VQKV: HIGH-FIDELITY KV CACHE COMPRESSION FOR MEMORY-LIMITED LONG-CONTEXT INFERENCE

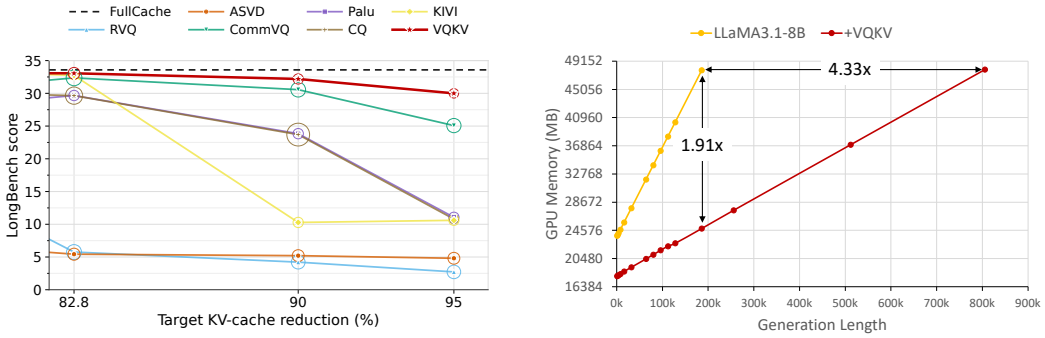
Anonymous authors
Paper under double-blind review

ABSTRACT

A compact key–value (KV) cache that represents every token reduces memory requirements for long-context inference. We present VQKV, which combines representation learning, output-aware calibration, and fused attention using shared post-RoPE codebooks. With residual SimVQ and asymmetric key/value capacity, VQKV retains 98.4% of the full-cache LongBench average on LLaMA3.1-8B at 82.8% code-length compression. It achieves the highest aggregate scores among the evaluated baselines at 90% and 95% compression. With the quantizer fixed, static value calibration improves 32K RULER by 3.22 points in the primary paired evaluation while preserving LongBench quality. It uses 128 KiB of parameters and leaves the encoder and cache format unchanged. Matched iterative and analytic MSE controls show the benefit of optimizing model outputs in addition to reconstruction. Fused FlashAttention integrates the correction into value lookup and computes attention directly from compact codes. By trading encoding and lookup computation for cache capacity, VQKV supports frozen-model inference on evaluated workloads whose full-cache memory requirements exceed GPU capacity.

1 INTRODUCTION

The key–value (KV) cache stores the states needed to attend to previous tokens. Its memory footprint grows with sequence length and batch size. Compression reduces this footprint, allowing longer sequences and larger workloads. Its practical value depends on prediction quality and the computation required to use the compressed representation. We study this quality–memory–computation tradeoff with frozen pretrained language models.



(a) Quality at high compression.

(b) Over 4.3× longer generation.

Figure 1: VQKV compresses the KV cache while preserving quality and extending generation capacity on LLaMA3.1-8B. (a) LongBench quality versus cache reduction. Circle area indicates measured generation time (smaller is faster). (b) Maximum generation length with a 65,536-token prompt on a 48GB GPU, FP16 and FlashAttention: approximately 190k to over 824k tokens. Timing protocol: Appendix F.

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.

We compress stored representations while retaining every token. Token-selection methods save memory by retaining a subset of past states (Zhang et al., 2023; Li et al., 2024b); low-rank representations and quantization reduce the storage cost of retained features (Wang et al., 2024; Chang et al., 2024; Yuan et al., 2025; Liu et al., 2024; Hooper et al., 2024; Zhang et al., 2024). Vector quantization (VQ) jointly encodes groups of features as indices into shared codebooks (Kumar, 2024; Li et al., 2025a; Zhang et al., 2025; Li et al., 2025b). Codebook capacity can be allocated flexibly while preserving access to the full sequence.

Residual VQ approximates a vector through successive codebooks (Gersho & Gray, 1991; Zeghidour et al., 2022). For KV-cache compression, the representation must capture positional structure, preserve the information used by attention, and remain compact during inference. Quantizer design must therefore account for both model behavior and the attention implementation.

Position is particularly relevant when quantizing keys. KVQuant quantizes keys before rotary position embedding (RoPE), whereas CommVQ uses RoPE-commutative codebooks (Hooper et al., 2024; Li et al., 2025a). Direct post-RoPE quantization learns the position-dependent states used by attention. An accurate shared quantizer can reuse codebooks across layers and heads, with separate capacity allocations for keys and values.

We introduce **VQKV**, which uses a common compressed representation for reconstruction, calibration, and attention. VQKV learns shared codebooks with residual SimVQ (RSimVQ) (Zhu et al., 2025), encoding keys after RoPE and allocating capacity separately to keys and values. Query–codebook products provide key-logit lookups, while value codewords are decoded in tiles. Fusing these operations with FlashAttention keeps intermediate KV states compressed during decoding, while small initial and recent regions remain in full precision.

We calibrate decoded values while keeping the codebooks and encoder fixed. A static per-channel correction is integrated into value lookup, using 128 KiB of shared parameters and no additional per-token codes or correction states. Because attention weights reconstruction errors according to the query, we optimize the model's output distribution alongside reconstruction. Matched iterative and analytic MSE controls isolate the effect of this objective. Figure 1 summarizes the quality, storage, and generation-capacity trade-offs.

Our main findings are:

- **High quality with compact post-RoPE codes.** At 82.8% code-length compression, VQKV retains 98.4% of the full-cache LongBench average and scores 100.00 on NIAH for LLaMA3.1-8B. It achieves the highest aggregate scores among the evaluated baselines at 90% and 95%, with strong LongBench retention across five models.
- **Output-aware calibration of a fixed quantizer.** A 128 KiB correction improves the primary paired 32K RULER score by 3.22 points while preserving LongBench quality. Iterative and analytic MSE controls establish the benefit of the calibration objective. Evaluations across six model/codec settings and the complete LLaMA3.2 input collection assess its broader applicability.
- **A practical memory–computation trade-off.** FlashAttention–VQ integrates code lookup and calibration into attention. Measurements show lower cache storage across batch sizes and successful inference on A100 40GB workloads for which resident FullCache runs out of memory. Encoding and lookup require additional computation while retaining every token in a smaller cache.

2 Related Work

Compression for long-context inference. Streaming and token-selection methods use recency, attention, or retrieval signals to reduce cache size (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024b; Cai et al., 2025). Latent attention and low-rank methods compress stored representations (DeepSeek-AI, 2024; Wang et al., 2024; Chang et al., 2024; Yuan et al., 2025), while quantization reduces the precision of keys and values (Liu et al., 2024; Hooper et al., 2024; Zhang et al., 2024). VQKV targets inference under memory constraints with frozen model weights and retains a representation of every token.

Vector quantization and positional information. VQ and residual VQ provide compact representations for vision, speech, and retrieval (van den Oord et al., 2017; Esser et al., 2021; Baevski et al., 2020; Zeghidour et al., 2022; Jégou et al., 2011; Johnson et al., 2021). SimVQ parameterizes code vectors with a learned linear transformation to improve codebook utilization (Zhu et al., 2025). Its residual implementation refines the approximation over successive stages (Wang, 2026). VQKV uses shared residual SimVQ codebooks with asymmetric key/value capacity to encode the states consumed by attention. Related KV methods use residual VQ (Kumar, 2024), RoPE-commutative additive quantization in CommVQ (Li et al., 2025a), product-quantization-based retrieval in PQCache (Zhang et al., 2025), or anchor-aware precision allocation in AnTKV (Li et al., 2025b). VQKV's codebooks learn post-RoPE keys directly, and the fused attention path operates on the resulting uniform code sequence.

Attention over compressed representations. FlashAttention reduces memory traffic through tiled computation and online softmax normalization (Dao et al., 2022; Dao, 2024). In VQKV, tiled attention uses query–codebook products and value-tile lookup to compute attention directly from compressed states. The intermediate cache remains in code form.

Calibration beyond reconstruction. KVLinC uses learned feature maps and input-dependent correction states (Saxena & Roy, 2025). OptR learns K/V rotations before quantization using attention and output-projection objectives (Yun et al., 2026). VQKV calibrates decoded values after fixing its quantizer. The codebooks, encoder, and cache format remain unchanged. Static channel-wise parameters are applied within the existing value lookup without sequence-dependent correction states. Matched iterative and analytic MSE controls measure the contribution of fitting the model's final output. The calibration operates on the same representation used by compressed attention.

3 VQKV: Accurate Compression and Fused Attention

VQKV separates representation learning from readout adaptation. Shared codebooks encode post-RoPE keys and unrotated values. A static correction refines decoded values, and a fused kernel integrates lookup and correction into attention. Small initial and recent regions remain in full precision. Every token remains available to attention. Figure 2 illustrates the process from encoding to inference.

