# BALANCING QUERY COVERAGE IN KV CACHE COMPRESSION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

During autoregressive decoding, the KV cache supports queries that may depend on different parts of a long context. Under a limited memory budget, selecting tokens by aggregate attention can leave some queries poorly covered even when the total retained attention is high. We study this imbalance through a distributional formulation of cache eviction, measuring query coverage by the attention probability mass assigned to retained tokens. For fixed queries and keys, restricting attention to a retained token subset is equivalent to conditioning the original attention distribution on that subset. The KL divergence from the conditioned distribution to the original distribution equals the negative logarithm of the retained probability mass, yielding an objective that maximizes the geometric mean of query coverage. Its marginal scores give greater weight to attention from queries with less retained mass. We instantiate this principle as FIDKV, a lightweight online eviction policy that estimates retained attention mass from the current cache without additional training or calibration. Our analysis bounds changes in token rankings by the spread of retained mass across queries, recovering cumulative-attention rankings when coverage is uniform. Experiments on LongBench and RULER across three language-model families support this prediction. On six LongBench tasks spanning all task categories, the advantage of FIDKV is most pronounced under aggressive compression: at 2% and 5% cache budgets, it outperforms the strongest evaluated baseline on most tasks, while remaining competitive at moderate budgets. Paired cache measurements further show improved coverage for the worst-served queries and reduced coverage imbalance, while deployment retains the memory and latency profile of standard KV eviction.

## 1 INTRODUCTION

KV cache eviction is usually presented as a token-ranking problem. A method constructs an importance signal for cached tokens, aggregates evidence across queries, and keeps the highest-scoring subset under a memory budget. Cumulative attention (Zhang et al., 2023), pooled attention over an observation window (Li et al., 2024; Cai et al., 2024; Feng et al., 2024), page-level summaries (Tang et al., 2024), persistence-weighted recency (Liu et al., 2023), and Gumbel-regularized key-token scoring (Adnan et al., 2024) all fit this template. Their differences matter, but they share a premise: token importance is the primitive objective.

This premise is convenient but incomplete. Eviction does not perturb generation by changing an abstract token score; it perturbs each query's attention distribution. Once a retained set $S$ is chosen, the query can only attend to tokens in $S$, and the retained-cache distribution is the full-cache distribution conditioned on this event. The normalizer of that conditional distribution is the retained attention mass $w_t(S)$, which we call *attention fidelity* (Figure 1). If $w_t(S)$ is small, query $t$ is under-served even if the retained tokens look important on average. A unit of attention from an already well-covered query and a unit from an under-served query should therefore not have the same marginal value.

This view makes the objective natural. Because retained-cache attention is a conditioning of full-cache attention, the Kullback–Leibler (KL) divergence between the two has the simple form $\mathrm{KL}(q_t \| p_t) = -\log w_t(S)$. Losing retained attention mass is exactly the information loss, and summing over queries gives a log-fidelity objective. Relaxing the keep/drop decision and applying the Karush–Kuhn–Tucker (KKT) conditions gives the marginal token score $\sum_t \alpha_{t,j}/w_t$: attention from low-fidelity queries is



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

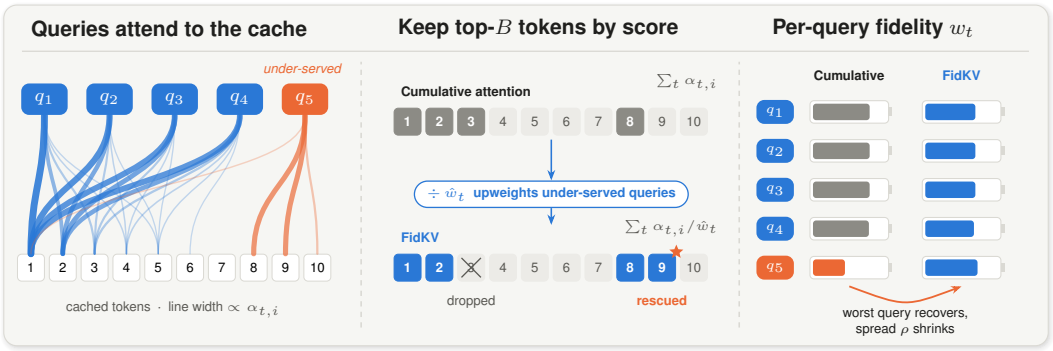



Figure 1: **How fidelity weighting rebalances the retained cache** (worked example, budget $B = 4$). **Left:** queries $q_1$–$q_4$ concentrate on tokens 1–3, while the under-served query $q_5$ attends mainly to tokens 8–9; line width is proportional to attention $\alpha_{t,i}$. **Middle:** cumulative attention $\sum_t \alpha_{t,i}$ keeps tokens 1, 2, 3, and 8. Dividing each query's attention by its estimated fidelity $\hat{w}_t$ upweights $q_5$, so FIDKV drops token 3 and rescues token 9. **Right:** per-query fidelity under the two rules. The retained mass of $q_5$ rises from the lowest to the highest among all queries, while $q_1$–$q_4$ decrease only slightly, so the worst-served query recovers and the fidelity spread $\rho$ shrinks.

amplified, while attention from already well-covered queries is discounted. Setting every $w_t = 1$ recovers cumulative attention.

We instantiate this score as FIDKV. The policy maintains a running fidelity-weighted score per cached token and estimates the current attention fidelity from a provisional top-$B$ set chosen by cumulative attention. A natural concern is that the online policy only sees attention over the already-compressed cache, not the full history. We show that this does not distort the weighting itself: current-cache attention and the estimated fidelity share the same normalizer, which cancels in the update (Section 4.2), and a dual-cache analyzer confirms the correspondence numerically.

The same formulation predicts *when* the correction can matter. A simple bound (Proposition 1) shows that the ratio between the FIDKV score and cumulative attention is controlled by the fidelity spread $\rho = \max_t w_t / \min_t w_t$. When fidelities are nearly uniform, FIDKV collapses toward cumulative attention; when some queries are clearly under-served, it can change which tokens survive. This predicts selective rather than uniform effects, and we evaluate it that way.

Our contributions are:

1. **Query fidelity as an explicit eviction objective.** We identify each query's retained attention mass as the quantity directly changed by eviction and show that joint attention KL equals negative average log-fidelity.
2. **A derived score and a faithful online policy.** KKT on the relaxed objective yields the fidelity-weighted score; we show that the deployed current-cache update reproduces the full-history weighting exactly up to a shared normalizer, and verify this with a dual-cache analyzer.
3. **A principled characterization of when weighting matters.** The re-ranking power of the correction is bounded by the fidelity spread $\rho$, so its effect should concentrate where retained mass is uneven.
4. **Empirical evidence across models and budgets.** Across LongBench, RULER, and three models, FIDKV is competitive at 10%–20% budgets and strongest under 2%–5% stress tests. A matched cumulative-attention control with paired bootstrap intervals shows that the $1/w_t$ correction preserves answer accuracy while improving the fidelity objective and attention-output approximation under repeated eviction.

## 2 Related Work

**Token-scoring eviction.** Most KV eviction methods define an importance signal for cached tokens and evict the low-scoring ones. *H2O* (Zhang et al., 2023) uses cumulative attention, *StreamingLLM* (Xiao et al., 2024) combines attention sinks with a recent window, *Scissorhands* (Liu et al., 2023)

uses persistence-weighted recency, *Keyformer* (Adnan et al., 2024) scores key tokens with a Gumbel-regularized attention signal, and *FastGen* (Ge et al., 2024) profiles heads to choose among predefined compression policies. Prefill-window methods such as *SnapKV* (Li et al., 2024) and *PyramidKV* (Cai et al., 2024) pool attention over a short observation window before selecting keys, while *AdaKV* (Feng et al., 2024) changes the head-level budget allocation. The closest recent method in spirit is *DefensiveKV* (Feng et al., 2025), which also revisits the aggregation step, replacing mean aggregation of observed importance with worst-case risk control. The two methods target different failure modes: DefensiveKV guards against fragile importance estimates in the aggregation, whereas FIDKV corrects a query-wise imbalance in retained mass that accumulates during decoding. The two are in principle composable.

**Complementary cache-efficiency methods.** Other methods improve long-context efficiency by changing the cache unit, precision, or access path rather than the token-scoring objective. *Quest* (Tang et al., 2024) selects pages using key summaries; *MInference* (Jiang et al., 2024) exploits structured sparse prefill; and *HeadKV* (Fu et al., 2024) allocates at the head level. Post-training sparse-attention approaches operate on a different axis: *DuoAttention* (Xiao et al., 2025) keeps full history only for identified retrieval heads and a bounded cache for streaming heads, and DeepSeek Sparse Attention (DeepSeek-AI, 2025) learns an indexer and adapts the model to perform query-dependent sparse access. These methods require calibration or training and differ in memory objective, so we do not treat them as memory-matched baselines; fidelity weighting could serve as the bounded-cache retention rule inside such systems. Quantization and memory-augmentation methods such as *KIVI* (Liu et al., 2024), *GEAR* (Kang et al., 2024), Landmark Attention (Mohtashami & Jaggi, 2023), and retrieval-augmented long-context models such as Unlimiformer (Bertsch et al., 2023) are likewise compatible: they decide how the cache is represented or accessed, while our formulation specifies what query-level quantity a retained token set should preserve.

# 3 Attention Fidelity and the Induced Score

## 3.1 Problem Setup and Fidelity

We work with one attention layer and one head at a time; the same rule is applied independently across heads under their assigned budgets. Let $\alpha_{t,i}$ be the full-cache attention paid by query $t$ to cached token $i$. A token eviction method maintains a retained set $S$ of size $B$ and answers the query using only tokens in $S$. The retained-cache attention distribution is therefore the full-cache distribution conditioned on the event that the attended token lies in $S$.

The state variable induced by this operation is the retained attention mass $w_t(S) = \sum_{i \in S} \alpha_{t,i}$, which we call the *attention fidelity* of query $t$. It measures how well the retained cache covers the query and is a query-level quantity, not another token score: $w_t(S) = 1$ means query $t$'s attention distribution is fully covered, while small $w_t(S)$ means the query is reconstructed from a small surviving fraction of its original attention mass. Fidelity also controls the attention output: if $\|v_i\| \le M$, then $\|o_t - \tilde{o}_t\| \le 2M(1 - w_t)$, with a tighter variance-aware refinement in Appendix A.

