COMPKV: COMPENSATION-AWARE KV SELECTION FOR LONG-CONTEXT LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite their strong performance, large language models (LLMs) are bottlenecked by KV cache memory traffic during long-context inference. Sparse attention is widely used to accelerate LLM inference by computing exact attention over a selected subset of tokens. To recover the contribution of tokens excluded from exact attention, recent methods apply coarse-grained compensation to the omitted attention tail. However, existing methods typically select tokens based on attention mass and only then compensate for the unselected tokens. This decoupled design overlooks their interaction: selection should prioritize tokens that would leave the largest compensation error if omitted. To address this limitation, we introduce CompKV, the first compensation-aware sparse attention framework that divides tokens into blocks and explicitly optimizes selection for the downstream compensation mechanism. Our theoretical analysis shows that the residual left by block-level mean compensation is governed by both block attention mass and within-block logit variation. We approximate this residual using compact block-level statistics, yielding a deployable selection criterion. We further develop an efficient asynchronous implementation. Experiments on RULER and LongBench-Pro show that CompKV performs best among the evaluated sparse baselines while delivering up to a $6.85\times$ self-attention speedup over full attention.

1 INTRODUCTION

Large language models increasingly operate over long and evolving contexts for document understanding, software engineering, and agentic reasoning (Grattafiori et al., 2024; Yang et al., 2025; Jimenez et al., 2024; Bai et al., 2024; Zhou et al., 2024). During autoregressive decoding, however, the KV cache grows with context length, and dense attention reads the entire cache at every step, making long-context inference increasingly bottlenecked by KV cache memory traffic.

To alleviate this bottleneck, query-aware sparse attention reduces this traffic by reading only a small set of tokens selected for the current query (Xiao et al., 2024a; Ribar et al., 2024; Wang et al., 2026). More recent methods divide tokens into blocks (Tang et al., 2024; Liu et al., 2025; Deng et al., 2026) and further compensate for the attention contributions of omitted blocks using compact token summaries (Hooper et al., 2025; Yang et al., 2026; Fan et al., 2026). Despite recent progress in compensation, selection and reconstruction remain largely **decoupled**: blocks are ranked by predicted relevance or attention mass before reconstruction is considered. This design overlooks how well each block can be reconstructed by the compensator. A high-mass block may be easy to reconstruct, whereas a lower-mass block with greater within-block logit variation may leave a larger error. Therefore, selection should prioritize the blocks that have a larger estimated reconstruction error.

Motivated by this mismatch, we introduce CompKV, a training-free, compensation-aware KV selection method that prioritizes blocks that are difficult to reconstruct, as shown in Figure 1. To the best of our knowledge, CompKV is the first sparse attention framework to make block selection explicitly aware of the error induced by the downstream compensation mechanism. We formulate full block attention and its summary-based approximation as distributions whose KL divergence reduces to a log-partition gap. **For block-mean reconstruction, our analysis reveals that the residual contributed by an omitted block depends jointly on its attention mass and within-block logit variation.** Based on this result, CompKV estimates the post-compensation residual and then converts compensation-aware selection into a query-dependent block-ranking problem.

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.

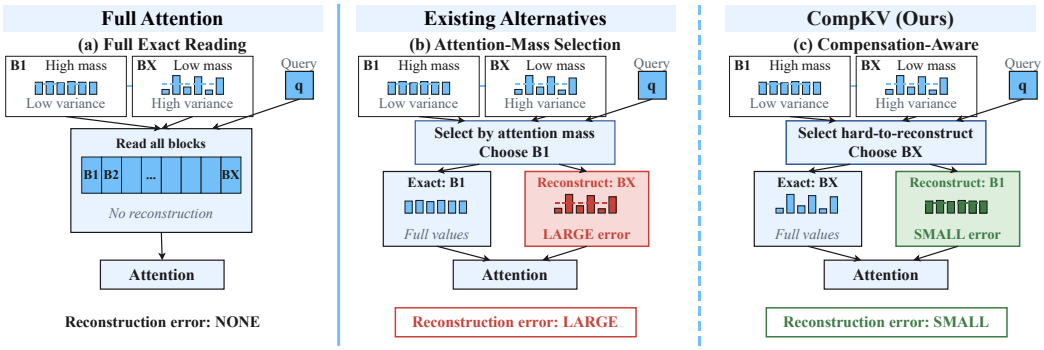


Figure 1: Comparison between full attention (a), attention-mass selection (b), and CompKV (c) on two illustrative blocks B1 and BX. Mass-based selection may spend exact reads on blocks already well approximated, whereas CompKV targets blocks with the largest estimated reconstruction error.

CompKV maintains compact mean and grouped-variance statistics for each KV block to estimate its post-compensation residual. For each query, these statistics approximate the block attention mass and within-block logit variation, which are combined into a selection score. Blocks with the largest scores receive exact attention, while the rest contribute through their mean summaries; both contributions are jointly normalized to produce the final output. For efficient long-context inference, we further develop an asynchronous CPU-offload implementation that overlaps summary transfer and compensation with block selection, CPU-side KV gathering, and selected-KV transfer.

We evaluate CompKV in terms of both accuracy and efficiency. For accuracy, we benchmark CompKV on RULER and LongBench-Pro using Llama-3.1-8B-Instruct, Qwen3-8B, and Qwen3-32B. Across both benchmarks, CompKV achieves the highest average scores among the evaluated sparse methods for all three models. For efficiency, we benchmark the CPU-offload single-layer pipeline on an NVIDIA H100 GPU. CompKV achieves the lowest mean decoding-step latency among the evaluated methods for all nine combinations of 32K-128K context lengths and 512-2,048 token budgets, delivering up to $6.85\times$ speedups over full attention.

Our contributions are:

- We identify the mismatch between selection and compensation and present the first formulation that explicitly models selection with respect to compensation. Our analysis shows that the post-compensation residual is jointly governed by attention mass and within-block logit variation.
- Based on this analysis, we introduce CompKV, the first training-free, compensation-aware KV selection method that constructs a query-dependent score from compact block statistics, along with an efficient asynchronous implementation to accelerate long-context inference.
- We comprehensively evaluate the accuracy and efficiency of CompKV. Across three widely used models and two long-context benchmarks, CompKV achieves the highest average scores among the evaluated baselines while delivering up to $6.85\times$ self-attention speedups over full attention.

2 Related Work

KV Cache Reduction for Long-Context Inference. To accelerate long-context inference, eviction methods bound the cache using recency, accumulated attention, heavy-hitter statistics, or prompt-side importance estimates (Xiao et al., 2024b; Liu et al., 2023; Zhang et al., 2023; Li et al., 2024; Cai et al., 2025; Feng et al., 2025). Complementary approaches reduce retained-state costs through low-rank compression (Saxena et al., 2024; Singhania et al., 2024; Chang et al., 2025), quantization (Liu et al., 2024; Hooper et al., 2024; Kang et al., 2024; Li et al., 2025), or memory management (Kwon et al., 2023; Qu et al., 2025). These methods primarily reduce the amount or representation cost of retained KV states. In contrast, our work focuses on deciding which KV blocks should be read exactly for each query under a fixed per-step read budget.

Query-Aware and Block-Level Sparse Attention. Query-aware sparse attention saves all KVs and conditions exact KV reads on the current decoding query. For example, Quest (Tang et al., 2024)

ranks pages with query-dependent upper bounds derived from compact key statistics, while InfiniGen (Lee et al., 2024), ShadowKV (Sun et al., 2025), SpeCache (Jie et al., 2025) and RetrievalAttention (Liu et al., 2025) retrieve candidate blocks using different scoring mechanisms. However, these methods primarily rank blocks according to their predicted attention mass for the current query.

Tail Compensation and Reconstruction. Tail-compensation methods approximate omitted KV states or their aggregate attention contribution. SparQ (Ribar et al., 2024) reallocates estimated omitted mass to a running mean value, RESA (Yang et al., 2026) reconstructs the tail from a low-rank logit prior, and ResKV (Zhan et al., 2026) retains compact moment or residual summaries for compressed caches. These methods partly reconstruct the tail without extra exact KV reads.

Nevertheless, existing block selection and tail compensation are largely treated as separate stages, as mentioned in Section 1. In contrast, CompKV explicitly selects blocks according to their estimated post-compensation residual, rather than simply those with the largest predicted attention mass.

3 Compensation-Aware Selection Formulation

In this section, we formulate sparse attention selection and derive a compensation-aware selection criterion. We define Mean compensation as **replacing every logit in an unselected block by its block mean**. We use Mean compensation because its compact summaries yield an exact log-partition identity, making the interaction between selection and compensation analytically tractable. The compensation-aware formulation itself is not restricted to Mean compensation. Based on this, we analyze the residual introduced by compensation and show that the optimal selection priority depends jointly on a block's attention mass and the variation of its within-block attention logits.

3.1 Mean Compensation Model and Selection Objective

Attention and Mean Compensation. At each decoding step, we consider one KV head shared by G query heads under grouped-query attention (Ainslie et al., 2023). Let $\boldsymbol{q}_g, \boldsymbol{k}_j \in \mathbb{R}^d$ denote the query of head $g \in \{1, \dots, G\}$ and the key at token position j, respectively, where d is the head dimension. The corresponding attention logit is $z_{g,j} = \boldsymbol{q}_g \boldsymbol{k}_j^\top / \sqrt{d}$. Following Tang et al. (2024), we divide the KV cache into contiguous blocks of a fixed size, starting from the first token. Let $\mathcal{B}$ denote the set of nonempty blocks, where each block $b \in \mathcal{B}$ contains $|b|$ tokens. Its unnormalized attention weight is $Z_{g,b} = \sum_{j \in b} e^{z_{g,j}}$, giving the full attention probability $P_g = \mathrm{softmax}_j(z_{g,j})$.

Given a selected block set $\mathcal{S} \subseteq \mathcal{B}$ shared by the G query heads, for each token position $j \in b$, the logit under Mean compensation is

$$\hat{z}_{g,j} = \begin{cases} z_{g,j}, & b \in \mathcal{S}, \\ \bar{z}_{g,b}, & b \notin \mathcal{S}, \end{cases} \tag{1}$$

where $\bar{z}_{g,b} = |b|^{-1} \sum_{j \in b} z_{g,j}$ is the block-mean logit. Therefore, applying softmax jointly over all token positions defines the compensated distribution $\widehat{P}_g(\mathcal{S}) = \mathrm{softmax}_j(\hat{z}_{g,j})$.