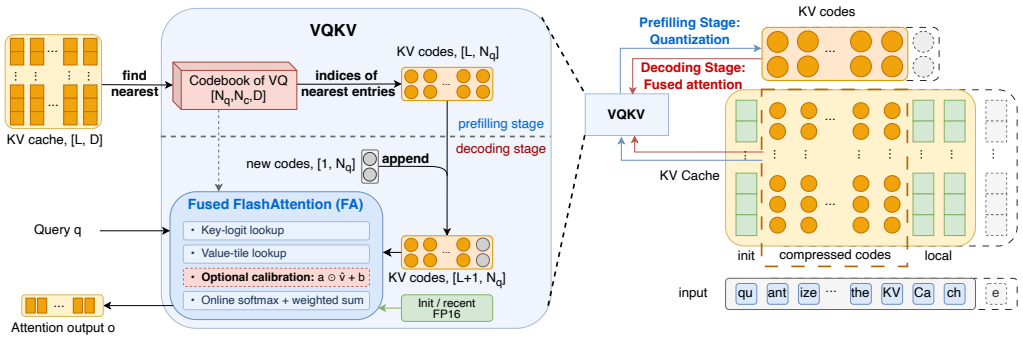


Figure 2: VQKV encoding, storage, and fused FlashAttention. Shared residual codebooks represent intermediate KV states, while initial and recent regions remain in full precision. During decoding, the kernel integrates key-logit and value-tile lookup, online softmax, and optional value calibration while retaining the existing cache format. N_q and N_c denote the number and size of the codebooks.

3.1 High-fidelity reconstruction of post-RoPE keys

RoPE applies position-dependent rotations to keys (Su et al., 2024). A shared quantizer must therefore capture both content variation and positional structure. KVQuant quantizes keys before RoPE (Hooper et al., 2024), whereas VQKV learns the post-RoPE representation directly. For a query

q, key reconstruction error $e_t = k_t - \hat{k}_t$ changes the attention logit by at most $\|q\|_2\|e_t\|_2/\sqrt{D^k}$. Reducing key reconstruction error therefore helps preserve attention with a small code budget.

Let $x \in \mathbb{R}^D$ denote a key or value vector, and let Q_i contain S basis entries for residual stage i. SimVQ parameterizes an effective code vector as $c_{i,j} = W_i Q_i[j]$, where W_i is a learned linear projection. Starting from $r_1 = x$, encoding and reconstruction are defined by

$$z_i = \underset{j\in\{1,\dots,S\}}{\operatorname{argmin}} \|r_i - c_{i,j}\|_2^2, \qquad r_{i+1} = r_i - c_{i,z_i}, \qquad \hat{x} = \sum_{i=1}^{N} c_{i,z_i}. \tag{1}$$

The tuple $(z_1, \dots, z_N)$ replaces the floating-point vector. SimVQ parameterizes its codebooks with learned transformations, and successive residual stages refine the approximation (Zhu et al., 2025; Zeghidour et al., 2022). The effective entries $W_i Q_i[j]$ are fixed at inference; neither independent subspaces nor RoPE-commutative codebooks are required.

Keys and values use separate (N^k, S^k) and (N^v, S^v) configurations, each shared across all layers and KV heads. The asymmetric allocation assigns more residual stages to position-dependent keys, while sharing amortizes codebook storage across the growing cache. The auxiliary quantizers are trained on approximately 10M cached tokens while the LLM and basis entries Q_i remain frozen. The objective includes the final reconstruction loss and two auxiliary VQ losses averaged across residual stages:

$$\begin{aligned}\mathcal{L} = \frac{1}{D}\|x - \hat{x}\|_2^2 + \frac{\beta}{ND}\sum_{i=1}^{N}\|W_i Q_i[z_i] - \mathrm{sg}(r_i)\|_2^2 \\ + \frac{\gamma}{ND}\sum_{i=1}^{N}\|r_i - \mathrm{sg}(W_i Q_i[z_i])\|_2^2,\end{aligned} \tag{2}$$

where sg denotes the stop-gradient operator, and β, γ weight the codebook-fitting and input-commitment terms. The first term measures reconstruction error after all codewords are summed; the auxiliary terms compare each selected codeword with the corresponding stage residual. With frozen inputs, detached residual updates, and the quantizer's input-directed gradient surrogate, only the fitting term backpropagates to W_i. We evaluate fidelity and select configurations using held-out final reconstruction MSE. Figure 6 plots RSimVQ's post-RoPE key reconstruction error against codebook parameter count. Section 4.6 evaluates the downstream effect of key/value capacity.

3.2 Token-preserving compression throughout generation

The initial L_{init} tokens and recent tokens remain in FP16. During prefill, each layer first attends to its full-precision states, then compresses the intermediate keys and values and releases their full-precision copies. Earlier layers retain compact codes; the current layer, attention workspace, and encoding buffers still contribute to peak memory.

During decoding, new states enter the recent region; states leaving this region are encoded and appended to the code sequence. Block-wise distance calculations and encoding in batches over L_{local} steps amortize nearest-codeword search. States remain in full precision until encoded, preserving token coverage. Token tiles bound the encoding workspace, and attention reads stored indices without repeatedly copying the growing sequence.

Reconstructed keys already incorporate RoPE; values retain their standard attention representation. The compressed region uses a uniform code format, with initial and recent tokens exempt from quantization.

3.3 Fusing VQ lookup with FlashAttention

For a decoding query q, define $c^k_{i,j} = W^k_i Q^k_i[j]$. Query–codebook products can be computed once for that query and reused across compressed tokens:

$$P^k_i[j] = q^\top c^k_{i,j}, \qquad \ell_t = \frac{q^\top \hat{k}_t}{\sqrt{D^k}} = \frac{1}{\sqrt{D^k}}\sum_{i=1}^{N^k} P^k_i[z^k_{t,i}]. \tag{3}$$

For uncompressed tokens, the kernel computes the standard query–key product. If α_t is the softmax weight over *all* eligible tokens, the output is

$$o = \sum_{t \in \mathcal{U}} \alpha_t v_t + \sum_{t \in \mathcal{C}} \alpha_t \left(\sum_{i=1}^{N^v} c^v_{i, z^v_{t,i}} \right), \tag{4}$$

where $\mathcal{U}$ and $\mathcal{C}$ denote the uncompressed and compressed positions. Both sets require a shared softmax normalization; separately normalized block outputs cannot be directly added.

The fused kernel integrates these lookups with FlashAttention's tiling and online softmax (Dao et al., 2022; Dao, 2024). It uses paged storage for recent tokens and reconstructs only the value tile currently being accumulated, avoiding a separate full-cache reconstruction pass during decoding. Section 4.5 reports end-to-end generation time, including online encoding.

3.4 Compression and memory accounting

We define compression in terms of the fixed-length binary codes that represent intermediate KV states. FP16 keys and values use $16(D^k + D^v)$ bits per token and KV head; the code length is $N^k \lceil \log_2 S^k \rceil + N^v \lceil \log_2 S^v \rceil$. Their relative reduction is

$$r_{\text{nom}} = 1 - \frac{N^k \lceil \log_2 S^k \rceil + N^v \lceil \log_2 S^v \rceil}{16(D^k + D^v)}. \tag{5}$$

With $D^k = D^v = 128$, the LLaMA3.1-8B configuration represents each key/value pair with $56 \cdot 10 + 16 \cdot 9 = 704$ code bits versus 4096 FP16 bits, giving 82.8% compression. This code-length definition is used throughout the quality comparisons.

The inference implementation stores indices in 16-bit containers and retains the initial/recent regions, shared codebooks, and allocated buffers. Section 4.5 reports measurements of the complete cache. Peak GPU memory also includes model weights and temporary tensors used in computation. Appendix E details the storage accounting.

3.5 Output-aware readout calibration

Reconstruction MSE weights errors in stored vectors uniformly, whereas attention combines these errors using query-dependent weights. For full-cache and compressed outputs $o = \alpha^\top V$ and $\widehat{o} = \widehat{\alpha}^\top \widehat{V}$,

$$\widehat{o} - o = (\widehat{\alpha} - \alpha)^\top V + \widehat{\alpha}^\top (\widehat{V} - V). \tag{6}$$

For fixed attention weights, let $E = \widehat{V} - V$. The readout error is $\|\alpha^\top E\|^2$, whereas reconstruction MSE is $\|E\|_F^2/(Td)$. Equal-norm error matrices can align with α or be orthogonal to it, producing different readout errors. This dependence on error direction motivates jointly optimizing output-distribution agreement and reconstruction error. A static value correction adjusts decoded values within the existing attention lookup. With codebooks, encoder, and model weights fixed, we apply a channel-wise affine correction to compressed values in each layer and KV head:

$$\widetilde{v}_{\ell h t} = a_{\ell h} \odot \widehat{v}_{\ell h t} + b_{\ell h}, \qquad \mathcal{L}_{\text{cal}} = \overline{D}_{\text{KL}}(p_{\text{full}} \| p_{\text{cal}}) + 0.1\, \overline{\mathcal{L}}_{V\text{-MSE}}. \tag{7}$$

Here, p denotes the final vocabulary distribution, and overbars indicate normalization by the uncalibrated error. We fit bounded scales and biases on eight retrieval-augmented OpenWebText prompts (Gokaslan & Cohen, 2019) for 200 updates, using the full-cache model as the teacher. At inference, the correction is fused into value lookup and requires neither a teacher model nor additional per-token codes. LLaMA3.1-8B uses 65,536 parameters (128 KiB in FP16). The MSE-only control uses the same parameter family and fitting budget. Appendix H gives the training and numerical protocols.