## 3.2 From Fidelity to an Objective

Let $p_t$ denote the full-cache attention distribution and $q_t$ the retained-cache distribution obtained by conditioning on $S$. Since the only normalizer introduced by this conditioning is $w_t(S)$, the per-query KL cost is exactly $\mathrm{KL}(q_t \| p_t) = -\log w_t(S)$ (Appendix A). If $P$ and $Q$ are the joint full-cache and retained-cache attention distributions over query-token pairs with a uniform query marginal, the joint KL is the average of the per-query terms:

$$S^\star = \arg\min_{|S|=B} \mathrm{KL}(Q\|P) = \arg\min_{|S|=B} \frac{1}{T} \sum_{t=1}^{T} \mathrm{KL}(q_t\|p_t) = \arg\max_{|S|=B} \sum_{t=1}^{T} \log w_t(S). \tag{1}$$

Equivalently, the objective maximizes the geometric mean $\left(\prod_t w_t(S)\right)^{1/T}$ of attention fidelity, i.e., of query coverage. It changes what the cache is asked to preserve: not total retained mass alone, but query-level fidelity across the decoding process.

### 3.3 Fidelity Imbalance

Let $\bar{w}(S) = T^{-1} \sum_t w_t(S)$. Jensen's inequality gives $T^{-1} \sum_t \log w_t(S) \le \log \bar{w}(S)$, with equality only when all queries have the same fidelity. For a fixed average retained mass, any imbalance in $(w_1, \ldots, w_T)$ lowers the objective, so recovering attention mass for a query with small $w_t$ is worth more than recovering the same mass for a well-covered query. We summarize this with the tail fidelity and fidelity spread

$$w_{\min}(S) = \min_t w_t(S), \qquad w_{\max}(S) = \max_t w_t(S), \qquad \rho(S) = w_{\max}(S)/w_{\min}(S). \tag{2}$$

Two retained sets can have similar average mass but very different low-fidelity tails; the log objective prefers the one that avoids such tails (Appendix A.4).

### 3.4 Fidelity-Weighted Marginal Score

Replace the binary retained-set indicator by weights $x \in [0,1]^n$ with $\sum_i x_i = B$. Differentiating log-fidelity gives the marginal value of increasing token $j$'s retention weight:

$$\frac{\partial}{\partial x_j} \sum_{t=1}^{T} \log\Big(\sum_i x_i \alpha_{t,i}\Big) = \sum_{t=1}^{T} \frac{\alpha_{t,j}}{w_t(x)} \;\triangleq\; \mathrm{score}(j). \tag{3}$$

Appendix A gives the full KKT statement: at the relaxed optimum, retained tokens are exactly those whose score exceeds a threshold. Unweighted cumulative attention counts every query equally, whereas the derived score divides by the query's current fidelity. Setting all fidelities to one recovers cumulative-attention scoring, so FidKV can be viewed as its fidelity correction.

### 3.5 When Can the Correction Matter?

If all queries have similar fidelity, dividing by $w_t$ rescales every query by nearly the same factor and the ranking barely changes. The following elementary bound makes this precise.

**Proposition 1** (Bounded re-ranking). *Let $a_j = \sum_t \alpha_{t,j}$, assume $a_j, a_{j'} > 0$ and $w_{\min} > 0$. Then*

$$\frac{1}{\rho} \cdot \frac{a_j}{a_{j'}} \;\le\; \frac{\mathrm{score}(j)}{\mathrm{score}(j')} \;\le\; \rho \cdot \frac{a_j}{a_{j'}}. \tag{4}$$

The proof follows from $w_{\min} \le w_t \le w_{\max}$ (Appendix A). At $\rho = 1$ the two scores induce the same ranking; as $\rho$ grows, the bound opens and fidelity weighting gains room to change which tokens survive. This gives a testable prediction about where the correction acts, which Section 5 tests.

## 4 Method

### 4.1 FidKV Cache Policy

The ideal score in Eq. (3) uses the fidelity of the optimal retained set, which is unavailable online. FidKV (Algorithm 1) keeps two accumulators per head and cached token: the unweighted cumulative attention $a_{h,i}$ and the fidelity-weighted score $s_{h,i}$. At each decoding step it computes attention over the current cache, forms a provisional retained set $\hat{S}_t$ from the $B$ tokens with the largest cumulative attention, estimates each head's current attention fidelity $\hat{w}_{t,h}$ as the attention mass on $\hat{S}_t$, and adds $\tilde{\alpha}_{t,h,j} / \max(\hat{w}_{t,h}, \epsilon)$ to every token's score. The retained set is always chosen by the fidelity-weighted scores; $a$ and $\hat{S}_t$ serve only to estimate fidelity. Scores are reduced across heads by a maximum, and the first $n_{\mathrm{sink}} = 4$ tokens are always kept. Eviction is batched with slack $\Delta = 256$, and the first eviction takes place after the first decoding query has been scored, so the initial selection over the prompt is already query-aware.

### 4.2 From the Current Cache to Full-History Fidelity

After earlier evictions, the policy cannot see full-history attention $\alpha_{t,\cdot}$; it sees $\tilde{\alpha}_{t,\cdot}$, the attention renormalized over the current cache. For a fixed head and step, let $c_t = \sum_{i \in C_t} \alpha_{t,i}$ be the full-history

**Algorithm 1** FIDKV for one layer with heads $h = 1, \dots, H$.

**Require:** budget $B$, sink size $n_{\text{sink}}$, eviction slack $\Delta$, stability constant $\epsilon$

**Prefill** (prompt of length $n$)

store all $n$ keys and values in $C$; $a_{h,i} \leftarrow 0$, $s_{h,i} \leftarrow 0$ for all $h, i \in C$ ▷ no eviction yet

**Decoding step** $t$ (query $q_{t,h}$ per head)

**for** $h = 1, \dots, H$ **do**

  $\tilde{\alpha}_{t,h,\cdot} \leftarrow \mathrm{softmax}\big(q_{t,h}^{\top} K_h / \sqrt{d}\big)$ over $C$ ▷ $= \alpha_{t,h,\cdot}/c_{t,h}$, Eq. (5)

  $a_{h,j} \leftarrow a_{h,j} + \tilde{\alpha}_{t,h,j}$ for all $j \in C$ ▷ cumulative attention

**end for**

$\hat{S}_t \leftarrow \mathrm{TopB}\big(\max_h a_{h,\cdot}\big)$ ▷ provisional retained set

**for** $h = 1, \dots, H$ **do**

  $\hat{w}_{t,h} \leftarrow \sum_{i \in \hat{S}_t} \tilde{\alpha}_{t,h,i}$ ▷ estimated attention fidelity

  $s_{h,j} \leftarrow s_{h,j} + \tilde{\alpha}_{t,h,j} / \max(\hat{w}_{t,h}, \epsilon)$ for all $j \in C$ ▷ fidelity-weighted score

**end for**

**if** $|C| > B + \Delta$ **then**

  $C \leftarrow$ first $n_{\text{sink}}$ tokens $\cup\ \mathrm{Top}_{B - n_{\text{sink}}}\big(\max_h s_{h,\cdot}\big)$; compact $C$, $a$, $s$

**end if**

mass held by the current cache. Then $\tilde{\alpha}_{t,i} = \alpha_{t,i}/c_t$ for $i \in C_t$, and because $\hat{S}_t \subseteq C_t$, the estimated fidelity is $\hat{w}_t = w_t^{\text{full}}(\hat{S}_t)/c_t$. The shared factor cancels in the update:

$$\frac{\tilde{\alpha}_{t,j}}{\hat{w}_t} = \frac{\alpha_{t,j}/c_t}{w_t^{\text{full}}(\hat{S}_t)/c_t} = \frac{\alpha_{t,j}}{w_t^{\text{full}}(\hat{S}_t)}. \tag{5}$$

So $\hat{w}_t$ need not equal the absolute retained mass in isolation: the quantity FIDKV actually accumulates is exactly the full-history fidelity-weighted attention evaluated at the provisional set $\hat{S}_t$. The remaining approximation is the substitution of $\hat{S}_t$ for the ideal set, with sensitivity $|1/\hat{w} - 1/w^{\star}| = |\hat{w} - w^{\star}|/(\hat{w}\, w^{\star})$ (Appendix A.6).

We verify this correspondence with the dual-cache analyzer of Section 5.5, which runs the full cache alongside the evicted one and therefore observes $\alpha_{t,\cdot}$ directly. After accounting for $c_t$, the full-history retained mass of the deployed provisional set is reconstructed with residual below $1.7 \times 10^{-7}$, and the deployed GQA-aggregated update matches the same-set full-history reference update with Pearson correlation above $0.99$ and mean absolute error below $3 \times 10^{-6}$. With $\epsilon = 10^{-6}$, clipping never triggered in the analyzed runs, so the floor did not affect any measured update. Moreover, because FIDKV performs its first eviction only after the first decoding query has been scored, no token has been removed at that point: the current cache is the full history, $c_t = 1$, and $\hat{w}_t$ equals the full-history fidelity $w_t^{\text{full}}(\hat{S}_t)$ exactly. Under the official generation limits, this first selection is the only eviction for most LongBench tasks, so the selection that determines the retained cache in these tasks uses exact fidelity values.

**Cost.** FIDKV adds one scalar sum and division per head and decoding step on top of cumulative-attention scoring, plus ordinary top-$B$ selection over maintained scores. It needs no learned gate, calibration pass, or auxiliary model.

## 5 EXPERIMENTS

We ask three questions. (i) Is the complete FIDKV policy competitive with representative eviction methods at practical budgets, and how does it behave under extreme compression? (ii) What does the $1/w_t$ correction itself contribute, measured against a matched cumulative-attention control? (iii) Do the cache-state effects predicted by Section 3 appear inside the cache?