Optimization objective. For Mean compensation, we define the selection loss as $\mathcal{L}(\mathcal{S}) = \sum_{g=1}^{G} D_{\mathrm{KL}}\Big(\widehat{P}_g(\mathcal{S}) \| P_g\Big)$, summing over the query heads that share the selected set. With a budget of $K \le |\mathcal{B}|$ exact block reads, our objective is

$$\min_{\mathcal{S} \subseteq \mathcal{B},\, |\mathcal{S}| = K} \mathcal{L}(\mathcal{S}). \tag{2}$$

3.2 Deriving the Compensation-Aware Selection Criterion

For each query head g, define the attention mass and logit variance of block b as

$$p_{g,b} = \frac{Z_{g,b}}{\sum_{c \in \mathcal{B}} Z_{g,c}}, \qquad \sigma_{g,b}^2 = \frac{1}{|b|} \sum_{j \in b} (z_{g,j} - \bar{z}_{g,b})^2, \tag{3}$$

where c ranges over all available blocks and $\sigma_{g,b} \ge 0$ denotes the corresponding standard deviation.

To solve Equation 2, we expand the KL divergence defined in Section 3.1, yielding

$$\mathcal{L}(\mathcal{S}) = \sum_{g=1}^{G} \log \frac{\sum_{b\in\mathcal{B}} Z_{g,b}}{\sum_{b\in\mathcal{S}} Z_{g,b} + \sum_{b\notin\mathcal{S}} |b| e^{\bar{z}_{g,b}}} = -\sum_{g=1}^{G} \log \left[1 - \sum_{b\notin\mathcal{S}} p_{g,b} \left(1 - \frac{|b| e^{\bar{z}_{g,b}}}{Z_{g,b}}\right)\right]. \tag{4}$$

Appendix A.1 provides the derivation. The ratio $|b|e^{\bar{z}_{g,b}}/Z_{g,b}$ measures the fraction of block b's unnormalized attention weight recovered by Mean compensation. Appendix B.1 provides empirical support for this selection principle: ranking blocks by their exact compensation residuals reduces average attention KL and output error relative to ranking by exact attention mass.

A second-order Taylor expansion around the block-mean logit yields

$$p_{g,b}\left(1 - \frac{|b| e^{\bar{z}_{g,b}}}{Z_{g,b}}\right) = p_{g,b}(\frac{1}{2}\sigma^2_{g,b} + O(\sigma^3_{g,b})). \tag{5}$$

Thus, the leading contribution of each block is proportional to its attention mass multiplied by its logit variance. Appendix A.2 derives this expansion and specifies the expansion regime.

Let $R_g(\mathcal{S})$ denote the inner sum in Equation 4. Applying $-\log(1 - R_g) = R_g + O(R_g^2)$ to each head in Equation 4 then yields

$$\mathcal{L}(\mathcal{S}) = \frac{1}{2}\sum_{g=1}^{G}\sum_{b\notin\mathcal{S}} p_{g,b}(\sigma^2_{g,b} + O\big(\sigma^3_{g,b}\big)). \tag{6}$$

The remainder collects the blockwise expansion errors and the outer-logarithm remainder, which is absorbed into the cubic term in the small-variance regime. Appendix A.3 provides the derivation.

The second-order term is additive across unselected blocks. Since the sum over all blocks is independent of $\mathcal{S}$, minimizing the second-order term of Equation 6 is equivalent to

$$\max_{\mathcal{S}\subseteq\mathcal{B},\,|\mathcal{S}|=K} \sum_{b\in\mathcal{S}}\sum_{g=1}^{G} p_{g,b}\sigma^2_{g,b}. \tag{7}$$

Thus, selecting the K blocks with the largest $\sum_{g=1}^{G} p_{g,b}\sigma^2_{g,b}$ minimizes the second-order objective. The resulting score combines a block's attention mass with the variation that Mean compensation leaves unreconstructed. A block with nearly constant logits contributes little to the loss, even when its attention mass is large. Appendix A.5 also proves that removing Mean compensation from our KL objective recovers the standard Top-K selection rule based on attention mass for a single query head. Section 4.1 estimates both factors from compact block statistics.

4 CompKV

Building on the criterion derived in Section 3, CompKV turns the oracle selection rule into a practical decoding pipeline. It first estimates block attention mass and logit variance from compact statistics, uses their product to allocate exact reads, and represents the remaining blocks with mean summaries under a shared normalization. To realize it efficiently, we develop an implementation that overlaps compensation with KV retrieval and exact attention.

4.1 Block Scoring from Compact Statistics

Grouped key-variance estimation. Computing the exact score in Equation 7 requires reading the keys of every candidate block. CompKV estimates this score from the mean key and grouped key variances stored for each block. The mean key $\bar{\boldsymbol{k}}_b = |b|^{-1}\sum_{j\in b}\boldsymbol{k}_j$ gives the block-mean logit $\bar{z}_{g,b} = \boldsymbol{q}_g\bar{\boldsymbol{k}}_b^\top/\sqrt{d}$ defined in Section 3.1. We partition the d key coordinates into r disjoint groups $\mathcal{G}_t$, indexed by $t = 0, \ldots, r-1$. The vector $\bar{\mathbf{D}}_b \in \mathbb{R}^r$ stores the average coordinate variance in each group. At each decoding step, query head g estimates the block's logit variance as

$$\widehat{\sigma}^2_{g,b} = \frac{1}{d}\sum_{t=0}^{r-1}\left(\sum_{i\in\mathcal{G}_t} q^2_{g,i}\right)(\bar{\mathbf{D}}_b)_t, \qquad (\bar{\mathbf{D}}_b)_t = \frac{1}{|b|\,|\mathcal{G}_t|}\sum_{j\in b}\sum_{i\in\mathcal{G}_t}(k_{j,i} - \bar{k}_{b,i})^2. \tag{8}$$

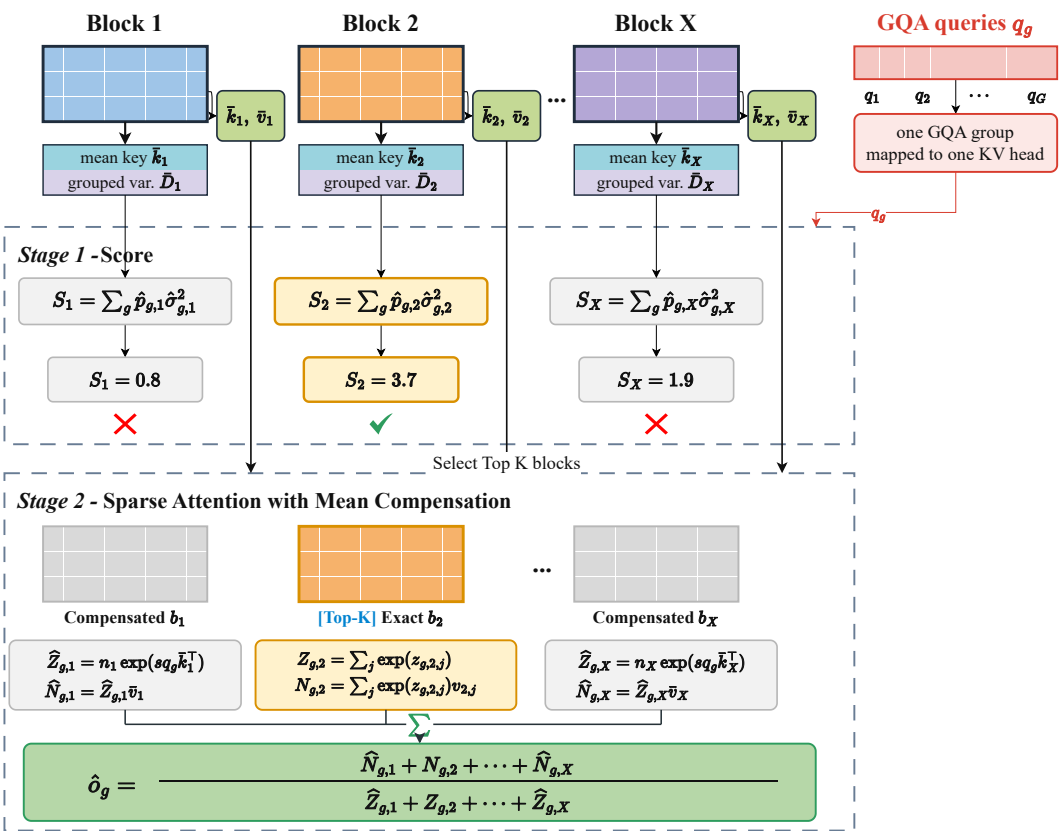


Figure 2: End-to-end CompKV decoding, where $X = |\mathcal{B}|$ is the number of available blocks. Stage 1 ranks blocks with Equation 10. Stage 2 evaluates selected blocks exactly, compensates the remaining blocks with their means, and normalizes all contributions jointly.

Here, $q_{g,i}$, $k_{j,i}$, and $\bar{k}_{b,i}$ denote the ith coordinates of $\boldsymbol{q}_g$, $\boldsymbol{k}_j$, and $\bar{\boldsymbol{k}}_b$, respectively. The parameter r controls the granularity of the stored variances. Appendix A.4 specifies the coordinate groups and derives this estimate from the key covariance within a block along the query.

Attention-mass estimation. The log-partition expansion in Equation 23 gives $\log|b| + \bar{z}_{g,b} + \sigma^2_{g,b}/2$ as the second-order approximation to $\log Z_{g,b}$. Substituting $\widehat{\sigma}^2_{g,b}$ and normalizing the estimated block weights within each query head gives the attention-mass estimate $\widehat{p}_{g,b}$:

$$\widehat{p}_{g,b} = \frac{|b| \exp\left(\bar{z}_{g,b} + \frac{1}{2}\widehat{\sigma}^2_{g,b}\right)}{\sum_{c\in\mathcal{B}} |c| \exp\left(\bar{z}_{g,c} + \frac{1}{2}\widehat{\sigma}^2_{g,c}\right)}. \tag{9}$$

The variance correction accounts for within-block logit variation when estimating the block's unnormalized attention weight. A detailed derivation is provided in Appendix A.4.