4 Experiments

We evaluate task quality at high compression, output-aware calibration with a fixed quantizer, and the memory–computation trade-off of fused inference. LongBench, NIAH, and RULER measure task quality; paired reconstruction controls isolate the effect of the calibration objective.

Table 1: Quality at target KV-cache reductions of 80%, 90%, and 95% on LLaMA3.1-8B. LB denotes the LongBench average, N denotes NIAH, and R denotes the RULER average over 4K–32K; higher is better. Methods use supported configurations near each target. Bold indicates the best compressed-cache score in each column, including ties. The 95% group includes configurations at 94% reduction.

	80% target			90% target			95% target		
Method	LB	N	R	LB	N	R	LB	N	R
Full cache	33.58	100.00	92.79	33.58	100.00	92.79	33.58	100.00	92.79
KIVI	**33.16**	**100.00**	**88.27**	10.28	2.91	2.46	10.60	2.69	1.78
ASVD	6.42	3.29	0.56	5.21	3.75	0.08	4.81	2.54	0.04
Palu	28.47	**100.00**	83.98	23.84	88.79	53.82	11.07	9.09	7.53
RVQ	12.61	39.58	4.95	4.22	3.56	0.62	2.73	3.46	0.16
CommVQ	31.17	94.93	58.56	30.56	93.23	45.99	25.07	23.36	11.69
CQ	30.00	**100.00**	87.87	23.70	46.00	25.09	10.79	4.43	3.40
VQKV	33.09	**100.00**	88.09	**32.19**	**99.71**	**73.44**	**30.00**	**37.36**	**23.70**

Table 2: Selected configurations across five models. Reduction is computed as described in Section 3.4; Full uses the uncompressed cache. Scores cover LongBench, NIAH, and RULER. The selected LLaMA configurations differ from those in the target-ratio sweep in Table 1; Table 7 compares their baselines.

		LongBench		NIAH		RULER	
Model	Red. (%)	Full	VQKV	Full	VQKV	Full	VQKV
LLaMA3.1-8B	82.80	33.58	33.06	100.00	100.00	92.79	87.11
LLaMA3.2-3B	82.40	32.30	32.20	100.00	100.00	84.40	77.10
Qwen2.5-1.5B	82.20	32.83	32.38	98.84	98.41	74.86	64.20
Qwen2.5-3B	82.90	38.21	37.73	100.00	99.56	80.95	74.40
Qwen3-8B	82.80	37.78	36.14	99.71	99.85	92.16	89.11

4.1 Experimental setup

We evaluate LLaMA3.1-8B, LLaMA3.2-3B (Grattafiori et al., 2024; Meta AI, 2024), and Qwen2.5-1.5B/3B and Qwen3-8B (Yang et al., 2024; 2025), training codebooks separately for each model. Quantizers are trained on approximately 10M OpenWebText KV tokens with a learning rate of 0.001 and a batch size of 65,536. We select codebooks by held-out reconstruction loss at the target bit budget. LLaMA3.1-8B uses K56×1024/V16×512; LLaMA3.2-3B uses K56×1024/V10×65536. Both use initial and recent windows of 4 and 1,024 tokens. Appendix A provides baseline configurations and implementation details.

Using OpenCompass (OpenCompass Contributors, 2023), we evaluate the 16 LongBench tasks with a 32K limit (Bai et al., 2024), NeedleBench NIAH through 32K (Li et al., 2024a), and RULER at 4K/8K/16K/32K (Hsieh et al., 2024). LLaMA3.1 results at 80%–95% use the FP16 fused VQKV implementation; Appendix A specifies the implementation used in each study. Within each calibration study, variants are evaluated on identical inputs and cache windows against the same full-cache control. Baselines include SnapKV, KIVI, ASVD, Palu, RVQ, CommVQ, and CQ, with implementations based on publicly released code. SnapKV NIAH scores are from Liu et al. (2025).

4.2 Maintaining quality at high compression

Table 1 compares LLaMA3.1-8B at 80%, 90%, and 95% compression, using supported configurations near each target. Appendix D summarizes the sweep. The results supplement provides task and length breakdowns, including the 60% setting.

VQKV performs particularly well at high compression. At 90%, its LongBench, NIAH, and RULER scores are 32.19, 99.71, and 73.44, respectively. The best baseline scores are 30.56, 93.23, and 53.82. VQKV also leads all three aggregate comparisons at 95%, retaining 89.3% of the full-cache

LongBench average (30.00 versus 33.58). At the 80% target, it retains 98.5% of the full-cache LongBench average and scores 100.00 on NIAH and 88.09 on RULER.

4.3 HIGH-FIDELITY COMPRESSION ACROSS MODELS

Table 2 reports results for the same codebook-learning procedure applied to five models. At 82.8%/82.4% compression, LLaMA3.1/3.2 retain 98.4%/99.7% of the full-cache LongBench average and score 100.00 on NIAH. The Qwen models retain 95.7%–98.7% of LongBench at 82.2%–82.9% compression. These learned representations provide the basis for subsequent readout calibration. Appendix B compares the selected LLaMA configurations with baselines; Appendix C reports length-specific results.

4.4 FROM RECONSTRUCTION FIDELITY TO LONG-CONTEXT CALIBRATION

Improving predictions with a fixed quantizer. We compare uncalibrated, MSE-calibrated, and output-calibrated VQKV with identical codebooks, compression, and model weights. The paired FullCache control provides the reference scores (Table 3).

Table 3: Calibration results for LLaMA3.1-8B at 82.8% compression. RULER and LongBench comparisons pair methods on the same 48 GB GPU with common value-reconstruction arithmetic. Annotations indicate changes from VQKV; MSE denotes reconstruction-only calibration. Appendix I describes the protocol and provides the full results.

Method	RULER 4K	RULER 32K	LongBench
Full cache	96.08	87.64	33.52
VQKV	92.83	76.35	33.02
VQKV + MSE	93.63 +0.80	78.47 +2.12	33.02 +0.00
VQKV + Calibration	93.63 +0.80	79.57 +3.22	33.07 +0.05

VQKV with calibration raises 32K RULER to 79.57, a paired gain of 3.22 points (95% confidence interval [2.20, 4.27]). MSE-only calibration gains 2.12 points; the output-aware objective adds a further 1.10 points [0.30, 1.92]. All compressed variants use the same value-reconstruction arithmetic.

Preserving LongBench quality. The LongBench average changes from 33.02 to 33.07: $+0.05$ points $[-0.11, 0.21]$, meeting the prespecified preservation criteria. The mean change and the lower bound of its 95% confidence interval are each at least -0.30 points, and every task change is at least -2.00. Calibration thus improves 32K retrieval while maintaining aggregate LongBench quality. Appendix I reports task-specific changes and additional input coverage.

Stronger multi-item retrieval. Seven tasks improve and six are unchanged. Multi-query retrieval rises from 68.50 to 95.50, closing 90% of its gap to FullCache; multi-value retrieval gains 6.75 points. The gains are concentrated in tasks that retrieve multiple items from the compressed context. Appendix I, Table 14, gives all 13 tasks and their full-cache controls.

Effect of the calibration objective. Table 4 compares calibration objectives using the same bounded affine family and fixed codebooks. The analytic MSE fit achieves the lowest training and held-out reconstruction error for both RSimVQ and RVQ. Output calibration achieves the highest 32K RULER scores, exceeding the analytic fit by 1.31 and 4.91 points. Both paired gains have confidence intervals above zero after multiple-comparison correction. Together with the iterative-MSE control, these results support fitting the decoded representation to model predictions (Appendix J).

Table 4: Reconstruction fidelity and 32K RULER results with fixed LLaMA3.1-8B codebooks. Top: normalized V-MSE (lower is better) and RULER (higher is better), evaluated on separate data distributions. Bottom: all four prespecified paired RULER contrasts, with individual 95% intervals and Bonferroni-adjusted intervals (four comparisons). Analytic-MSE optimizes the continuous objective before FP16 export; Appendix J describes the protocol.

