextualisation alone keeps adjacent repeats 2–3.5× above break-even; RoPE roughly doubles the gap at long distances. On GPT-2 medium, which uses absolute positions, the mergeable fraction is 2–20× larger.
3. **Fixed-key compensation and its generalisation** (§6). Attention-mass compensation (Tian et al., 2026; Bouyahiaoui et al., 2026) lies outside the theorem's function class. For compensated key averaging with a matched merge weight, the mass residual is second order, while the value residual remains first order. We keep keys fixed and jointly fit the mass correction and value mixture to attention-output error, recovering 8–11% of eviction error on real decode queries. The construction gains +0.70 LongBench on the development split, with 7.9% lower perplexity and unchanged retrieval. The gain falls to an unresolved +0.18 on held-out data and reverses to −0.74 on Qwen2.5-7B-Instruct. Neither channel alone yields a resolved gain.
4. **A matched-protocol re-evaluation** (§7). Each ported operator is compared with its own eviction base. CaM (Zhang et al., 2024) yields −0.14 LongBench; KVMerger yields −0.58 on a strong selector and −1.35 on its own, despite improving perplexity; SelKV has no resolved gain; and our KeepKV port yields −9.75. Across 1,200 paired samples, two splits and two budgets, competitive selectors show no statistically detectable pairwise differences. Detecting a ±0.3-point difference with 80% power requires ≈5,800 paired samples, more than the English suite contains.
5. **Improving selection** (§8). The decomposition motivates *value-weighted selection* (VWS): window attention × value magnitude (VATP's factor; Guo et al., 2024), token-axis pooling and cross-layer water-filling. Its LongBench differences from SnapKV and PyramidKV are unresolved, but it retrieves 98%/90% of needles at 5%/2% budgets over 100 positions (pooled SnapKV 88%/78%, PyramidKV 82%/48%). It exceeds pooled SnapKV by +6.5 (CI $[+3.0, +10.5]$) on a RULER-style multi-needle suite at 2%, within 4% of SnapKV's decode speed.

2 Related work

Eviction. StreamingLLM keeps sinks and a recency window (Xiao et al., 2024); H2O accumulates attention over all queries (Zhang et al., 2023); SnapKV scores with a recent observation window and pools along the token axis (Li et al., 2024); PyramidKV shapes budgets by depth (Cai et al., 2024); Ada-KV and CAKE adapt budgets across heads and layers (Feng et al., 2025; Qin et al., 2025); VECTOR approximates values from retained keys (Lin et al., 2026); VATP scores tokens by attention × value norm (Guo et al., 2024), the factor VWS adopts (§8). Quest retains the full cache and retrieves the highest-scoring pages for each decode query (Tang et al., 2024), trading memory for bandwidth. Its query-dependent selection lies outside the fixed-coefficient setting of our theorem (§9). Concurrent work also targets the quantities in Lemma 1 — CAOTE's closed-form output-change score (Goel et al., 2025), ReST-KV's layer-wise output reconstruction (An et al., 2026), LAVa's residual-stream budgets (Shen et al., 2025), CompressKV's retrieval-head scores with layer-adaptive budgets (Lin et al., 2025), ChunkKV's contiguous chunks (Liu et al., 2025a), CompilerKV's offline policies (Yang et al., 2026) — providing related perspectives on the decomposition (§8). We do not evaluate these methods under our protocol.

Merging. Class A edits values only: CaM folds evicted values into following positions with attention-derived Bernoulli weights (Zhang et al., 2024); WeightedKV convexly merges them into neighbours (Yuan et al., 2025). D2O (Wan et al., 2025), KVMerger (Wang et al., 2024) and LOOK-M (Wan et al., 2024) additionally *average the keys* of a merge set into a pivot, so the merged slot carries the mean logit of its members and no mass channel (§6); AsymKV

(Cui and Xu, 2025) and GRKV (Peng et al., 2026) also move keys, by adjacent fusion and by output-fitted ridge updates. Compensated methods add a mass channel. KeepKV combines per-slot "electoral votes" with ZIP-Merging, preserving an EMA score surrogate while rewriting the retained key (Tian et al., 2026). SelKV uses a value-similarity gate and attention-ratio bias while keeping keys fixed (Bouyahiaoui et al., 2026). Other examples include KVSlimmer's asymmetric adjacent-pair merge (Liu et al., 2026) and ZSMerge's compensated residual slots (Liu et al., 2025b).

Outside our scope. Trained compression (Nawrot et al., 2024; DeepSeek-AI, 2024), merging across layers (Liu et al., 2024) or model scales (Zhao et al., 2025), global carriers for the evicted set such as MomentKV's moment statistics (Li et al., 2026), and retrieval over the full cache (Sun et al., 2025; Qi et al., 2026; Tang et al., 2024) escape our analysis by construction (§9).

3 Setup

For a single attention head, the cache contains post-RoPE keys $k_t \in \mathbb{R}^d$ and values $v_t \in \mathbb{R}^d$; a query q produces $o(q) = \sum_t a_t(q)\, v_t$ with $a_t(q) = e^{q\cdot k_t/\sqrt{d}}/Z(q)$. Compression keeps a set A, evicts E, and may replace retained values $v_j \to v_j + \delta_j$ with query-independent δ_j and keys unchanged (class A), or additionally equip slot j with a bias b_j so that its logit becomes $q \cdot k_j/\sqrt{d} + b_j$ (class B; equivalently KeepKV's multiplicative votes). A third family *moves keys*: a merge set S becomes one slot $(\bar{k}_S, \bar{v}_S) = \sum_{i\in S} w_i(k_i, v_i)$, without a bias (D2O, KVMerger, LOOK-M) or with one and a rescaled $\bar{k}_S$ (KeepKV; §6). We write $\rho(q) = \sum_{i\in E} a_i(q)$ for evicted mass and $\bar{v}_S(q) = \sum_{i\in S} a_i(q)\, v_i \big/ \sum_{i\in S} a_i(q)$ for the attention-weighted value centroid of a set S. Throughout, Σ_q is the *centred* covariance of the query distribution that will read the cache.

4 The limits of value-coefficient merging

Lemma 1 (Eviction error)**.** $o_{\text{evict}}(q) - o(q) = \rho(q)\,[\bar{v}_A(q) - \bar{v}_E(q)]$.

The error therefore depends on both the evicted attention mass and the gap between the retained and evicted value centroids. For nonzero evicted mass, eviction is exact when these centroids coincide. Selection reduces the error by choosing which tokens to retain; merging attempts to compensate for the residual error.

Theorem 1 (Exactness)**.** *Suppose $E \neq \emptyset$ and the cached keys are in general position (all pairwise sums $k_a + k_b$ over unordered index pairs, $a = b$ allowed, are distinct). If a class-A merge satisfies $o_{\text{merged}}(q) = o(q)$ for every $q \in \mathbb{R}^d$, then $\delta_j = 0$ for all j and all cached values are identical — i.e., the eviction error already vanished identically.*

The proof matches coefficients of the vector-valued exponential polynomial obtained by cross-multiplying the two softmax normalisers (Appendix A). General position forces all edits to vanish and all cached values to coincide. Without this assumption, the theorem does not apply. For a single coincident retained–evicted key pair, with other keys in general position, a value-only edit is exact only when every other value equals the pair's midpoint, making the dense output constant. Otherwise, exact absorption requires a mass correction: bias $\log 2$ on the averaged value (§6).

The practical question is therefore approximate recovery. Published class-A operators use *routing*: each evicted token contributes $\delta_j = \sum_i T_{ij} v_i$ through query-independent weights T. For a route $i \to j$, let $\Delta = k_i - k_j$. The true relative weight $R(q) = e^{q\cdot\Delta/\sqrt{d}}$ is replaced by a constant. We model the logit gap as Gaussian over queries, supported by the sampled routes' median $|\text{skew}| = 0.16$ and excess kurtosis $0.03 - -0.06$.

Proposition 1 (Ceiling under the Gaussian gap model)**.** *Fix a route $i \to j$ and suppose $X = q\cdot\Delta/\sqrt{d} \sim \mathcal{N}(\mu, \sigma^2)$ over the query distribution, with $\sigma^2 = \Delta^\top \Sigma_q \Delta/d$ and Σ_q the centred query covariance. Then the best constant coefficient $c^\star = \mathbb{E}[R]$ removes exactly a fraction $e^{-\sigma^2}$ of the squared coefficient error $\mathbb{E}[R^2]$ of not merging.*

This result has two qualifications. First, it holds under the Gaussian model and is not a distribution-free bound (Appendix A). Second, it assumes that $c^\star$ is optimal for the queries that will read the cache. A coefficient estimated from prefill queries incurs additional error when the optimal coefficient shifts during decoding (§6). At $\sigma = 1$, the optimal constant removes only 37% of the coefficient error; at $\sigma = 1.5$, it removes 11%.

To relate this route-level result to eviction, we use two observations. All merged tokens are evaluated at the *same* decode query, so their coefficient errors need not average out. Meanwhile, the relative bias that merging aims to remove, $\|\bar{v}_A - \bar{v}_E\|/\|v\|$, is small because centroids of many values concentrate (measured: 0.291). This motivates the following calibrated condition for merging to improve on eviction:

$$\mathrm{CV} = \sqrt{e^{\sigma^2} - 1} \;<\; \frac{\|\bar{v}_A - \bar{v}_E\|}{\|v\|} \quad \Longrightarrow \quad \sigma \leq 0.285. \tag{1}$$

(1) is a *calibrated criterion*, not a theorem: its constant inherits the measured centroid gap (0.291 on Qwen3-1.7B, 0.27–0.32 on Llama-3.1-8B), so break-even sits at $\sigma \approx 0.25$–0.32; we use 0.285, and Table 1 is insensitive to the choice (Appendix B). The criterion depends on a measurable key gap under the query covariance. The cosine similarity used to gate D2O, KVMerger and LOOK-M is only weakly correlated with this quantity ($r \approx 0.3$–0.4).

Beyond routing: the unrestricted class-A oracle. Proposition 1 analyses one route; a general class-A operator edits every retained value with an arbitrary vector, and joint edits could in principle cancel errors that a route-level analysis does not capture. We assess this possibility empirically. For a fixed anchor set the merge that minimises true squared output error over real queries is a linear least-squares problem in $\delta \in \mathbb{R}^{m\times d}$, $\min_\delta \|\hat{A}\,\delta - Y\|_F^2 + \lambda\|\delta\|_F^2$, with $\hat{A}$ the calibration queries' attention renormalised over the retained slots and $Y = o - o_{\text{evict}}$ (one row per query, the correction the edit must supply); its solution is the ridge-regularised in-sample optimum over this class. The class includes the value edits produced by any routing scheme or transport plan fitted on the same queries. We measure how well this estimator transfers to subsequent queries. Under a deployment-faithful protocol (Qwen3-1.7B, budget 0.15, 112 heads, window-attention anchors; fitted on the W prefill queries preceding decode, evaluated on the next 128 positions; Appendix B.1), at $W = 128$ it removes 66% of squared eviction error in sample and 21%–28% held out; $W \geq 512$ *lowers* the held-out figure to 10–13%, because older calibration queries are off-distribution for the decode queries (§6). The held-out reduction is therefore comparable to the route-level value $e^{-\sigma^2} \approx 21\%$ at the median σ; joint vector-valued edits provide only a modest improvement in this experiment. Two further checks clarify the result (Appendix B.1). With H2O anchors, the edited cache never outperforms plain window-based eviction in absolute error. An anti-causal oracle fitted on the answer's own decode queries, unavailable to a deployable method, removes 38–59% of the error. Even this favourable fit leaves about 40% or more of the eviction error uncorrected.

5 Measuring σ across model families

We measure σ with *oracle routing*: each evicted token is paired with the retained anchor that minimises σ. This gives the smallest gap available within the retained set, independently of a practical routing rule. Queries are captured post-RoPE from held-out continuation positions.

Table 1 reports the results for four rotary models from three families. Median σ is 3–5× above break-even across these models, with substantial variation across heads (0.4–3.2). Mistral's 5.1% mergeable fraction is concentrated in four layer-0 heads (Appendix B.3). Additional Qwen3-1.7B sweeps show the same pattern across the tested budgets, context lengths and text types, including *one passage repeated verbatim* 8× (σ 1.21 in that sweep; Appendix B).