Compensation-aware block scoring. Replacing the mass and variance in Equation 7 with these estimates defines the CompKV score S_b for block b:

$$S_b = \sum_{g=1}^{G} \widehat{p}_{g,b}\widehat{\sigma}^2_{g,b}. \tag{10}$$

The two uses of $\widehat{\sigma}^2_{g,b}$ follow from the preceding derivations: the correction inside $\widehat{p}_{g,b}$ estimates the block's attention mass, and the outer variance factor captures the leading dependence of the

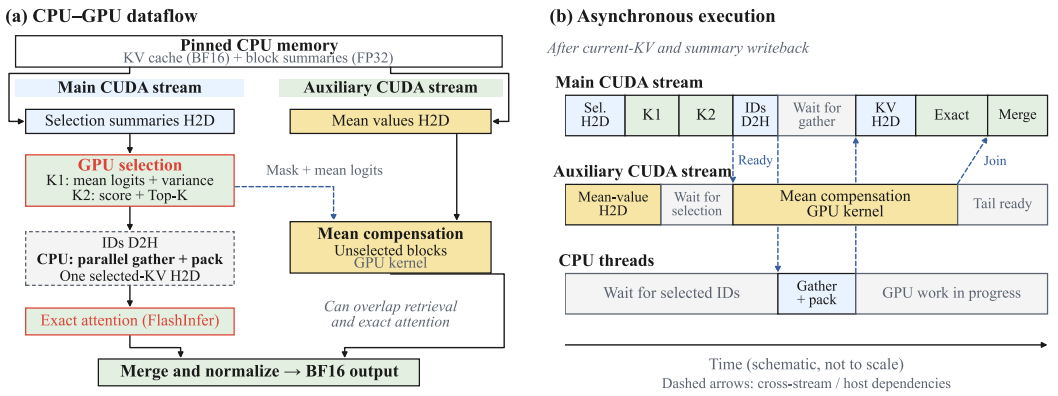


Figure 3: Implementation of CPU-offload CompKV. (a) Dataflow across the main stream, auxiliary stream, and CPU. (b) Asynchronous schedule: after mean values and selection are ready, Mean compensation runs on an auxiliary stream while the main path calculates the exact attention. The branches join at the final merge. Dashed arrows indicate dependencies; durations are schematic.

compensation residual on logit variation. **CompKV computes $\widehat{\sigma}^2_{g,b}$ only once and then reuses it.** To evaluate the effectiveness of the score, we measure how closely it follows exact residual selection using Spearman correlation and residual capture. The results are shown in Appendix B.2.

4.2 Block Selection and Compensated Decoding

Figure 2 shows the end-to-end decoding workflow of CompKV. Mean compensation additionally stores $\bar{\boldsymbol{v}}_b = |b|^{-1} \sum_{j \in b} \boldsymbol{v}_j$, where $\boldsymbol{v}_j \in \mathbb{R}^d$. The complete summary $\{|b|, \bar{\boldsymbol{k}}_b, \bar{\mathbf{D}}_b, \bar{\boldsymbol{v}}_b\}$ contains $2d + r + 1$ scalars per block and is updated incrementally as new K/V states enter the active block.

The mandatory set $\mathcal{F}$ contains the first sink block and the two most recent blocks, counting any overlap once. These blocks count toward the total budget, and Equation 10 ranks the other blocks:

$$\mathcal{S} = \mathcal{F} \cup \mathrm{TopK}_{K-|\mathcal{F}|} \{S_b : b \in \mathcal{B} \setminus \mathcal{F}\}, \qquad |\mathcal{S}| = K. \tag{11}$$

Here, TopK returns the blocks with the highest scores. Selected blocks use their original K/V states, while unselected blocks use their mean summaries. Both contributions are normalized together:

$$\widehat{\mathbf{o}}_g = \frac{\sum_{b \in \mathcal{S}} \sum_{j \in b} e^{z_{g,j}} \boldsymbol{v}_j + \sum_{b \notin \mathcal{S}} |b| e^{\bar{z}_{g,b}} \bar{\boldsymbol{v}}_b}{\sum_{b \in \mathcal{S}} Z_{g,b} + \sum_{b \notin \mathcal{S}} |b| e^{\bar{z}_{g,b}}}. \tag{12}$$

4.3 Efficient Implementation of CompKV

In practical LLM deployments, the KV cache may need to be stored outside GPU memory. Therefore, CompKV keeps the complete BF16 KV cache and FP32 block summaries in pinned CPU memory. Each step uses a fused GPU kernel to update the current block's statistics and pack the new K/V, followed by one device-to-host (D2H) transfer and CPU writeback. The summary buffer separates selection statistics from mean values. The main CUDA stream transfers the selection region and runs two kernels for block statistics and normalized Top-K scoring, while an auxiliary stream independently transfers mean values. The selector reuses grouped variance projection in both factors of Equation 10 to estimate both block mass and residual magnitude.

After selection, the CPU gathers the selected K/V into a contiguous pinned buffer for one host-to-device (H2D) transfer and FlashInfer exact attention. Meanwhile, the auxiliary stream computes Mean compensation once the mean values and selected-block mask are ready, overlapping this computation with CPU gathering, selected-KV transfer, and exact attention. The main stream waits for compensation immediately before merging both contributions under the shared normalization in Equation 12. Figure 3 shows the schedule; Appendix E.1 details implementation and measurement.

Table 1: **RULER accuracy (%).** Bold values mark the best sparse result per model and column, including ties. AVG annotations show drops (↓) from Full in percentage points.

Method	SG1	SG2	SG3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	AVG
Llama-3.1-8B-Instruct														
Full	100.0	100.0	100.0	100.0	99.8	99.2	94.9	99.1	99.1	9.9	93.3	81.4	54.4	87.0
Quest	**100.0**	**99.6**	99.6	**99.8**	92.0	22.4	86.9	96.8	94.4	1.7	80.7	76.2	50.8	77.0↓ 10.0
InfLLM	**100.0**	95.4	78.2	91.0	92.0	38.0	64.0	83.0	90.6	**2.7**	84.4	80.0	51.4	73.1↓ 13.9
Quest+RESA	**100.0**	99.4	99.6	**99.8**	93.8	29.4	**91.8**	**97.2**	97.1	0.1	82.5	75.4	50.6	78.2↓ 8.8
CompKV	**100.0**	**99.6**	**99.8**	98.6	**97.8**	**73.0**	91.2	97.1	**97.2**	0.5	**91.7**	**80.8**	**54.0**	**83.2**↓ 3.8
Qwen3-8B														
Full	100.0	100.0	100.0	98.6	97.4	98.8	96.2	97.5	100.0	82.7	92.3	71.8	54.0	91.5
Quest	**100.0**	**100.0**	97.6	98.2	77.8	3.4	82.0	95.3	99.7	34.8	93.3	64.0	49.0	76.5↓ 15.0
InfLLM	**100.0**	94.0	76.0	94.0	89.6	42.8	81.9	85.9	**99.8**	20.0	88.4	70.6	52.6	76.6↓ 14.9
Quest+RESA	**100.0**	**100.0**	97.2	97.6	75.8	4.8	85.6	95.7	99.6	21.1	93.9	65.2	48.6	75.8↓ 15.7
CompKV	**100.0**	**100.0**	**100.0**	**98.6**	**96.4**	**75.0**	**95.7**	**97.2**	**99.8**	**35.2**	**94.9**	**71.8**	**53.4**	**86.0**↓ 5.5
Qwen3-32B														
Full	100.0	98.4	100.0	99.8	99.8	100.0	99.4	100.0	99.7	87.5	93.3	80.0	60.4	93.7
Quest	**100.0**	**99.2**	94.4	**100.0**	35.2	2.4	95.8	**99.8**	98.4	35.5	93.1	74.8	55.8	75.7↓ 18.0
InfLLM	**100.0**	95.8	82.4	93.4	91.6	68.0	85.3	90.1	**99.4**	34.9	93.3	81.0	57.0	82.5↓ 11.2
Quest+RESA	97.6	98.0	93.0	**100.0**	38.2	3.2	94.8	99.4	98.1	39.5	91.3	74.6	55.4	75.6↓ 18.1
CompKV	**100.0**	**99.2**	**100.0**	99.8	**99.4**	**93.0**	**98.5**	99.6	99.2	**56.0**	**96.7**	**81.4**	**60.2**	**91.0**↓ 2.7

5 Experiments

5.1 Setup

Models. To comprehensively evaluate our proposed CompKV, we consider three models that are widely used in sparse attention studies: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-8B, and Qwen3-32B (Yang et al., 2025). Qwen3-8B and Qwen3-32B are used to assess generalization across model scales, while Llama-3.1-8B-Instruct is included to further examine whether CompKV generalizes across different model families and architectural designs.

Methods. To verify the effectiveness of CompKV, we compare it against Quest (Tang et al., 2024), InfLLM (Xiao et al., 2024a), and Quest+RESA with $\lambda = 0.25$ (Yang et al., 2026). Quest and InfLLM are state-of-the-art training-free, query-aware baselines for block selection. CompKV differs from these methods in two aspects: how blocks are selected for exact attention computation and how the unselected blocks are handled. RESA is a plug-in compensation method that can be integrated with different sparse-attention methods; following its original evaluation setup, we use Quest+RESA as the RESA baseline. As defined in Section 4.1, r denotes the number of variance groups, and we set $r = 4$ for all main CompKV results. Full attention is included as the dense reference.

Benchmarks and metrics. RULER (Hsieh et al., 2024) provides controlled retrieval, tracing, and aggregation tasks to test whether CompKV preserves long-context capabilities under tight exact-read budgets. LongBench-Pro (Chen et al., 2026) assesses whether these gains extend to more realistic tasks requiring either localized evidence retrieval or global context integration. We evaluate RULER at 32K and LongBench-Pro at context lengths up to 128K using their respective official scorers.

Additional configuration. All evaluations are conducted on NVIDIA H100 80GB HBM3 GPUs with a batch size of one. For a consistent comparison, all four sparse methods partition consecutive tokens into blocks of size 16 and retain the first block as an attention sink together with the two most recent blocks as a local attention window. These mandatory blocks are always included in the

selected set and count toward the same per-step exact-read budget. Appendix C provides further details on the complete experimental setup and evaluation configurations.

Table 2: **LongBench-Pro scores.** Mod. and Extr. abbreviate Moderate and Extreme. Bold marks the best sparse result per model and column. AVG arrows show point drops from Full.