	RSimVQ			RVQ		
Readout	Train	Held-out	RULER	Train	Held-out	RULER
Base	1.0000	1.0000	79.68	1.0000	1.0000	79.10
Iterative-MSE	0.9114	0.9123	81.88	0.9537	0.9552	77.34
Analytic-MSE	0.8928	0.8938	81.67	0.9482	0.9498	76.28
Output	0.9978	0.9978	82.99	1.0016	1.0017	81.20

Codec	Difference	Mean	95% CI	Bonferroni4 CI
RSimVQ	Analytic-MSE−Iterative-MSE	-0.21	$[-0.82, 0.39]$	$[-0.98, 0.60]$
RSimVQ	Output−Analytic-MSE	+1.31	$[0.43, 2.24]$	$[0.18, 2.51]$
RVQ	Analytic-MSE−Iterative-MSE	-1.06	$[-1.69, -0.46]$	$[-1.86, -0.32]$
RVQ	Output−Analytic-MSE	+4.91	$[3.66, 6.16]$	$[3.35, 6.51]$

Results across models, codecs, and input collections. Table 5 extends the paired evaluation to six settings. For both LLaMA3.1 codecs, LLaMA3.2-3B, and Qwen3-8B, confidence intervals for the gains remain above zero after correction. LLaMA3.2 and Qwen3 gain 4.76 and 2.84 points. A separate fixed-checkpoint LLaMA3.2 evaluation covers 5,200 RULER and 660 NIAH inputs: calibration raises the RULER average across four lengths from 77.14 to 80.59 while maintaining a NIAH score of 100.00 (Appendix I). For the fixed KIVI and xKV configurations, confidence intervals for the gains also lie entirely above zero after correction. Appendix K reports all paired contrasts and model-specific LongBench tests.

Table 5: 32K RULER results across six model/codec settings (13 tasks, 100 examples per task). Cal. uses output-aware calibration; Δ and its interval compare Cal. with Base. Intervals are Bonferroni-adjusted over all twelve contrasts in Appendix K. These runs use a collection separate from that in Table 3.

Model	Codec	Base	Cal.	Δ	Adjusted CI
LLaMA3.1-8B	RSimVQ	79.59	82.99	+3.40	[1.91, 5.04]
LLaMA3.1-8B	RVQ	78.83	81.09	+2.26	[0.71, 3.81]
LLaMA3.2-3B	RSimVQ	65.96	70.72	+4.76	[3.12, 6.55]
Qwen2.5-1.5B	RSimVQ	56.15	55.46	-0.69	[-1.71, 0.29]
Qwen2.5-3B	RSimVQ	67.26	67.83	+0.58	[-0.53, 1.74]
Qwen3-8B	RSimVQ	70.50	73.33	+2.84	[1.48, 4.39]

4.5 The memory–computation trade-off

Longer generation. VQKV increases the memory-limited generation capacity by over $4.3\times$ under the same GPU budget (Figure 1b). This capacity measurement complements task-quality evaluations up to 32K; Appendix F specifies the protocol.

Trading computation for cache capacity. VQKV accommodates workloads whose full cache exceeds GPU memory by encoding and looking up a compact representation of every token. On A100 40GB, VQKV with calibration completes batch 8 with 16K prompts and batch 16 with 8K prompts, while resident FullCache runs out of memory (Table 10). Table 9 reports the runtime cost when both representations fit in memory. Fused attention operates directly on the compressed cache, avoiding full reconstruction during decoding.

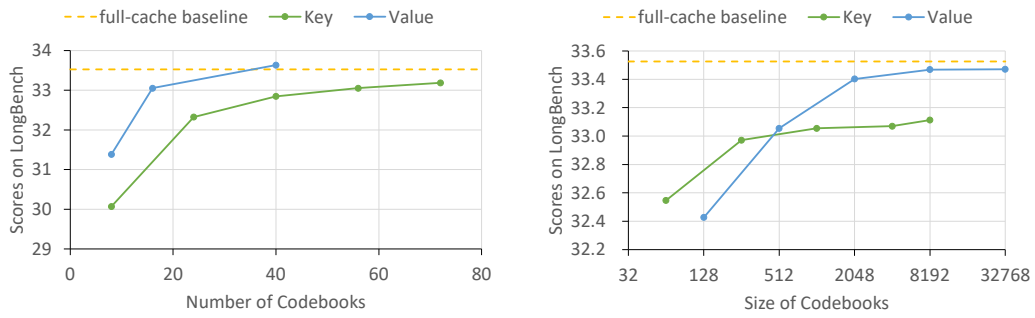
Table 6: Actual cache storage on RTX 6000 Ada (48 GB), with peak allocations for reference (GiB): 8192 input tokens and 64 output tokens. VQKV+Cali uses the 82.8% configuration and windows 4/1024. After the final forward pass, cache storage includes allocated buffers, shared codebooks, and 128 KiB of calibration parameters; peaks also include weights and workspace. Measurements use three timed runs after one warmup.

BS	**Cache storage**		Prefill peak		Decode peak	
	Full	VQKV+Cali	Full	VQKV+Cali	Full	VQKV+Cali
1	1.01	**0.42**	16.88	16.31	15.99	15.47
2	2.02	**0.83**	18.78	17.58	17.01	15.88
4	4.03	**1.64**	22.59	20.14	19.06	16.71
8	8.06	**3.27**	30.22	25.25	23.15	18.39
16	16.12	**6.52**	45.47	35.47	31.34	21.75

Calibration uses the same inference path. With matched reconstruction arithmetic, it changes mean generation time from 26.79 to 26.82 seconds (0.12%) and adds 128 KiB of static parameters. Appendix L reports the workload and matched controls.

Cache savings across batch sizes. Table 6 reports complete cache storage: indices, full-precision regions, shared codebooks, and allocated buffers. Calibrated VQKV uses approximately 58%–60% less cache storage across batch sizes 1–16 with 8K inputs. Peak allocations also include model weights and temporary workspace.

4.6 Codebook capacity ablations



(a) Number of codebooks.

(b) Size of codebooks.

Figure 3: Codebook ablations on LLaMA3.1-8B. Each curve varies a single codebook count (a) or size (b); all other codebook settings follow the main 82.8% configuration. Varying capacity also varies the storage budget. Appendix G, Table 13, reports configurations and scores.

Increasing the number of key codebooks from 8 to 24 raises LongBench from 30.07 to 32.33 and NIAH from 37.51 to 99.86; increasing the number of value codebooks from 8 to 16 raises RULER from 61.16 to 87.27 (Figure 3). These sweeps inform codebook capacity allocation; calibration improves the decoded output while keeping the quantizer fixed. Appendix G includes reconstruction errors and full configurations.

5 Discussion and Conclusion

VQKV combines shared post-RoPE codebooks, a small static value correction, and fused attention for high-quality inference with a compact, token-preserving cache. Matched reconstruction controls demonstrate the benefit of output-aware calibration with a fixed encoder and cache format. Encoding and lookup add computation while reducing cache storage, enabling workloads beyond resident full-cache capacity.

AI USE STATEMENT

Generative AI tools assisted with writing and coding. The authors are responsible for experimental validation and the final content.

ETHICS STATEMENT

This work evaluates KV-cache compression using existing pretrained models, corpora, and benchmarks. It does not collect new data from human participants. Compression can introduce generation errors, and applications retain the limitations of the underlying models.

REPRODUCIBILITY STATEMENT

Sections 3 and 4.1 describe the method and principal experiment settings. Appendices B–G provide detailed scores, compression sweeps, and ablations. Appendix A specifies evaluation and aggregation conventions, and Appendix F describes the timing protocols. Timing repetitions for Figure 1 are documented in Appendix F.

REFERENCES

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html.

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. *arXiv preprint arXiv:2308.14508*, 2024. URL https://arxiv.org/abs/2308.14508.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2025. URL https://arxiv.org/abs/2406.02069.

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: Compressing KV-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024. URL https://arxiv.org/abs/2407.21118.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/98ed250b203d1ac6b24bbcf263e3d4a7-Abstract-Conference.html.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. doi: 10.52202/068431-1189.

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

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021. doi: 10.1109/cvpr46437.2021.01268.

Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 1991. doi: 10.1007/978-1-4615-3626-0.

Aaron Gokaslan and Vanya Cohen. OpenWebText corpus, 2019. URL https://skylion007.github.io/OpenWebTextCorpus/.

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

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024. doi: 10.52202/079017-0040.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024. URL https://arxiv.org/abs/2404.06654.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/tpami.2010.57.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/tbdata.2019.2921572.

Ankur Kumar. Residual vector quantization for KV cache compression in large language model. In *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pp. 485–490. PMLR, 2024. URL https://proceedings.mlr.press/v262/kumar24a.html.

Junyan Li, Yang Zhang, Muhammad Yusuf Hassan, Talha Chafekar, Tianle Cai, Zhile Ren, Pengsheng Guo, Foroozan Karimzadeh, Colorado Reed, Chong Wang, and Chuang Gan. CommVQ: Commutative vector quantization for KV cache compression. *arXiv preprint arXiv:2506.18879*, 2025a. URL https://arxiv.org/abs/2506.18879.

Mo Li, Songyang Zhang, Taolin Zhang, Haodong Duan, Yunxin Liu, and Kai Chen. NeedleBench: Evaluating LLM retrieval and reasoning across varying information densities. *arXiv preprint arXiv:2407.11963*, 2024a. URL https://arxiv.org/abs/2407.11963.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024b. doi: 10.52202/079017-0722.

Zeyu Li, Chuanfu Xiao, Yang Wang, Xiang Liu, Zhenheng Tang, Baotong Lu, Mao Yang, Xinyu Chen, and Xiaowen Chu. AnTKV: Anchor token-aware sub-bit vector quantization for KV cache in large language models. *arXiv preprint arXiv:2506.19505*, 2025b. URL https://arxiv.org/abs/2506.19505.