Why repeated tokens remain distinct. Splitting same-token pairs by positional distance and re-rotating the later key onto the earlier position (removing RoPE exactly for identical content) separates the two mechanisms (Figure 1, left). At distance one, where RoPE contributes least, post-RoPE σ is 0.74 (Llama) and 1.08 (Qwen3) against 0.62 and 0.99

Table 1: Oracle-routing σ at a 15% budget (wikitext, three documents of 2560 tokens, probe = final 128 queries, every fourth layer). Mean over (layer, head) of the per-head median σ and of the per-head fraction of evicted tokens under the value-only bar (0.285) and the looser compensated-merge bar (0.9, §6); σ of a passage repeated 8×; σ between positions holding the *same token id* at distance 0–8 and >512. Last row: an absolute-position model (no RoPE, 896-token prefill, all 24 layers). Reading notes: Appendix B.4.

model	median σ	frac ≤ 0.285	frac ≤ 0.9	verbatim 8× σ	same-token σ, 0–8 → far
Qwen3-1.7B	1.24	0.4%	18%	1.07	1.38 → 2.00
Qwen2.5-7B-Instruct	1.38	0.1%	15%	1.37	1.48 → 2.25
Llama-3.1-8B-Instruct	0.98	0.4%	44%	0.90	1.06 → 1.55
Mistral-7B-Instruct-v0.3	0.94	5.1%	46%	0.88	1.07 → 1.55
GPT-2 medium (absolute positions)	0.83	10.0%	62%	0.64 (26%)	0.56 → 2.32

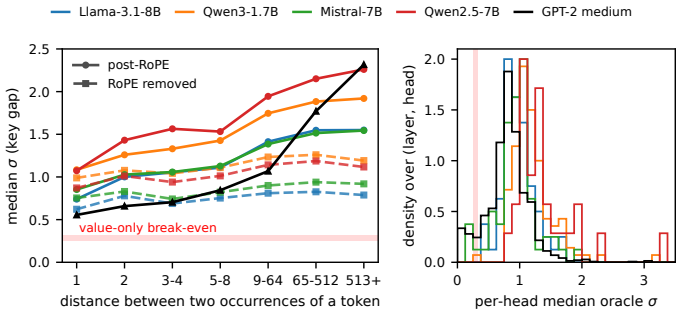

Figure 1: Left: median σ between two occurrences of the same token against their positional distance, with the keys as the model uses them (post-RoPE, solid) and with the later key re-rotated onto the earlier position (RoPE removed, dashed); GPT-2 medium has no RoPE and one curve. Shaded band: value-only break-even ($\sigma \leq 0.25$–0.32). Right: distribution of the per-(layer, head) median oracle σ at a 15% budget; two GPT-2 heads ($\sigma \approx 5.1$–5.4) lie off the axis. Colour identifies the model in both panels. Qwen3 is sampled every second layer here, so its values differ slightly from Table 1 (1.92 vs. 2.00 beyond 512 positions).

with RoPE removed: contextualisation alone keeps adjacent repeats 2–3.5× above break-even, because keys are functions of the whole prefix. With distance RoPE roughly doubles the gap (0.79–0.83 against 1.55 beyond 512 positions on Llama; 1.19–1.26 against 1.92 on Qwen3): rotary embeddings preserve positional distinctions between occurrences, reducing the redundancy available to merging.

Without RoPE. The mechanism suggests greater mergeability in an absolute-position model. Consistent with this prediction, on GPT-2 medium (Table 1, last row) median per-head σ is 0.83, 10% of evicted tokens clear break-even (20× the Llama and Qwen3 figures), 16% of heads have a median σ below 0.5, and verbatim repetition raises the mergeable fraction to 26%, whereas it has little effect on the rotary models. The limitation is therefore less severe in this absolute-position model: 90% of evicted tokens remain above the criterion, compared with approximately 99.5% on Llama and Qwen3 (§9, Appendix B).

6 Compensated merging: benefits and limits

KeepKV and SelKV equip merged slots with an attention-mass channel: votes p_j multiplying the softmax numerator, or equivalently a logit bias $b_j = \log p_j$. This extends the function class beyond Theorem 1: the slot now represents $e^{q \cdot k + b} v$, with an additional degree of freedom.

Second-order mass residual. Compensation addresses two sources of error differently. With a suitable averaged key, a constant bias matches the pair's *total* log-mass up to a second-order residual in the logit gap. The value *mixture*, however, follows the sigmoid of that gap and retains a first-order residual. Formally (Appendix A), merge a pair into $\bar{k} = wk_1 + (1-w)k_2$, $\bar{v} = wv_1 + (1-w)v_2$ and fit $b = \mathbb{E}_q[\log(e^{\ell_1}+e^{\ell_2}) - \bar{\ell}]$ on observed queries (from log-attention columns, where the normaliser cancels); with $x = \Delta\ell(q)$ of mean μ the per-query error $\log(1+e^x) - wx - b$ is $[\mathrm{sigmoid}(\mu) - w](x-\mu) + O((x-\mu)^2)$, *second order* exactly when $w = \mathrm{sigmoid}(\mu)$ (our construction uses the pair's mean attention share, which agrees to first order, and gates on the measured residual). For equal masses, the residual's standard deviation is $\leq 0.18\sigma^2$, and equals 0.17 at the measured adjacent-pair $\sigma \approx 1.1$. This is below the calibrated break-even reference, even though the key gap is about $4\times$ the class-A threshold. The value channel has no such property: the best constant replacing $w(q) = \mathrm{sigmoid}(\Delta\ell)$ still leaves $\mathrm{std}_q[w]\,\|v_1 - v_2\| \approx 0.22\,\|v_1 - v_2\|$ at $\sigma = 1.1$, so compensation makes a pair viable only to the extent that its two values agree, and the construction below gates on the per-pair mass error, measurable from stored window attention rows.

Key averaging without compensation (D2O, KVMerger, LOOK-M) under-represents a merge set's mass by at least $\log|S|$ in log-mass for equal weights, so its residual exceeds the compensated construction's by $(\mathbb{E}_q\Lambda)^2$ (Appendix A.1).

Mass compensation with a fixed value mixture. We implement this construction (Appendix C): adjacent selected pairs are scored by the measured $\hat{\sigma}$-residual, non-overlapping pairs below threshold are fused into a slot carrying $(\bar{k}, \bar{v}, b)$, the bias is injected at decode as an additive mask, and every freed slot admits one more token at the same budget. Relative to the identical selection without merging, its LongBench change is -1.83 $[-3.00, -0.80]$ at a 2% budget and -0.72 $[-1.58, +0.09]$, unresolved, at 5% (Table 2), leaves needle retrieval at 1.00 at 5% and improves continuation perplexity and KL by 0.3–2.5% (Table 7).

Calibration transfer. The bias is fitted on prefill-window queries but applied to subsequent queries. On Qwen3-1.7B over 64 decode steps (Appendix B) the *spread* of the logit gap under decode queries equals the prefill-probe σ (median ratio 0.91–1.01) but its *mean* shifts by 0.39–$0.44\,\sigma$; within the prefill the same statistic is 0.25–$0.40\,\sigma$ between windows, suggesting that changing the calibration window already introduces ≈ 0.3–0.4σ of mean shift, with decoding adding 0.04–0.13σ in these measurements. Losses concentrate on multi-hop retrieval (2wikimqa -2.5, hotpotqa -1.7, triviaqa -8.6 at 2%).

Fitting both channels against the output. Value-only edits cannot adjust attention mass, while key averaging changes future logits. SelKV preserves keys and adds a mass channel; KeepKV adds the channel but rescales keys. Both use merge-time formulas, without jointly fitting the channels to attention-output error or validating merges on held-out queries. We instead keep keys fixed and jointly fit the mixture $(1-\theta)v_j + \theta v_i$ and logit bias $\log c$ for each candidate pair within eight positions. We retain edges that reduce output error in every quarter of the calibration window, assign receivers by min-cost matching with a drop option, and apply the plan only if it reduces error on held-out selection queries by $\geq 5\%$ without worsening any query head (Appendix B.2). On Qwen3-1.7B and Llama-3.1-8B (six prompts each, 336 and 384 heads, 64 real decode queries, pooled-SnapKV eviction), it recovers 8% and 11% of eviction error at a 5% budget and 14–16% at 10% (intervals and worst-head results: Appendix B.2).

Deployed at the 5% budget of Table 3, the construction gains +0.70 LongBench (task-clustered CI $[+0.27, +1.15]$) over 400 paired development samples (permutation $p = 0.017$ over samples, 0.023 over task means; seven of eight tasks positive; the sign test remains unresolved at 120/96; Table 5). It also lowers wikitext perplexity by 7.9% (4.13 vs. 4.49) and leaves needle retrieval unchanged. This is the only evaluated merge whose development-split gain over its own base is statistically resolved. After freezing the method, a single run on the 800-sample held-out split yields +0.18 $[-0.27, +0.62]$, with six of eight tasks positive but no resolved gain ($p = 0.46$ over samples; Appendix B.7). Pooling all 1,200 paired samples gives +0.35 $[-0.01, +0.73]$, $p = 0.059$. The development estimate therefore exceeds the gain observed on held-out data. On Qwen2.5-7B-Instruct, which has the least favourable measured key geometry in Table 1, the same construction yields a resolved loss of -0.74 $[-1.26, -0.27]$ (Appendix B.8).

Table 2: Merging operators against their *own* eviction bases under one protocol (Llama-3.1-8B, LongBench dev, 400 paired samples, 5% budget unless stated; paired bootstrap CIs, task-clustered intervals in the text). The "on pooled SnapKV" rows and the last block share one base; CaM, own-selector and 25% rows are paired against their own eviction runs, the consolidation rows against VWS. Columns: what happens to retained *keys*, the attention-*mass* channel a merged slot carries, and what the merge coefficients are *fitted* to. Per-task rows: Table 5; further checks: Tables 8 and 9.

operator	keys	mass	fitted to	Δ vs. base	95% CI
CaM (Zhang et al., 2024), published cadence	fixed	—	formula	−0.14	[−0.25, −0.04]
KVMerger (Wang et al., 2024), own selector (H2O)	averaged	—	formula	−1.35	[−2.50, −0.22]
KVMerger on pooled SnapKV, published $\delta = 0.75$	averaged	—	formula	−0.58	[−1.22, +0.05]
KVMerger on pooled SnapKV, $\delta = 0.5$	averaged	—	formula	−0.06	[−0.61, +0.49]
our consolidation, keys averaged @5%	averaged	bias	window mass	−0.72	[−1.58, +0.09]
our consolidation, keys averaged @2%	averaged	bias	window mass	−1.83	[−3.00, −0.80]
SelKV (Bouyahiaoui et al., 2026) on pooled SnapKV	fixed	bias	formula	+0.02	[−0.26, +0.31]
SelKV, own selector	fixed	bias	formula	+0.10	[−0.40, +0.58]
at its published 25% budget	fixed	bias	formula	+0.31	[+0.01, +0.66]
KeepKV (Tian et al., 2026) on pooled SnapKV	rewritten	votes	EMA score	−9.75	[−11.98, −7.66]
same merges, retained keys left in place	fixed	votes	EMA score	+0.37	[−0.31, +1.15]
at a 25% budget	rewritten	votes	EMA score	−4.24	[−5.66, −2.89]
keys fixed, mass and mixture (§6)	fixed	bias	attn. output	**+0.70**	[+0.14, +1.37]
held out: test split, 800 samples	fixed	bias	attn. output	+0.18	[−0.27, +0.62]
mass channel only ($\theta = 0$)	fixed	bias	attn. output	+0.08	[−0.47, +0.75]
value mixture only ($c = 1$)	fixed	—	attn. output	+0.20	[−0.18, +0.63]

Channel ablations and the output-error surrogate. Removing either channel leaves the gain over the base unresolved: forcing the mixture to zero gives +0.08 [−0.47, +0.75], and disabling the bias gives +0.20 [−0.18, +0.63] (Table 2). The joint configuration exceeds each ablation in direct comparisons (+0.62 [+0.11, +1.16] and +0.50 [+0.10, +0.94], with task-clustered intervals). However, these contrasts do not survive exact randomisation over the eight task means ($p = 0.13$, 0.07), and the interaction is unresolved at +0.42 [−0.31, +1.12]. Thus, only the joint configuration has a resolved gain over the base; the evidence does not establish synergy between the channels. The attention-output metric ranks the individual channels differently: the mass channel alone recovers 6–18% of eviction error, close to the full construction, whereas the mixture alone recovers ≤1.6%. Reducing this surrogate, also used by recent selectors (Goel et al., 2025; Shen et al., 2025), therefore need not rank interventions in the same order as task performance. Simulated decode-time re-calibration recovers a further 1–13 points of output-error reduction (Appendix B.2). Task performance at the 10% budget, where the mechanism-level gain is largest, remains unevaluated.