Method	Difficulty				Context length					AVG
	Easy	Mod.	Hard	Extr.	8k	16k	32k	64k	128k	
Llama-3.1-8B-Instruct										
Full	28.30	18.88	25.61	25.72	30.46	25.55	25.73	22.59	21.98	25.26
Quest	**28.30**	17.71	22.66	24.75	28.09	**25.54**	24.47	21.55	**21.75**	24.28↓ 0.98
InfLLM	27.56	17.69	21.84	23.72	28.31	24.53	24.07	20.72	20.35	23.60↓ 1.66
Quest+RESA	27.51	**18.09**	**24.38**	24.54	27.95	24.34	**25.94**	**21.73**	21.54	24.30↓ 0.96
CompKV	27.91	18.05	23.94	**24.83**	**28.63**	25.36	25.30	21.50	21.39	**24.44**↓ 0.82
Qwen3-8B										
Full	39.63	29.36	32.36	31.62	38.68	35.86	34.88	29.53	32.21	34.23
Quest	34.54	27.14	28.68	28.89	33.83	29.54	30.50	27.84	31.19	30.58↓ 3.65
InfLLM	36.18	29.09	29.82	28.83	35.01	30.87	33.43	27.13	32.31	31.75↓ 2.48
Quest+RESA	34.54	27.82	28.63	28.46	33.90	29.96	31.37	27.25	30.50	30.60↓ 3.63
CompKV	**37.17**	**30.09**	**30.68**	**29.89**	**35.74**	**32.82**	**33.77**	**28.17**	**33.19**	**32.74**↓ 1.49
Qwen3-32B										
Full	50.88	39.11	39.17	33.63	48.20	44.45	41.56	38.72	37.09	42.00
Quest	44.93	33.93	35.76	31.43	41.67	35.15	39.00	36.22	36.06	37.62↓ 4.38
InfLLM	48.07	**36.44**	37.18	31.80	44.63	38.75	**42.02**	36.54	36.15	39.62↓ 2.38
Quest+RESA	43.61	31.93	35.11	29.69	41.87	35.24	38.35	35.07	30.28	36.16↓ 5.84
CompKV	**49.56**	35.23	**38.62**	**32.36**	**45.49**	**39.52**	41.09	**38.28**	**37.15**	**40.30**↓ 1.70

5.2 Accuracy Results

We evaluate how well CompKV preserves task quality by comparing overall scores, task-level behavior, and sensitivity to the KV budget. With $r = 4$, CompKV achieves the highest AVG for all three models on RULER at a 32K length and a 512-token budget (Table 1), as well as on LongBench-Pro (Table 2). It also obtains the best or tied-best sparse scores in most task-level comparisons.

The gains are particularly pronounced on MK3, where the accuracy of CompKV is significantly higher than that of other baselines. This improvement suggests that compensation-aware selection helps preserve the information needed for difficult retrieval tasks. However, on Llama-3.1, Full attention scores only 9.9% on CWE, and all evaluated sparse methods remain low, which means the aggregate improvements coexist with tasks that remain challenging even for the dense model.

The advantage persists as the exact-read budget varies. We further evaluate CompKV with token budgets ranging from 128 to 1024 for both 8B models on RULER. Results are shown in Appendix D. The curves show that several baselines approach Full as the budget increases, but CompKV remains the best across most budgets, tasks and models.

5.3 Efficiency Evaluation

To further evaluate the efficiency of our CompKV implementation, we follow the profiling setup of Quest (Tang et al., 2024) and measure the complete CPU-offload attention step on a single NVIDIA H100 GPU with a batch size of one. We benchmark Full, Quest, InfLLM, and CompKV with $r \in \{4, 128\}$ under the same evaluation setting. We do not include Quest+RESA in the latency evaluation because its original paper reports higher latency than Quest. The evaluation covers endpoint lengths of 32K, 64K, and 128K and selected-token budgets of 512, 1,024, and 2,048. CompKV

($r = 4$) achieves the lowest mean latency in all nine settings, with up to $6.85\times$ speedups over Full. Figure 4 shows the 512-token budget; Appendix E.2 gives the full protocol and results.

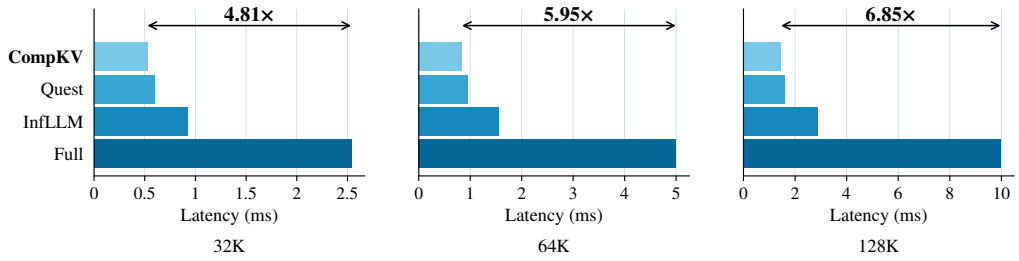


Figure 4: **CPU-offload attention latency.** Mean host wall time per single-layer attention step at a 512-token budget. Lengths denote the endpoints of the 256-step decode windows.

5.4 Ablation Studies

Ablation objectives. We examine Mean compensation, the outer variance factor, and the second-order mass correction to determine how tail reconstruction and residual-aware scoring contribute to the final task quality. We also vary the number of variance groups r to assess how compact block statistics affect accuracy and metadata cost.

Factor ablations. Table 3 compares four configurations with $r = 4$. Here, $\widehat{p}^{(2)}$ includes the variance correction in Equation 9, while $\widehat{p}^{(1)}$ uses mean-key logits alone. Adding Mean compensation improves AVG by 0.48–2.47 points, while the outer variance factor provides a further 0.35–1.19 points. The second-order mass correction contributes an additional 1.96–3.39 points over its first-order counterpart. Overall, the results indicate that compensation, residual-aware selection, and second-order mass estimation all contribute consistently, with CompKV achieving the highest AVG in all four settings.

Variance-group granularity. Figure 5 shows that accuracy generally improves with finer variance groups, with $r = 128$ achieving the highest AVG at both budgets. Our default $r = 4$ trails this endpoint by 1.87 and 1.08 points at budgets 256 and 512, respectively, while requiring only $1/32$ of its variance metadata. This trade-off motivates the use of compact grouped statistics in the main experiments.

Table 3: **Factor ablations on RULER.** Values are 13-task AVG (%) for Llama-3.1-8B-Instruct and Qwen3-8B with $r = 4$. Column labels denote selected-token budgets.

Variant	Llama		Qwen	
	256	512	256	512
CompKV	**78.6**	**83.2**	**80.9**	**86.0**
$\widehat{p}^{(2)}$ only	75.6	81.8	79.2	83.5
$\widehat{p}^{(2)}$ + Mean	78.1	82.8	79.7	85.1
$\widehat{p}^{(1)}\widehat{\sigma}^2$ + Mean	76.1	81.2	77.5	83.8

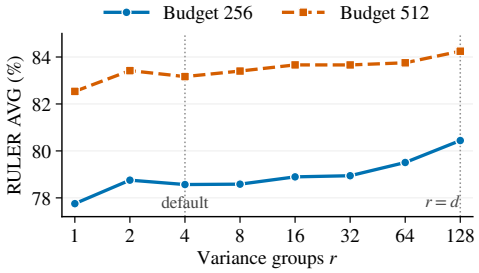


Figure 5: **Variance-group granularity on RULER.** Curves show the 13-task AVG for Llama-3.1-8B-Instruct at selected-token budgets of 256 and 512.

6 Conclusion

In this paper, we identify the mismatch between KV block selection and downstream compensation and introduce CompKV, a compensation-aware method that selects blocks according to their estimated post-compensation residual. Our analysis shows that, under Mean compensation, this residual is jointly governed by block attention mass and within-block logit variation, which CompKV estimates from compact block statistics. Across three models and two long-context benchmarks, CompKV achieves the highest average scores among the evaluated sparse methods, while its asynchronous implementation delivers up to $6.85\times$ self-attention speedups over full attention.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs, implement methods, and assist with translation. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, clean or reformat datasets, support qualitative or thematic data analysis, or interpret results. Generating synthetic datasets was not applicable to this work.

Additionally, we used generative AI tools to create or modify scientific figures, create, edit, and debug software code, summarize and identify relevant literature, source and search for information, and edit portions of the manuscript to improve readability and language.

We have reviewed all AI-assisted work. In particular, AI-assisted implementations and code modifications were manually reviewed and tested for correctness by the authors; AI-assisted proofs were checked against the authors' intended mathematical arguments; references and information identified with AI assistance were independently verified against their original sources; and AI-assisted text and figures were manually reviewed for correctness and consistency. We take responsibility for the final content of this work, including text, claims, code, figures, and other artifacts produced or modified with the aid of generative AI.

ETHICS STATEMENT

CompKV is designed to reduce the memory traffic and computation required for long-context LLM inference. By lowering the per-query resource requirements of sparse decoding, it may reduce inference costs and energy use, making long-context systems more accessible with limited resources.

Inference-efficiency techniques are dual use: reducing the cost of serving existing models may also lower the cost of deploying them at scale for any purpose. CompKV changes KV cache selection and attention computation. It does not modify the training data, generation objectives, or safety mechanisms of the underlying model, and it is not designed to introduce new content-generation capabilities. It also does not mitigate biases, privacy and security risks, or harmful-content risks inherited from those models. Deployments should therefore evaluate these risks after acceleration and retain safeguards appropriate to the underlying model. Our experiments use the publicly available RULER and LongBench-Pro benchmarks and publicly released model checkpoints. We did not recruit human participants or collect or release new datasets.

REPRODUCIBILITY STATEMENT

All model checkpoints and benchmark datasets used in this work are publicly available. Section 5.1 summarizes the evaluated models, baselines, benchmarks, token budgets, and hardware. Appendix C provides the full model configurations and evaluation protocols, including benchmark composition, prompt formatting, context handling, decoding parameters, random seeds, token budgets, official scorers, and score aggregation. Section 5.3 specifies the latency workload, while Appendix E.1 gives the CPU-offload implementation and Appendix E.2 reports the latency protocol and complete results. The anonymized source code will be provided with the supplementary materials.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=hmOwOZWzYE.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August

2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ayi7qezU87.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. Palu: KV-cache compression with low-rank projection. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=LWMS4pk2vK.

Ziyang Chen, Xing Wu, Junlong Jia, Chaochen Gao, Qi Fu, Debing Zhang, and Songlin Hu. Longbench pro: A more realistic and comprehensive bilingual long-context evaluation benchmark, 2026. URL https://arxiv.org/abs/2601.02872.

Keqi Deng, Shaoshi Ling, Ruchao Fan, and Jinyu Li. Unique: Universal top-k sparse attention for training-free inference and sparsity-aware training, 2026. URL https://arxiv.org/abs/2605.27740.

Qihang Fan, Huaibo Huang, Zhiying Wu, Bingning Wang, and Ran He. Flashprefill v2: Block-sparse prefill attention for long-context llm serving, 2026. URL https://arxiv.org/abs/2608.19758.

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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=tcisuhGsQZ.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 1270–1303. Curran Associates, Inc., 2024. doi: 10.52202/079017-0040. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/028fcbcf85435d39a40c4d61b42c99a4-Paper-Conference.pdf.