### 5.1 SETTING

We evaluate on LongBench (Bai et al., 2024) (16 tasks) and RULER-64K (Hsieh et al., 2024) with LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and Qwen3-8B (Yang et al., 2025) (standard, non-thinking mode) from $10\%$ to $50\%$ retained cache. We

Table 1: **Task-level LongBench comparison at KV budgets 10% and 20% (LLaMA-3.1-8B-Instruct).** Ours denotes FIDKV. Scores are single-run greedy-decoding F1; bold marks the best eviction method per column within a budget (FIDKV within 0.2 points of the best is treated as a tie). The first row is the uncompressed full cache, given as a reference rather than a memory-matched competitor. Paired uncertainty for the fidelity correction is reported in Appendix Table 9.

| Budget | Method | Single-Doc QA | | | Multi-Doc QA | | | Summarization | | | Few-shot | | | Synthetic | | Code | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Qasper | MFQA | NrtvQA | HpQA | 2WkMQ | MuSQ | GovRpt | QMSum | MNws | TREC | TrvQA | SAMSum | PC | PR | LCC | RB-P | Avg |
| 100% | Full cache | 33.7 | 55.8 | 31.4 | 54.2 | 44.9 | 29.5 | 41.6 | 16.6 | 32.3 | 13.9 | 49.2 | 34.3 | 9.1 | 100.0 | 24.7 | 29.8 | 37.6 |
| 10% | StreamingLLM | 23.2 | 35.4 | 25.0 | 49.2 | 39.7 | 22.8 | 31.6 | 12.6 | 22.8 | 6.2 | 48.5 | 33.3 | **10.0** | 93.5 | 19.9 | 23.1 | 31.1 |
| | H2O | 23.1 | 34.8 | 29.5 | 49.1 | 40.1 | 22.9 | 31.6 | 12.4 | 22.7 | 6.9 | 49.1 | 33.2 | 9.0 | 93.5 | 19.5 | 23.2 | 31.3 |
| | SnapKV | 29.8 | 52.7 | **31.8** | 53.5 | 41.6 | 29.2 | 35.2 | 14.6 | 19.1 | 11.8 | 48.2 | 34.1 | 7.3 | 99.5 | 20.2 | **28.4** | 34.8 |
| | AdaKV | 30.9 | 54.4 | 31.5 | **54.2** | 41.8 | 28.9 | 35.6 | 14.8 | 19.6 | 12.3 | 48.1 | 33.6 | 9.1 | 100.0 | 20.8 | 27.5 | 35.2 |
| | PyramidKV | 30.7 | 54.4 | 30.6 | 53.9 | 41.7 | 28.1 | 34.9 | 14.1 | 20.0 | 12.3 | 48.6 | **34.3** | 7.8 | 99.5 | 19.5 | 27.5 | 34.9 |
| | Quest | 32.7 | 55.2 | 31.5 | 54.0 | 44.0 | 29.3 | 36.0 | 14.9 | 25.4 | 13.6 | 48.7 | 33.0 | 7.5 | 99.5 | 21.3 | 26.4 | 35.8 |
| | Keyformer | 24.5 | 41.4 | 30.2 | 50.9 | 43.8 | 27.4 | 33.2 | 13.7 | 26.1 | 9.1 | 48.7 | 32.4 | 9.9 | 98.5 | 19.0 | 22.5 | 33.2 |
| | **Ours** | **32.6** | **56.4** | 31.2 | 54.0 | **44.5** | **29.3** | **37.3** | **14.8** | **27.4** | **14.1** | **49.1** | 33.0 | 9.1 | **100.0** | **21.1** | 26.7 | **36.3** |
| 20% | StreamingLLM | 23.3 | 37.0 | 25.2 | 50.8 | 40.8 | 22.2 | 34.1 | 13.1 | 26.7 | 8.3 | 48.3 | 32.9 | 10.0 | 90.5 | 22.0 | 26.4 | 32.0 |
| | H2O | 23.7 | 37.0 | 28.5 | 50.5 | 40.7 | 22.3 | 34.2 | 12.9 | 26.7 | 8.6 | 48.2 | 33.7 | 10.5 | 90.5 | 22.4 | 26.2 | 32.3 |
| | SnapKV | 32.0 | 55.5 | 31.3 | 53.8 | 43.3 | 29.3 | 37.0 | 15.7 | 25.4 | 13.8 | 48.3 | 33.7 | 8.5 | 100.0 | 23.6 | **29.2** | 36.3 |
| | AdaKV | 32.7 | 55.5 | **31.7** | **54.7** | 44.0 | 28.9 | 37.9 | **15.8** | 26.1 | **14.5** | 48.3 | **34.4** | **10.6** | 100.0 | 24.1 | 29.1 | **36.8** |
| | PyramidKV | 31.9 | 55.4 | 31.6 | 54.0 | 43.6 | 28.4 | 37.0 | 15.6 | 26.3 | 14.1 | 48.2 | 33.9 | 9.1 | 100.0 | **24.2** | 29.1 | 36.4 |
| | Quest | 33.2 | 56.0 | 31.2 | 54.6 | 44.2 | 28.6 | 37.1 | 15.5 | 28.6 | 14.4 | 48.2 | 34.1 | 8.7 | 100.0 | 24.1 | 28.6 | 36.7 |
| | Keyformer | 26.3 | 43.3 | 30.9 | 53.1 | 44.3 | 26.7 | 35.3 | 14.3 | 27.5 | 11.3 | 48.8 | 33.3 | 8.8 | 99.5 | 21.3 | 26.5 | 34.5 |
| | **Ours** | **33.5** | **56.6** | 31.1 | 54.1 | **44.6** | **29.8** | **37.9** | 15.6 | **29.4** | 14.1 | **48.8** | 33.1 | 8.8 | **100.0** | 23.4 | 28.5 | **36.8** |

Table 2: **LongBench average F1 across models and budgets.** FIDKV vs. the strongest of the seven evaluated eviction methods in each setting (full per-task results in Table 1 and Appendix Table 5).

| | 10% cache | | 20% cache | |
|---|---|---|---|---|
| Model | FIDKV | Best baseline | FIDKV | Best baseline |
| LLaMA-3.1-8B-Instruct | **36.3** | 35.8 | **36.8** | **36.8** |
| Mistral-7B-Instruct-v0.3 | **30.1** | 29.5 | **30.7** | 30.4 |
| Qwen3-8B | **33.4** | 33.1 | 34.1 | **34.2** |

compare with StreamingLLM, H2O, AdaKV, SnapKV, PyramidKV, Quest, and Keyformer. The primary operating points are $10\%$ and $20\%$ ($10\times$ and $5\times$ compression); $2\%$/$5\%$ budgets are an explicit stress test, and $30\%$–$50\%$ results are in Appendix B. All runs use `bfloat16` and greedy decoding with the official task-specific generation caps. Because decoding is greedy and the policy is training-free, there is no seed-dependent randomness; we quantify uncertainty with paired bootstrap intervals over examples (Appendix D).

### 5.2 Main Comparison at 10% and 20% Budgets

Table 1 compares methods on LLaMA-3.1, and Appendix Table 5 extends this to all three models. At the model-budget level, FIDKV has the best or tied-best average in five of six settings and is within $0.1$ points of the best in the sixth, with margins of up to $0.6$ points over the strongest baseline. Per task, FIDKV tends to be strongest on multi-document QA and summarization columns and close to the competitive cluster on synthetic and code tasks. All methods remain below the full cache, most visibly on summarization. Table 2 summarizes the averages across all three models.

### 5.3 What Does the Fidelity Correction Contribute?

Differences against other eviction methods mix the scoring rule with other design choices (observation windows, head allocation, page granularity). To isolate the correction, we compare FIDKV with a *matched cumulative-attention control* (CumAttn) that removes only the $1/\hat{w}_t$ weighting while keeping the model, cache size, examples, accumulator, and eviction schedule fixed (Appendix C). On all 16 LongBench tasks, the correction preserves the answer accuracy of the matched control at both 10% and 20% budgets. Where corrected scores feed later evictions, it improves the cumulative log-fidelity objective by 6.0% and 8.6% and reduces full-attention output-approximation error, with paired bootstrap intervals excluding zero at both budgets.

Table 3: **Extreme-budget stress test** (LLaMA-3.1-8B-Instruct). Sample-weighted F1 on six LongBench tasks, one per category (MultiFieldQA, 2WikiMQA, MultiNews, SAMSum, PassageCount, LCC). $\Delta$ is the gap to the strongest baseline with its paired 95% bootstrap interval. Per-task results are in Table 4.

| Budget | StreamingLLM | H2O | SnapKV | AdaKV | PyramidKV | **FIDKV** | $\Delta$ [95% CI] |
|---|---|---|---|---|---|---|---|
| 2% (50×) | 21.62 | 21.89 | 10.52 | 11.04 | 17.63 | **27.00** | +5.11 [+4.09, +6.14] |
| 5% (20×) | 23.28 | 23.27 | 21.46 | 21.79 | 23.05 | **28.36** | +5.08 [+4.03, +6.15] |

Table 4: **Per-task breakdown of the extreme-budget stress test (Table 3).** LLaMA-3.1-8B-Instruct. One task is taken from each LongBench category. "Best base." is the strongest of StreamingLLM, H2O, SnapKV, AdaKV, and PyramidKV *on that task*, so it can be a different method in each row. Aggregate results are in Table 3.

| | | 2% budget | | | 5% budget | | |
|---|---|---|---|---|---|---|---|
| Task | Category | FIDKV | Best base. | $\Delta$ | FIDKV | Best base. | $\Delta$ |
| MultiFieldQA | Single-Doc QA | 54.37 | 31.63 | +22.74 | 55.65 | 44.51 | +11.14 |
| 2WikiMQA | Multi-Doc QA | 44.42 | 40.08 | +4.34 | 44.39 | 40.52 | +3.87 |
| MultiNews | Summarization | 22.02 | 17.81 | +4.21 | 24.63 | 19.17 | +5.46 |
| SAMSum | Few-shot | 32.59 | 30.22 | +2.37 | 32.62 | 33.42 | −0.80 |
| PassageCount | Synthetic | 0.09 | 0.12 | −0.03 | 0.13 | 0.14 | −0.01 |
| LCC | Code | 15.36 | 14.13 | +1.23 | 19.59 | 18.28 | +1.31 |

### 5.4 Extreme-Budget Stress Test

Table 3 probes 50× and 20× compression. These budgets are not recommended operating points; they show how methods fail when memory pressure is extreme. The six tasks were chosen by a fixed rule, one representative per LongBench category, and Table 4 gives the per-task breakdown. FIDKV beats the strongest per-task baseline on five of six tasks at 2% and four of six at 5%, and its aggregate exceeds the strongest aggregate baseline by about 5 points with paired intervals excluding zero. The gains are largest on MultiFieldQA and remain positive on multi-document QA, summarization, and code; PassageCount is near zero for every method. Because prefill-window methods degrade sharply at 2%, we report $\Delta$ against the strongest baseline in each setting, which remains stable at both budgets.