7 Published operators under a matched protocol

CaM and KVMerger. CaM is a streaming operator defined for one eviction per decode step; at the prefill-end mass eviction its perplexity increases sharply (wikitext ppl 77.7 against 4.42), so we run it at its own cadence on the decode-time evictions only, where it changes 30 of 400 samples and yields −0.14. KVMerger averages the keys of a high-cosine set into its highest-attention retained member; the published method reaches its 35–50% budgets by merging alone, so our port evaluates its operator at a more restrictive 5% budget; at its native 35% budget, where it merges 67% of the evicted tokens, it loses −0.48 [−0.95, −0.04] in one run and −0.40 [−0.85, +0.01] in a rerun against pooled SnapKV at that budget, negative in both runs, resolved in one and not the other (Appendix B.7). At 5% it loses −1.35 on its published selector (H2O), −0.58 on pooled SnapKV at $\delta = 0.75$ and −0.06 at $\delta = 0.5$, is retrieval-neutral, and improves wikitext perplexity by 7.7%, a metric evaluated on continuation queries following the calibration window (Appendix C).

SelKV and KeepKV. For both methods, no released code was available for our evaluation. We implement the published equations and compare each merge-on run with its matching eviction base; disabling merging reproduces that base bit for bit (Appendix C). *SelKV* merges values only, routing an evicted token to the kept token of highest prefill attention *inside its own 32-token bucket*. At a 5% budget that bucket usually holds no kept token, so only 3.8% of evicted tokens are routed (5.4% on pooled SnapKV), with neither gain statistically

resolved: $+0.10$ $[-0.40, +0.58]$ and $+0.02$ $[-0.26, +0.31]$; routing the empty-bucket tokens to the global argmax instead, so that over 99.99% of evicted tokens are routed, turns these into -0.23 $[-0.83, +0.26]$ and -0.37 $[-0.75, -0.04]$ (Appendix B.7). At its published 25% budget it routes 21.5% and estimates $+0.31$ $[+0.01, +0.66]$, unresolved ($p = 0.059$, 0.078). *KeepKV* moves keys: the retained key becomes a rescaled mixture of itself and the evicted key, and the slot carries a $\log p$ bias for the p tokens it now represents. At 5% it merges 8.3% of evicted tokens, 1.20 per retained slot. It changes 330 of 400 samples and yields -9.75 $[-11.98, -7.66]$, with losses on all eight tasks. This is the largest loss we measure. A controlled intervention isolates the contribution of the key rewrite: keeping keys fixed while applying the same value mixture and votes changes the result from -9.75 to $+0.37$ $[-0.31, +1.15]$, a paired contrast of $+10.12$ $[+8.00, +12.38]$ positive on all eight tasks; removing the vote bias or restricting merge targets changes little (Appendix C). This is an Algorithm-1 port on pooled SnapKV, not KeepKV's published configuration (PyramidInfer is absent, the EMA constants and GQA handling are unstated; Appendix C lists every choice). The same pattern holds with a second selector: on SelKV's own selector the port loses -10.40 $[-12.71, -8.24]$ with keys rewritten and $+0.10$ $[-0.81, +0.98]$ with keys fixed (Appendix B.7). This pattern is consistent with the geometric account in §4: rewriting a key affects every subsequent query, whereas ZIP-Merging preserves the score used to construct the merge. On Qwen2.5-7B-Instruct, all evaluated operators have negative point estimates: KeepKV -6.85 with keys rewritten and -0.62, unresolved, with keys fixed; KVMerger -0.66 and the fixed-key construction -0.74, both resolved losses; SelKV -0.45 (Appendix B.8).

Statistical resolution among selectors. For twelve competitive selector configurations (Appendix B.6), every pairwise 95% CI includes zero. The largest point gap is 0.76, although per-task profiles vary by 3–7 points. This indicates limited statistical resolution, not equivalence: with a 7–8-point per-sample spread, detecting $\Delta = 0.3$ with 80% power requires $\approx$5,800 paired samples, exceeding the full English suite's 3,350. Sub-point margins, including the scale of many published merging gains, can therefore be difficult to detect under our protocol. Generating 64 tokens per task (versus the official 128–512 for four tasks) and middle-truncating at 8000 tokens may further compress recall-based differences and leave cumulative merge drift unmeasured. This analysis concerns *selectors*, which disagree on most samples. An operator applied to a fixed selection often changes far fewer samples and thus has lower paired variance: the fixed-key construction's $+0.70$ development gain is resolved at 400 samples, whereas its $+0.18$ held-out gain remains unresolved at 800.

8 Improving selection

Lemma 1 expresses eviction error as $\rho(q)$ times a value gap. This motivates retaining tokens that future queries will attend to, weighted by the potential contribution of their values. Our *value-weighted selection* method (VWS) scores middle tokens using window attention aggregated across KV heads (Li et al., 2024), max-pools along the token axis (kernel 7), and multiplies by $\|v_t\|$ (VATP's factor, Guo et al., 2024). The value norm serves as an inexpensive proxy for the value gap. We allocate slots across layers by greedy water-filling on normalised marginal gains, the exact solution to the separable allocation problem with a floor and cap. Sink and recency windows remain protected, and re-compression occurs every four decode steps (Appendix C). The individual components are established ideas—SnapKV's pooling, VATP's value norm and Ada-KV's budget adaptivity. The decomposition motivates their combination (Figure 2).

Table 3 summarises the results. On LongBench, the differences among VWS, SnapKV and PyramidKV are statistically unresolved; VWS exceeds H2O by $+2.3$ and StreamingLLM by $+3.1$. On retrieval over 100 needle positions it leads pooled SnapKV and PyramidKV at both budgets (paired exact test bounded over the possible pairings: $p \in [0.002, 0.013]$ at 5%, $[0.0005, 0.050]$ at 2%; Appendix B); on Qwen2.5-7B-Instruct it retrieves 0.99/0.78 against pooled SnapKV's 0.93/0.72 and PyramidKV's 0.91/0.60 with an unresolved LongBench difference (-0.12 $[-0.75, +0.52]$; Appendix B.8). At 2% VWS leads pooled SnapKV on the multi-needle suite by $+6.5$ ($[+3.0, +10.5]$, 21 wins to 5) and by $+21$ on multi-query, and performs comparably at 5%. These results suggest that value-weighted selection can better preserve multiple needles when slots are scarce. The available needle-retrieval ablations

Table 3: Main comparison (Llama-3.1-8B-Instruct, 5% KV budget, max_length 8000). LongBench: 800 paired *test-split* samples run once after the method was frozen; every pairwise CI in the top block includes zero. NIAH: 100 needle positions (ten context lengths 1k–8k × ten depths), or [25] a 25-position run; retrieval: four RULER-style tasks × 25 seeded samples; PPL: six wikitext documents; efficiency: H200, 8k prompt, eager attention. —: not run. 2% retrieval and 25-position figures: Appendix B.4.

method	LongBench (test)	NIAH 5%/2%	retrieval @5%	wiki ppl	TTFT (s)	decode tok/s	peak KV (MB)
Dense	28.77	1.00[25]	87.5	3.32	1.40	24.0	1016
SnapKV (global, unpooled)	26.13	0.88[25]/—	—	4.31	—	—	50.1
SnapKV (global, pooled)	26.00	0.88/0.78	83.2	4.49	1.40	17.8	50.1
PyramidKV (global)	25.89	0.82/0.48	74.0	4.32	1.39	18.1	50.1
VWS	25.71	**0.98/0.90**	82.0	4.43	1.44	17.2	**50.0**
H2O	23.38	0.40[25]/—	0.0	4.30	1.43	18.0	50.1
StreamingLLM	22.64	0.20[25]/—	—	3.90	—	—	—

indicate that the value-norm factor is the main contributor (1.00 → 0.84 without it); PyramidKV and unpooled SnapKV lead VWS on perplexity by 2–3%, consistent with the known anti-correlation between perplexity and retrieval; pooled SnapKV and PyramidKV lead it on decode speed by 3–5% and on time to first token by 3% (every compressed method decodes 25–30% slower than dense under this eager-attention harness, Appendix B.4).

9 Discussion and conclusion

Our findings concern *training-free, token-axis merging in frozen rotary-position transformers*, identifying geometric limits on value-only edits and calibration-transfer limits on compensation. Several approaches lie outside this setting. **Trained merging** (DMC; MLA-style architectures) can change key geometry and calibration behaviour through learning. **Cross-layer or cross-scale merging** (Liu et al., 2024; Zhao et al., 2025) exploits a different redundancy axis; **global carriers** (Li et al., 2026) replace per-token slots. Neither is covered by the theorem. AsymKV and GRKV modify keys and report gains we do not test; our controlled intervention concerns local merge formulas, leaving output-fitted key updates open (Appendix C). **Query-dependent coefficients** also escape the theorem, although they may require retaining the information being compressed; Quest's per-query retrieval (Tang et al., 2024) retains the full cache. **Decode-time re-calibration** recovers a further 1–13 points of attention-output error (§6), with end-task benefit untested. Other open directions are a **per-head policy** targeting low-σ heads (Appendix B.3) and **non-rotary models**, including the unmeasured ALiBi and NoPE settings.

Limitations. End-task evaluation covers two models up to 8B (Llama-3.1-8B and Qwen2.5-7B), using eager attention and English benchmarks (LongBench, needle retrieval and perplexity). It does not include a multi-step reasoning suite such as GSM8K. We measure σ on four instruction-tuned rotary models and one small absolute-position model, calibrate the main break-even criterion on two models, and measure decode drift on Qwen3-1.7B. The checks in Appendix B.7 use H100 runs paired with H100 baselines. The KeepKV and SelKV results are from our ports of their published equations, with implementation choices documented in Appendix C.

Conclusion. Repetition in the input does not generally produce mergeable keys in the frozen rotary transformers tested here. Contextualisation and rotary embeddings keep the measured key gap 3–5× above the calibrated threshold for value-only merging. The ported operators that rewrite keys have negative point estimates on both evaluated models (up to −10.40); fixed-key published operators show no resolved gain. Jointly fitting the two compensated channels to attention-output error produces the only resolved development-split gain (+0.70), which falls to +0.18 on held-out data and reverses on a second model. Under this protocol, improving token selection provides the more consistent benefit, particularly for retrieval.

Reproducibility statement

Paired comparisons reuse the corresponding baseline run. We independently re-ran the development baseline ten days later, after the code changes described in this paper, and reproduced every sample score on all eight tasks (`runs/modal/dev8k_fresh/`). All measurements derive from released code (anonymous repository: https://anonymous.4open.science/r/kv-cache-merging-limits-4202): the evaluation framework (`evaluation/`, `baselines/`, `core/`), the mechanism scripts named in Appendix B, the KVMerger and CaM ports, the per-sample score files of every LongBench and retrieval-suite run, the summary files of every NIAH run (per-position files for runs made after the 100-position re-run), and the scripts that regenerate the underlying statistics, reports, figures and macros from those files (Table 9 is assembled from `runs/qwen_checks/REPORT.md` and `experiments/table9_llama_dev.py`). Models are public checkpoints (Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, Qwen2.5-7B-Instruct, Qwen3-1.7B, GPT-2 medium); LongBench and wikitext are public; the retrieval-suite generator and seeds are released. The checks of Appendix B.7 ran on H100-80GB GPUs of an academic cluster; their per-sample files and report are under `runs/h100_checks/`. Compute: about 30 H200-hours for the end-task runs and about 12 H100-hours for those checks and 5 GPU-hours on one RTX PRO 6000 for the Qwen2.5 runs of Appendix B.8 (`runs/qwen_checks/`, `experiments/qwen_checks_report.py`); every σ, oracle and drift measurement runs on a laptop.

Ethics statement

This work analyses inference-time compression of public language models on public English benchmarks; it introduces no new data collection, no human subjects, and no deployment. By characterising when merging provides limited benefit, the study may help guide the allocation of research and evaluation compute.

Use of large language models

LLM-based agents were used, under the authors' direction and review, to run experiments, draft and revise text, and audit the manuscript (proof checking, experiment-integrity and literature audits by a model family different from the drafting model). All derivations, numbers and claims were verified by the authors against the released code and result files; the authors take full responsibility for the content.

References

Yongqi An, Chang Lu, Kuan Zhu, Tao Yu, Chaoyang Zhao, Hong Wu, Ming Tang, and Jinqiao Wang. ReST-KV: Robust KV cache eviction with layer-wise output reconstruction and spatial-temporal smoothing. In *International Conference on Learning Representations*, 2026. arXiv:2605.08840.