Xiaoran Liu, Siyang He, Qiqi Wang, Ruixiao Li, Yuerong Song, Zhigeng Liu, Linlin Li, Qun Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Beyond homogeneous attention: Memory-efficient LLMs via fourier-approximated KV cache. *arXiv preprint arXiv:2506.11886*, 2025. URL https://arxiv.org/abs/2506.11886.

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. *arXiv preprint arXiv:2402.02750*, 2024. URL https://arxiv.org/abs/2402.02750.

Meta AI. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models, 2024. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

OpenCompass Contributors. OpenCompass: A universal evaluation platform for foundation models, 2023. URL https://github.com/open-compass/opencompass.

Utkarsh Saxena and Kaushik Roy. KVLinC: KV cache quantization with hadamard rotation and linear correction, 2025. URL https://arxiv.org/abs/2510.05373.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

Luning Wang, Shiyao Li, Xuefei Ning, Zhihang Yuan, Shengen Yan, Guohao Dai, and Yu Wang. CSKV: Training-efficient channel shrinking for KV cache in long-context scenarios. *arXiv preprint arXiv:2409.10593*, 2024. URL https://arxiv.org/abs/2409.10593.

Phil Wang. vector-quantize-pytorch, 2026. URL https://github.com/lucidrains/vector-quantize-pytorch.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2024. URL https://arxiv.org/abs/2309.17453.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

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. URL https://arxiv.org/abs/2505.09388.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2025. URL https://arxiv.org/abs/2312.05821.

Vincent-Daniel Yun, Woosang Lim, Minsoo Cheong, Sunwoo Lee, Murali Annavaram, Sai Praneeth Karimireddy, and Sungjoo Yoo. Output-aware rotation for INT2 KV-cache quantization, 2026. URL https://arxiv.org/abs/2608.02691.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. SoundStream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2022. doi: 10.1109/taslp.2021.3129994.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. PQCache: Product quantization-based KVCache for long context LLM inference. *arXiv preprint arXiv:2407.12820*, 2025. URL https://arxiv.org/abs/2407.12820.

Tianyi Zhang, Jonah Yi, Zhaozhuo Xu, and Anshumali Shrivastava. KV cache is 1 bit per channel: Efficient large language model inference with coupled quantization. *arXiv preprint arXiv:2405.03917*, 2024. URL https://arxiv.org/abs/2405.03917.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023. doi: 10.52202/075280-1506.

Yongxin Zhu, Bocheng Li, Yifei Xin, Zhihua Xia, and Linli Xu. Addressing representation collapse in vector quantized models with one linear layer. *arXiv preprint arXiv:2411.02038*, 2025. URL https://arxiv.org/abs/2411.02038.

A EXPERIMENTAL PROTOCOLS

Quality experiments use the pretrained models and quantizer settings in Section 4.1. Each model's key and value codebooks are trained offline with the language-model weights frozen, then shared across its layers and KV heads. Training uses activations for approximately 10M tokens from a 0.1% sample of OpenWebText.

Baseline configurations. The selected baselines use 2048 recalled middle tokens for SnapKV, ASVD ratio 0.2, RVQ shape $(32, 5, 1024, 16)$, 22 key residuals for CommVQ, and CQ 4c11b. The selected KIVI LongBench configuration uses 2-bit keys and values, group size 64, and a 1024-token residual window. Compression targets use the code-length definition in Section 3.4; the selected configurations and retention windows are specified above. Within each calibration study, branches share the same cache windows and compressed representation.

The RVQ calibration controls in Tables 4 and 5 use a separately trained K56/V16 configuration at 82.8% target reduction, with windows 4/1024. These controls differ from the RVQ baseline in Table 7 in both configuration and evaluation: their 32K scores use different input collections from the baseline's four-length RULER average. Each comparison evaluates calibration within a fixed codec configuration.

Scores and aggregation. LongBench averages its 16 task scores with equal weights; RULER equally weights the four evaluated context lengths. NIAH averages NeedleBench retrieval scores over the evaluated lengths and needle positions. Scores and timings are rounded for display; relative changes and paired differences are computed before rounding. LLaMA3.1-8B LongBench retention is computed from unrounded task means: the 82.8% configuration retains 98.4% of the full-cache score. Quality tables report point estimates; timing repetitions are specified in Appendix F.

Implementation and hardware. LLaMA3.1-8B VQKV quality results at 80%, 82.8%, 90%, and 95% use the optimized encoding and fused-attention implementation. Each configuration covers all 3,750 LongBench examples, 5,200 RULER examples, and 660 NIAH examples. Implementation checks pair each example with its reference and full-cache branches on the same GPU, using identical FP16 inputs, generation settings, and scorers. Tables use the complete RTX 6000 Ada full-cache evaluation as a common reference. Other models, baseline quality results, the 60% sweep, and codebook ablations use the reference evaluation setup. Timing comparisons use the same GPU model and workload within each study. In Table 10, Full/Offloaded and VQKV/calibration each share one A100; the two pairs run concurrently on separate devices of the same model. The accompanying timing protocols specify device allocation and run settings.

Artifacts. The experiments build on pretrained models, OpenWebText, LongBench, NeedleBench, RULER, OpenCompass, PyTorch, Transformers, FlashAttention, and Triton. Use and redistribution follow the original providers' terms. The accompanying *Complete Results* supplement contains full task scores, additional studies, and implementation settings; its index identifies each study and protocol.

B Detailed LongBench Results

Table 7 compares the selected high-fidelity LLaMA configurations with the baselines. The selected reductions vary by method; the target-ratio comparison appears in Table 1.

Table 7: Selected high-fidelity configurations on two LLaMA models, distinct from the target-ratio sweep in Table 1. Reduction uses the code-length definition in Section 3.4. LB denotes LongBench; R denotes the RULER average over 4K–32K. Bold indicates the best compressed-cache score for each model and metric.

	LLaMA3.1-8B				LLaMA3.2-3B			
Method	Red. (%)	LB	NIAH	R	Red. (%)	LB	NIAH	R
Full cache	0.00	33.58	100.00	92.79	0.00	32.30	100.00	84.40
SnapKV	80.00	32.69	88.35	74.20	80.00	31.40	90.29	68.80
KIVI	82.80	30.61	99.43	83.38	82.80	29.30	98.70	41.30
ASVD	80.00	6.42	3.29	0.56	80.00	14.90	36.40	18.40
Palu	80.00	28.47	**100.00**	83.98	80.00	28.10	35.99	63.00
RVQ	80.00	12.61	39.58	4.95	80.00	7.52	25.14	2.17
CommVQ	82.80	32.36	97.82	65.63	82.40	31.22	99.85	71.63
CQ	82.80	29.65	99.71	83.15	82.40	29.15	97.71	64.23
VQKV	82.80	**33.06**	**100.00**	**87.11**	82.40	**32.20**	**100.00**	**77.10**

The results supplement provides the complete 16-task LongBench breakdowns.

C Length-Specific RULER and NIAH Results

Table 8 reports RULER scores at 4K, 8K, 16K, and 32K, their average, and NIAH scores. SnapKV NIAH scores are from Liu et al. (2025); the remaining results use the evaluation setup in Section 4.1.

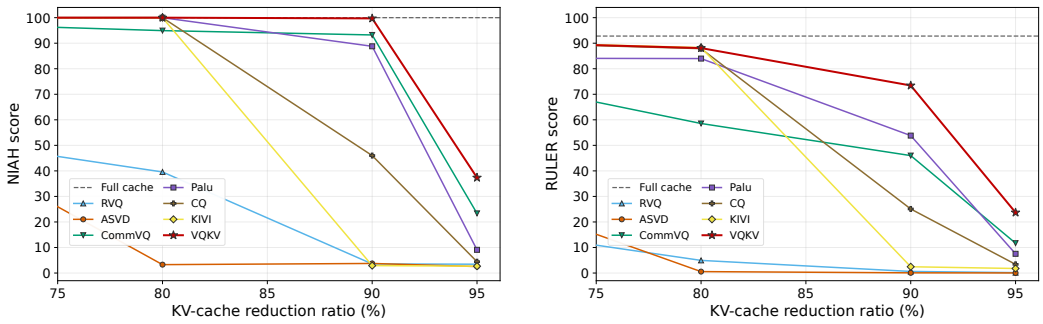
Table 8: NIAH and RULER results; Ratio denotes KV-cache reduction.