### 5.5 Mechanism Analysis

We instrument eviction with a dual-cache analyzer that runs the evicted cache and the full cache in parallel and records retained mass $w_t$, spread $\rho$, and attention-output perturbation. The reference ranks tokens by unweighted cumulative attention $\sum_t \alpha_{t,i}$, whereas FIDKV uses $\sum_t \alpha_{t,i}/w_t$. The analyzer is not part of deployment.

**Spread and effect size.** Proposition 1 predicts that the correction can only re-rank tokens when $\rho$ is large. Figure 2 (left) is consistent with this on the four measured LLaMA-3.1 settings: the MFQA points lie in the high-spread region and show large differences from the reference, while the TriviaQA points lie in the low-spread region and stay close to it, with both budgets following the same ordering.

**Cache state.** If FIDKV is correcting fidelity imbalance, it should raise the worst-served retained states and contract the spread. Figure 2 (right) shows both effects in paired sample-level measurements: the change in $\min_{\ell,t} w_{\ell,t}$ is positive and the change in $\log \rho$ is negative.

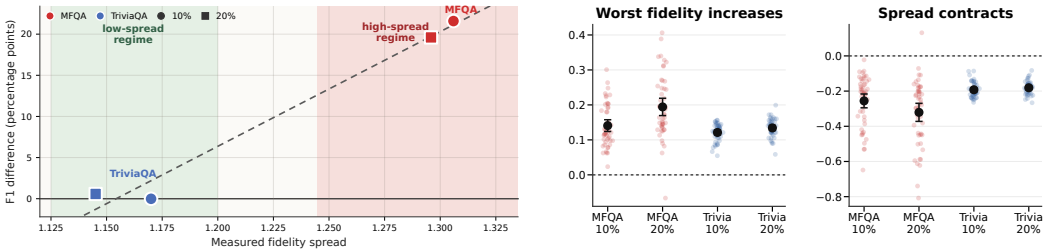



Figure 2: **Left: fidelity spread and the effect of the correction.** Red and blue denote MFQA and TriviaQA; circles and squares denote $10\%$ and $20\%$ budgets. The x-axis is the spread measured under the cumulative-attention reference; the y-axis is the F1 difference between FIDKV and that reference. **Right: cache-state measurements.** Paired change in the worst retained fidelity and in log fidelity spread, relative to the cumulative-attention reference.

**Output perturbation.** Renormalization replaces the original attention mixture with the retained-value mixture, and $o_t - \tilde{o}_t = (1 - w_t)(\bar{v}_{\bar{S}} - \bar{v}_S)$ (Appendix A): fidelity loss sets the scale while value geometry determines how much becomes output error. Figure 3 shows the same pattern empirically: low-loss states are stable, whereas high-loss states produce larger and more variable perturbations. The worst-case $2M(1 - w_t)$ envelope is conservative, but the fidelity variable remains predictive.

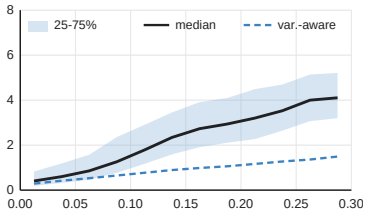



Figure 3: **Fidelity loss vs. output perturbation** $\|o_{\ell,t} - \tilde{o}_{\ell,t}\|_2$. Solid: binned median with IQR; dashed: variance-aware bound.

### 5.6 RULER AND EFFICIENCY

On RULER-64K (Figure 4), FIDKV departs from the comparison methods on tasks where budget pressure makes evidence selection difficult and stays within the competitive cluster otherwise, mirroring the LongBench pattern.

FIDKV keeps the retained KV budget unchanged and adds negligible state, so it is memory-neutral (peak memory within about 0.3% of SnapKV; Appendix Table 8). The per-step score update adds little overhead: end-to-end wall-clock speed relative to SnapKV is $1.08\times$/$1.03\times$ on LLaMA-3.1, $0.96\times$/$0.93\times$ on Mistral, and $0.95\times$/$0.93\times$ on Qwen3 at 10%/20% budgets, and latency follows the same length trend as other attention-scoring methods (Appendix Figure 5). FIDKV is faster than AdaKV and Quest in all settings.

## 6 DISCUSSION

**When to expect an effect.** The analysis predicts that fidelity weighting matters when retained mass is uneven across queries and is close to neutral when it is balanced. The experiments follow this pattern: the complete policy separates most clearly from the comparison methods under aggressive compression and on tasks with dispersed evidence, while at moderate budgets it stays within the competitive cluster. The fidelity spread $\rho$ is cheap to monitor online, so it also offers a practical diagnostic of whether a workload is in the regime where the choice of eviction score is consequential.

**Composability.** FIDKV changes only how evidence from different queries is weighted inside the token score; it leaves the cache representation, the budget, and the access path unchanged. It can therefore be combined with head-level budget allocation such as AdaKV, with quantized caches such as KIVI, and with page- or head-structured systems such as Quest and DuoAttention, where it would determine which tokens a bounded cache retains. Because the method adds no parameters and no calibration pass, such combinations do not require retraining.

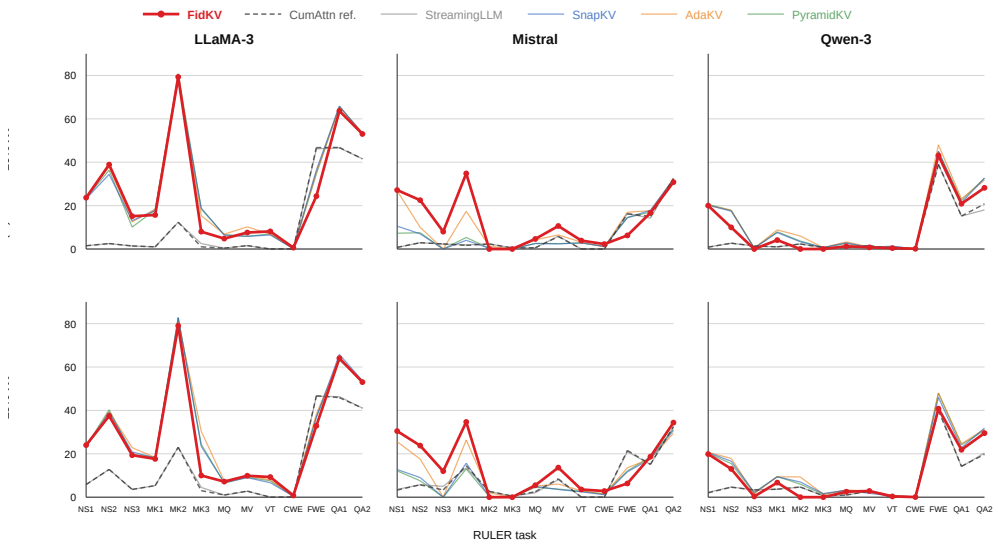



Figure 4: **RULER-64K task profiles under budget pressure.** Each panel is one model-budget setting; the x-axis lists RULER tasks, the y-axis reports F1, and each curve is an eviction method.

## 7 Limitations

Our evaluation focuses on standard long-context benchmarks with short-to-moderate generation lengths. Long reasoning traces, multi-turn dialogue, and agent workloads involve many more eviction steps, where fidelity imbalance can accumulate over time; extending the evaluation to these settings is a natural next step. All results use greedy decoding with a single run per configuration, and we report paired bootstrap intervals for the key comparisons.

## 8 Conclusion

We reframed KV eviction around attention fidelity: the retained attention mass of each query is the quantity eviction directly changes, and it determines the attention KL exactly. This view yields a log-fidelity objective, a KKT-derived fidelity-weighted score, and a bound identifying when the score can differ from cumulative attention. The deployed policy implements this score faithfully despite seeing only the compressed cache, and a matched control shows that it improves the fidelity objective and output approximation under repeated eviction while preserving accuracy. Long-generation workloads, where eviction is repeated many times and fidelity imbalance can accumulate, are a natural setting in which to extend this analysis.

## AI Use Statement

AI assistants were used only for LaTeX formatting. All technical content, experiments, analysis, and results are entirely the authors' own work.

## Ethics Statement

FidKV is an inference-efficiency method. Its main positive impact is to reduce the memory cost of long-context language-model deployment, which can make long-context tools more accessible. The same efficiency can make both benign and harmful uses cheaper. The method does not release a new model or dataset and adds no new capability beyond cache management; deployments should rely on the safeguards appropriate for the underlying model and application. All benchmarks used are public.

## Reproducibility Statement

The derivations and proofs are in Appendix A. The policy is fully specified by Algorithm 1 and Section 4.2, including $\epsilon = 10^{-6}$. Models, benchmarks, budgets, decoding settings, and hardware are given in Section 5.1 and Appendix D; the task-selection rule for the stress test is in Section 5.4; and the bootstrap procedure is in Appendix D. Code, evaluation outputs, aggregation scripts, and the dual-cache analyzer will be released.

## References

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: KV cache reduction through key tokens selection for efficient generative inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. Unlimiformer: Long-range transformers with unlimited length input. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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. *arXiv preprint arXiv:2407.11550*, 2024.

Yuan Feng, Haoyu Guo, Junlin Lv, S. Kevin Zhou, and Xike Xie. Taming the fragility of KV cache eviction in LLM inference. *arXiv preprint arXiv:2510.13334*, 2025.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. *arXiv preprint arXiv:2410.19258*, 2024.

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

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

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

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

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An efficient KV cache compression recipe for near-lossless generative inference of LLM. *arXiv preprint arXiv:2403.05527*, 2024.

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

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

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

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

# A Theoretical Derivations and Proofs

This appendix collects the derivations used in Sections 3 and 4. They follow one sequence: eviction conditions the full attention distribution on the retained set; the conditioning normalizer gives the KL cost; summing this cost gives the log-fidelity objective; Jensen's inequality exposes the resulting imbalance penalty; differentiating the objective gives the FIDKV score.