Soumia Bouyahiaoui, Manel Kara laouar, Aicha Boutorh, and Mohamed Hadj Ameur. SelKV: Selective KV cache merging with per-token merge-or-drop and attention compensation. *arXiv preprint arXiv:2607.16213*, 2026. URL https://arxiv.org/abs/2607.16213.

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

Wanyun Cui and Mingwei Xu. Homogeneous keys, heterogeneous values: Exploiting local KV cache asymmetry for long-context LLMs. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2506.05410.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024. doi: 10.48550/arXiv.2405.04434. URL https://arxiv.org/abs/2405.04434.

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*, 2025. doi: 10.52202/085713-3773.

Raghavv Goel, Junyoung Park, Mukul Gagrani, Dalton Jones, Matthew Morse, Harper Langston, Mingu Lee, and Chris Lott. CAOTE: KV cache selection for LLMs via attention output error-based token eviction. *arXiv preprint arXiv:2504.14051*, 2025.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention score is not all you need for token importance indicator in KV cache reduction: Value also matters. In *Proceedings of EMNLP*, pages 21158–21166, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1178. URL https://aclanthology.org/2024.emnlp-main.1178/.

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

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*, volume 37, pages 22947–22970, 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/28ab418242603e0f7323e54185d19bde-Abstract-Conference.html.

Xiaolin Lin, Jingcun Wang, Olga Kondrateva, Yiyu Shi, Bing Li, and Grace Li Zhang. CompressKV: Semantic retrieval heads know what tokens are not important before generation. *arXiv preprint arXiv:2508.02401*, 2025.

Yuping Lin, Jiayuan Ding, Yue Xing, Pengfei He, Jiliang Tang, and Subhabrata Mukherjee. A simple plug-in for improving eviction-based KV cache compression. *arXiv preprint arXiv:2605.23258*, 2026.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. MiniCache: KV cache compression in depth dimension for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 139997–140031, 2024. doi: 10.52202/079017-4443. URL https://arxiv.org/abs/2405.14366.

Lianjun Liu, Hongli An, Weiqi Yan, Xin Du, Shengchuan Zhang, Huazhong Liu, and Yunshan Zhong. KVSlimmer: Theoretical insights and practical optimizations for asymmetric KV merging. *arXiv preprint arXiv:2603.00907*, 2026. URL https://arxiv.org/abs/2603.00907.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. ChunkKV: Semantic-preserving KV cache compression for efficient long-context LLM inference. In *Advances in Neural Information Processing Systems*, 2025a. arXiv:2502.00299.

Xin Liu, Xudong Wang, Pei Liu, and Guoming Tang. ZSMerge: Zero-shot KV cache compression for memory-efficient long-context LLMs. *arXiv preprint arXiv:2503.10714*, 2025b.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 37396–37412. PMLR, 2024. URL https://proceedings.mlr.press/v235/nawrot24a.html.

Junjie Peng, You Wu, Haoyi Wu, Jialong Han, Xiaohua Xie, Kewei Tu, and Jianhuang Lai. GRKV: Global regression for training-free KV cache compression in long-context LLMs. *arXiv preprint arXiv:2605.31105*, 2026. URL https://arxiv.org/abs/2605.31105. Accepted to EMNLP 2026 (Main).

Yanlin Qi, Xinhang Chen, Huiqiang Jiang, Qitong Wang, Botao Peng, and Themis Palpanas. ParisKV: Fast and drift-robust KV-cache retrieval for long-context LLMs. In *International Conference on Machine Learning*, 2026. arXiv:2602.07721.

Ziran Qin, Yuxin Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, and Weiyao Lin. CAKE: Cascading and adaptive KV cache eviction with layer preferences. In *International Conference on Learning Representations*, 2025.

Yiqun Shen, Song Yuan, Zhengze Zhang, Xiaoliang Wang, Daxin Jiang, and Nguyen Cam-Tu. LAVa: Layer-wise KV cache eviction with dynamic budget allocation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025. arXiv:2509.09754.

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*, 2025. arXiv:2410.21465.

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*, 2024. arXiv:2406.10774.

Yuxuan Tian, Zihan Wang, Yebo Peng, Aomufei Yuan, Zhiming Wang, Bairen Yi, Xin Liu, Yong Cui, and Tong Yang. KeepKV: Achieving periodic lossless KV cache compression for efficient LLM inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33259–33267, 2026. doi: 10.1609/aaai.v40i39.40611. URL https://doi.org/10.1609/aaai.v40i39.40611. arXiv v1 (April 2025) was titled "KeepKV: Eliminating Output Perturbation in KV Cache Compression for Efficient LLMs Inference".

Zhongwei Wan, Ziang Wu, Che Liu, Jinfa Huang, Zhihong Zhu, Peng Jin, Longyue Wang, and Li Yuan. LOOK-M: Look-once optimization in KV cache for efficient multimodal long-context inference. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4065–4078, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.235. URL https://aclanthology.org/2024.findings-emnlp.235/.

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 *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=HzBfoUdjHt.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive KV cache merging for LLMs on long-context tasks. *arXiv preprint arXiv:2407.08454*, 2024. URL https://arxiv.org/abs/2407.08454.

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

Ning Yang, Chengzhi Wang, Yibo Liu, Baoliang Tian, and Haijun Zhang. CompilerKV: Risk-adaptive KV compression via offline experience compilation. *arXiv preprint arXiv:2602.08686*, 2026.

Jian Yuan, Ziwei He, Haoli Bai, Jingwen Leng, and Bo Jiang. WeightedKV: Attention scores weighted key-value cache merging for large language models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025. doi: 10.1109/ICASSP49660.2025.10889583. URL https://doi.org/10.1109/ICASSP49660.2025.10889583.

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*, volume 235 of *Proceedings of Machine Learning Research*, pages 58840–58850. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhang24n.html.

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*, volume 36, pages 34661–34710, 2023. doi: 10.52202/075280-1506. URL https://arxiv.org/abs/2306.14048.

Yi Zhao, Yajuan Peng, Cam-Tu Nguyen, Zuchao Li, Xiaoliang Wang, Hai Zhao, and Xiaoming Fu. SmallKV: Small model assisted compensation of KV cache compression for efficient LLM inference. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2508.02751.

A Derivations

Lemma 1. Renormalisation gives $\hat{a}_j = a_j/(1-\rho)$, so $o_{\text{evict}} = \bar{v}_A$; also $o = (1-\rho)\bar{v}_A + \rho\bar{v}_E$. Subtracting gives the result. □

Theorem 1. Write $\ell_t(q) = q \cdot k_t/\sqrt{d}$. Exactness for every q means, after cross-multiplying the two (positive) softmax normalisers,

$$\Big[\sum_{j\in A} e^{\ell_j}(v_j+\delta_j)\Big]\Big[\sum_{t\in A\cup E} e^{\ell_t}\Big] = \Big[\sum_{t\in A\cup E} e^{\ell_t} v_t\Big]\Big[\sum_{j\in A} e^{\ell_j}\Big] \qquad \text{for all } q.$$

Both sides are finite sums $\sum_{a,b} c_{ab}\, e^{q\cdot(k_a+k_b)/\sqrt{d}}$ with vector coefficients. The functions $q \mapsto e^{q\cdot s}$ are linearly independent for distinct s, and under general position the frequency $k_a + k_b$ identifies the unordered pair $\{a, b\}$, so coefficients match pairwise. Frequency $2k_j$ ($j \in A$) gives $v_j + \delta_j = v_j$, hence $\delta_j = 0$. Frequency $k_j + k_i$ ($j \in A$, $i \in E$) then gives $v_i = v_j$ for every such pair, so all retained and evicted values equal one common c. By Lemma 1, $\bar{v}_A \equiv \bar{v}_E \equiv c$ and the eviction error was identically zero. (The A–A cross frequencies give $\delta_j + \delta_{j'} = 0$, consistent.) If an evicted key coincides with a retained one, $k_i = k_j$ ($i \in E$, $j \in A$), the frequencies $k_i + k_t$ and $k_j + k_t$ merge and the matching above no longer separates them. Repeating it with the merged frequencies (the remaining keys in general position): frequency $2k_t$, $t \in A \setminus \{j\}$, gives $\delta_t = 0$ as before; frequency $2k_j$ collects $2(v_j + \delta_j)$ on the left against $v_j + v_i$ on the right, so $\delta_j = (v_i - v_j)/2$; frequency $k_j + k_t$ collects $(v_j + \delta_j) + 2(v_t + \delta_t)$ against $v_j + v_i + v_t$, so $\delta_j = v_i - v_t$; and frequencies $k_{i'} + k_t$ and $k_{i'} + k_j$ for any other evicted i' give $v_{i'} = v_t$ and $v_{i'} = v_j + \delta_j$. Together, $\delta_j = (v_i - v_j)/2$ and $v_t = v_{i'} = (v_i + v_j)/2 =: m$ for every other cached token. A value edit alone can therefore absorb a duplicate exactly, but only when every other cached value equals the pair's midpoint: the edit then sets the whole retained cache to m, and the dense output was already the constant m for every query. In this degenerate configuration eviction itself is inexact (its error is $\frac{a_j}{1-a_j}(v_j - m)$, non-zero unless $v_i = v_j$), so the theorem's conclusion does not extend verbatim to coincident keys. Outside it the duplicate is absorbed exactly only with a mass channel: bias $\log 2$ on the slot with value $(v_i + v_j)/2$ reproduces $e^{\ell_j}(v_i + v_j)$ in every coefficient (§6). The case $A = \{j, t\}$, $E = \{i\}$ was also checked symbolically. How close real keys come to this degenerate regime is measured rather than assumed: templated or repeated text is the case most likely to produce near-duplicate keys, and the verbatim column of Table 1 (one passage repeated eight times) keeps the median σ at 0.88–1.37 on the rotary models, far from coincident keys ($\sigma = 0$). □

Proposition 1. With $R = e^X$, $X \sim \mathcal{N}(\mu, \sigma^2)$, the best constant is $c^\star = \mathbb{E}[R]$ with residual $\mathrm{Var}(R)$; not merging is $c = 0$ with error $\mathbb{E}[R^2]$. Their ratio is $(e^{\sigma^2} - 1)/e^{\sigma^2} = 1 - e^{-\sigma^2}$. The fraction is a property of the Gaussian model, not a distribution-free bound: for a gap law with the same variance, lighter tails allow more removal (a two-point gap removes $\cosh^2\sigma/\cosh 2\sigma$, 0.63 at $\sigma = 1$) and heavier tails less: a Laplace gap of the same variance removes 65% at $\sigma = 0.5$ and 42% at $\sigma = 0.6$ against the Gaussian 78% and 70%, and nothing at all once $\sigma \geq 1/\sqrt{2}$, where $\mathbb{E}[R^2] = \mathbb{E}[e^{2X}]$ diverges (the Laplace moment generating function exists only below $\sqrt{2}/\sigma$). □

Criterion (1) (heuristic, as labelled in the text). Every merged token's coefficient error $c - R(q)$ is evaluated at the same decode query, so the errors need not average out across the n merged tokens (the criterion assumes they add like the removed bias, a heuristic);

net positivity then requires the coefficient's relative error $\mathrm{CV} = \sqrt{\mathrm{Var}(R)}/\mathbb{E}[R] = \sqrt{e^{\sigma^2} - 1}$ to be below the relative bias $\|\bar{v}_A - \bar{v}_E\|/\|v\|$, measured at 0.291 across 144 (layer, head) samples; solving gives $\sigma \leq 0.285$. This heuristic neglects value-vector alignment and cross-token cancellation. The unrestricted oracle in §4 makes neither assumption and provides an empirical check on the scale of the predicted improvement.