Coleman Richard Charles Hooper, Sebastian Zhao, Luca Manolache, Sehoon Kim, Michael W. Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. Multipole attention for efficient long context reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=5Qe7AGO3Eq.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Shibo Jie, Yehui Tang, Kai Han, Zhi-Hong Deng, and Jing Han. Specache: Speculative key-value caching for efficient generation of LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=PQIrsaIQdn.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

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, 2024. URL https://arxiv.org/abs/2403.05527.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. Infinigen: efficient generative inference of large language models with dynamic kv cache management. In *Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation*, OSDI'24, USA, 2024. USENIX Association. ISBN 978-1-939133-40-3.

Xing Li, Zeyu XING, Yiming Li, Linping Qu, Hui-Ling Zhen, Yiwu Yao, Wulong Liu, Sinno Jialin Pan, and Mingxuan Yuan. KVTuner: Sensitivity-aware layer-wise mixed-precision KV cache quantization for efficient and nearly lossless LLM inference. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=zDwipF6h06.

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 *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=poE54GOq2l.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. Retrievalattention: Accelerating long-context LLM inference via vector retrieval. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=8z3cOVER4z.

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 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=JZfg6wGi6g.

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 Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24bz.html.

Guanqiao Qu, Qiyuan Chen, Wei Wei, Zheng Lin, Xianhao Chen, and Kaibin Huang. Mobile edge intelligence for large language models: A contemporary survey. *IEEE Communications Surveys & Tutorials*, 27(6):3820–3860, 2025.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. SparQ attention: Bandwidth-efficient LLM inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42558–42583. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/ribar24a.html.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for KV cache compression. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15332–15344, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.899. URL https://aclanthology.org/2024.findings-emnlp.899/.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=raABeiV71j.

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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=oa7MYAO6h6.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: query-aware sparsity for efficient long-context llm inference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Georgios Tzachristas, Lei Deng, Ioannis Tzachristas, Gong Zhang, and Renhai Chen. A mathematical theory of top-k sparse attention via total variation distance. In *2026 IEEE International Symposium on Information Theory (ISIT)*, pp. 1–6. IEEE, 2026.

Yifei Wang, Yueqi Wang, Zhenrui Yue, Huimin Zeng, Yong Wang, Ismini Lourentzou, Zhengzhong Tu, Xiangxiang Chu, and Julian McAuley. FASA: FREQUENCY-AWARE SPARSE ATTENTION. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=FnSgecCEwg.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=bTHFrqhASY.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Weihao Yang, Hao Huang, Ningke Li, Shihao Wang, Darong Yang, Yanqi Pan, Wen Xia, Shiyi Li, and Xiangyu Zou. RESA: Bringing back what sparse attention ignores with residual estimation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ktcq26hMCH.

Yuhang Zhan, Lisi Chen, and Shuo Shang. Reskv: Reconstructing omitted attention contributions for fixed-budget kv cache compression, 2026. URL https://arxiv.org/abs/2607.29591.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=RkRrPp7GKO.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf.

A Detailed CompKV Derivations

A.1 KL Identity for Mean Compensation

Mean compensation replaces the exact weight of each unselected block with its mean-based weight. Consequently, the selection loss admits both a log-partition form and a form that isolates the unre-

constructed weight of each omitted block (Equation 4):

$$\mathcal{L}(\mathcal{S}) = \sum_{g=1}^{G} \log \frac{\sum_{b\in\mathcal{B}} Z_{g,b}}{\sum_{b\in\mathcal{S}} Z_{g,b} + \sum_{b\notin\mathcal{S}} |b| e^{\bar{z}_{g,b}}} = -\sum_{g=1}^{G} \log \left[1 - \sum_{b\notin\mathcal{S}} p_{g,b} \left(1 - \frac{|b| e^{\bar{z}_{g,b}}}{Z_{g,b}}\right)\right].$$

To establish this identity, fix a query head g and a selected block set $\mathcal{S}$. Write the full and compensated normalization constants as

$$D_g = \sum_{b\in\mathcal{B}} Z_{g,b}, \qquad \widehat{D}_g(\mathcal{S}) = \sum_{b\in\mathcal{S}} Z_{g,b} + \sum_{b\notin\mathcal{S}} |b| e^{\bar{z}_{g,b}}. \tag{13}$$

The token probabilities are $P_g(j) = e^{z_{g,j}}/D_g$ and $\widehat{P}_g(j;\mathcal{S}) = e^{\hat{z}_{g,j}}/\widehat{D}_g(\mathcal{S})$. Substituting them into the KL divergence gives

$$\begin{aligned} D_{\mathrm{KL}}\left(\widehat{P}_g(\mathcal{S}) \| P_g\right) &= \sum_{b\in\mathcal{B}} \sum_{j\in b} \widehat{P}_g(j;\mathcal{S}) \left[\hat{z}_{g,j} - z_{g,j} + \log \frac{D_g}{\widehat{D}_g(\mathcal{S})}\right] \\ &= \log \frac{D_g}{\widehat{D}_g(\mathcal{S})} + \sum_{b\notin\mathcal{S}} \sum_{j\in b} \widehat{P}_g(j;\mathcal{S}) \left(\bar{z}_{g,b} - z_{g,j}\right). \end{aligned} \tag{14}$$

The second equality follows from $\hat{z}_{g,j} = z_{g,j}$ in selected blocks and normalized probabilities.

Within each unselected block, every token has the same compensated probability. The remaining term therefore vanishes:

$$\sum_{b\notin\mathcal{S}} \sum_{j\in b} \widehat{P}_g(j;\mathcal{S}) \left(\bar{z}_{g,b} - z_{g,j}\right) = \frac{1}{\widehat{D}_g(\mathcal{S})} \sum_{b\notin\mathcal{S}} e^{\bar{z}_{g,b}} \sum_{j\in b} \left(\bar{z}_{g,b} - z_{g,j}\right) = 0, \tag{15}$$

because $\bar{z}_{g,b}$ is the arithmetic mean of the original logits in block b. Summing the resulting log-partition ratio over the G query heads yields

$$\mathcal{L}(\mathcal{S}) = \sum_{g=1}^{G} \log \frac{D_g}{\widehat{D}_g(\mathcal{S})} = \sum_{g=1}^{G} \log \frac{\sum_{b\in\mathcal{B}} Z_{g,b}}{\sum_{b\in\mathcal{S}} Z_{g,b} + \sum_{b\notin\mathcal{S}} |b| e^{\bar{z}_{g,b}}}. \tag{16}$$

To obtain the residual form, write the compensated normalization constant as the full weight minus the unreconstructed weight:

$$\widehat{D}_g(\mathcal{S}) = D_g - \sum_{b\notin\mathcal{S}} \left(Z_{g,b} - |b| e^{\bar{z}_{g,b}}\right). \tag{17}$$

Dividing by D_g and using $p_{g,b} = Z_{g,b}/D_g$ gives

$$\frac{\widehat{D}_g(\mathcal{S})}{D_g} = 1 - \sum_{b\notin\mathcal{S}} p_{g,b} \left(1 - \frac{|b| e^{\bar{z}_{g,b}}}{Z_{g,b}}\right). \tag{18}$$

Substitution into the log-partition ratio establishes

$$\mathcal{L}(\mathcal{S}) = -\sum_{g=1}^{G} \log \left[1 - \sum_{b\notin\mathcal{S}} p_{g,b} \left(1 - \frac{|b| e^{\bar{z}_{g,b}}}{Z_{g,b}}\right)\right], \tag{19}$$

which is the second equality in Equation 4.

A.2 Expansion of the Blockwise Compensation Residual

The residual form above shows that an omitted block contributes to the loss only through the weight that Mean compensation fails to recover. In the small-variance regime, this unreconstructed fraction has the expansion in Equation 5:

$$p_{g,b}\left(1-\frac{|b|e^{\bar{z}_{g,b}}}{Z_{g,b}}\right)=p_{g,b}\left(\frac{1}{2}\sigma_{g,b}^2+O(\sigma_{g,b}^3)\right).$$

To establish this expansion, we hold the maximum block size fixed and let $\max_{g,b}\sigma_{g,b}\to 0$, where $\sigma_{g,b}$ is the standard deviation of the logits within block b for query head g. We start from the block's exact weight relative to its mean-based weight.

Averaging the second-order Taylor expansion of the exponential over a block gives

$$\frac{Z_{g,b}}{|b|e^{\bar{z}_{g,b}}}=\frac{1}{|b|}\sum_{j\in b}e^{z_{g,j}-\bar{z}_{g,b}}=1+\frac{1}{2}\sigma_{g,b}^2+O(\sigma_{g,b}^3). \tag{20}$$

The linear term vanishes because $\sum_{j\in b}(z_{g,j}-\bar{z}_{g,b})=0$, and the quadratic term averages to $\sigma_{g,b}^2/2$. The fixed maximum block size makes the averaged third-order remainder $O(\sigma_{g,b}^3)$.

The deviation of this ratio from one is $O(\sigma_{g,b}^2)$, so taking its reciprocal gives

$$\frac{|b|e^{\bar{z}_{g,b}}}{Z_{g,b}}=\frac{1}{1+\frac{1}{2}\sigma_{g,b}^2+O(\sigma_{g,b}^3)}=1-\frac{1}{2}\sigma_{g,b}^2+O(\sigma_{g,b}^3). \tag{21}$$

The reciprocal expansion introduces a quadratic remainder of order $O(\sigma_{g,b}^4)$, which combines with the existing cubic remainder. Subtracting the reciprocal from one and multiplying by $p_{g,b}$ yields

$$p_{g,b}\left(1-\frac{|b|e^{\bar{z}_{g,b}}}{Z_{g,b}}\right)=p_{g,b}(\frac{1}{2}\sigma_{g,b}^2+O(\sigma_{g,b}^3)). \tag{22}$$

Taking the logarithm of Equation 20 also gives the log-partition expansion used for attention-mass estimation:

$$\log Z_{g,b}=\log|b|+\bar{z}_{g,b}+\frac{1}{2}\sigma_{g,b}^2+O(\sigma_{g,b}^3). \tag{23}$$

A.3 Second-Order Expansion of the Selection Loss

The preceding subsection expands the residual of a single omitted block. We now combine these residuals within each query head and account for the outer logarithm in the exact selection loss. This gives the second-order approximation in Equation 6:

$$\mathcal{L}(\mathcal{S})=\frac{1}{2}\sum_{g=1}^{G}\sum_{b\notin\mathcal{S}}p_{g,b}\left(\sigma_{g,b}^2+O\big(\sigma_{g,b}^3\big)\right).$$