## A.1 Notation

For a retained set $S$, write $w_t = w_t(S) = \sum_{i\in S} \alpha_{t,i}$. The full-cache attention distribution is $p_t(i) = \alpha_{t,i}$. When $w_t > 0$, the retained-cache distribution is

$$q_t(i) = \begin{cases} \alpha_{t,i}/w_t, & i \in S, \\ 0, & i \notin S. \end{cases}$$

The case $w_t = 0$ gives infinite log-fidelity cost and is excluded from finite-objective derivations. The full and retained outputs are $o_t = \sum_i \alpha_{t,i} v_i$ and $\tilde{o}_t = \sum_{i\in S} \alpha_{t,i} v_i / w_t$ when $w_t > 0$. For $0 < w_t < 1$, define the retained and evicted value means

$$\bar{v}_S = \frac{1}{w_t} \sum_{i\in S} \alpha_{t,i} v_i, \qquad \bar{v}_{\bar{S}} = \frac{1}{1-w_t} \sum_{i\notin S} \alpha_{t,i} v_i.$$

If $w_t = 1$, no attention mass is evicted, $o_t = \tilde{o}_t$, and the output bounds below hold trivially.

## A.2 Conditioning on the Retained Cache

The retained-cache distribution $q_t$ is exactly $p_t$ conditioned on $i \in S$. Hence the same normalizer $w_t$ appears in both the information loss and the output perturbation.

**Lemma 2** (KL identity)**.** *If $w_t > 0$, then* $\mathrm{KL}(q_t \| p_t) = -\log w_t$.

*Proof.* Because $q_t(i) = p_t(i)/w_t$ on $S$ and $q_t(i) = 0$ outside $S$,

$$\mathrm{KL}(q_t \| p_t) = \sum_{i\in S} \frac{\alpha_{t,i}}{w_t} \log \frac{\alpha_{t,i}/w_t}{\alpha_{t,i}} = \log \frac{1}{w_t} \sum_{i\in S} \frac{\alpha_{t,i}}{w_t} = -\log w_t.$$

**Lemma 3** (Exact output decomposition)**.** *For $0 < w_t < 1$, $o_t - \tilde{o}_t = (1-w_t)(\bar{v}_{\bar{S}} - \bar{v}_S)$.*

*Proof.* Write the full output as the mixture of the retained and evicted conditional means: $o_t = w_t \bar{v}_S + (1-w_t)\bar{v}_{\bar{S}}$ and $\tilde{o}_t = \bar{v}_S$. Subtracting the second identity from the first yields the claim.

**Corollary 4** (Worst-case output bound)**.** *If $\|v_i\| \le M$, then $\|o_t - \tilde{o}_t\| \le 2M(1-w_t)$.*

*Proof.* Apply the triangle inequality to the exact decomposition and use $\|\bar{v}_S\| \le M$, $\|\bar{v}_{\bar{S}}\| \le M$.

**Corollary 5** (Variance-aware output bound)**.**

$$\|o_t - \tilde{o}_t\| \le \sqrt{\frac{1-w_t}{w_t}} \cdot \sqrt{\mathrm{Var}_{p_t}[v]}, \qquad \mathrm{Var}_{p_t}[v] = \sum_i \alpha_{t,i} \|v_i - o_t\|^2.$$

*Proof.* From $o_t = w_t \bar{v}_S + (1-w_t)\bar{v}_{\bar{S}}$, $\bar{v}_S - o_t = (1-w_t)(\bar{v}_S - \bar{v}_{\bar{S}})$ and $\bar{v}_{\bar{S}} - o_t = w_t(\bar{v}_{\bar{S}} - \bar{v}_S)$. Dropping the within-cluster variance terms and keeping only the between-cluster component gives

$$\begin{aligned} \mathrm{Var}_{p_t}[v] &\ge w_t \|\bar{v}_S - o_t\|^2 + (1-w_t)\|\bar{v}_{\bar{S}} - o_t\|^2 \\ &= w_t(1-w_t)^2 \|\bar{v}_{\bar{S}} - \bar{v}_S\|^2 + (1-w_t) w_t^2 \|\bar{v}_{\bar{S}} - \bar{v}_S\|^2 = w_t(1-w_t)\|\bar{v}_{\bar{S}} - \bar{v}_S\|^2. \end{aligned} \tag{6}$$

Combining this with the exact decomposition,

$$\|o_t - \tilde{o}_t\| = (1-w_t)\|\bar{v}_{\bar{S}} - \bar{v}_S\| \le (1-w_t)\sqrt{\frac{\mathrm{Var}_{p_t}[v]}{w_t(1-w_t)}} = \sqrt{\frac{1-w_t}{w_t}} \sqrt{\mathrm{Var}_{p_t}[v]}.$$

### A.3 Log-Fidelity Objective

Let $P(t,i) = \alpha_{t,i}/T$ and $Q(t,i) = q_t(i)/T$. Then

$$\begin{aligned}\mathrm{KL}(Q\|P) &= \sum_{t=1}^{T}\sum_{i=1}^{n} Q(t,i)\log\frac{Q(t,i)}{P(t,i)} = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n} q_t(i)\log\frac{q_t(i)}{p_t(i)} \\ &= \frac{1}{T}\sum_{t=1}^{T}\mathrm{KL}(q_t\|p_t) = -\frac{1}{T}\sum_{t=1}^{T}\log w_t(S). \end{aligned} \tag{7}$$

Thus minimizing joint attention KL is equivalent to maximizing $\sum_t \log w_t(S)$.

### A.4 Fidelity Imbalance and Jensen Bound

The log objective is not a monotone reparameterization of total retained mass. Define $J_{\mathrm{lin}}(S) = \sum_t w_t(S)$ and $J_{\log}(S) = \sum_t \log w_t(S)$. Assume every $w_t(S) > 0$; if some $w_t(S) = 0$, then $J_{\log}(S) = -\infty$. Jensen's inequality for the concave log gives

$$\frac{1}{T}J_{\log}(S) \le \log\left(\frac{1}{T}\sum_{t=1}^{T} w_t(S)\right) = \log\left(\frac{J_{\mathrm{lin}}(S)}{T}\right), \tag{8}$$

with equality if and only if $w_1(S) = \cdots = w_T(S)$. Among retained sets with the same linear fidelity, the log objective is maximized by the one whose attention fidelitys are most balanced.

The two-query case makes the penalty explicit. Fix an average fidelity $m$ and write $w_1 = m + \delta$, $w_2 = m - \delta$ with $0 \le \delta < m$. Then

$$\tfrac{1}{2}\big(\log(m+\delta) + \log(m-\delta)\big) = \log m + \tfrac{1}{2}\log\big(1 - \delta^2/m^2\big) \le \log m. \tag{9}$$

As $\delta$ grows, the average retained mass stays fixed but the objective decreases. For a small fidelity gain $\Delta > 0$ assigned to query $t$, $\log(w_t + \Delta) - \log w_t = \Delta/w_t + O(\Delta^2/w_t^2)$, so the same recovered mass has larger first-order value when $w_t$ is small. The KKT score below is the token-level form of this rule.

### A.5 KKT Score

Relax the retained-set indicator to $x \in [0,1]^n$ with $\sum_i x_i = B$, on the domain where $w_t(x) = \sum_i x_i\alpha_{t,i} > 0$ for every query. Define $F(x) = \sum_t \log\big(\sum_i x_i\alpha_{t,i}\big)$, which is concave as a sum of logs of positive affine functions, with $\partial F/\partial x_j = \sum_t \alpha_{t,j}/w_t(x)$. The Lagrangian of the relaxed problem is

$$\mathcal{L}(x,\lambda,\mu,\nu) = F(x) - \lambda\Big(\sum_i x_i - B\Big) + \sum_i \mu_i x_i + \sum_i \nu_i(1 - x_i), \tag{10}$$

with $\mu_i, \nu_i \ge 0$. Stationarity at $x^\star$ gives $\sum_t \alpha_{t,j}/w_t(x^\star) - \lambda + \mu_j - \nu_j = 0$. Together with complementary slackness $\mu_j x_j^\star = 0$ and $\nu_j(1 - x_j^\star) = 0$, this yields the threshold form

$$\sum_{t=1}^{T}\frac{\alpha_{t,j}}{w_t(x^\star)}\begin{cases}\ge \lambda, & x_j^\star = 1,\\ = \lambda, & x_j^\star \in (0,1),\\ \le \lambda, & x_j^\star = 0.\end{cases}$$

The KKT conditions therefore rank tokens by the fidelity-weighted score used in Section 3.4.

### A.6 Runtime Fidelity Surrogate

By Eq. (5), the deployed update equals $\alpha_{t,j}/w_t^{\mathrm{full}}(\hat{S}_t)$, so the only approximation relative to the ideal score is the use of the provisional set $\hat{S}_t$ in place of the relaxed optimum $x^\star$. Writing $\hat{w}_t$ for $w_t^{\mathrm{full}}(\hat{S}_t)$, the per-query weight perturbation without clipping is

$$\left|\frac{1}{\hat{w}_t} - \frac{1}{w_t(x^\star)}\right| = \frac{|\hat{w}_t - w_t(x^\star)|}{\hat{w}_t\, w_t(x^\star)}.$$

The implemented update uses the clipped inverse $\psi_\epsilon(w) = 1/\max(w, \epsilon)$, which is $1/\epsilon^2$-Lipschitz on $w \geq 0$; if $|\hat{w}_t - w_t(x^\star)| \leq \eta_t$, then $|\psi_\epsilon(\hat{w}_t) - \psi_\epsilon(w_t(x^\star))| \leq \eta_t/\epsilon^2$. The update is therefore accurate when the provisional set has fidelity close to the ideal set, and most sensitive in the low-fidelity regime where the correction is meant to act. Section 4.2 verifies the normalization part of this approximation numerically, and shows that it is exact at the first eviction.

### A.7 Proof of Proposition 1

Let $a_j = \sum_t \alpha_{t,j}$ and $s_j = \sum_t \alpha_{t,j}/w_t$. Since $w_{\min} \leq w_t \leq w_{\max}$, we have $a_j/w_{\max} \leq s_j \leq a_j/w_{\min}$. Applying the lower bound to $s_j$ and the upper bound to $s_{j'}$ gives $s_j/s_{j'} \geq (a_j/w_{\max})/(a_{j'}/w_{\min}) = \rho^{-1} a_j/a_{j'}$, and the reverse pairing gives $s_j/s_{j'} \leq \rho\, a_j/a_{j'}$. □

## B Extended Empirical Results

### B.1 LongBench: All Models at Budgets 10% and 20%

Table 5 expands the main LongBench comparison to all three models, and Table 6 gives the uncompressed full-cache reference for each model.