Second-order residual (§6). With $\bar{\ell} = w\ell_1 + (1-w)\ell_2$ and $x = \Delta\ell$, $\log(e^{\ell_1}+e^{\ell_2}) - \bar{\ell} = \log(1+e^x) - wx$, whose Taylor expansion about $\mu = \mathbb{E}[x]$ has the linear coefficient $\mathrm{sigmoid}(\mu) - w$; the merge weight $w = \mathrm{sigmoid}(\mu)$ removes it and leaves $\frac{1}{2}\mathrm{sigmoid}'(\mu)(x-\mu)^2 \leq (x-\mu)^2/8$, centred to $\frac{1}{2}\mathrm{sigmoid}'(\mu)[(x-\mu)^2 - \sigma^2]$ once the constant b is fitted. The implemented weight is the mass share $\mathbb{E}a_1/\mathbb{E}(a_1+a_2)$ (Appendix C), equal to $\mathrm{sigmoid}(\mu)$ to first order in the gap spread, so the linear term is small but not zero; the construction gates on the measured residual. For equal masses ($\mu = 0$, $w = \frac{1}{2}$), $\log(e^{\ell_1}+e^{\ell_2}) = \bar{\ell} + \log 2\cosh(x/2)$; the bias absorbs $\mathbb{E}[\log 2\cosh(x/2)]$ and the residual is the fluctuation of $\mathrm{logcosh}(x/2) \leq x^2/8$, whose std for $x \sim \mathcal{N}(0, \sigma^2)$ is bounded by $\sqrt{2}\,\sigma^2/8 \approx 0.18\sigma^2$ (tight for small σ; the quadratic proxy over-estimates the std by 19% at $\sigma = 1.1$, where the exact value is 0.17). The value channel is not second order: with $w(q) = \mathrm{sigmoid}(\Delta\ell)$ the pair contributes $wv_1 + (1-w)v_2$, a constant mixture leaves $(w - \bar{w})(v_1 - v_2)$, and std$[w]$ is 0.12, 0.19, 0.22, 0.27 at $\sigma = 0.5$, 0.9, 1.1, 1.5 (numerically; $\sigma/4$ to first order). □

A.1 Key averaging without compensation

This expands the statement in §6. The same algebra as the second-order residual covers D2O, KVMerger and LOOK-M, which average the keys of a merge set S into $\bar{k} = \sum_{i\in S} w_i k_i$ (values likewise) with no mass channel. The merged slot carries the mean logit $\bar{\ell} = \sum_i w_i \ell_i$ and represents the set with mass $e^{\bar{\ell}} = e^{-\Lambda(q)} \sum_{i\in S} e^{\ell_i}$, where $\Lambda(q) = \log \sum_i e^{\ell_i - \bar{\ell}} \geq H(w) \geq \log(1/\max_i w_i)$ by Gibbs' inequality, $H(w)$ the entropy of the merge weights ($= \log|S|$ for equal weights), with equality in the first bound exactly when $\ell_i - \bar{\ell} = \log w_i + H(w)$ for every member, which for equal weights is when all member logits coincide. A compensated merge with bias $b = \mathbb{E}_q\Lambda$ leaves the residual $\Lambda - \mathbb{E}_q\Lambda$, the curvature term of §6; the uncompensated key average leaves Λ itself, i.e., the same residual plus a systematic under-weighting of at least $\log 2$ per equal-mass pair relative to the unmerged slots that share its softmax. Its mean squared log-mass residual therefore exceeds the compensated construction's by $(\mathbb{E}_q\Lambda)^2 \geq (\log 1/\max_i w_i)^2$, which reduces to $(\log|S|)^2$ for equal weights. Thus, for a fixed merge set, compensation lowers the mean squared log-mass residual, although it need not lower the residual for every query. This comparison does not imply a corresponding ordering in output error or task score; Table 7 reports the end-task results of the compensated construction.

B Experimental details

Protocol. Unless otherwise stated, experiments use Llama-3.1-8B-Instruct with eager attention and greedy decoding. We generate 64 new tokens for every LongBench task and 96 for the retrieval suite. The official LongBench configuration uses 128 tokens for qasper and samsum and 512 for gov_report and multi_news; we shorten generation to reduce compute, with the implications discussed in §7. Budgets are fractions of the prefill length, with 4 sink tokens and a recency window equal to 10% of the budget. Comparisons are paired and reported with 20,000-resample bootstrap CIs, two-sided sign tests and paired permutation tests. A LongBench gain is termed *resolved* when its bootstrap CI excludes zero and both permutation tests—over samples and over the eight task means—give $p < 0.05$. The development split uses samples 0–50 of eight English tasks; the test split uses samples 50–150 and is evaluated once after the method is frozen. We release all per-sample scores, configurations and exact commands.

σ measurement. Post-RoPE keys and queries captured by forward pre-hooks; oracle routing pairs each of 192 sampled evicted tokens per head with its globally best anchor at a 15% budget; probes are the final 128 query positions; every fourth layer, all KV heads. Anchors are evenly spaced, which under RoPE gives every evicted token a nearby anchor and

therefore biases σ downward, against our conclusion. Per-head figures and the RoPE-removed same-token σ (Figure 1) use the same protocol with every fourth layer on Llama, Mistral and Qwen2.5 and every second layer on Qwen3; the later key of a pair is re-rotated onto the earlier position with the model's own cosine and sine tables, which cancels RoPE exactly for identical content (`experiments/sigma_perhead_derot.py`, `experiments/sigma_nonrope.py`). The Qwen3-1.7B sweeps (every ninth layer): budgets 5% → 95% move median σ 1.40 → 0.98 and the fraction under break-even 0.05% → 0.40%; context lengths 1k→16k saturate σ at ≈1.18; text types give prose 1.16, code 1.22 and one passage repeated verbatim 8× 1.21. GPT-2 medium is measured with its keys as used (no rotation), all 24 layers, 896-token prefills; it is a 355M-parameter model with a 1024-token context, and larger absolute-position or ALiBi models are not measured here. Decode-time σ and the drift statistic (§6) use the post-RoPE queries of 64 decode steps on retrieval, continuation and summarisation prompts (context 2500; six prompts per kind for continuation and summarisation, two for retrieval; greedy decoding with repeated trigrams blocked, since plain greedy decoding degenerated on two of six prompts and supplied the extremes of an earlier version of this measurement); the within-prefill nulls use the 64 positions preceding the calibration window (mean shift 0.31–0.40 σ) and the two halves of the window (0.25–0.33 σ).

Class-A oracle. Anchors are the top-mass tokens of the last 128 prefill queries under causal attention at a 15% budget; the ridge edit $\delta \in \mathbb{R}^{m\times d}$ is fitted on the W prefill queries preceding the cut (both query heads of a GQA group) and evaluated on the 128 positions after it, which read only the prefill cache; every fourth layer, all KV heads, two wikitext documents of 2560 tokens (112 heads); medians over heads. Splitting the probe by query head rather than by position, without a causal mask, inflates the held-out figure and makes it grow with calibration size (19% → 34%); we report the position split. The anti-causal variant fits the same edit on the decode queries of a prompt themselves, at the 5% budget (121 unknowns per output dimension) with 256 decode steps (repeated trigrams blocked; 512 rows over both query heads), evaluated on the other half of the steps under a first/second-half and an interleaved split; two retrieval and two continuation prompts of 2500 tokens. The selector comparison reports, per head, the held-out eviction error and the edited error normalised by $\|o\|^2$ over the test rows for anchors chosen by the last-128-query mass (window rule) and by the mass accumulated over all W calibration queries (H2O rule), $W \in \{128, 1024, 2048\}$.

Break-even constants. $\|\bar{v}_A - \bar{v}_E\|/\|v\|$ at a 15% budget with all-query top-k anchors: Qwen3-1.7B 0.291 (144 heads), Llama-3.1-8B 0.324 (112 heads; 0.275 with window anchors), Mistral-7B 0.280 (168 heads; 0.277); the implied σ_{be} are 0.285, 0.316 (0.270) and 0.275 (0.272). The fractions of Table 1 are insensitive to the constant: under its own threshold (0.316) 0.64% of Llama's evicted tokens clear break-even under oracle routing (0.39% at 0.285; Qwen3-1.7B: 0.65% vs. 0.44%); Mistral's own gap was not re-measured for this sensitivity check.

Matched-protocol ports. CaM and KVMerger are evaluated on the LongBench dev split (400 paired samples, 5% budget), NIAH (25 positions) and six wikitext documents, each against the identical selection without the operator. The 100-position NIAH run uses ten context lengths from 1k to 8k and ten depths; per-position results were not stored for that run, so the paired exact test in §8 is bounded over all pairings consistent with the two accuracies: the minimum number of discordant positions (the weaker method's successes a subset of the stronger's) gives the smallest p, the maximum (disjoint failure sets) the largest; for 98 vs 88 of 100 the range is 0.002–0.013, for 90 vs 78 it is 0.0005–0.050.

Per-task LongBench results (test split). Table 4 gives the per-task scores behind Table 3; the eight tasks are qasper, multifieldqa_en, hotpotqa, 2wikimqa, gov_report, multi_news, triviaqa and samsum (F1 for QA, ROUGE-L for summarisation). At max_length 8000 the fraction of samples middle-truncated is 7%, 40%, 88%, 24%, 47%, 4%, 34% and 30% respectively, showing that truncation affects a substantial fraction of hotpotqa and gov_report samples before cache compression. We abandoned an earlier 4096-token protocol because it truncated 70–96% of samples.

Consolidation. Window rows (last 32 queries, GQA groups folded) are retained alongside the selection scores; the per-pair residual and bias are computed from log-attention columns, in which the softmax normaliser cancels; the decode-time bias is injected as an additive attention

Table 4: Per-task LongBench scores on the test split (samples 50–150 of each task, 800 paired), Llama-3.1-8B-Instruct, 5% budget; Δ and bootstrap CI against pooled SnapKV. †An earlier configuration of this project: evicted values are folded into up to four retained anchors by a near-hard entropic transport plan (a class-A merge), on a selector with the regime detector and pyramid budgets. It differs from the other rows in more than the merge, so it is not an operator ablation. The *regime detector* is a per-layer switch: when window attention is not markedly more peaked than all-query attention (ratio of normalised entropies $\geq$0.73) it treats the prompt as a continuation and selects tokens per head from all-query attention, half by rank and half by a k-means cover of the keys. VWS runs without it, because on Llama the statistic does not separate question prompts from continuation prompts.

method	2wiki	gov	hotpot	mnews	mfqa	qasper	samsum	trivia	avg	Δ [CI]
Dense	12.9	15.5	14.5	16.3	37.3	20.0	29.5	84.1	28.77	+2.77 [1.98, 3.57]
SnapKV (global, unpooled)	10.0	12.7	13.2	12.4	34.1	14.1	29.6	83.0	26.13	+0.13 [−0.44, 0.72]
SnapKV (global, pooled)	11.0	12.0	13.7	12.4	32.0	14.2	30.1	82.7	26.00	(ref)
VWS − water-filling	10.9	12.1	12.6	12.8	32.3	14.8	29.5	82.9	25.96	−0.04 [−0.54, 0.47]
VWS − pooling	9.8	12.8	12.2	12.1	33.5	14.3	29.8	82.7	25.90	−0.10 [−0.67, 0.48]
PyramidKV (global)	10.2	12.7	12.9	12.4	32.4	13.9	29.5	83.1	25.89	−0.10 [−0.67, 0.46]
VWS	10.1	12.2	12.4	12.0	32.9	14.4	29.3	82.3	25.71	−0.29 [−0.79, 0.19]
VWS − value weighting	10.3	11.9	11.7	12.0	31.9	13.2	28.9	82.9	25.36	−0.64 [−1.15, −0.14]
VWS + regime detector	10.0	13.6	13.7	15.5	27.7	15.9	25.6	77.7	24.97	−1.03 [−1.93, −0.17]
transport merge†	10.7	14.2	12.4	15.8	27.3	14.0	24.8	77.0	24.51	−1.48 [−2.45, −0.55]
H2O (per-head)	10.2	14.9	12.3	13.0	26.8	15.1	20.7	74.0	23.38	−2.62 [−3.62, −1.66]
StreamingLLM	8.5	12.1	10.4	11.6	22.2	9.5	31.5	75.3	22.64	−3.35 [−4.36, −2.38]

mask built after any pending maintenance compression, so mask and cache always agree. Identical-pair losslessness and channel alignment through twenty decode-time compressions are unit-tested (88 assertions across the suite).

B.1 Class-A oracle: transfer, selector-conditional and anti-causal checks

We use the class-A oracle protocol described above and provide the detailed results underlying §4. With $W = 128$, the ridge edit removes 66% of squared eviction error in sample and 21%–28% on held-out queries ($\lambda \in [10^{-3}, 1]$). Increasing the calibration window to $W \geq 512$ lowers the held-out median to 10–13%. The corresponding means are strongly negative (-5 to -9), indicating large errors in some heads. These measurements do not exclude estimators that generalise better than ridge, but show that additional prefill calibration does not improve this estimator's transfer.