	Ratio	**NIAH**	**RULER**				
			4K	8K	16K	32K	Avg.
LLaMA3.1-8B	0	100.00	94.74	92.86	93.43	90.16	92.79
+ SnapKV	80.00%	88.35	91.20	77.10	69.80	58.50	74.20
+ KIVI	82.80%	99.43	89.04	85.90	82.85	75.71	83.38
+ ASVD	80.00%	3.29	0.92	0.92	0.21	0.21	0.56
+ Palu	80.00%	**100.00**	89.86	84.67	83.44	77.96	83.98
+ RVQ	80.00%	39.58	9.17	4.93	4.20	1.48	4.95
+ CommVQ	82.80%	97.82	76.71	70.34	62.35	53.10	65.63
+ CQ	82.80%	99.71	91.44	87.35	81.57	72.22	83.15
+ VQKV	82.80%	**100.00**	**92.73**	**89.57**	**86.74**	**79.39**	**87.11**
+ VQKV + Calibration	82.80%	**100.00**	**93.24**	**90.69**	**88.26**	**82.94**	**88.78**
LLaMA3.2-3B	0	100.00	90.00	85.90	83.30	78.30	84.40
+ SnapKV	80.00%	90.29	85.30	69.60	61.70	58.70	68.80
+ KIVI	82.80%	98.70	37.90	49.40	41.40	36.30	41.30
+ ASVD	80.00%	36.40	27.20	20.80	16.00	9.50	18.40
+ Palu	80.00%	35.99	71.80	65.40	59.70	55.10	63.00
+ RVQ	80.00%	25.14	3.95	2.31	1.30	1.13	2.17
+ CommVQ	82.40%	99.85	80.60	74.95	67.64	63.33	71.63
+ CQ	82.40%	97.71	79.11	68.07	58.55	51.20	64.23
+ VQKV	82.40%	**100.00**	**88.20**	**78.70**	**73.60**	**67.90**	**77.10**
+ VQKV + Calibration	82.40%	**100.00**	**89.03**	**82.04**	**78.64**	**72.65**	**80.59**

Full NIAH heatmaps across context lengths and needle positions are available in the results supplement.

D Compression-Ratio Sweeps

Figure 4 complements the main aggregate comparison in Table 1. The results supplement provides the complete 60%–95% sweeps, task-level scores, and eight KIVI configurations. Target groups use supported method-specific configurations under the compression definition in Section 3.4.



(a) Nearly perfect NIAH retrieval at 90%.

(b) Strongest RULER averages at 90% and 95%.

Figure 4: LLaMA3.1-8B NIAH and RULER compression sweeps. VQKV leads the evaluated NIAH and RULER comparisons at both 90% and 95% target reduction. Figure 1 jointly visualizes LongBench quality, compression, and generation time.

E Hybrid Compression and Memory

Combining VQKV with token selection. Combining VQKV with SnapKV yields 96.6% reduction and a 32.8 LongBench average on LLaMA3.1-8B, and 96.5% reduction and 31.4 on LLaMA3.2-3B. This hybrid removes tokens through selection; VQKV alone retains every token, as in the main experiments.

Persistent cache storage is

$$M_{\text{cache}} = M_{\text{codes}} + M_{\text{init}} + M_{\text{local}} + M_{\text{codebooks}} + M_{\text{projections}} + M_{\text{metadata}}. \tag{8}$$

The inference format uses 16-bit index containers: $(56 + 16) \cdot 16 = 1152$ bits per intermediate key/value pair for the main configuration, a 71.9% reduction in that storage component. The shared codebook cost is amortized over a batch, while token-dependent storage scales with batch size. Appendix L reports the current cache and peak-allocation measurements; the results supplement provides the historical memory breakdown.

F Additional Efficiency Experiments

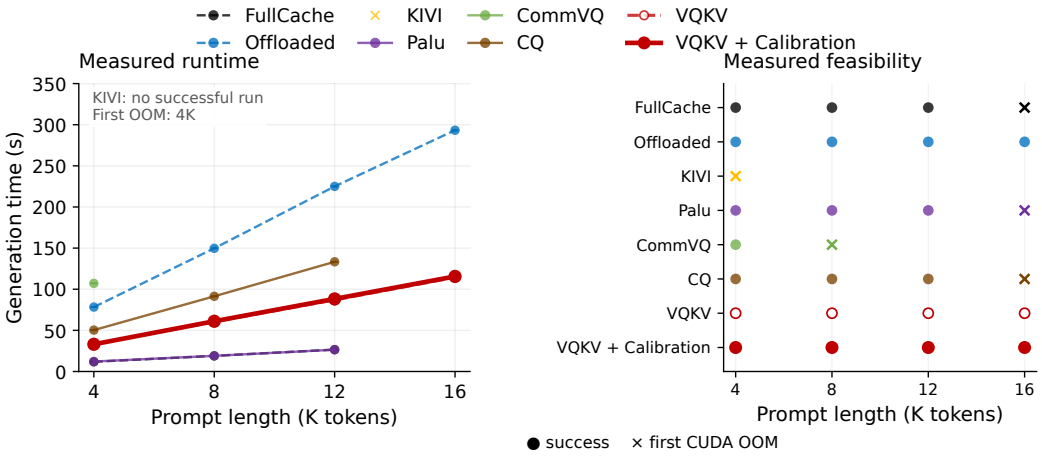


Figure 5: Generation time and feasibility on A100 (40 GB), batch 8, with 256 outputs and identical NarrativeQA prefixes. Left: mean prefill-plus-decode time with standard deviation. Right: successes and the first measured CUDA OOM per method. The 4K–16K window is shown; complete scans are in the results supplement.

We evaluate baseline implementations based on publicly released code, using one warmup and three timed repetitions on the same GPU model and inputs. Timings include synchronization and exclude model loading and validation. Both VQKV variants use the 82.8% configuration and windows 4/1024; KIVI uses 2-bit keys and values, group size 64, and residual length 128. Both VQKV variants complete 16K prompts, compared with 12K for FullCache. OOM marks the first CUDA allocation failure of each tested inference implementation, including its prefill workspace. Figure 5 displays the 4K–16K window. The accompanying protocol specifies implementation versions, logit policies, and correctness fixes; the results supplement provides complete scans and alternative implementation paths.

Table 9 evaluates LLaMA3.1-8B in FP16 on RTX 6000 Ada (48 GB). Each method uses identical seeded synthetic prompts and 1024 decode steps, with two timed runs after one warmup. Both VQKV variants use the final 82.8% configuration, initial/recent windows of 4/1024, and fused decoding. End-to-end time (E2E) includes prefill, decoding, and synchronization between stages; decode throughput counts tokens across all sequences.

Table 9: Generation latency on one RTX 6000 Ada (48 GB). Both VQKV variants use the final 82.8% configuration and 1024 decode steps. Means are computed from two timed runs after one warmup. E2E includes prefill and compression; throughput counts all sequences. FullCache fits both workloads.

Method	BS	Prompt	Prefill (s)	Decode (ms/step)	Throughput (tok/s)	E2E (s)
FullCache	8	16384	28.09	79.97	100.04	109.98
Offloaded	8	16384	26.20	1123.15	7.12	1176.31
VQKV	8	16384	59.19	154.93	51.64	217.84
VQKV + Calibration	8	16384	59.17	132.53	60.37	194.87
FullCache	16	8192	23.97	81.81	195.57	107.74
Offloaded	16	8192	22.91	1123.29	14.24	1173.17
VQKV	16	8192	50.89	187.44	85.36	242.83
VQKV + Calibration	16	8192	50.97	176.63	90.58	231.84

Tables 9 and 10 characterize the time cost and capacity benefit of compression. Both cache formats fit on the 48GB device, where calibrated VQKV takes $1.77\times$ and $2.15\times$ the resident full-cache time. On A100 40GB, the same batch/prompt shapes exceed FullCache capacity, while calibrated VQKV completes them in 270.75 and 289.31 seconds. The complete timings include all methods and both workloads. The protocol uses identical seeded synthetic prompts, prefill, and 1024 decode-forward steps; the first decode input repeats the final prompt token, as in the 48GB timing loop. The fixed-step workloads evaluate runtime and capacity, while the benchmark suites evaluate task quality.

Table 10: Memory-limited timing on A100 (40 GB). FullCache exhausts memory at both workloads. All runs use identical seeded prompts, FP16, and 1024 fixed decode-forward steps, with one warmup and two timed runs. Reported time includes prefill and synchronization. Appendix A specifies the timing setup.

Batch / prompt	Method	Prefill (s)	Decode (s)	Total (s)
8 / 16K	FullCache		OOM	
	Offloaded	13.02	1282.69	1295.70 ± 0.99
	VQKV	58.77	209.97	268.74 ± 0.06
	VQKV + Calibration	58.84	211.91	270.75 ± 0.06
16 / 8K	FullCache		OOM	
	Offloaded	11.52	1323.95	1335.47 ± 10.32
	VQKV	54.59	231.41	286.00 ± 0.20
	VQKV + Calibration	54.75	234.56	289.31 ± 0.04

Table 11 evaluates the updated kernel with calibration across batch sizes, using 8192 input tokens and 64 outputs, including 63 decode forwards. Table 12 separately compares fused and non-fused attention in the reference implementation for 50 outputs.

Table 11: Batch scaling on RTX 6000 Ada (48 GB): 8192 input tokens and 64 generated tokens, with three timed runs after one warmup. Full denotes FullCache; VQKV+Cal denotes VQKV + Calibration at 82.8%. Decode latency is measured per batch forward pass; generated-token throughput is $B \times 64/T_{\mathrm{E2E}}$.