Table 5: LongBench F1 (%) at KV budgets 10% and 20% for all three LLMs. **FIDKV** (ours). Best per column (within a sub-block) bold; FIDKV within 0.2 percentage points of the best is treated as a tie.

| Method | Single-Doc QA | | | Multi-Doc QA | | | Summarization | | | Few-shot | | | Synthetic | | Code | | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Qasper | MFQA | NrtvQA | HpQA | 2WkMQ | MuSQ | GovRpt | QMSum | MNws | TREC | TrvQA | SAMSum | PC | PR | LCC | RB-P | |
| *LLaMA-3 @ 10%* | | | | | | | | | | | | | | | | | |
| StreamingLLM | 23.2 | 35.4 | 25.0 | 49.2 | 39.7 | 22.8 | 31.6 | 12.6 | 22.8 | 6.2 | 48.5 | 33.3 | **10.0** | 93.5 | 19.9 | 23.1 | 31.1 |
| H2O | 23.1 | 34.8 | 29.5 | 49.1 | 40.1 | 22.9 | 31.6 | 12.4 | 22.7 | 6.9 | 49.1 | 33.2 | 9.0 | 93.5 | 19.5 | 23.2 | 31.3 |
| SnapKV | 29.8 | 52.7 | **31.8** | 53.5 | 41.6 | 29.2 | 35.2 | 14.6 | 19.1 | 11.8 | 48.2 | 34.1 | 7.3 | 99.5 | 20.2 | **28.4** | 34.8 |
| AdaKV | 30.9 | 54.4 | 31.5 | **54.2** | 41.8 | 28.9 | 35.6 | 14.8 | 19.6 | 12.3 | 48.1 | 33.6 | 9.1 | 100.0 | 20.8 | 27.5 | 35.2 |
| PyramidKV | 30.7 | 54.4 | 30.6 | 53.9 | 41.7 | 28.1 | 34.9 | 14.1 | 20.0 | 12.3 | 48.6 | **34.3** | 7.8 | 99.5 | 19.5 | 27.5 | 34.9 |
| Quest | 32.7 | 55.2 | 31.5 | 54.0 | 44.0 | 29.3 | 36.0 | 14.9 | 25.4 | 13.6 | 48.7 | 33.0 | 7.5 | 99.5 | 21.3 | 26.4 | 35.8 |
| Keyformer | 24.5 | 41.4 | 30.2 | 50.9 | 43.8 | 27.4 | 33.2 | 13.7 | 26.1 | 9.1 | 48.7 | 32.4 | 9.9 | 98.5 | 19.0 | 22.5 | 33.2 |
| **FIDKV** | **32.6** | **56.4** | 31.2 | 54.0 | **44.5** | **29.3** | **37.3** | **14.8** | **27.4** | **14.1** | **49.1** | 33.0 | 9.1 | **100.0** | **21.1** | 26.7 | **36.3** |
| *LLaMA-3 @ 20%* | | | | | | | | | | | | | | | | | |
| StreamingLLM | 23.3 | 37.0 | 25.2 | 50.8 | 40.8 | 22.2 | 34.1 | 13.1 | 26.7 | 8.3 | 48.3 | 32.9 | 10.0 | 90.5 | 22.0 | 26.4 | 32.0 |
| H2O | 23.7 | 37.0 | 28.5 | 50.5 | 40.7 | 22.3 | 34.2 | 12.9 | 26.7 | 8.6 | 48.2 | 33.7 | 10.5 | 90.5 | 22.4 | 26.2 | 32.3 |
| SnapKV | 32.0 | 55.5 | 31.3 | 53.8 | 43.3 | 29.3 | 37.0 | 15.7 | 25.4 | 13.8 | 48.3 | 33.7 | 8.5 | 100.0 | 23.6 | **29.2** | 36.3 |
| AdaKV | 32.7 | 55.5 | **31.7** | **54.7** | 44.0 | 28.9 | 37.9 | **15.8** | 26.1 | **14.5** | 48.3 | **34.4** | **10.6** | 100.0 | 24.1 | 29.1 | 36.8 |
| PyramidKV | 31.9 | 55.4 | 31.6 | 54.0 | 43.6 | 28.4 | 37.0 | 15.6 | 26.3 | 14.1 | 48.2 | 33.9 | 9.1 | 100.0 | **24.2** | 29.1 | 36.4 |
| Quest | 33.2 | 56.0 | 31.2 | 54.6 | 44.2 | 28.6 | 37.1 | 15.5 | 28.6 | 14.4 | 48.2 | 34.1 | 8.7 | 100.0 | 24.1 | 28.6 | 36.7 |
| Keyformer | 26.3 | 43.3 | 30.9 | 53.1 | 44.3 | 26.7 | 35.3 | 14.3 | 27.5 | 11.3 | 48.8 | 33.3 | 8.8 | 99.5 | 21.3 | 26.5 | 34.5 |
| **FIDKV** | **33.5** | **56.6** | 31.1 | 54.1 | **44.6** | **29.8** | **37.9** | 15.6 | **29.4** | 14.1 | **48.8** | 33.1 | 8.8 | **100.0** | 23.4 | 28.5 | **36.8** |
| *Mistral @ 10%* | | | | | | | | | | | | | | | | | |
| StreamingLLM | 13.2 | 28.7 | 23.8 | 34.3 | 20.2 | 15.7 | 23.9 | 18.0 | 21.6 | 3.1 | 44.6 | 46.7 | **7.5** | 57.5 | 12.0 | 13.1 | 24.0 |
| H2O | 13.3 | 29.0 | 19.4 | 34.8 | 19.6 | 15.3 | 24.1 | 17.8 | 21.4 | 3.3 | 44.7 | 46.5 | **7.5** | 57.5 | 11.9 | 13.0 | 23.7 |
| SnapKV | 19.1 | 41.6 | 23.0 | 39.6 | 24.9 | 19.6 | 29.6 | 20.2 | 19.9 | 3.0 | **45.5** | 46.6 | 4.5 | 98.5 | 12.3 | 14.6 | 28.9 |
| AdaKV | 20.2 | 42.2 | 23.4 | 40.6 | 24.0 | 19.8 | 28.2 | 19.9 | 19.7 | 3.0 | 45.2 | 47.3 | 5.5 | 98.0 | **13.0** | **15.4** | 29.1 |
| PyramidKV | 18.6 | 41.4 | **25.7** | 39.5 | 23.7 | 19.6 | 28.3 | 19.8 | 19.8 | 3.0 | 44.4 | 46.3 | 5.5 | 98.0 | 12.4 | 14.1 | 28.8 |
| Quest | 21.3 | 40.8 | 24.2 | **41.6** | **26.5** | 19.6 | 31.0 | 20.5 | 25.3 | **3.6** | 44.3 | 45.9 | 5.5 | 97.5 | 11.3 | 13.9 | 29.5 |
| Keyformer | 15.8 | 32.3 | 19.8 | 35.2 | 23.1 | 17.0 | 26.1 | 18.5 | 24.2 | 3.5 | 43.6 | 45.3 | 4.5 | 37.0 | 10.8 | 12.3 | 23.0 |
| **FIDKV** | **21.3** | **43.1** | 23.2 | 40.6 | 26.3 | **20.3** | **31.1** | **21.4** | **27.6** | 3.1 | 44.8 | **47.8** | 6.5 | **98.5** | 11.8 | 13.4 | **30.1** |
| *Mistral @ 20%* | | | | | | | | | | | | | | | | | |
| StreamingLLM | 15.6 | 30.4 | 25.7 | 35.2 | 21.2 | 16.2 | 27.7 | 18.9 | 25.0 | 3.3 | 44.3 | 47.1 | **7.0** | 69.0 | 12.9 | 13.3 | 25.8 |
| H2O | 15.7 | 30.6 | 20.6 | 35.2 | 21.1 | 16.6 | 28.0 | 18.9 | 24.9 | 3.6 | 44.4 | 47.1 | **7.0** | 69.5 | 12.8 | 13.6 | 25.6 |
| SnapKV | 21.5 | 44.1 | 23.3 | 39.9 | 26.0 | 20.1 | 31.0 | 21.3 | 25.2 | 3.0 | 44.0 | 46.5 | 6.0 | 98.0 | 13.7 | 14.9 | 29.9 |
| AdaKV | 22.1 | 43.5 | 22.9 | 40.6 | 25.7 | 20.0 | 31.5 | 21.4 | 25.1 | 3.0 | 44.6 | 46.9 | 6.0 | 98.0 | **14.1** | 14.7 | 30.0 |
| PyramidKV | 21.1 | **44.5** | **26.8** | 40.3 | 25.7 | 20.2 | 30.9 | 20.9 | 24.9 | 3.1 | 44.5 | 46.6 | 5.5 | 98.0 | 13.8 | 14.7 | 30.1 |
| Quest | 23.1 | 43.6 | 24.4 | **40.9** | 26.2 | 20.2 | 33.2 | 21.0 | 27.8 | 3.5 | 44.3 | 46.9 | 6.5 | 97.5 | 12.9 | 13.8 | 30.4 |
| Keyformer | 18.1 | 35.4 | 21.3 | 36.7 | 24.1 | 18.0 | 29.2 | 19.8 | 26.2 | 3.3 | 44.5 | 45.7 | 6.5 | 46.0 | 11.9 | 13.4 | 25.0 |
| **FIDKV** | **23.6** | 43.8 | 23.3 | 40.7 | **26.2** | **20.2** | **33.3** | **22.3** | **28.6** | **3.6** | **45.0** | **47.3** | 6.0 | **98.0** | 13.7 | **14.9** | **30.7** |
| *Qwen-3 @ 10%* | | | | | | | | | | | | | | | | | |
| StreamingLLM | 23.8 | 32.1 | 22.1 | 43.0 | 30.3 | 19.3 | 23.0 | 11.8 | 19.5 | 2.3 | 45.1 | 39.2 | 1.0 | 41.5 | 19.3 | 24.7 | 24.9 |
| H2O | 23.8 | 32.4 | 21.7 | 43.1 | 30.3 | 19.4 | 22.9 | 11.9 | 19.8 | 2.3 | 44.9 | 39.1 | 1.5 | 43.5 | 19.2 | 24.3 | 25.0 |
| SnapKV | 33.3 | **51.1** | 25.7 | 54.9 | 37.0 | **29.9** | 26.6 | **14.2** | 17.1 | 2.4 | 46.8 | 39.9 | 2.0 | 100.0 | 20.1 | **28.6** | 33.1 |
| AdaKV | **33.8** | 50.4 | **26.2** | 55.1 | **37.4** | 29.1 | 27.3 | **14.2** | 17.3 | 2.5 | 46.8 | 39.7 | 1.5 | 100.0 | 20.3 | **28.6** | 33.1 |
| PyramidKV | 33.1 | 50.0 | 25.7 | **55.2** | 37.1 | 29.7 | 26.0 | 13.5 | 17.2 | 2.4 | 47.1 | 39.6 | 1.5 | 100.0 | 20.1 | 27.3 | 32.9 |
| Quest | 33.1 | 47.3 | 25.3 | 54.1 | 35.3 | 27.8 | 27.4 | 13.3 | 22.3 | 2.5 | 46.8 | 39.2 | 1.5 | 98.0 | 21.3 | 24.0 | 32.5 |
| Keyformer | 25.1 | 37.2 | 23.4 | 46.7 | 33.1 | 24.1 | 24.4 | 12.6 | 21.8 | 2.2 | 45.9 | 37.9 | 2.0 | 70.0 | 20.1 | 23.2 | 28.1 |
| **FIDKV** | 33.0 | 49.4 | 26.0 | 54.6 | 37.0 | 28.2 | **28.7** | 14.0 | **24.5** | **2.4** | **47.1** | **39.7** | **2.1** | **100.0** | **21.8** | 26.3 | **33.4** |
| *Qwen-3 @ 20%* | | | | | | | | | | | | | | | | | |
| StreamingLLM | 25.5 | 32.8 | 23.3 | 45.0 | 30.3 | 19.9 | 26.3 | 12.1 | 23.0 | 2.1 | 46.2 | 39.1 | **2.0** | 51.0 | 20.4 | 26.2 | 26.6 |
| H2O | 25.4 | 32.9 | 23.4 | 45.3 | 30.3 | 20.2 | 26.7 | 12.1 | 23.0 | 2.1 | 45.9 | 39.2 | 1.5 | 51.0 | 20.2 | 26.6 | 26.6 |
| SnapKV | 34.9 | 49.7 | 26.4 | 55.2 | **39.1** | 29.3 | 28.6 | 14.1 | 23.4 | 2.5 | 47.2 | 39.7 | 1.0 | 100.0 | 22.8 | 28.6 | 33.9 |
| AdaKV | **35.7** | 50.2 | 26.4 | **55.8** | 39.0 | 29.3 | 29.2 | **14.4** | 23.6 | 2.4 | 46.8 | 40.0 | 1.5 | 100.0 | **23.4** | **29.0** | 34.2 |
| PyramidKV | 34.1 | 50.1 | **26.6** | 55.0 | 38.6 | **29.8** | 27.4 | 14.2 | 22.8 | 2.4 | 46.8 | 40.6 | 1.5 | 100.0 | 23.1 | 28.4 | 33.8 |
| Quest | 35.0 | 49.2 | 25.4 | 54.4 | 37.0 | 28.1 | 28.9 | 14.2 | 25.8 | 2.5 | 46.9 | 40.5 | 1.5 | 99.5 | 23.2 | 25.9 | 33.6 |
| Keyformer | 27.6 | 39.2 | 23.8 | 47.8 | 35.2 | 23.8 | 27.1 | 12.8 | 24.3 | 2.5 | 46.1 | 39.1 | **2.0** | 77.5 | 22.3 | 25.5 | 29.8 |
| **FIDKV** | 35.3 | **50.2** | 26.1 | 54.9 | 38.1 | 28.9 | **30.1** | 14.0 | **26.9** | **2.3** | **47.1** | **40.8** | 1.0 | **100.0** | 22.4 | 27.7 | **34.1** |