Two checks examine more favourable conditions. First, the recovered fraction depends on the selector. Anchors chosen by attention accumulated over all W calibration queries (the H2O rule) allow a larger fraction of a larger initial error to be removed, but the edited cache never beats plain window-attention eviction in absolute error. Normalised by $\|o\|^2$, held-out error with window anchors is 0.113 before the edit and 0.065–0.086 afterwards. With accumulated-attention anchors, it is 0.19–0.27 before and 0.12–0.20 afterwards. The edited H2O cache therefore remains less accurate than the unedited window cache. Second, an anti-causal oracle fits on the answer's own decode queries, which are unavailable at deployment. At the 5% budget, the fit is determined (121 unknowns, 256 fit rows, interleaved split, repetition-free decoding of 256 tokens). It removes 38–59% of error on retrieval and continuation prompts ($\lambda \in [10^{-3}, 1]$), or 35–50% with a first-half/second-half split. Splitting the probe by query head rather than by position, without a causal mask, instead gives 19% rising to 34% with calibration size. We report the position-split, causally masked results as the deployment-faithful estimate.

B.2 The fixed-key intervention

Protocol. We evaluate three wikitext continuation prompts and three needle prompts of ≈2,500 tokens, sampling every fourth layer and every KV head. The retained set is the pooled-SnapKV set of the Table 3 baseline (sink 4, last 64 positions kept verbatim, top-k middle by the last 32 queries, kernel 7), shared across the layer's KV heads, at 5% and 10% of the prefill. For each evicted, unprotected token, we consider up to four nearest

Table 5: Per task: LongBench dev scores of pooled SnapKV (Llama-3.1-8B, 5% budget, 50 samples per task) and the paired change each operator makes to that identical selection. "keys fixed" is the construction of §6; its two channels are then ablated. The last column counts the samples the full construction changes at all (KeepKV changes 330, SelKV 163). Bootstrap CIs for the means are in Table 2. The triviaqa row rests on one changed sample for the full construction; dropping that task leaves it at +0.56 and KeepKV at −6.23.

task	pooled SnapKV	keys fixed, mass+mixture	mass only	mixture only	keys averaged	KeepKV	SelKV	samples changed
2wiki	9.65	+0.68	-0.36	-0.57	+0.14	-5.07	+0.06	18
gov	12.10	+0.51	+0.61	+0.86	-0.14	-3.89	+0.04	48
hotpot	15.98	+0.23	-0.75	+0.06	-1.70	-9.51	+0.09	18
mnews	12.06	-0.40	-0.37	-0.46	-0.52	-3.62	-0.86	47
mfqa	32.07	+0.58	-0.96	+0.54	-1.46	-14.00	+0.57	24
qasper	13.69	+0.76	+0.96	+0.16	-0.56	-5.54	+0.23	33
samsum	26.85	+1.55	-0.27	+0.02	-1.23	-2.01	-0.08	27
trivia	77.77	+1.73	+1.81	+1.00	+0.86	-34.38	+0.08	1
mean	25.02	**+0.70**	+0.08	+0.20	-0.58	-9.75	+0.02	216

retained neighbours within eight positions, provided no sentence boundary separates the pair (a boundary is a token whose decoded string ends in ".", "!" or "?", optionally followed by whitespace; no entity or clause detection). For each edge $i \to j$, the keys remain fixed, the receiver's value becomes $(1-\theta)v_j + \theta v_i$ and its slot receives a logit bias $\log c$. On the *fit* rows (the first 32 positions of the calibration window, using all query heads in the group), we first estimate $c^\star = \sum_q a_j(a_i{+}a_j)/\sum_q a_j^2$ from the pair's mass, where a denotes dense attention. We then search a nine-point logarithmic grid over $c \in [c^\star/2, 2c^\star]$ (clipped to $[1, 4]$) with the conditional least-squares optimum $\theta(c)$, keeping the c with the smallest total squared error of the full attention output over the retained set (the joint fit; "mass-fit c" below is the mass value alone; identical keys take the exact $c = 2$, $\theta = \frac{1}{2}$). The edge cost is the largest, over four contiguous time blocks of the fit rows, of the mean change in squared output error, and only edges that lower it in every block are kept. Receivers are assigned by maximum-weight bipartite matching with a zero-cost drop option (each evicted token at most once, each receiver at most once), or greedily by most-negative cost for the comparison. The plan is applied at the scale $\gamma \in \{0.25, 0.5, 1\}$ that most reduces the error on the next 16 positions (*select* rows) if that reduction is ≥5% and no query head of the group is worse, otherwise, the plan is rejected.

Evaluation and ablations. We use 64 greedy decode steps from the dense model, with repeated-trigram blocking. Queries attend causally to the compressed prefill cache and the generated tokens' own keys and values. For each head, the metric is the mean of $\|o' - o\|^2$ over the mean of $\|o_{\text{evict}} - o\|^2$ per head, over all 64 steps and over the last 16 ("late" steps, held out from both prefill and online calibration, but included in the in-sample reference). Ablations use the same pairs or candidate sets and the same gate: no bias ($c = 1$, θ refitted); bias only ($\theta = 0$, joint c); random pairing among the same candidates with the matching's proposed count (applied counts differ after the gate); greedy instead of matching. We also evaluate two calibration interventions: *online*, fitted on decode steps 1–32 with γ chosen by the same gate on steps 33–48 (what a method that re-calibrates during decoding could do); and the in-sample *decode-fitted reference*, fitted and γ-selected (minimum gain 0) on all 64 steps — an achieved score of this construction, not a bound. Confidence intervals resample whole prompts because heads within a prompt share its text, retained set and decode (prompt-clustered CIs at 5%: $[4.2, 11.2]$ on Qwen3-1.7B and $[9.0, 12.5]$ on Llama-3.1-8B; the worst head is 1.4–4.8× worse than eviction, Table 6).

Implementation and validation. The implementation is in `core/otkv_v8.py`, with 77 unit checks in `experiments/test_otkv_v8.py` — including a brute-force per-edge grid search that the joint (c, θ) fit must reproduce — and `experiments/v8_offline.py`, whose self-test reruns the whole protocol against an independent per-edge implementation; an adversarial code review by a second model family preceded the runs. The resulting checks

enforce a shared fitter across conditions, the best feasible γ, equal proposed counts for random pairing, inclusion of generated tokens in the decode cache, prompt-clustered intervals, and reporting of applied rather than proposed merge counts. The joint fit and online intervention were added following a manuscript review by a second model family. The exact-branch and additive-cost algebra was checked independently (`docs/otkv_v8_math_check.md`); per-edge costs form an additive surrogate, whereas the joint softmax couples their effects. Validation on the selection rows checks the combined plan against this coupling. The deployable version (`core/otkv_v8_cache.py`: the same merge at prefill end on the pooled-SnapKV cache of Table 3, from the stored attention rows of the last 64 prefill queries, bias injected at decode through the attention mask) reproduces the harness plan on random tensors (18 checks) and is the method evaluated end to end in §6 and Table 5; its two channels are ablated there by `force_theta=0` (mass only) and `force_c=1` (mixture only), which change nothing else about the selection, the candidates or the gate. All four LongBench columns of Table 5 share one baseline run, which we re-executed from scratch after this paper's code changes and which reproduced sample for sample.

Table 6: The fixed-key intervention: reduction of the eviction error on the 64 decode rows (one minus the ratio of mean $\|o' - o\|^2$ to mean $\|o_{\text{evict}} - o\|^2$, mean over heads, in %), over all 64 steps / over the last 16 steps (held out from the online calibration too), per model and budget; six prompts per model, 336 (Qwen3) and 384 (Llama) heads; prompt-clustered CIs in the text. The lower block describes the prefill-window plan.

construction	Qwen3-1.7B		Llama-3.1-8B	
	5%	10%	5%	10%
prefill-window fit, matching	+7.6 / +6.8	+14.3 / +10.3	+10.7 / +9.2	+15.6 / +16.2
greedy instead of matching	+7.5 / +6.8	+14.0 / +9.9	+10.5 / +9.1	+15.4 / +15.8
no bias channel ($c = 1$)	+0.2 / +0.1	+0.5 / +0.3	+0.6 / +0.5	+1.6 / +1.4
bias only ($\theta = 0$)	+6.2 / +5.6	+17.0 / +15.9	+8.8 / +7.7	+18.1 / +17.3
random pairing, same proposed count	+6.9 / +6.0	+12.5 / +9.1	+8.1 / +7.4	+15.3 / +14.1
c from the mass fit only (no joint search)	+3.2 / +2.5	+6.8 / +5.2	+4.0 / +3.3	+7.1 / +7.1
online: fit on decode steps 1–32, γ on 33–48	+11.0 / +8.1	+26.9 / +23.6	+14.9 / +12.8	+27.4 / +24.2
decode-fitted reference (in sample)	+13.4 / +11.5	+29.1 / +27.7	+17.1 / +15.3	+30.3 / +29.6
applied merges per head (prefill fit)	15	63	23	73
applied evicted mass (prefill fit)	2.6%	7.2%	2.4%	5.8%
heads >5% worse (prefill fit)	1%	6%	3%	5%
worst head, error ratio (prefill fit)	2.22	4.76	1.38	4.79

B.3 Per-head structure of σ

The head-level measurements underlying §5 show substantial variation around the head-averaged results (Figure 1, right): per-head median σ runs from 0.42 to 1.70 on Llama-3.1-8B (10th–90th percentile 0.70–1.35) and from 0.37 to 3.23 on Qwen3-1.7B (0.88–1.63). One head per model has more than 5% of its evicted tokens under break-even (a layer-0 head on Llama, a layer-4 head on Qwen3), so a per-head merging policy could act there; but weighting tokens by the attention mass the probe queries place on them, the mergeable share of evicted mass is 0.5% on Llama and 0.3% on Qwen3. Mistral's 5.1% is the largest mergeable fraction among the rotary models and is concentrated in a small number of heads: four of its 64 measured heads, all in layer 0, have median σ of 0.15–0.29 and 42–97% of their evicted tokens under break-even, while the other 60 heads look like Llama's (Figure 1, right); weighted by attention mass those four heads hold 5.9% of the evicted mass. Its own break-even constant is 0.25–0.27 (centroid gap 0.26–0.28), below the reference. Qwen2.5-7B has no such head (0.09% under break-even, minimum per-head median 0.87). A per-head policy that merges only in such heads therefore remains a possible, though limited, option on some models (§9).

B.4 Reading notes for Tables 1, 7 and 3

Table 7: Compensated key-averaging consolidation, paired against the identical selection without merging (Llama-3.1-8B, LongBench dev, 400 paired samples; bootstrap CIs). It preserves retrieval and improves continuation metrics, but reduces long-context QA performance at 2%; the change at 5% is unresolved (the LongBench rows also appear in Table 2); the cosine-gate row is discussed below.

	Δ LongBench	95% CI	NIAH
consolidation @5% budget	−0.72	[−1.58, +0.09]	1.00 (unchanged)
consolidation @2% budget	−1.83	[−3.00, −0.80]	—
gate on cosine instead of $\hat{\sigma}$ (@5%)	−0.73	[−1.68, +0.15]	—
wikitext ppl (@5%)	4.419 vs. 4.431 unmerged (−0.3%)		
Qwen3-1.7B local KL / ppl	−2.5% / −2.0%		

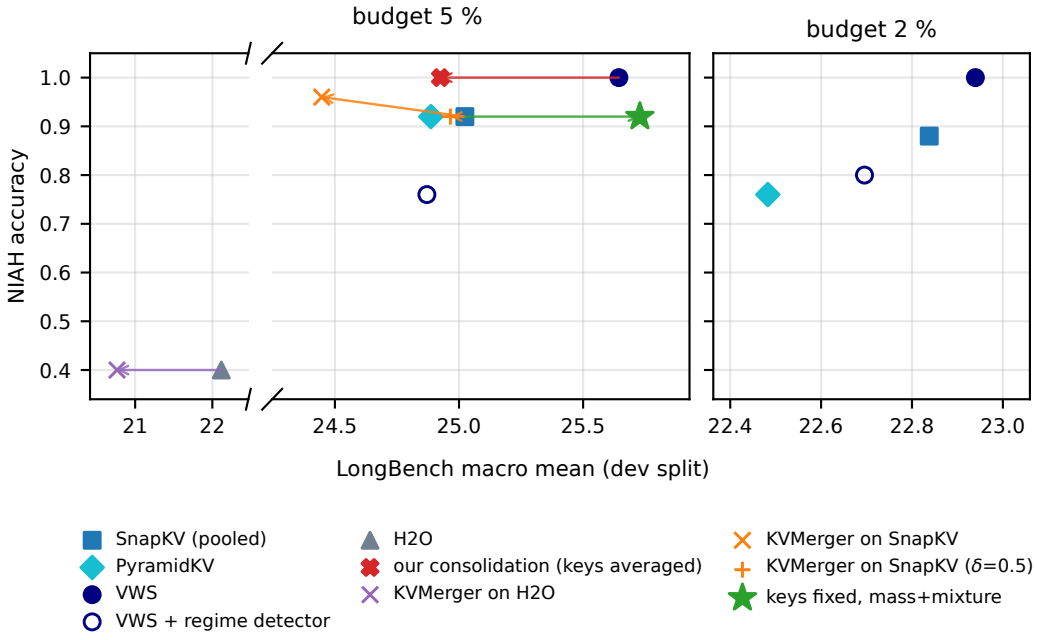

Figure 2: LongBench macro mean (dev split, 400 paired samples) against needle-retrieval accuracy at 5% and 2% budgets, for the configurations with both measurements: selectors (blue and grey) and merging operators (5% only; crosses and star, each joined by an arrow to its own eviction base; KVMerger at δ=0.5 moves its base by only −0.06 and sits on the pooled-SnapKV square). The 5% panel's x-axis is broken between the H2O pair and the rest. CaM, the cosine-gated consolidation, the two channel ablations of §6 and the KeepKV and SelKV ports have no needle run and are therefore absent. Of the merging operators only the fixed-key construction of §6 (green star) moves its base to the right; the competitive selectors span <1 LongBench point while retrieval separates them.

