μ-ZERO: ZERO-ANCHORED LOG COMPANDING FOR KV CACHE QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context LLM inference is increasingly limited by KV cache memory traffic, making low-bit cache quantization essential for efficient deployment. Most calibration-free methods use per-group affine min–max quantization, uniformly allocating levels across the observed range. We show that this rule is poorly aligned with attention: after standard key/value grouping, online cache groups exhibit dense zero-centered mass and sparse, approximately log-tailed extremes, causing uniform grids to waste codes in low-density regions. Moreover, query–key logits are often residuals of cancelling signed products, so small perturbations near zero can induce large relative errors or sign flips despite low entrywise reconstruction error. These observations motivate protecting the zero neighborhood rather than only the range endpoints. We propose μ-ZERO, a zero-anchored piecewise log-companded quantizer that applies a monotone μ-law transform to the positive and negative sides around an exact zero level. μ-ZERO concentrates resolution near zero, preserves asymmetric side extents, and uses coarser logarithmic spacing in the tails. A lightweight prefill-time rule selects the compander strength per layer and cache type from scale-free statistics and reuses it during decoding. Across reasoning, code generation, and long-context understanding, μ-ZERO achieves the best quantized results across all evaluated 2–4-bit settings against representative scalar, transform/sketch-domain, and mixed-precision baselines. Further evaluations support long-output reasoning and applicability to models up to 70B and selected MQA/MLA architectures. System-level evaluations show up to $2.38\times$ higher throughput and 69% lower peak memory usage than BF16, while extending feasible batch size by at least $3.3\times$ and context length by $4.7\times$. Code available at: https://github.com/muZeroQuant/muZero.

1 INTRODUCTION

Long-context large language models are increasingly limited by the memory traffic of autoregressive decoding (Pope et al., 2023; Shazeer, 2019; Xiao et al., 2024; Miao et al., 2025; Lee et al., 2024; Wang et al., 2024; Li et al., 2024a). At each decoding step, every layer reuses the accumulated key and value (KV) cache from all previous tokens (Ainslie et al., 2023; Liu et al., 2024a; Bhaskar et al., 2025). This cache is essential for efficient inference, yet its size grows linearly with context length and batch size, quickly becoming a dominant memory and bandwidth cost (Kwon et al., 2023; Sheng et al., 2023; Feng et al., 2024; Chen et al., 2024). Compressing the KV cache is therefore one of the most direct routes to longer contexts, larger batches, and lower serving cost (Zhang et al., 2023; Liu et al., 2023; Xiao et al., 2023b; Li et al., 2024b; Tang et al., 2024).

KV cache quantization is attractive because it can reduce this cost without changing model weights. The most deployable methods are calibration-free scalar quantizers: they partition cache entries into small groups, store low-bit integer codes, and keep a small amount of per-group range metadata (Liu et al., 2024b; Li et al., 2025b; Su et al., 2025a; Sheng et al., 2023). This interface is simple, online, and compatible with standard attention kernels, which makes it useful not only as a standalone compressor but also as a primitive inside broader cache-management systems (Kwon et al., 2023; Lin et al., 2025; Liu et al., 2025; Chen et al., 2025). However, the local quantization rule in many such systems remains an affine min–max quantizer. It preserves the endpoints of each group, but allocates integer levels uniformly across the range. It falls short in meeting what attention needs.

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.

Precision misallocation is a classical quantization concern. For weights, offline optimization can effectively expose nonuniform level allocation (Frantar et al., 2022; Lin et al., 2024; Dettmers et al., 2022; Nagel et al., 2020; Li et al., 2021); KV cache methods pursue related goals with outlier handling, transforms or sketches, and learned or tuned codebooks (Liu et al., 2024b; Hooper et al., 2024; Su et al., 2025a; Li et al., 2025b; Zandieh et al., 2025; Su et al., 2025b; Li et al., 2025a). However, KV entries are generated dynamically *online*, vary with the prompt, and are consumed immediately by attention, so their local statistics cannot be fixed in advance. Complex remedies therefore often compensate with adaptation or exceptions, such as outlier selection and protection (Zandieh et al., 2024; Hooper et al., 2024), per-layer or per-token tuning (Li et al., 2025b; Chen et al., 2025), transforms/sketches (Zandieh et al., 2024; 2025; Su et al., 2025b), or learned codebooks (Li et al., 2025a), which can introduce offline calibration/training/storage costs or online selection, metadata, and kernel overhead. These mechanisms still do not directly fit the nonuniform statistics of each newly generated cache group during streaming decode. We therefore study the mismatch that remains after common key/value grouping layouts: most entries concentrate near zero, while the tails are sparse and approximately log-linear after normalization; uniform bins thus spend many levels on low-density regions and leave too few levels where most entries lie. We quantify this mismatch with scale-free group statistics and connect it to attention sensitivity: query-key logits are signed sums, and when positive and negative contributions nearly cancel, small perturbations in near-zero key components can cause large relative logit errors or sign flips. Thus the sensitive region is not merely the extrema of the observed range, but the dense zero neighborhood that governs cancellation.