Table 6: **Full-cache (uncompressed) LongBench reference for all three models**, under the same protocol as Tables 1 and 5.

| Model | Qasper | MFQA | NrtvQA | HpQA | 2WkMQ | MuSQ | GovRpt | QMSum | MNws | TREC | TrvQA | SAMSum | PC | PR | LCC | RB-P | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LLaMA-3.1-8B | 33.73 | 55.75 | 31.42 | 54.21 | 44.86 | 29.48 | 41.61 | 16.56 | 32.33 | 13.89 | 49.17 | 34.31 | 9.10 | 100.00 | 24.67 | 29.79 | 37.55 |
| Mistral-7B | 25.91 | 43.99 | 23.34 | 41.79 | 27.07 | 20.84 | 39.54 | 23.27 | 32.59 | 3.30 | 44.47 | 48.16 | 5.50 | 97.50 | 18.21 | 18.66 | 32.13 |
| Qwen3-8B | 35.73 | 51.16 | 26.17 | 55.05 | 38.51 | 29.52 | 31.78 | 14.25 | 30.72 | 2.42 | 47.01 | 40.40 | 1.00 | 100.00 | 23.36 | 28.30 | 34.71 |

### B.2 LONGBENCH: LLAMA-3.1 AT 30%–50% RETAINED CACHE

As the budget increases, most queries already retain sufficient attention mass, and FIDKV moves toward the strongest comparison methods rather than creating a larger gap (Table 7).

Table 7: LongBench F1 (%) on LLaMA-3.1-8B-Instruct at 30%–50% retained cache. **FIDKV** remains at or near the strongest comparison methods. Best per column (within sub-block) bold; FIDKV within 0.2 percentage points of the best is treated as a tie.

| Method | Single-Doc QA | | | Multi-Doc QA | | | Summarization | | | Few-shot | | | Synthetic | | Code | | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Qasper | MFQA | NrtvQA | HpQA | 2WkMQ | MuSQ | GovRpt | QMSum | MNws | TREC | TrvQA | SAMSum | PC | PR | LCC | RB-P | |
| *LLaMA-3 @ 30%* | | | | | | | | | | | | | | | | | |
| StreamingLLM | 26.7 | 36.5 | 27.2 | 52.9 | 42.7 | 24.1 | 35.6 | 13.0 | 28.5 | 10.8 | 48.4 | 33.2 | **9.5** | 92.0 | 22.3 | 27.2 | 33.1 |
| H2O | 27.2 | 36.6 | 28.5 | 52.5 | 42.2 | 23.5 | 35.7 | 13.3 | 28.4 | 10.2 | 48.4 | 33.6 | 7.8 | 92.0 | 22.1 | 27.4 | 33.1 |
| SnapKV | 32.6 | 55.0 | 31.5 | 53.7 | 44.1 | 28.3 | 38.0 | 15.8 | 27.5 | 14.0 | 48.2 | 33.5 | 8.0 | 100.0 | 24.5 | 29.0 | 36.5 |
| AdaKV | 33.7 | 55.7 | **31.6** | **54.7** | 43.6 | **29.7** | 38.7 | **16.1** | 28.8 | 14.9 | 48.6 | **33.8** | 8.9 | 100.0 | **24.8** | 28.9 | 37.0 |
| PyramidKV | 32.6 | 55.7 | 31.1 | 54.6 | 43.1 | 28.4 | 38.0 | 15.8 | 27.4 | 13.8 | 48.6 | **33.8** | 9.0 | 100.0 | 24.4 | 28.8 | 36.6 |
| Quest | 33.5 | 56.3 | 31.4 | 54.5 | 44.6 | 29.3 | 38.2 | 15.6 | 29.6 | 14.6 | 48.8 | 33.6 | 8.8 | 99.5 | **24.8** | 29.0 | 37.0 |
| Keyformer | 28.3 | 46.4 | 30.5 | 53.5 | 42.4 | 27.7 | 35.8 | 14.5 | 28.9 | 12.7 | **48.9** | 33.5 | 9.3 | 99.0 | 22.3 | 26.9 | 35.0 |
| **FIDKV** | **34.3** | **57.5** | 31.4 | 53.9 | **45.3** | 29.2 | **38.6** | 15.9 | **30.6** | **15.1** | 48.3 | 33.3 | 8.1 | **100.0** | 24.2 | **29.2** | **37.2** |
| *LLaMA-3 @ 40%* | | | | | | | | | | | | | | | | | |
| StreamingLLM | 27.8 | 37.6 | 29.2 | 51.8 | 42.4 | 24.5 | 36.5 | 14.2 | 29.6 | 12.0 | 48.3 | 33.8 | **10.2** | 88.5 | 22.4 | 28.1 | 33.6 |
| H2O | 28.2 | 37.0 | 29.6 | 52.2 | 41.3 | 24.9 | 37.0 | 14.0 | 29.9 | 11.2 | 48.2 | **34.0** | 8.8 | 89.0 | 22.6 | 28.1 | 33.5 |
| SnapKV | 33.8 | 56.7 | **31.7** | 54.3 | 43.6 | 29.0 | 38.7 | 15.9 | 29.4 | **14.7** | 48.3 | 33.8 | 9.2 | 100.0 | **24.9** | **29.3** | 37.1 |
| AdaKV | 33.8 | 55.6 | 31.3 | 54.0 | 44.6 | 29.6 | 38.8 | 16.3 | 29.6 | 14.5 | 48.3 | 33.6 | 9.9 | 100.0 | 24.2 | 29.2 | 37.1 |
| PyramidKV | 33.7 | 55.6 | 30.0 | **54.7** | 43.2 | 29.7 | 38.7 | 16.1 | 29.4 | **14.7** | 48.3 | 33.5 | 9.3 | 100.0 | 24.5 | 29.2 | 36.9 |
| Quest | 32.8 | 56.8 | 31.5 | 54.4 | 44.8 | 29.5 | **39.4** | 16.0 | 30.6 | 14.2 | 47.9 | 33.6 | 8.7 | 99.5 | 24.3 | 28.8 | 37.0 |
| Keyformer | 27.9 | 50.7 | 29.3 | 53.8 | 42.8 | 27.5 | 36.7 | 15.4 | 29.8 | 13.1 | **49.4** | 33.2 | 9.7 | 99.5 | 24.2 | 27.4 | 35.6 |
| **FIDKV** | **34.1** | **57.6** | 31.2 | 53.9 | **44.9** | **29.6** | 38.7 | **16.4** | **30.9** | 14.3 | 48.8 | 33.7 | 8.6 | **100.0** | 23.8 | 28.7 | **37.2** |
| *LLaMA-3 @ 50%* | | | | | | | | | | | | | | | | | |
| StreamingLLM | 30.6 | 39.8 | 28.7 | 53.1 | 42.5 | 26.7 | 37.9 | 14.4 | 30.5 | 13.8 | 48.3 | 33.8 | 8.2 | 85.5 | 23.0 | 27.8 | 34.0 |
| H2O | 30.7 | 39.7 | 30.0 | 52.7 | 42.2 | 26.5 | 37.9 | 14.6 | 30.5 | 13.7 | 48.0 | 33.7 | 10.2 | 84.5 | 22.9 | 27.9 | 34.1 |
| SnapKV | 33.6 | 56.0 | 31.1 | 54.3 | 44.1 | 29.1 | 39.2 | 16.3 | 30.3 | 14.6 | 48.6 | 34.3 | 9.8 | 100.0 | 24.0 | 29.5 | 37.2 |
| AdaKV | 33.6 | 56.5 | 31.0 | 54.2 | 44.0 | 29.5 | 39.8 | **16.4** | 31.0 | 14.5 | 48.6 | 33.7 | 8.9 | 100.0 | **24.4** | 29.4 | 37.2 |
| PyramidKV | 33.5 | 55.3 | 30.1 | **54.5** | **44.9** | 29.4 | 39.9 | 16.2 | 30.4 | 13.8 | 48.7 | 34.1 | 9.5 | 100.0 | 23.7 | 29.1 | 37.1 |
| Quest | 33.5 | 56.6 | **31.5** | 54.0 | 44.4 | **29.6** | **40.2** | **16.4** | 31.1 | **15.2** | 48.8 | 33.5 | **11.0** | 100.0 | 24.2 | 28.5 | 37.4 |
| Keyformer | 29.9 | 51.1 | 31.1 | 54.3 | 43.4 | 26.7 | 37.6 | 15.8 | 30.2 | 14.4 | **49.2** | 33.9 | 9.8 | 99.5 | 24.1 | 28.3 | 36.2 |
| **FIDKV** | **33.7** | **56.7** | 31.1 | 54.2 | 44.4 | 29.3 | 39.8 | 15.8 | **31.8** | 14.2 | 48.7 | **34.3** | 9.3 | **100.0** | 23.5 | **29.5** | **37.3** |