Table 1. The value-only break-even fraction never exceeds 5.1% on a rotary model; verbatim repetition does not create mergeable keys; and σ between positions holding the *same token id* grows with positional distance (0–8 → >512). "≤ 0.9" is a looser reference: at $\sigma = 0.9$ the second-order residual of a compensated pair (§6) is ≈0.13, under break-even, so this column provides a looser reference for the tokens that compensation might make viable. The last row is an absolute-position model (no RoPE, 896-token prefill), where the picture changes in degree: ten percent of evicted tokens clear break-even, and repetition raises it to 26%.

Table 7. Gating on cosine instead of $\hat{\sigma}$ gives an indistinguishable result (−0.73 vs. −0.72): adjacent pairs are already the lowest-σ pairs (Table 1), so the two gates have little scope

to differentiate candidates. This comparison provides no evidence that the choice of gate explains the loss.

Table 3. NIAH: the top three methods were re-run over 100 positions after the 25-position run saturated (VWS 1.00/1.00, pooled SnapKV 0.92/0.88, PyramidKV 0.92/0.76 at 5%/2%; unpooled SnapKV 0.88, H2O 0.40, StreamingLLM 0.20 at 5%, which the table marks [25]); the 100-position figures are the ones the text uses. Component ablations of VWS were run at 25 positions and 5% only: without the value norm 0.84, without token-axis pooling 0.96, with uniform layer budgets instead of water-filling 1.00 (`runs/modal/niah8k`); the 25-position grid saturates, so these ablations identify a contribution from the value norm but do not separate the effects of the other two components, and the 100-position and 2% ablations were not run. Blank cells: unpooled SnapKV and StreamingLLM were not profiled, and StreamingLLM's retrieval suite and the 2% NIAH of the weaker methods were not run. Speed: the columns compare the compressors with each other under one eager-attention harness that re-compresses every four steps; an earlier profiling pass of the same harness found every compressed method ≈30% slower than dense at a 4k prompt and dense unable to run at 16k, where the compressed methods decode at ≈26 tok/s. Retrieval suite at 2%: VWS 73.8 vs. pooled SnapKV 67.2, paired CI on the difference $[+3.0, +10.5]$; multi-query 63 vs. 42.

B.5 Two protocol pitfalls

Beyond the resolution argument of §7, we identified two implementation details that can change method rankings by more than the reported margins: budget accounting and baseline fidelity. *Budget accounting*: a method that re-compresses every N decode steps can let the cache grow to budget + N between compressions, so it holds up to N more slots than one that reserves N slots and keeps its peak *at* budget; an earlier version of our own selector scored ≈0.6 higher this way, matching its apparent advantage. Every method evaluated here uses the reserve. *Baseline fidelity*: the widely re-implemented SnapKV without its token-axis pooling loses 0.92 → 0.88 NIAH.

B.6 Statistical resolution among selectors

We compare the window-scored selectors without a regime detector or value-weight ablation: pooled SnapKV, PyramidKV and VWS on the dev split at 5% and 2%, plus unpooled SnapKV and VWS's pooling and water-filling ablations on the 800-sample test split — twelve configurations. Every pairwise difference among them has a 95% CI that includes zero; the largest point gap is 0.76 (dev, 5%) and the six test-split configurations span 0.42, while per-task profiles swing by 3–7 points; removing value weighting (-0.64) or adding the regime detector (-1.03) produces a detectable difference (Table 4). Two protocol choices affect the interpretation: every task generates 64 tokens (the official configuration generates up to 512) and middle truncation at 8000 tokens cuts most hotpotqa and gov_report contexts (Appendix B), so the protocol may itself compress the differences it fails to resolve. Nondetection does not establish equivalence (the VWS–SnapKV interval is $[-0.79, 0.19]$): with a 7–8 point per-sample spread, a 95% interval of half-width 0.3 needs ≈2,800 paired samples and 80% power at $\Delta = 0.3$ needs ≈5,800, against the 800 of our test split and the 3,350 of the full English suite. Sub-point LongBench margins in this family, including the scale of many published merging gains, can fall below the benchmark's resolution under our protocol, and two protocol pitfalls (budget accounting and baseline fidelity; Appendix B.5) reorder tables by more than the claimed margins. This resolution analysis concerns *selectors*, which disagree on most samples and so carry the full spread; an operator applied to a fixed selection leaves most samples untouched, which is why the +0.70 of Table 2 resolves at 400 samples where the same *selector* margin would not, and why the same operator's +0.18 on 800 held-out samples does not (Appendix B.7).

B.7 Held-out, second-selector, fallback and native-budget checks

Four checks were run after the first version of this paper, on H100-80GB GPUs of an academic cluster, under the protocol of Table 2 (bf16, eager attention; `runs/h100_checks/`,

`experiments/h100_checks_report.py`). Pooled SnapKV re-run on the H100 differs from the H200 files on 79 of 400 dev samples and about a fifth of the test samples, with task means within 0.1 (25.12 vs. 25.02 on dev), so every standalone row of Table 8 is paired with a base produced on the same hardware as its method run (the pooled row combines H200/H200 development pairs with H100/H100 test pairs). Eviction-only runs are bit-identical across H100 jobs; KeepKV and the fixed-key construction do not use the atomic accumulation responsible for the noise below, whereas the KVMerger and SelKV merges accumulate with CUDA `index_add_` (float atomics) and carry run-to-run noise, measured directly for KVMerger by a rerun ($+0.08$ $[-0.11, +0.29]$ against itself, 50 of 400 samples changed). *Held out.* The fixed-key construction on LongBench samples 50–150 of the eight tasks (800 paired, run once after freezing): $+0.18$ $[-0.27, +0.62]$, 226 wins to 238 losses, six of eight tasks positive, permutation $p = 0.46$ over samples and 0.24 over task means (26.12 vs. 25.94). Pooled with the 400 development samples (`experiments/pooled_devtest.py`: seed 0, 200,000 resamples of the concatenated paired differences, as for Table 2): $+0.35$ $[-0.01, +0.73]$, $p = 0.059$ and 0.047. *Second selector.* KeepKV ported onto SelKV's own selector (`keepkv;selector=selkv`; with merging off it is bit-identical to `selkv;selector=selkv;merge=false`, the base, 25.12): keys rewritten -10.40, keys fixed $+0.10$, direct contrast $+10.50$ $[+8.32, +12.79]$ positive on all eight tasks; 7.6% of evicted tokens merged (8.3% on pooled SnapKV). *Fallback.* SelKV with an evicted token whose 32-token bucket holds no kept token routed to the kept token of highest prefill attention anywhere in the head's kept set (`fallback=global`; same gate and compensation): over 99.99% of evicted tokens are now routed (5,047 and 9,729 fallback misses out of 5.9×10^8; 5.4% and 3.8% before), and the operator loses -0.37 on pooled SnapKV (CI excludes zero, $p = 0.035$ over samples but 0.078 over task means, one task positive) and -0.23, unresolved, on its own selector. *Native budget.* KVMerger and pooled SnapKV both at a 35% budget: 66.8% of evicted tokens are merged at prefill (1.7 million merge sets over 400 samples), and the operator loses -0.48 $[-0.95, -0.04]$ ($p = 0.038$, 0.047; resolved) in the first run and -0.40 $[-0.85, +0.01]$ ($p = 0.067$, 0.070) in the rerun, with run-to-run variation attributable to the atomic accumulation described above; the base scores 27.95.

Table 8: H100 checks (Llama-3.1-8B-Instruct, LongBench, 5% budget unless stated; same-hardware bases: standalone rows H100 method against H100 base, the pooled row H200/H200 development pairs plus H100/H100 test pairs). Δ and paired bootstrap CI as in Table 2; perm-s / perm-t: permutation p over samples / over the eight task means; changed: samples whose score moved.

comparison	base	n	Δ	95% CI	perm-s	perm-t	tasks>0	changed
keys fixed, mass and mixture, test split	pooled SnapKV	800	+0.18	[−0.27, +0.62]	0.46	0.24	6/8	464
dev and test pooled	pooled SnapKV	1200	+0.35	[−0.01, +0.73]	0.059	0.047	6/8	680
KeepKV on SelKV's selector, keys rewritten	merge off	400	−10.40	[−12.71, −8.24]	<0.001	0.008	0/8	335
same merges, keys fixed	merge off	400	+0.10	[−0.81, +0.98]	0.82	0.88	3/8	254
keys fixed vs. rewritten	—	400	+10.50	[+8.32, +12.79]	<0.001	0.008	8/8	336
SelKV, global fallback, on pooled SnapKV	pooled SnapKV	400	−0.37	[−0.75, −0.04]	0.035	0.078	1/8	162
SelKV, global fallback, own selector	merge off	400	−0.23	[−0.83, +0.26]	0.48	0.48	5/8	153
KVMerger @35%	pooled SnapKV @35%	400	−0.48	[−0.95, −0.04]	0.038	0.047	2/8	194
rerun	pooled SnapKV @35%	400	−0.40	[−0.85, +0.01]	0.067	0.070	2/8	188
rerun vs. first run	—	400	+0.08	[−0.11, +0.29]	0.44	0.38	4/8	50

B.8 Second model: Qwen2.5-7B-Instruct

We repeat the protocol of Table 2 on Qwen2.5-7B-Instruct (28 layers, 4 KV heads, GQA groups of 7), using one RTX PRO 6000 Blackwell (96 GB, bf16, eager attention, torch 2.11.0, transformers 5.7.0). The LongBench development split contains 400 paired samples at a 5% budget; each method is compared with a baseline run on the same GPU (`runs/qwen_checks/`, `experiments/qwen_checks_report.py`). With merging disabled, the KeepKV port reproduces the pooled-SnapKV baseline bit for bit (400/400).

Absolute performance under this harness. Qwen scores substantially below Llama under this protocol (dense 19.50 against 28.77 on Llama's test split; triviaqa 42.8 against 84.1). The cause remains undiagnosed. The run stored scores but not generated answers, so we cannot distinguish the effects of shortened generation, truncation and answer formatting; the pre-run plan had expected higher Qwen scores. Compression does not explain the low absolute scores: dense Qwen itself scores 19.50 and exceeds pooled SnapKV by +2.59

[+1.61, +3.63], compared with +3.84 on Llama's development split. The paired within-Qwen differences remain interpretable under this harness, but these runs do not establish typical Qwen performance or support absolute cross-model comparisons. Middle-truncation counts are similar to Llama's development split (hotpotqa 46/50 for both; gov_report 38/50 against 35/50).

Merging results. Qwen2.5 has the least favourable measured key geometry in Table 1: median σ is 1.38, with 0.1% of evicted tokens below break-even. That measurement uses wikitext at a 15% budget, whereas the task evaluation uses LongBench at 5%. The operator results in Table 9 are qualitatively consistent with the geometric account, but two models do not establish a monotone cross-model relationship. Relative to Llama, the fixed-key estimate falls by 1.45 points, KVMerger's changes by only -0.08, and rewritten KeepKV is 2.9 points less damaging. The fixed-key construction yields -0.74 $[-1.26, -0.27]$ ($p = 0.002$ over samples, 0.039 over task means; six of eight tasks negative; 260 samples changed). Its merge-counter files were not written, but applied-edge counts are recorded in the run log. KeepKV with rewritten keys yields -6.85, with losses on all eight tasks, while merging 22% of evicted tokens. Fixing its keys yields an unresolved -0.62; the direct contrast is +6.23, positive on all eight tasks. KVMerger merges 22% of evicted tokens and yields a resolved -0.66, whereas Llama's -0.58 is unresolved. SelKV routes 4.9% of evicted tokens and yields -0.45, with $p = 0.18$ over task means.