To derive this expression, hold the maximum block size fixed and set $\varepsilon=\max_{g,b}\sigma_{g,b}\to 0$. Define the total normalized residual for head g as

$$R_g(\mathcal{S})=\sum_{b\notin\mathcal{S}}p_{g,b}\left(1-\frac{|b|e^{\bar{z}_{g,b}}}{Z_{g,b}}\right). \tag{24}$$

Summing the blockwise expansion from Appendix A.2 gives

$$R_g(\mathcal{S})=\frac{1}{2}\sum_{b\notin\mathcal{S}}p_{g,b}\sigma_{g,b}^2+O\left(\sum_{b\notin\mathcal{S}}p_{g,b}\sigma_{g,b}^3\right). \tag{25}$$

Since $\sum_{b\notin\mathcal{S}}p_{g,b}\sigma_{g,b}^3\le\varepsilon\sum_{b\notin\mathcal{S}}p_{g,b}\sigma_{g,b}^2$, we have $R_g=O(\sum_{b\notin\mathcal{S}}p_{g,b}\sigma_{g,b}^2)$. Moreover, $\sum_{b\notin\mathcal{S}}p_{g,b}\le 1$ implies $R_g\to 0$. Expanding the outer logarithm therefore yields

$$\begin{aligned}-\log(1-R_g)=&\frac{1}{2}\sum_{b\notin\mathcal{S}}p_{g,b}\sigma_{g,b}^2\\&+O\left(\sum_{b\notin\mathcal{S}}p_{g,b}\sigma_{g,b}^3+\left[\sum_{b\notin\mathcal{S}}p_{g,b}\sigma_{g,b}^2\right]^2\right).\end{aligned} \tag{26}$$

The quadratic remainder can be absorbed into the cubic term. By the Cauchy–Schwarz inequality,

$$\begin{aligned}\left[\sum_{b\notin\mathcal{S}} p_{g,b}\sigma_{g,b}^2\right]^2 &\le \left(\sum_{b\notin\mathcal{S}} p_{g,b}\right)\left(\sum_{b\notin\mathcal{S}} p_{g,b}\sigma_{g,b}^4\right)\\ &\le \varepsilon \sum_{b\notin\mathcal{S}} p_{g,b}\sigma_{g,b}^3.\end{aligned} \tag{27}$$

Substituting this bound and summing over the query heads gives

$$\begin{aligned}\mathcal{L}(\mathcal{S}) &= \frac{1}{2}\sum_{g=1}^{G}\sum_{b\notin\mathcal{S}} p_{g,b}\sigma_{g,b}^2 + O\left(\sum_{g=1}^{G}\sum_{b\notin\mathcal{S}} p_{g,b}\sigma_{g,b}^3\right)\\ &= \frac{1}{2}\sum_{g=1}^{G}\sum_{b\notin\mathcal{S}} p_{g,b}\left(\sigma_{g,b}^2 + O(\sigma_{g,b}^3)\right),\end{aligned} \tag{28}$$

which establishes Equation 6. The remainder bounds hold uniformly over the selected sets throughout the stated small-variance regime with a fixed maximum block size.

The leading term depends on $\mathcal{S}$ only through the unselected blocks. Because its sum over all blocks is independent of $\mathcal{S}$, minimizing the unselected contribution under a budget of K exact block reads is equivalent to

$$\max_{\mathcal{S}\subseteq\mathcal{B},\,|\mathcal{S}|=K}\sum_{b\in\mathcal{S}}\sum_{g=1}^{G} p_{g,b}\sigma_{g,b}^2.$$

Thus, the second-order objective selects the K blocks with the largest $\sum_{g=1}^{G} p_{g,b}\sigma_{g,b}^2$, as stated in Equation 7.

A.4 Grouped Variance and Attention-Mass Estimation

The oracle selection rule above requires each block's exact attention mass and logit variance, which would require reading its keys. CompKV instead estimates the variance from grouped key statistics. The first quantity to derive is Equation 8:

$$\widehat{\sigma}_{g,b}^2 = \frac{1}{d}\sum_{t=0}^{r-1}\left(\sum_{i\in\mathcal{G}_t} q_{g,i}^2\right)(\bar{\mathbf{D}}_b)_t, \qquad (\bar{\mathbf{D}}_b)_t = \frac{1}{|b|\,|\mathcal{G}_t|}\sum_{j\in b}\sum_{i\in\mathcal{G}_t}(k_{j,i}-\bar{k}_{b,i})^2.$$

To obtain this estimate, let $\mathbf{\Sigma}_{k,b}\in\mathbb{R}^{d\times d}$ denote the key covariance within block b. Its definition and projection along the query give

$$\mathbf{\Sigma}_{k,b} = \frac{1}{|b|}\sum_{j\in b}(\boldsymbol{k}_j-\bar{\boldsymbol{k}}_b)^\top(\boldsymbol{k}_j-\bar{\boldsymbol{k}}_b), \qquad \sigma_{g,b}^2 = \frac{1}{d}\boldsymbol{q}_g\mathbf{\Sigma}_{k,b}\boldsymbol{q}_g^\top. \tag{29}$$

The projection follows from $z_{g,j}-\bar{z}_{g,b} = \boldsymbol{q}_g(\boldsymbol{k}_j-\bar{\boldsymbol{k}}_b)^\top/\sqrt{d}$.

For the settings where r divides $d/2$, we divide the ordered RoPE frequency pairs into r equal, consecutive groups, keeping both coordinates of each pair together. Each group $\mathcal{G}_t$ contains d/r coordinates. At $r=d$, each group contains one coordinate. The statistic stored for each group is the average of its diagonal covariance entries:

$$(\bar{\mathbf{D}}_b)_t = \frac{1}{|\mathcal{G}_t|}\sum_{i\in\mathcal{G}_t}(\mathbf{\Sigma}_{k,b})_{ii} = \frac{1}{|b|\,|\mathcal{G}_t|}\sum_{j\in b}\sum_{i\in\mathcal{G}_t}(k_{j,i}-\bar{k}_{b,i})^2. \tag{30}$$

We approximate $\mathbf{\Sigma}_{k,b}$ by a diagonal matrix, assigning $(\bar{\mathbf{D}}_b)_t$ to every diagonal entry whose coordinate belongs to $\mathcal{G}_t$. Projecting this approximation along the query yields

$$\widehat{\sigma}_{g,b}^2 = \frac{1}{d}\sum_{t=0}^{r-1}\sum_{i\in\mathcal{G}_t} q_{g,i}^2(\bar{\mathbf{D}}_b)_t = \frac{1}{d}\sum_{t=0}^{r-1}\left(\sum_{i\in\mathcal{G}_t} q_{g,i}^2\right)(\bar{\mathbf{D}}_b)_t. \tag{31}$$

Together, these expressions recover both parts of Equation 8. Each group's average key variance is weighted by the sum of squared query coordinates in that group, with attention scaling $1/d$.

The same variance estimate also determines the attention-mass estimate in Equation 9. Exponentiating the log-partition expansion in Equation 23 gives

$$Z_{g,b} = |b| \exp\left(\bar{z}_{g,b} + \frac{1}{2}\sigma^2_{g,b} + O(\sigma^3_{g,b})\right). \tag{32}$$

Retaining the second-order terms and replacing $\sigma^2_{g,b}$ with $\widehat{\sigma}^2_{g,b}$ gives the estimated block weight $|b| \exp(\bar{z}_{g,b} + \widehat{\sigma}^2_{g,b}/2)$. Since the full attention mass is $p_{g,b} = Z_{g,b} / \sum_{c \in \mathcal{B}} Z_{g,c}$, normalizing these estimated weights over all available blocks yields

$$\widehat{p}_{g,b} = \frac{|b| \exp\left(\bar{z}_{g,b} + \frac{1}{2}\widehat{\sigma}^2_{g,b}\right)}{\sum_{c \in \mathcal{B}} |c| \exp\left(\bar{z}_{g,c} + \frac{1}{2}\widehat{\sigma}^2_{g,c}\right)}. \tag{33}$$

This recovers Equation 9, with normalization performed separately for each query head.

A.5 Comparison with Uncompensated Sparse Attention

For comparison, Drop removes unselected blocks and renormalizes over the tokens in a nonempty selected set $\mathcal{S}$, giving $P_g^{\mathrm{Drop}}(\mathcal{S})$. Unselected positions in the original token space receive zero probability. The ratio between the approximate and full probabilities is constant on selected positions:

$$D_{\mathrm{KL}}\big(P_g^{\mathrm{Drop}}(\mathcal{S}) \| P_g\big) = \log \frac{\sum_{c \in \mathcal{B}} Z_{g,c}}{\sum_{b \in \mathcal{S}} Z_{g,b}} = -\log\left(\sum_{b \in \mathcal{S}} p_{g,b}\right). \tag{34}$$

This identity agrees with the KL analysis of Top-k sparse attention (Tzachristas et al., 2026). For one head and a fixed block budget, retaining the blocks with the largest attention mass exactly minimizes this KL divergence. With a shared selected set across heads, the objective sums the headwise logarithmic terms. When the omitted attention mass is small for each head, its first-order additive score is $\sum_g p_{g,b}$.

For the same nonempty selected set, adding positive mean-compensation weights increases the normalization constant. The inequality $|b| e^{\bar{z}_{g,b}} \le Z_{g,b}$ also ensures that this constant does not exceed that of full attention. Consequently,

$$0 \le D_{\mathrm{KL}}\Big(\widehat{P}_g(\mathcal{S}) \| P_g\Big) \le D_{\mathrm{KL}}\big(P_g^{\mathrm{Drop}}(\mathcal{S}) \| P_g\big). \tag{35}$$

The second inequality is strict whenever an unselected block exists. The first is also strict if at least one unselected block has nonconstant logits. This comparison holds for a fixed selected set; the two operators can induce different optimal block sets under the same exact-read budget.

B Attention Fidelity of Residual-Aware Selection

B.1 Exact Residual versus Attention Mass

We compare exact mass-based and residual-based selection on Llama-3.1-8B-Instruct across all 13 RULER tasks, randomly sampling 64 prompts per task with seed 42 (832 total). We collect full-attention states under greedy decoding with batch size one at some layers. Both selectors use identical Q/K/V states, Mean compensation, and a 512-token budget with 16-token blocks; the first block and two most recent blocks are mandatory and included in the budget.