We propose μ-ZERO, a novel zero-centered log-companded quantizer for KV caches. It keeps the interface of drop-in scalar quantization: packed integer codes, per-group minimum and scale, no offline calibration or learned codebooks, and fused QK and AV kernels. μ-ZERO applies a monotone μ-law compander to each side of zero, quantizes uniformly in the transformed domain, and reconstructs through the inverse map. This provides fine resolution near zero and progressively coarser spacing toward the tails. A zero-centered code split ensures that the highest code density is anchored at the real value zero, while preserving asymmetric side extents. Finally, a lightweight promotion rule selects the compander strength per layer and cache type from scale-free prefill statistics, then reuses the selected value throughout decoding.

This design occupies a useful point in the KV-compression landscape. Unlike transform- or sketch-domain attention methods, μ-ZERO does not change the attention algebra. Unlike learned codebooks or tuning-based systems, it does not require a calibration set, training procedure, or model-specific search. Because its code space is defined by $L = 2^b - 1$, the same rule supports any integer bit budget $b \geq 1$, including both hardware-friendly widths such as 2, 4, and 8 bits and intermediate budgets such as 3 bits. This flexibility is important for cache-management policies that assign different memory budgets across layers, tokens, requests, or deployment targets.

Our contributions are as follows. First, we introduce KV-cache-specific metrics that expose precision misallocation in online cache groups under uniform affine quantization. Second, we connect this distributional mismatch to attention sensitivity through cancellation in query-key logits. Third, we introduce μ-ZERO, a zero-centered adaptive log-companded quantizer that preserves the scalar KV-cache interface while supporting arbitrary integer bit widths. Fourth, we derive an attention-weighted α objective, showing why reconstruction NMSE is a poor selection criterion and motivating lightweight layer-wise prefill-time α selection. Finally, μ-ZERO achieves the best quantized results across all evaluated 2–4-bit settings against representative scalar, transform/sketch-domain, and mixed-precision baselines on reasoning, code generation, and long-context tasks. Extended evaluations cover long-output reasoning, models up to 70B, and selected MQA/MLA architectures (Appendix E). System-level evaluations further show up to $2.38\times$ higher throughput and 69% lower peak reserved memory than BF16, while supporting at least $3.3\times$ larger batches and $4.7\times$ longer contexts.

2 RELATED WORK

KV cache in autoregressive decoding. Autoregressive LLMs cache the key and value vectors of preceding tokens for reuse by subsequent queries (Vaswani et al., 2017). The cache avoids recomputation but grows linearly with context length, making long-context decoding increasingly constrained by memory capacity and bandwidth (Pope et al., 2023; Kwon et al., 2023). KV cache quantization reduces both storage and decode-time traffic, but unlike weight quantization, it must

operate online on prompt-dependent tensors whose errors directly perturb query–key logits and value aggregation (Liu et al., 2024b; Hooper et al., 2024; Frantar et al., 2022; Xiao et al., 2023a). This motivates lightweight, calibration-free quantizers that preserve the standard attention interface.

Drop-in scalar KV cache quantization. A common deployment-friendly approach leaves attention unchanged and stores historical keys and values as low-bit scalar codes. KIVI (Liu et al., 2024b), for example, quantizes keys channel-wise and values token-wise; subsequent methods protect outlier channels or tokens, retain short full-precision residual windows, or introduce lightweight selection rules (Hooper et al., 2024; Su et al., 2025a; Li et al., 2025b). Despite these extensions, the underlying local primitive is often per-group affine min–max quantization, which preserves range endpoints but allocates levels uniformly. As analyzed in Section 3, this allocation is poorly matched to the within-group cache distribution. μ-ZERO retains the same scalar interface while replacing uniform spacing with a closed-form, zero-centered nonuniform allocation.