BS	E2E (s)		Decode (ms)		Throughput (tok/s)	
	Full	VQKV+Cal	Full	VQKV+Cal	Full	VQKV+Cal
1	2.86	5.50	24.09	38.45	22.38	11.63
2	4.67	9.50	26.55	49.06	27.39	13.48
4	8.20	17.66	34.63	75.79	31.22	14.50
8	15.44	34.40	50.15	129.66	33.16	14.88
16	30.05	70.42	79.58	271.44	34.07	14.54

Table 12: Generation time and throughput of the reference implementation with separate KV-cache reconstruction (Non-Fused) and fused VQ attention (Fused) on a 48 GB GPU, for 50 generated tokens. These measurements predate the updated kernel. Speedup is Non-Fused time divided by Fused time.

Prompt	**Time (s)**		**Throughput (tok/s)**		**Speedup** ($\times$)
	Non-Fused	**Fused**	**Non-Fused**	**Fused**	
1K	8.76	1.83	5.71	27.34	4.79
2K	42.40	2.75	1.18	18.19	15.42
4K	62.49	5.13	0.80	9.74	12.18
8K	88.79	10.13	0.56	4.94	8.77
16K	114.39	20.38	0.44	2.45	5.61
32K	230.84	41.30	0.22	1.21	5.59

Generation capacity. Figure 1b measures the maximum generated-token count accommodated by a fixed GPU-memory budget, using a 65,536-token prompt, FP16, and FlashAttention on RTX 6000 Ada (48 GB). It measures storage capacity, with task quality evaluated separately through 32K. The pretrained model retains its original supported context window; VQKV provides cache compression rather than context extrapolation.

LongBench quality, compression, and efficiency. In Figure 1, circle area encodes measured generation time and center symbols identify methods. Efficiency measurements use RTX 6000 Ada GPUs, with identical input data and generation lengths for all methods. Each configuration uses three timed runs after one warmup. Timings include cache encoding and exclude model loading and warmup. VQKV uses optimized encoding and fused VQ attention. Baselines use evaluation implementations based on their publicly released code; the supplement records the implementation settings. Quality and latency use method-specific configurations; the aligned-window KIVI result in Table 7 is a separate setting. Latency is measured on a single shared long request rather than averaged across the benchmark. Target groups include supported configurations near the target, including 94% settings in the 95% group.

G Detailed Codebook Ablations

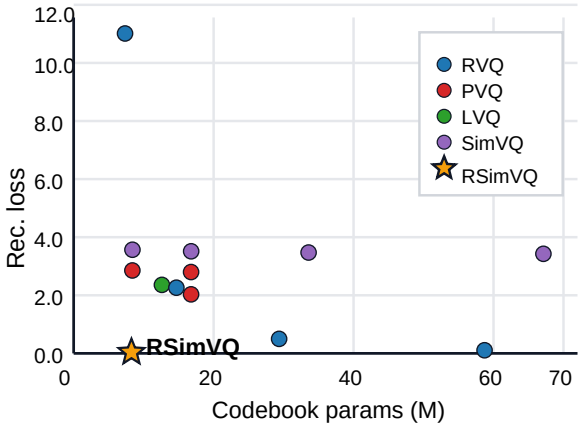


Figure 6: Post-RoPE key reconstruction error versus codebook parameters on LLaMA3.1-8B. The quantizer comparison motivates the RSimVQ representation used in VQKV.

Figure 6 shows the reconstruction–parameter trade-off that motivates residual SimVQ. Table 13 reports the codebook configurations and benchmark scores underlying Figure 3.

Table 13: Detailed results for different VQKV codebook combinations on LLaMA3.1-8B (Grattafiori et al., 2024). The **Ratio** column reports KV-cache reduction ratios.

N^k	S^k	N^v	S^v	Ratio	LongBench	NIAH	RULER				
							4K	8K	16K	32K	Avg.
56	1024	16	128	83.60%	32.43	99.86	90.58	86.28	80.58	73.35	82.70
56	*1024*	*16*	*512*	*82.80%*	*33.07*	*100.00*	*92.74*	*89.54*	*86.91*	*79.90*	*87.27*
56	1024	16	2048	82.00%	33.40	100.00	93.96	89.55	87.02	79.42	87.49
56	1024	16	8192	81.30%	33.47	100.00	94.08	91.70	90.09	85.45	90.33
56	1024	16	32768	80.50%	33.47	100.00	94.27	91.56	90.31	85.39	90.38
56	1024	8	512	84.60%	31.39	95.94	74.43	65.60	56.85	47.78	61.16
56	*1024*	*16*	*512*	*82.80%*	*33.07*	*100.00*	*92.74*	*89.54*	*86.91*	*79.90*	*87.27*
56	1024	40	512	77.50%	33.64	100.00	94.45	92.03	92.01	86.96	91.36
56	64	16	512	88.30%	32.55	100.00	87.72	82.03	76.28	69.62	78.91
56	256	16	512	85.50%	32.97	100.00	92.60	88.47	84.50	76.91	85.62
56	*1024*	*16*	*512*	*82.80%*	*33.07*	*100.00*	*92.74*	*89.54*	*86.91*	*79.90*	*87.27*
56	4096	16	512	80.10%	33.07	100.00	93.30	89.93	87.54	81.17	87.99
56	8192	16	512	78.70%	33.11	100.00	93.29	90.19	87.94	81.32	88.18
8	1024	16	512	94.50%	30.07	37.51	42.94	22.56	16.61	12.71	23.71
24	1024	16	512	90.60%	32.33	99.86	84.48	76.49	71.34	60.75	73.27
40	1024	16	512	86.70%	32.85	100.00	92.53	87.75	83.91	74.55	84.68
56	*1024*	*16*	*512*	*82.80%*	*33.07*	*100.00*	*92.74*	*89.54*	*86.91*	*79.90*	*87.27*
72	1024	16	512	78.90%	33.19	100.00	93.43	90.18	87.48	81.22	88.08

H Calibration Protocol

Calibration keeps the model, codebooks, and compression procedure fixed. LLaMA3.1 uses 65,536 affine parameters (128 KiB in FP16), with no additional per-token codes. Cross-model configurations and complete task results are provided in the results supplement.

Fitting the correction. Each of four OpenWebText documents, approximately 32K tokens long, is augmented with 12 identifier/code records and two four-code retrieval prompts. The resulting eight training prompts contain no benchmark text or answers; a fifth document provides two diagnostic prompts. Scales and biases are bounded as $a = 1 + 0.5\tanh(s)$ and $b = 0.25\rho\tanh(u)$, where ρ is the initial value RMS. We initialize the correction to the identity and run 200 Adam updates at learning rate 0.003. The final checkpoint is exported to FP16 without benchmark selection. Each forward uses 16 causal teacher positions and 512 sampled compressed positions per layer for V-MSE. Both losses are normalized by the corresponding uncalibrated error. The MSE-only control uses the same data, bounds, and update budget. Initial/recent windows 4/1,024 remain FP16.

Paired evaluation. Primary comparisons use all 13 RULER tasks with equal weights and the original scorers, with 20 examples per task at 4K and 100 at 32K. Cross-model synthetic inputs use each model's tokenizer and reserve generation space without cropping. The two Qwen2.5 sizes share an input collection, as do the two LLaMA3.1 codecs. The original-input replay and new-instance evaluations use different input collections; paired gains are computed within each collection.

I Calibration Gains and Quality Preservation

Section 4.4 reports the primary paired gains, intervals, and LongBench preservation criteria. Here we provide task-level results and evaluations on additional inputs.

In the complete results supplement, Base denotes the original implementation, and Matched denotes VQKV with the same value-tile reconstruction arithmetic as calibration. MSE and Output denote reconstruction-only and output-aware calibration. The original Base averages are 92.83/76.29/33.01 for RULER 4K/32K/LongBench; the corresponding Matched averages are 92.83/76.35/33.02. All full-cache scores come from the same paired study.

Table 14: Task-level calibration results from the same 32K evaluation as Table 3: all 13 tasks with 100 examples per task. Δ is VQKV + Calibration minus VQKV; both use common reconstruction arithmetic.

Task	Full	VQKV	VQKV + MSE	VQKV + Calibration	Δ
cwe	61.20	12.50	13.00	**13.60**	+1.10
fwe	76.67	74.67	72.33	**74.67**	+0.00
niah_multikey_1	99.00	95.00	96.00	**97.00**	+2.00
niah_multikey_2	99.00	91.00	93.00	**93.00**	+2.00
niah_multikey_3	100.00	72.00	72.00	**72.00**	+0.00
niah_multiquery	98.50	68.50	88.25	**95.50**	+27.00
niah_multivalue	98.75	89.25	93.50	**96.00**	+6.75
niah_single_1	100.00	100.00	100.00	**100.00**	+0.00
niah_single_2	100.00	99.00	99.00	**99.00**	+0.00
niah_single_3	100.00	99.00	100.00	**100.00**	+1.00
qa_hotpotqa	38.00	36.00	37.00	**36.00**	+0.00
qa_squad	70.00	62.00	61.00	**62.00**	+0.00
vt	98.20	93.60	95.00	**95.60**	+2.00
Average	**87.64**	**76.35**	**78.47**	**79.57**	**+3.22**

Multi-query retrieval increases from 68.50 to 95.50, compared with 98.50 for FullCache, closing 90% of the gap. Multi-value retrieval gains 6.75 points. These tasks match the multi-record structure of the fitting prompts and account for most of the aggregate recovery; CWE scores 13.60 versus 61.20 for FullCache. On LongBench, GovReport gains 1.18 points, while passage retrieval and Qasper decrease by 0.92 and 0.51. The supplement provides all 16 LongBench task scores.