The selectors rank blocks by $\sum_g p_{g,b}$ or $\sum_g \rho_{g,b}$, sharing each selected set across heads, where

$$\rho_{g,b} = p_{g,b}\left(1 - \frac{|b| e^{\bar{z}_{g,b}}}{Z_{g,b}}\right) \tag{36}$$

is the exact compensation residual in Equation 4. For each head h, we measure

$$E_{\mathrm{KL},h} = D_{\mathrm{KL}}(\widehat{P}_h \| P_h), \qquad E_{\mathrm{out},h} = \frac{\|\widehat{P}_h V_h - P_h V_h\|_2}{\|P_h V_h\|_2}. \tag{37}$$

KL compares post-softmax probabilities; relative L_2 compares outputs after multiplication by V_h. We plot $\Delta E_h = E_h^{(\mathrm{mass})} - E_h^{(\mathrm{res})}$, so **positive values favor residual-based selection**. Each point averages positions within prompts, prompts within tasks, and then all 13 tasks equally.

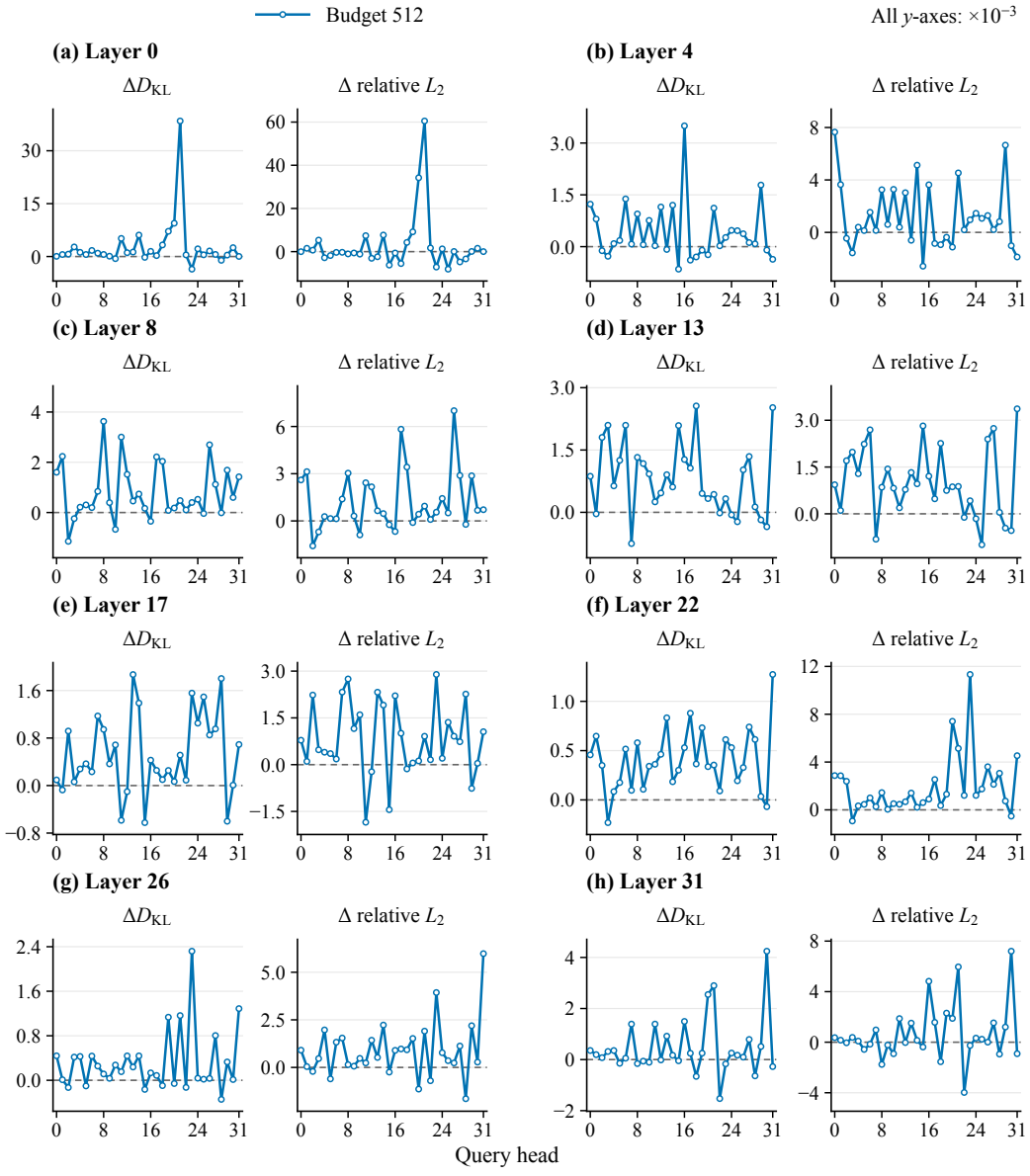


Figure 6: **Per-head error reduction from residual-based selection.** Mass-minus-residual differences in attention KL (left) and relative output L_2 (right), using the same Mean compensator.

Averaged over the eight layers and 32 query heads, residual-based selection reduces KL by 0.660% and relative L_2 by 0.582%. Both metrics improve in every layer-level mean, with improvements in most layer–head pairs. Thus, prioritizing the contribution left unreconstructed by Mean compensation lowers average attention and output error relative to mass-based selection.

Although the local gains are modest, attention-output errors propagate through subsequent layers and decoding steps and may be amplified, while reducing them may help limit accumulated deviations.

B.2 Fidelity of Practical CompKV Scores

On the same states and budget, we compare CompKV ($r \in \{4, 128\}$) with estimated-mass selection, $\sum_g \widehat{p}_{g,b}$, using the $r = 4$ mass estimate in Equation 9. This baseline retains the second-order mass correction and removes the external variance factor in Equation 10.

For each KV group, let $T_b = \sum_g \rho_{g,b}$, with $\rho_{g,b}$ defined in Equation 36. Spearman correlation compares scores with T_b over non-mandatory blocks. Let $\mathcal{S}_m$ and $\mathcal{S}_\rho$ contain the 29 non-mandatory blocks selected by method m and exact residual ranking. Residual capture is

$$\text{Capture}(m) = \frac{\sum_{b \in \mathcal{S}_m} T_b}{\sum_{b \in \mathcal{S}_\rho} T_b}. \tag{38}$$

We use the averaging in Appendix B.1. Layer curves additionally average KV heads, and overall results also average layers.

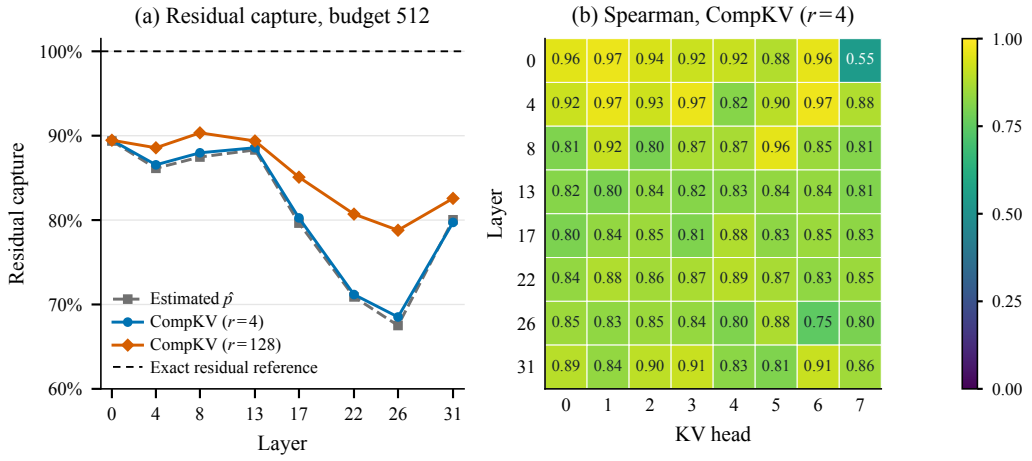


Figure 7: **Fidelity of practical CompKV scores.** (a) residual capture at budget 512, normalized by exact residual selection. (b) CompKV ($r = 4$) Spearman correlations by layer and KV head. Mandatory blocks are excluded from both metrics.

Figure 7 reports both metrics. Overall Spearman correlations are 0.8511, 0.8604, and 0.8782 for estimated mass, CompKV ($r = 4$), and CompKV ($r = 128$), respectively, with capture rates of 81.17%, 81.53%, and 85.61%. CompKV ($r = 4$) improves residual capture over estimated-mass selection in seven of eight layer means. Spearman correlations were saved as online scalars.

These results show that highly compressed block summaries preserve much of the residual-selection signal. With only four grouped-variance statistics per block, CompKV achieves strong ranking agreement and captures over 81% of the residual removable by the exact reference. Its modest improvements over estimated-mass selection use the same summary statistics, showing the benefit of compensation-aware scoring within a fixed metadata budget. Although a gap to exact selection remains, this provides a practical balance between summary compactness and selection fidelity.

C Evaluation Configuration

C.1 Model Configurations

- **Llama-3.1-8B-Instruct** has 32 Transformer layers, 32 query heads, 8 key–value (KV) heads, a head dimension of 128, and a native context length of 131,072 tokens.

- **Qwen3-8B** has 36 Transformer layers, 32 query heads, 8 KV heads, a head dimension of 128, and a native context length of 32,768 tokens.
- **Qwen3-32B** has 64 Transformer layers, 64 query heads, 8 KV heads, a head dimension of 128, and a native context length of 32,768 tokens.

C.2 Evaluation Benchmarks

RULER. RULER (Hsieh et al., 2024) is a synthetic long-context benchmark organized around retrieval, multi-hop tracing, aggregation, and question answering. We evaluate all 13 tasks in its 32K configuration, using 500 samples per task and 6,500 samples in total. Retrieval is measured by eight Needle-in-a-Haystack tasks: `niah_single_1/2/3`, `niah_multikey_1/2/3`, `niah_multivalue`, and `niah_multiquery`. `vt` evaluates variable tracking, `cwe` and `fwe` evaluate common- and frequent-word aggregation, and `qa_1` and `qa_2` evaluate question answering over the SQuAD and HotpotQA documents provided for these two tasks, respectively.

LongBench-Pro. LongBench-Pro (Chen et al., 2026) is constructed from natural long documents and contains 1,500 bilingual samples across 11 primary tasks and 25 secondary tasks. Its primary tasks are Retrieval & Ranking, Sequencing & Structure Reconstruction, Evidence-Grounded QA, Summarization & Synthesis, Attribution & Citation Alignment, Aggregation & Clustering, Consistency & Compliance Checking, Structured & Numeric Reasoning, Version & Code Diff Analysis, Rule Induction & In-Context Learning, and Dialogue Memory & Long-Horizon Tracking. We use the 625 English samples in the 8k, 16k, 32k, 64k, and 128k buckets, with 125 samples in each bucket. The selected samples cover both Full (global integration) and Partial (localized retrieval) context requirements and retain the Easy, Moderate, Hard, and Extreme difficulty labels.