Transform- and sketch-domain attention. Another line of work changes the representation in which attention is computed. QJL-style methods (Zandieh et al., 2024) approximate query–key inner products with randomized projections or Johnson–Lindenstrauss sketches, while TurboQuant (Zandieh et al., 2025) combines normalization, rotations, codebook quantization, and sketch-based residual compensation. Related methods apply Hadamard or rotation transforms, sometimes with channel reordering or query-side transformations, to regularize cache distributions (Son et al., 2025; Su et al., 2025b). These approaches can improve low-bit robustness, but require additional transformations, metadata, calibration or reordering, and specialized execution paths. In contrast, μ-ZERO preserves the attention algebra and applies a parametric scalar compander that can be fused into QK/AV kernels.

Learned codebooks and compressors. Learned KV compression replaces scalar rounding with vector quantization, additive codebooks, or learned encoder–decoder representations (Li et al., 2025a; Ge et al., 2024; Guan et al., 2022). Such models can exploit cross-channel correlations at very low bit rates, but introduce training or calibration requirements, codebook storage, and additional constraints from RoPE and cache layout. They are therefore less transparent and less immediately compatible with existing scalar-quantized cache systems.

System-level and tuning-based designs. System-oriented work allocates bit widths across layers (Li et al., 2025b; Yang et al., 2024; Hooper et al., 2024; Hariri et al., 2025), protects recent or sink tokens (Su et al., 2025b; Yue et al., 2024), progressively quantizes long traces (Liu et al., 2025; Zhang et al., 2025b;a), or jointly optimize compression with cache management and execution (Liu et al., 2025; Chen et al., 2025; Chen & Yang, 2025; Wang et al., 2025). These policies are largely orthogonal to the local quantization primitive, although many require offline calibration, model-specific search, or additional runtime policies. Because uniform affine quantization remains a common underlying representation, improving this primitive can benefit both standalone quantization and broader cache-management systems. μ-ZERO targets this layer: it requires no offline calibration, learned codebook, or model-specific search while preserving standard key-channel/value-token grouping.

3 ANALYSIS: MEASURING PRECISION ALLOCATION IN KV CACHE QUANTIZATION

Affine min–max quantization with key-channel/value-token grouping (Liu et al., 2024b) is a strong online baseline for KV caches: it is local, calibration-free, and often effective despite using uniformly spaced quantization levels within each group. Its remaining low-bit quality degradation suggests that preserving group endpoints is not enough to decide where the limited codes should be spent. Existing approaches address this with outlier handling (Zandieh et al., 2024; Su et al., 2025a; Hooper et al., 2024) or learned/tuned allocation (Li et al., 2025b; Chen et al., 2025; Liu et al., 2025), which typically requires prior calibration or online adaptation for prompt-dependent cache entries. We therefore revisit the grouped affine cache itself, asking what structure remains within each group and which quantization errors are most amplified by attention. We study these questions in two steps: first through within-group cache statistics, and then through their effect on the attention operator.

3.1 WITHIN-GROUP DENSITY IS NON-UNIFORM AND LOG-TAILED

We analyze the cache at a standard KV quantization granularity: channel-wise groups for keys and token-wise groups for values, with group size $G{=}64$. This controls for a widely supported grouping

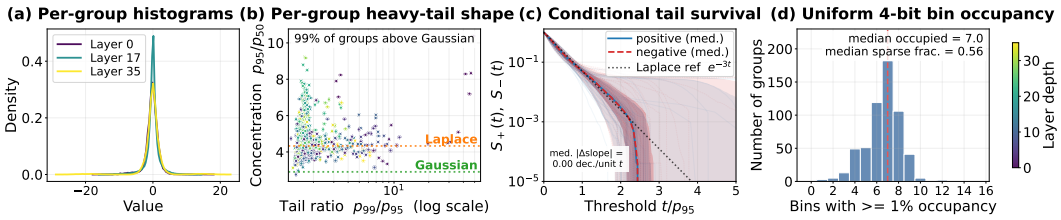

Figure 1: **Per-group distribution of Key cache in Qwen3-8B.** Groups are concentrated near zero, heavy-tailed, approximately log-linear in conditional tail survival, and poorly covered by uniform 4-bit bins. Appendix A gives observations on value cache, and on Llama-3.1-8B-Instruct.

choice and asks what structure remains inside each group. Figure 1 shows the key cache of Qwen3-8B on a held-out LongBench slice; Appendix A reports the value cache and a Llama-3.1-8B-Instruct replication with the same qualitative signatures.