Selection and retrieval. Selector differences remain unresolved (VWS -0.12, PyramidKV -0.45). Over 100 needle positions, dense retrieves 1.00, VWS 0.99/0.78, pooled SnapKV 0.93/0.72 and PyramidKV 0.91/0.60 at 5%/2% budgets. Comparing VWS with pooled SnapKV using the stored positions gives paired exact-test $p = 0.031$ at both budgets.

Table 9: Llama-3.1-8B-Instruct beside Qwen2.5-7B-Instruct, both on the LongBench development split (400 paired; Llama on H200, Qwen on one RTX PRO 6000), 5% budget, Δ against pooled SnapKV with paired bootstrap CI; perm-s / perm-t as in Table 8. Needle rows: accuracy over 100 positions (dense 1.00 on both models).

comparison	Llama-3.1-8B	Qwen2.5-7B	Qwen perm-s / perm-t
dense vs. pooled SnapKV	+3.84 [+2.77, +4.99]	+2.59 [+1.61, +3.63]	<0.001 / 0.031
fixed-key construction	+0.70 [+0.14, +1.37]	−0.74 [−1.26, −0.27]	0.002 / 0.039
KeepKV, keys rewritten	−9.75 [−11.98, −7.66]	−6.85 [−8.39, −5.37]	<0.001 / 0.008
same merges, keys fixed	+0.37 [−0.31, +1.15]	−0.62 [−1.50, +0.25]	0.165 / 0.062
keys fixed vs. rewritten	+10.12 [+8.00, +12.38]	+6.23 [+4.77, +7.76]	<0.001 / 0.008
SelKV on pooled SnapKV	+0.02 [−0.26, +0.31]	−0.45 [−0.91, −0.04]	0.043 / 0.180
KVMerger, $\delta = 0.75$	−0.58 [−1.22, +0.05]	−0.66 [−1.03, −0.30]	<0.001 / 0.008
VWS vs. pooled SnapKV	+0.62 [−0.16, +1.47]	−0.12 [−0.75, +0.52]	0.727 / 0.523
PyramidKV vs. pooled SnapKV	−0.14 [−1.00, +0.76]	−0.45 [−1.41, +0.52]	0.367 / 0.352
needles @5%: VWS / SnapKV / PyramidKV	0.98 / 0.88 / 0.82	0.99 / 0.93 / 0.91	—
needles @2%: VWS / SnapKV / PyramidKV	0.90 / 0.78 / 0.48	0.78 / 0.72 / 0.60	—

C Algorithms

Module and configuration names in the released code keep the project's original `otkv` prefix: `otkv7` (with `auto_regime=false`) is VWS, which involves no transport; `otkv8` is the fixed-key construction of §6, whose receivers are assigned by min-cost matching; `otkv4` is the transport merge of Table 4.

VWS (selection). At prefill end, per layer ℓ with middle scores $s^{(\ell)}$ (window attention of the last W=32 queries, GQA query heads folded onto their KV head, summed across KV heads):

1. pool: $\tilde{s}_t = \max_{|u-t|\le 3} s_u$ (token-axis max-pool, kernel 7);
2. value weight: $\tilde{s}_t \leftarrow \tilde{s}_t \cdot \|v_t\|$ for ranking, and $r_t = s_t \cdot \|v_t\|$ (unpooled) for allocation;
3. budgets: with uniform per-layer budget B and total $L \cdot B$, allocate B_ℓ by greedy water-filling on the marginal gains $r^{(\ell)}_{(r)}/M_\ell$, where $r^{(\ell)}_{(r)}$ is the r-th largest middle score of layer ℓ and M_ℓ is the layer's total value-weighted attention mass including the protected sink and window, subject to $0.25B \le B_\ell \le 2.5B$. Normalisation

expresses each marginal gain as the *fraction* of the layer's mass retained by a slot, consistent with the role of ρ in Lemma 1. Using raw r_t would favour layers with large value norms or sharply peaked window attention because of their scale;

4. keep the top-B_ℓ middle tokens (sink and recency window are always kept); re-run every 4 decode steps with a reserve so peak cache $\leq$ budget.

Cost. VWS reads the window attention rows the model already computes for the last W queries, so scoring is $O(Wn)$ per head over a prefill of n tokens, pooling and value norms $O(nd)$, and the allocation a per-layer sort plus a global selection of $L \cdot B$ gains, $O(Ln \log n)$; the extra state is one score per token. In Table 3 time to first token is 3% higher and decode speed is 3–5% lower than for pooled SnapKV. The fixed-key construction fits at most four edges per evicted token, each a nine-point grid with a closed-form θ over W fit rows, $O(|E|Wd)$ per head, and solves one assignment problem on the surviving edges (Hungarian, cubic in the number of candidates, on a matrix with at most four non-drop entries per row); it runs once at prefill end and adds one additive mask at decode. Its prefill cost is not profiled here.

Compensated consolidation (class-B probe). After selection, per layer and KV head, with window log-attention columns $z_t \in \mathbb{R}^W$:

1. for adjacent kept pairs $(t, t{+}1)$, with the mass share $w_t = \mathbb{E}_q e^{z_t} / \mathbb{E}_q(e^{z_t}{+}e^{z_{t+1}})$ and $\bar{z} = w_t z_t + (1{-}w_t) z_{t+1}$, estimate the pair residual $r_t = \mathrm{std}_q\big[\log(e^{z_t}{+}e^{z_{t+1}}) - \bar{z}\big]$ over the W window queries q (the $\hat{\sigma}$ gate of §6);
2. greedily fuse non-overlapping pairs with r_t below threshold: $\bar{k} = w_t k_t + (1{-}w_t) k_{t+1}$, $\bar{v} = w_t v_t + (1{-}w_t) v_{t+1}$, bias $b = \mathbb{E}_q\big[\log\big(e^{z_t}{+}e^{z_{t+1}}\big) - \bar{z}\big]$;
3. each freed slot admits one more token at the same physical budget;
4. at decode, inject b for merged slots as an additive attention mask (rebuilt after any maintenance compression). Merging an identical pair is exactly lossless for every future query.

KVMerger port (§7). After the base selector has chosen the retained set, per layer: (i) consecutive middle tokens whose keys have head-averaged cosine above δ (0.75 published; 0.5 ablated) form a set; (ii) the set's pivot is its highest-scoring retained member (the published max-attention rule restricted to slots that exist); (iii) each evicted member i of a set with a pivot p is folded into the pivot with Gaussian-kernel weights $w_i \propto \exp(-\|k_p - k_i\|^2 / 2s^2)$, s^2 the mean squared distance of the set's members to the pivot, applied to keys and values; (iv) evicted tokens whose set has no retained member are evicted. One token layout is shared across heads, as in the published code. The published method reaches 35–50% budgets by merging alone; the same merge sets do not achieve a 5% budget, so under the matched protocol the operator acts on tokens selected for eviction: each is merged into the highest-scoring retained member of its set, or evicted if its set has none. On pooled SnapKV at the published $\delta = 0.75$, the port yields -0.58 $[-1.22, +0.05]$ (105 wins, 114 losses of 400) and is retrieval-neutral (NIAH 0.96 vs 0.92); at the looser $\delta = 0.5$ it is -0.06 (115 wins, 116 losses); on H2O it loses -1.35 (Table 2).

KeepKV and SelKV ports (§7). No released code was available for either method in our evaluation. Both ports implement the published equations and, for KeepKV, Algorithm 1; every underspecified choice and every departure from the papers' complete deployment configurations is recorded with the code (`docs/compensated_ports.md`), together with a test suite that verifies the operators against their equations and checks that disabling merging reproduces each port's eviction base bit for bit. *KeepKV* gives every entry an electoral vote p (initially 1) and computes decode attention as $\sum_i p_i s_i v_i / \sum_i p_i s_i$ with $s_i = e^{q \cdot k_i / \sqrt{d}}$, i.e. a per-slot logit bias $\log p_i$; each evicted key is folded into the retained key of highest cosine when that cosine exceeds $T = 0.8$, by ZIP-Merging $k_r = \frac{\ln((w_e+w_c)/(p_e+p_c))}{w_e \ln \hat{s}_e + w_c \ln \hat{s}_c}(w_e k_e + w_c k_c)$, $v_r = (w_e v_e + w_c v_c)/(w_e + w_c)$, $p_r = p_e + p_c$, $w = p\hat{s}$, which makes $p_r e^{q \cdot k_r / \sqrt{d}} = w_e + w_c$ exactly at the query the scores came from. The paper does not state its EMA constants (we use $\alpha = 0.9$ over the protocol's 32-query window), does not discuss GQA (we average $\hat{s}$ over a KV head's query heads and keep votes per KV head), and selects with PyramidInfer,

which our framework does not implement. We therefore evaluate the port with pooled-SnapKV selection, as indicated in the table. Scores are handled in log space; the key scale is applied as written, negative values included, with two counted numerical guards (a near-zero denominator, and a merged key that overflows the cache dtype; together they skipped 2 of 4.9×10^7 merges). *SelKV* scores tokens by $\sum_{q \in W} A_{q,i} \|v_i\|$ over a 32-query window with average pooling (kernel 5), takes a per-KV-head top-m_s, unions the heads and trims back to the budget by mean score; each evicted token j is routed to $\pi_h(j) = \arg\max_{i \in S_h \cap B(j)} A_{h,j,i}$ inside its 32-token bucket $B(j)$, gated by $g = \max(\cos(v_j, v_{\pi(j)}), 0)$, merged into the target's *value* only, and compensated by a logit bias $\alpha \log R$, $R = (a_i + \sum_j g_j a_j)/a_i$ averaged over KV heads, $\alpha = 0.5$. The paper does not specify how to handle an evicted token whose bucket contains no member of S_h; our port drops it. This applies to most evicted tokens at a 5% budget.

KeepKV loss sensitivities. We conduct four paired 400-sample checks against the baseline used for the -9.75 result. (i) *Key rewriting accounts for the large loss.* Holding each retained key fixed while applying the same value mixture and votes yields $+0.37$ $[-0.31, +1.15]$, the only variant with a positive point estimate. The values and biases are bit-identical to those of the full operator (`experiments/test_compensated_ports.py`), isolating the key rewrite. The direct paired contrast is $+10.12$ $[+8.00, +12.38]$, positive on all eight tasks. (ii) *The vote bias alone does not explain the loss.* Disabling the $\log p$ bias while still merging keys and values yields -9.39 $[-11.50, -7.42]$. (iii) *Protected-region targets have little effect.* Only 0.20% of merges target a sink and 1.27% target the recent window. Restricting targets to the middle region yields -9.88 $[-12.11, -7.74]$. (iv) *Extreme key scales are uncommon.* The scale is 1.006 at the median layer and 1.264 at the 99th percentile; 0.12% of merges use a negative scale. The two numerical guards are triggered twice in 4.9×10^7 merges. The frequency of retained-key updates varies with the budget: 1.20 merges per retained slot at 5%, compared with 0.74 at 25%. At the latter budget, the loss decreases to -4.24 $[-5.66, -2.89]$ but remains negative.

Operators not ported. Three relevant methods were not evaluated under our matched protocol. *AsymKV* (Cui and Xu, 2025) fuses adjacent keys and keeps the values' mass by cardinality normalisation: a key-averaging merge with a mass channel, whose closest analogue here is the key-averaged compensated consolidation of §6 (-0.72 at 5%, -1.83 at 2% against the identical selection); its reported gains are against H2O, which is 2.6 points below pooled SnapKV under our protocol (Table 4). *GRKV* (Peng et al., 2026) updates retained keys and values by ridge regression against the attention output, with regularisation against over-smoothing. It therefore moves keys, but by an update fitted to the output rather than by a merge formula. The KeepKV intervention in §7 does not test this type of update: it attributes a 6–10-point loss on two models to a rewrite designed to preserve a score surrogate, without establishing that all key updates are harmful. Whether an output-fitted, regularised key update survives decode-time drift (§6) remains an open question. *VECTOR* (Lin et al., 2026) retains every key and approximates the values of reconstructable tokens from their keys by an offline least-squares map, a retain/approximate/evict triage that keeps the attention geometry intact and is consistent with the selection-side reading of Lemma 1.