C.3 Evaluation Details

InfLLM configuration. We adapt InfLLM to 16-token blocks and grouped-query attention. Each query head accumulates causal attention probabilities and represents each block by the FP32 mean of its four highest-weight post-RoPE keys. Representatives are frozen when blocks leave the recent window, before selection. Query–representative scores are summed within each KV group to obtain a shared selected set. We retain the model's positional encoding, including YaRN for Qwen3 on LongBench-Pro, to compare KV selection and compensation under the positional semantics of the same Full attention reference. InfLLM's distant-position remapping changes both retrieval scores and attention logits. Retaining the configured positions keeps this factor consistent across methods and between dense prefill and decoding, allowing us to evaluate its representative-based selection within our common decode setting. Original Q/K/V states are BF16; representatives, accumulated weights, scores, and attention accumulation use FP32.

Prompt formatting. Llama-3.1-8B-Instruct uses the official raw RULER prompt, in which the input is followed directly by the answer prefix. For Qwen3-8B and Qwen3-32B, the RULER input is placed in a user message and the answer prefix in the final assistant message, with generation continuing from that prefix by setting `continue_final_message=true`. For LongBench-Pro, we concatenate each context with the official `question_nonthinking` field using four newline characters and then render the model-native chat template. Llama uses its instruct template, while both Qwen3 models use their assistant-prefill template with the generation prompt enabled.

No-thinking policy. We evaluate both Qwen3 checkpoints in their supported non-thinking mode on RULER and LongBench-Pro, setting `enable_thinking=False` in the chat template and `add_special_tokens=False` during tokenization. This establishes a direct-answer setting for comparing KV-selection methods under fixed output budgets. In preliminary runs, some methods still generated thinking markers despite this configuration. We therefore additionally suppress the `<think>` and `</think>` token IDs, 151667 and 151668, at every decoding step before token selection. This policy applies identically to Full attention, all sparse baselines, and CompKV.

Context handling. RULER prompts are left-truncated only after reserving the task-specific completion length. On LongBench-Pro, both Qwen3 models use YaRN with a factor of 4 and an original maximum position of 32,768, giving an effective context length of 131,072 tokens, while Llama uses

its native context window. If a rendered prompt plus 1,024 completion tokens exceeds the model context, we retain equal-length prefix and suffix segments through middle truncation.

Sampling and stopping. RULER uses deterministic greedy decoding without sampling filters. Its official generation limits are 128 tokens for the retrieval tasks, 30 for `vt`, 120 for `cwe`, 50 for `fwe`, and 32 for each question-answering task. LongBench-Pro uses stochastic decoding: both Qwen3 models set $(\texttt{temperature}, \texttt{top_p}, \texttt{top_k}, \texttt{min_p}) = (0.7, 0.8, 20, 0)$, while Llama uses $(0.6, 0.9, 0, \texttt{None})$. All three models use `max_new_tokens=1024`, one beam, repetition penalty 1.0, and EOS-only stopping. RULER uses seed 42 for its single deterministic decode. Each LongBench-Pro case is decoded for three rounds with seeds 42, 43, and 44.

LongBench-Pro budgets. The two token budgets are $(256, 512)$ for 8k and 16k, $(512, 1024)$ for 32k, $(1024, 2048)$ for 64k, and $(2048, 4096)$ for 128k. For all sparse methods, these values cap the exact KV tokens read per decoding step, including the sink and recent blocks.

Scoring metrics. The official RULER scorer applies all-reference substring matching to the retrieval, tracing, and aggregation tasks and any-reference substring matching to the two question-answering tasks. The LongBench-Pro scorer selects the metric associated with each secondary task, including NDCG, Pairwise Accuracy, Accuracy, Summary, F1, and SubEM. Summary uses Qwen3-Embedding-8B as specified by the benchmark.

Score aggregation. For RULER, we first average over the 500 samples within each task and then report the unweighted mean of the 13 task scores. For LongBench-Pro, we average the scores from the three rounds for each sample and budget tier, average the two tiers with equal weight, and then take the arithmetic mean over all 625 samples to obtain the AVG score reported in Table 2. Difficulty and length-bucket scores apply the same aggregation to their corresponding sample subsets.

D Accuracy across KV Budgets

Figure 8 compares per-step exact-read budgets of 128–1,024 tokens. On MK3, CompKV reaches 73.0% for Llama and 75.0% for Qwen with 512 tokens, exceeding the strongest baselines at 1,024 tokens (70.4% and 62.4%, respectively). At 128 tokens, Qwen's MV and MQ scores exceed the strongest baseline by 37.2 and 31.0 percentage points, respectively. Several baselines approach Full on FWE, MV, and MQ as budgets increase, narrowing the gaps between methods.

E CPU-Offload Implementation and Detailed Latency Results

E.1 Implementation Details

The CPU-offload adapter stores BF16 KV states and FP32 summaries in pinned CPU memory, while retaining a 16-token active block and reusable workspaces on the GPU. Each append updates the active block's mean keys, mean values, and grouped population variances on the GPU, then copies the current KV states and updated summary to the CPU. Completed blocks' summaries remain unchanged. GPU scoring reuses projected variances for mass estimation and residual scoring, aggregating scores across query heads sharing a KV head. Stable Top-K selection includes one sink block and two recent blocks within the budget and returns block IDs in position order.

The CPU gathers selected KV blocks into a contiguous pinned buffer for one H2D transfer and exact attention with FlashInfer, passing the valid token count explicitly for a partial final block. An auxiliary stream transfers mean values and computes Mean compensation once the selected-block mask is ready, independently of the selected-KV retrieval and exact-attention path. CUDA events synchronize the two branches before merging their contributions under the shared normalization in Equation 12. Intermediate computations use FP32, and the final output is BF16.

E.2 Detailed Latency Results

Each case runs the pretrained Llama-3.1-8B-Instruct on one NVIDIA H100 80GB HBM3 GPU with four NUMA-local CPU threads and batch size one. Q/K/V and outputs use BF16; summaries

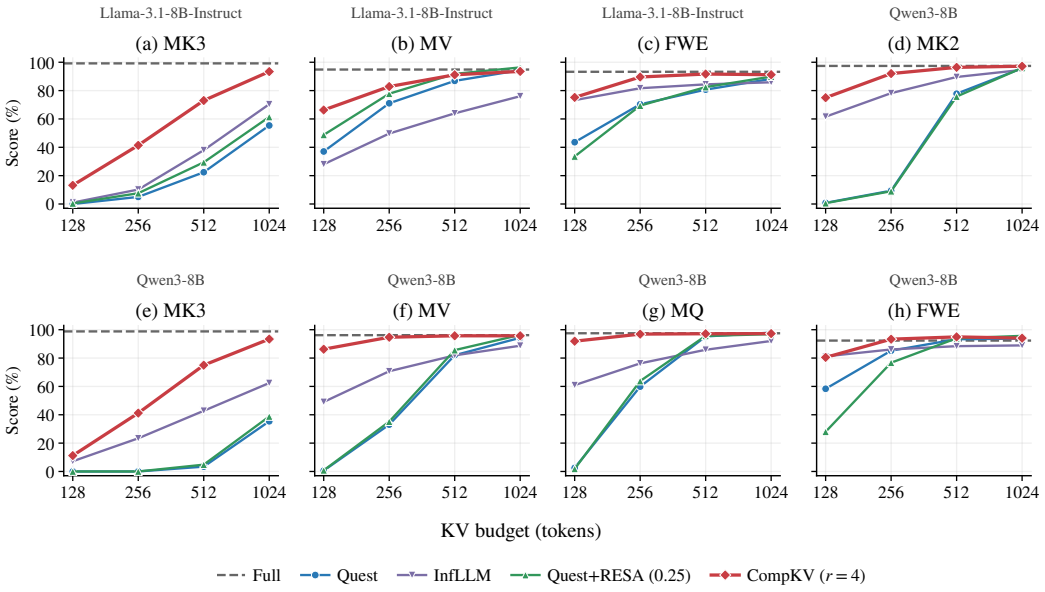


Figure 8: Accuracy across KV budgets on selected RULER tasks: (a–c) Llama-3.1-8B-Instruct and (d–h) Qwen3-8B. CompKV uses $r = 4$. Dashed gray lines denote Full attention.

and scores use FP32. Following the input construction and collection schedule in Quest's official profiling implementation (Tang et al., 2024), we use standard-normal random embeddings matched across methods by a saved RNG state. We prefill $L - 256$ tokens and decode 256 steps, defining L as the window endpoint. Steps 193–208 and layers 3–32 yield 480 calls per case. Sparse methods use 16-token blocks with one sink and two recent blocks included in the budget. Full transfers all valid CPU-resident KV at each step. Results are shown in Table 4.

A synchronized host timer starts after Q/K/V are ready and stops after the attention output and all state updates complete. It includes KV/summary updates, transfers, selection, CPU gathering, exact attention, compensation, merging, and InfLLM probability maintenance. Input generation, prefill, compilation, projections, RoPE, and MLP are excluded. Primary timing runs without a profiler. We report the mean and sample SD of all 480 calls. Speedups are ratios of reported means.

Table 4: CPU-offload attention-step latency, reported as mean ± sample SD (ms) over 480 calls per case. Lengths denote decode-window endpoints. Full is shared across the three budget rows at each length. Bold marks the lowest mean in each row.

Endpoint	Budget	Full	Quest	InfLLM	CompKV (r=4)	CompKV (r=128)
32K	512	2.536 ± 0.008	0.595 ± 0.013	0.919 ± 0.079	**0.527** ± 0.007	0.675 ± 0.006
	1024		0.750 ± 0.013	1.084 ± 0.596	**0.665** ± 0.017	0.806 ± 0.016
	2048		0.972 ± 0.025	1.293 ± 0.051	**0.891** ± 0.024	1.038 ± 0.015
64K	512	4.995 ± 0.007	0.944 ± 0.018	1.556 ± 0.041	**0.840** ± 0.006	1.134 ± 0.010
	1024		1.099 ± 0.016	1.706 ± 0.034	**0.916** ± 0.007	1.206 ± 0.019
	2048		1.331 ± 0.020	1.935 ± 0.056	**1.112** ± 0.021	1.391 ± 0.013
128K	512	9.966 ± 0.015	1.625 ± 0.024	2.877 ± 0.043	**1.456** ± 0.007	2.048 ± 0.033
	1024		1.795 ± 0.020	2.996 ± 0.020	**1.536** ± 0.010	2.119 ± 0.011
	2048		2.035 ± 0.029	3.269 ± 0.056	**1.618** ± 0.018	2.200 ± 0.021