Mass is concentrated near zero. Figure 1(a) shows representative groups from shallow, middle, and deep layers. Each has a sharp mode near zero and low-density tails. Uniform min–max quantization is therefore a limited fit to the empirical density: it spreads levels over $[\min, \max]$ although samples occupy a much smaller central region. Figure 1(d) quantifies this by counting how many of the 16 uniform 4-bit bins contain at least 1% of a group's samples. The median group has only 7 such bins; equivalently, the median fraction of bins below 1% occupancy is 0.56. Thus many intervals receive almost no samples, while high-density samples near the mode share too few levels.

Tail shape is heavy and structured across depth. Figure 1(b) summarizes each group by its tail ratio $r_{\text{tail}} = p_{99}(|X|)/p_{95}(|X|)$ and concentration ratio $r_{\text{conc}} = p_{95}(|X|)/p_{50}(|X|)$. In Qwen3-8B, 99% of groups lie above the half-normal concentration reference (r_{conc}=2.91), many exceed the half-Laplace reference (4.33), and a long right tail reaches r_{tail}>10, with several groups beyond 30. The broad spread shows that groups differ in shape, not only scale. Layer coloring indicates structured depth variation (especially for V cache), so a fixed nonuniform allocation would still be mismatched to many groups; the quantizer shape should adapt to group statistics.

To identify the tail form more directly, we compare conditional survival functions, i.e., the fraction of same-sign entries whose magnitude remains above a threshold:

$$S^+(t) = \Pr(x > t \mid x > 0), \qquad S^-(t) = \Pr(-x > t \mid x < 0). \tag{1}$$

In Figure 1(c), thresholds are normalized by each group's $p_{95}(|x|)$ before aggregation, and the vertical axis is logarithmic survival probability. The median positive and negative curves are nearly linear over the high-mass tail and track the Laplace reference e^{-3t} up to $t/p_{95} \approx 1$. This log-linear behavior suggests that equal value increments correspond to multiplicative changes in tail probability, motivating logarithmic companding: fine resolution near the dense center, coarser resolution in the low-density tail, and a coarsening rate adapted to group statistics.

Tail shapes are shared, but side extents are not. After p_{95} normalization, the positive and negative median survival curves in Figure 1(c) nearly overlap. Fitting log-survival slopes over $t/p_{95} \in [1, 3]$ gives a difference that rounds to 0.00 decades per unit threshold in the plotted statistic, suggesting that the two signs do not require unrelated tail-shape functions. The remaining question is where the highest resolution should be put given this shared shape. The next subsection examines it at the attention operator, where sign preservation and cancellation make zero especially sensitive.

3.2 Why precision–density mismatch damages attention at the operator level

We next ask how the distributional mismatch propagates through attention, where the effect of a cache perturbation depends on the operator's signed interactions rather than entrywise reconstruction alone.

Cancellation makes small perturbations consequential. For a query q and cached key k, the pre-softmax logit is

$$s = q^\top k = \sum_{j=1}^{d} q_j k_j. \tag{2}$$

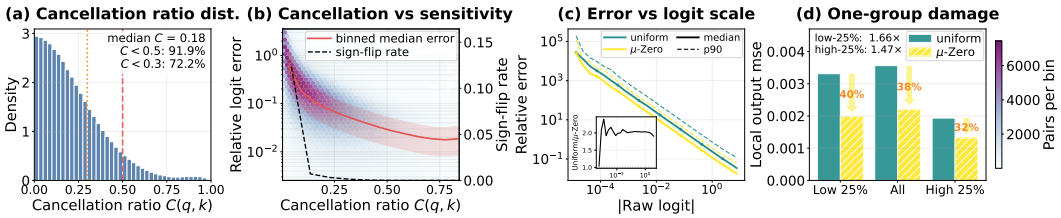

Figure 2: **Attention-level sensitivity to quantization error.** Cancellation is common in QK logits, predicts the relative error and sign flips of uniform key quantization, and exposes the benefit of μ-Zero in relative logit error and one-group output damage. Details are in Appendix B.

If k is quantized with component-wise error δ_j, the logit error is $\Delta s = \sum_j q_j \delta_j$. Many small errors can therefore accumulate, especially when the original dot product is produced by cancellation among positive and negative terms. We quantify this using

$$C(q,k) = \frac{\left|\sum_j q_j k_j\right|}{\sum_j |q_j k_j| + \varepsilon} \in [0,1], \tag{3}$$

where $C \approx 0$ indicates strong cancellation and $C \approx 1$ sign-aligned terms. Figure 2(a) shows that cancellation is typical in Qwen3-8B: the median score has $C = 0.18$, with 91.9% of scores below 0.5 and 72.2% below 0.3. Attention often operates on logits that are small residuals of larger opposing contributions. Appendix B shows that Qwen3 has more low-C mass than Llama, helping explain its larger low-bit degradation in Table 2.