### B.3 EFFICIENCY DETAILS

Table 8 gives the complete efficiency measurements and Figure 5 the latency by input length.

Table 8: **Efficiency on LongBench at 10% and 20% KV budgets.** Time is end-to-end elapsed time over the LongBench run, Mem is peak GPU memory, and Tok/s is aggregated input-token throughput. "×vs Snap" is the wall-clock speed ratio SnapKV time / method time, so values below 1 mean slower than SnapKV.

| | | Budget=10% | | | | Budget=20% | | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | Method | Time (s) | Mem (MB) | Tok/s | ×vs Snap | Time (s) | Mem (MB) | Tok/s | ×vs Snap |
| LLaMA3 | StreamingLLM | 4845 | 36103 | 4691 | 1.41 | 4734 | 36106 | 4856 | 1.36 |
| | H2O | 5368 | 34573 | 4110 | 1.27 | 5315 | 34576 | 4148 | 1.21 |
| | SnapKV | 6825 | 34571 | 3916 | 1.00 | 6416 | 34571 | 4112 | 1.00 |
| | AdaKV | 9432 | 34571 | 3541 | 0.72 | 9031 | 34572 | 3684 | 0.71 |
| | PyramidKV | 6088 | 36100 | 4499 | 1.12 | 5787 | 36101 | 4703 | 1.11 |
| | Quest | 9262 | 36512 | 3978 | 0.74 | 9411 | 36512 | 3898 | 0.68 |
| | Keyformer | 4756 | 36144 | 4686 | 1.43 | 4763 | 36144 | 4761 | 1.35 |
| | **FIDKV** | 6330 | 34652 | 3572 | 1.08 | 6212 | 34652 | 3686 | 1.03 |
| Mistral | StreamingLLM | 6819 | 35482 | 4272 | 1.36 | 6836 | 35486 | 4217 | 1.33 |
| | H2O | 7731 | 34499 | 3600 | 1.20 | 8163 | 34502 | 3447 | 1.11 |
| | SnapKV | 9293 | 34496 | 3473 | 1.00 | 9099 | 34497 | 3566 | 1.00 |
| | AdaKV | 12480 | 34496 | 3209 | 0.74 | 12928 | 34497 | 3132 | 0.70 |
| | PyramidKV | 8500 | 35479 | 4036 | 1.09 | 8614 | 35480 | 3972 | 1.06 |
| | Quest | 13253 | 36006 | 3200 | 0.70 | 13271 | 36006 | 3183 | 0.69 |
| | Keyformer | 6692 | 35562 | 4152 | 1.39 | 7225 | 35562 | 3987 | 1.26 |
| | **FIDKV** | 9684 | 34595 | 2958 | 0.96 | 9755 | 34595 | 2919 | 0.93 |
| Qwen3 | StreamingLLM | 5171 | 38168 | 4681 | 1.37 | 5478 | 38172 | 4530 | 1.27 |
| | H2O | 5993 | 36445 | 4082 | 1.18 | 6257 | 36449 | 3939 | 1.12 |
| | SnapKV | 7074 | 36443 | 3919 | 1.00 | 6981 | 36443 | 4012 | 1.00 |
| | AdaKV | 9576 | 36443 | 3555 | 0.74 | 9556 | 36444 | 3561 | 0.73 |
| | PyramidKV | 6190 | 38166 | 4517 | 1.14 | 6438 | 38166 | 4408 | 1.08 |
| | Quest | 9674 | 38638 | 3507 | 0.73 | 10464 | 38639 | 3263 | 0.67 |
| | Keyformer | 5413 | 38212 | 4343 | 1.31 | 5973 | 38212 | 4158 | 1.17 |
| | **FIDKV** | 7413 | 36536 | 3414 | 0.95 | 7471 | 36536 | 3347 | 0.93 |

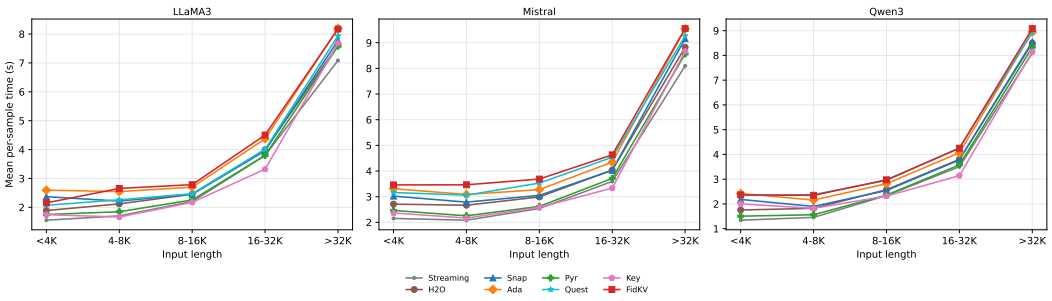



Figure 5: **End-to-end inference latency by input length (LongBench, budget 10%).** Timings include prefill and decoding.

## C MATCHED CONTROL FOR THE FIDELITY CORRECTION

Differences against other eviction methods mix the scoring rule with other design choices (observation windows, head allocation, page granularity). To isolate the correction, we compare FIDKV with a *matched cumulative-attention control* (CumAttn) that removes only the $1/\hat{w}_t$ weighting while fixing the model, cache size, prompts, examples, accumulator, and eviction schedule. Holding the schedule fixed also removes eviction frequency as a confound. Table 9 reports paired 95% bootstrap intervals.

On all 16 LongBench tasks, the correction preserves the answer accuracy of the matched control at both budgets (paired intervals include zero). Under the official generation limits, 13 of the 16 tasks finish decoding before scores updated with the correction can influence another eviction, so on most

Table 9: **Matched control: FIDKV vs. the same policy with the $1/\hat{w}_t$ weighting removed (CumAttn)**, LLaMA-3.1-8B-Instruct. All intervals are paired 95% bootstrap intervals. Top: all 16 LongBench tasks, 50 paired examples per task. Bottom: repeated-selection evaluation on the three LongBench tasks with the official 512-token generation cap, using fixed-teacher replay (equal cache sizes, a shared full-cache teacher trajectory, each policy evolving its own scores and retained positions), 60 paired examples per budget. pp = percentage points.

| | 10% cache | 20% cache |
|---|---|---|
| LongBench F1, FIDKV − CumAttn | $-0.051$ [$-0.163$, $+0.059$] | $-0.0004$ [$-0.333$, $+0.276$] |
| Log-fidelity objective (exp., rel. gain) | $+6.0\%$ [$+0.9\%$, $+11.2\%$] | $+8.6\%$ [$+5.6\%$, $+11.7\%$] |
| Attn.-output error reduction (pp) | 0.050 [0.026, 0.075] | 0.014 [0.003, 0.026] |

of LongBench the two policies make nearly the same selections. Where corrected scores do feed later evictions, the correction improves the exponentiated cumulative log-fidelity objective by 6.0% and 8.6% and reduces full-attention output-approximation error, with intervals excluding zero at both budgets. The matched control thus confirms that fidelity weighting is a faithful, accuracy-preserving implementation of the log-fidelity objective whose benefits appear once corrected scores influence later eviction decisions.

## D IMPLEMENTATION AND REPRODUCIBILITY

PyTorch 2.x, `bfloat16`, 8×A100-80GB GPUs. KV caches are allocated once and compacted in place at eviction events. The stability constant is $\epsilon = 10^{-6}$. The evaluation aggregates more than $4{,}400$ unique benchmark runs after removing duplicate records.

**Uncertainty.** All confidence intervals are paired 95% bootstrap intervals: the two policies are evaluated on the same examples, and examples are resampled with replacement (stratified by task for multi-task aggregates), recomputing the difference of the aggregate metric on each resample. No other significance test is used in this paper.

**Matched control and fixed-teacher replay.** The CumAttn control shares every component of FIDKV except the $1/\hat{w}_t$ factor. In fixed-teacher replay, both policies are driven by the same full-cache teacher trajectory, so differences in cache state are not caused by diverging generated text, while each policy maintains its own scores and retained positions across repeated evictions.