The original-input replay covers 5,200 RULER and 660 NIAH inputs, plus all 3,750 LongBench examples. Across 4K–32K, Matched/Output RULER scores are 92.98/93.24, 89.63/90.69, 86.89/88.26, and 79.68/82.94; both score 100.00 on NIAH. This coverage replay includes development inputs; the primary paired calibration results use the separately specified new-instance collection.

LLaMA3.2 original-input coverage. We evaluate the 82.4% configuration on 5,200 RULER and 660 NIAH inputs using the fixed calibration checkpoint. Full/Base/Output RULER scores are 90.03/87.65/89.03, 86.04/78.77/82.04, 83.30/74.13/78.64, and 78.27/67.99/72.65 from 4K to 32K; the four-length averages are 84.41/77.14/80.59. NIAH scores are 100.00/100.00/100.00. Output denotes VQKV with calibration in Table 8. These paired controls use a different input collection from the $+4.76$-point setting in Table 5, and results are reported separately for each collection.

Separate fixed transfer studies yield mean 32K gains of 2.59 points for ordinary RVQ and 4.68 for RSimVQ at 90% reduction. The supplement reports all task results and uncertainty for these transfer studies, including CommVQ and the earlier RVQ configurations.

J Stronger Reconstruction Control

The analytic-MSE control uses the same codebooks, training prompts, bounded affine family, and sampled reconstruction objective. Per-channel weighted moments define a convex quadratic. Evaluating the unconstrained stationary point, box edges, and corners yields the continuous optimum before FP16 export. This provides an exact control for the sampled reconstruction objective before FP16 rounding. Table 4 reports both codecs and all four prespecified contrasts. Differences are computed from paired per-instance scores with equal task weights. Intervals use 5,000 task-stratified bootstrap resamples with seed 2026092424; Bonferroni correction covers both comparisons for both codecs. These paired objective comparisons use a reused collection.

For fixed attention α, write $E = \widehat{V} - V$. Then $\|\alpha^\top E\|^2 = \mathrm{tr}(E^\top \alpha\alpha^\top E)$, whereas reconstruction loss uses $\mathrm{tr}(E^\top E)$. The attention-weighted quadratic depends on error direction as well as magnitude; a smaller unweighted error need not yield a smaller readout error. The output-sensitive objective accounts for this directional weighting. In this experiment, MSE is measured on held-out

OpenWebText and downstream quality on RULER, so the observed gain includes transfer from the fitting distribution. The supplement provides an extended derivation.

K Scope Across Models and Compressors

Table 5 summarizes the six 32K settings. Each setting evaluates 1,560 instances across 4K/32K with a frozen checkpoint. Table 15 reports all primary 32K contrasts, including both Qwen2.5 configurations. Full 4K and per-task results are in the supplement.

Table 15: All twelve primary 32K comparisons across six configurations. Intervals are computed from 5,000 paired within-task bootstrap resamples with seed 2026092224. Tasks and configurations retain their original weights; unconfirmed and negative results remain in the comparison family.

Configuration	Contrast	Change	95% interval	Bonferroni-12
LLaMA3.1/RSimVQ	Output–Base	+3.40	[2.42, 4.45]	[1.91, 5.04]
LLaMA3.1/RSimVQ	Output–MSE	+1.16	[0.27, 2.09]	[-0.13, 2.53]
LLaMA3.1/RVQ	Output–Base	+2.26	[1.24, 3.28]	[0.71, 3.81]
LLaMA3.1/RVQ	Output–MSE	+4.07	[2.91, 5.32]	[2.39, 5.96]
LLaMA3.2-3B	Output–Base	+4.76	[3.59, 6.03]	[3.12, 6.55]
LLaMA3.2-3B	Output–MSE	+4.33	[3.17, 5.52]	[2.64, 6.07]
Qwen2.5-1.5B	Output–Base	-0.69	[-1.37, -0.01]	[-1.71, 0.29]
Qwen2.5-1.5B	Output–MSE	-0.24	[-1.06, 0.54]	[-1.46, 0.90]
Qwen2.5-3B	Output–Base	+0.58	[-0.19, 1.34]	[-0.53, 1.74]
Qwen2.5-3B	Output–MSE	+0.71	[-0.21, 1.63]	[-0.66, 2.03]
Qwen3-8B	Output–Base	+2.84	[1.94, 3.79]	[1.48, 4.39]
Qwen3-8B	Output–MSE	+1.97	[0.82, 3.12]	[0.19, 3.62]

Cross-model LongBench. Each model uses its frozen checkpoint for all 3,750 examples across 16 tasks. Preservation depends on both model and control (Table 16). LLaMA3.2 meets the criteria against Base but does not meet the per-task tolerance against MSE; Qwen2.5-3B meets the criteria against both controls; Qwen2.5-1.5B and Qwen3 do not meet one or more conditions. RULER recovery and LongBench preservation are evaluated as separate outcomes.

Table 16: Output calibration minus each LongBench control. Individual 95% intervals are computed from 5,000 paired within-task resamples (seed 2026092014). M/I/T indicates whether the prespecified criteria are met: mean change ≥ -0.30, interval lower bound ≥ -0.30, and every task change ≥ -2.00, respectively. All three conditions must be met for tolerance-based preservation.

Model	Control	Change	95% interval	Worst task	M/I/T
Qwen2.5-1.5B	Base	-0.22	[-0.36, -0.09]	-1.11	Y/N/Y
Qwen2.5-1.5B	MSE	-0.02	[-0.20, 0.16]	-1.11	Y/Y/Y
Qwen2.5-3B	Base	-0.03	[-0.20, 0.12]	-1.31	Y/Y/Y
Qwen2.5-3B	MSE	+0.04	[-0.18, 0.25]	-0.50	Y/Y/Y
Qwen3-8B	Base	+0.19	[-0.05, 0.43]	-2.75	Y/Y/N
Qwen3-8B	MSE	+0.08	[-0.29, 0.44]	-3.19	Y/Y/N
LLaMA3.2-3B	Base	-0.04	[-0.22, 0.14]	-1.95	Y/Y/Y
LLaMA3.2-3B	MSE	+0.11	[-0.11, 0.33]	-2.09	Y/Y/N

Calibration with other compressed representations. At 32K, the adjusted confidence intervals for the gains lie entirely above zero for the fixed KIVI and xKV configurations, but not for CQ. The supplementary 4K Output–MSE contrast is also negative for CQ. These studies reuse the RULER collection and evaluate RULER only. All configurations and outcomes are in the supplement.

OptR optimizes key/value rotations before INT2 quantization; VQKV calibrates decoded values after fixing its representation. On the same Qwen3-8B examples, Output and OptR score 73.33 and 88.53 at reported physical budgets of 800 and 576 bits, respectively. The two methods operate on different representations and use different fitting procedures. VQKV fits the value readout while

retaining the trained codebooks, encoder, and cache format; it adds only static parameters and fused value arithmetic. Tables 3 and 4 isolate its quality benefit within that representation, and Table 17 measures its runtime cost.

L Updated Kernel and Calibration Cost

Table 17 uses the 82.8% LLaMA3.1 configuration, FP16, and windows 4/1024. With reconstruction arithmetic matched, calibration changes mean time from 26.79 to 26.82 seconds (0.12%) and adds 128 KiB of static parameters. Timings include encoding, prefill, decoding, and synchronization, excluding warmup and model loading.

Table 17: Generation time and peak allocation on RTX 6000 Ada (48 GB): 31,500 input tokens, batch size 1, 256 outputs, and identical 82.8% VQ codebooks. The matched row uses the calibration branch's reconstruction arithmetic without affine correction. Means and sample standard deviations are computed from three runs after one warmup. Peak memory includes model weights.

Implementation	Prefill (s)	Decode (s)	E2E (s)	Peak (GiB)
Full cache	7.07	8.52	15.59 ± 0.08	22.31
VQKV	13.99	13.66	27.65 ± 0.19	19.70
VQKV (matched)	14.21	12.57	26.79 ± 0.11	19.70
VQKV + Calibration	14.20	12.62	26.82 ± 0.11	19.70

Tables 9 and 11 report offloading and batch scaling. These VQKV variants use different reconstruction arithmetic; Table 17 provides the matched comparison for calibration cost.

Table 6 counts allocated cache buffers, shared codebooks, and calibration parameters. Peak allocations include the cache, model weights, and temporary computation tensors. VQKV reduces the cache component; these measurements use the existing prefill workspace. The results supplement provides all measurements and OOM outcomes.