Cancellation predicts sensitivity to uniform quantization. Figure 2(b) evaluates per-group uniform 4-bit quantization of keys on the same (q, k) pairs. Relative logit error rises sharply as C decreases: large-C scores with weak cancellation are stable, whereas small-C scores with strong cancellation can incur order-of-magnitude larger relative error. Sign flips follow the same pattern, concentrating in the low-C regime and vanishing as C increases. Thus allocating too few levels to dense small-magnitude values is harmful because they control the sign and cancellation pattern of the dot product.

Relative accuracy is the relevant target. The softmax is sensitive to relative logit changes, especially when top candidates have small margins. A useful attention quantizer should therefore reduce relative logit error across logit scales, rather than only minimizing entry-wise reconstruction error. Figure 2(c) compares uniform quantization with μ-Zero. Across scales, μ-Zero lowers both median and high-percentile relative error, with the largest absolute errors at small raw logits where relative perturbations are hardest to control. This supports applying nonuniform allocation to KV caches: fine spacing near the dense region and coarser spacing in the tails.

Local interventions confirm that the error channel matters. Figure 2(d) tests whether this logit-level mechanism causes output damage at the quantization granularity. We quantize one cache group at a time and keep the rest in full precision, isolating that group's local error. Relative to uniform quantization, μ-Zero reduces local output MSE over all sampled groups and within both the bottom and top quartiles of near-zero fraction. The gain is therefore not confined to one distributional slice.

Synthesis. Together, the data and operator analyses point to a specific principle: a KV quantizer should protect the zero neighborhood controlling cancellation in attention logits, not just endpoints. When QK scores are small remainders of signed products, errors on dense small-magnitude entries can change relative logits or flip signs. This motivates three design constraints: (i) a *nonuniform* grid with fine spacing near zero and coarser spacing in log-like tails; (ii) a highest-resolution point *anchored at real zero*, with an exact zero code and separate positive/negative side budgets; and (iii) a shape that *adapts efficiently* across layers and cache types, sharing one sign-wise compander while selecting its strength from scale-free statistics.

4 μ-ZERO QUANTIZATION

μ-ZERO instantiates the principles from Section 3 within the standard scalar KV-quantization interface. It keeps the packed codes and per-group affine metadata of uniform min–max quantization, but

interprets each group through a zero-anchored piecewise log compander, which has a single shape parameter selected without offline calibration but on the prefill statistics per layer and cache type.

4.1 COMPANDING AS DENSITY-MATCHED QUANTIZATION

For a group $x \in \mathbb{R}^G$ quantized to b bits, let $L = 2^b - 1$, $m = \min_i x_i$, $M = \max_i x_i$, and $s = \max(M - m, \epsilon)$. We first normalize

$$y_i = \frac{x_i - m}{s} \in [0, 1], \tag{4}$$

which removes the group scale using the standard per-group affine metadata: the stored pair (m, s), equivalently a minimum and scale. Section 3.1 shows that the resulting range-normalized groups remain highly nonuniform: samples concentrate around the real value zero, and the side-conditioned tails decay approximately linearly on a log-survival plot. This suggests a *companding* construction: quantize uniformly after a monotone change of coordinate Φ, so equal-width bins in $\Phi(y)$ become nonuniform intervals in the original value domain, then reconstruct with Φ^{-1}. Appendix C.1 gives the corresponding high-rate expression for the local reconstruction error induced by a compander.

μ-ZERO instantiates this idea with a one-parameter μ-law compander on each side of zero:

$$\Phi_\alpha(u) = \frac{\log(1 + \alpha u)}{\log(1 + \alpha)}, \qquad \Phi_\alpha^{-1}(v) = \frac{\exp(v \log(1 + \alpha)) - 1}{\alpha}, \tag{5}$$

where $u, v \in [0, 1]$. The logarithm gives dense bins near $u = 0$ and progressively wider bins toward the tail, matching the near-zero concentration and log-shaped decay observed in Section 3.1. The parameter α controls this concentration: $\alpha \to 0$ recovers a nearly uniform grid, while larger α increases the ratio between near-zero and tail resolution.

4.2 ZERO-CENTERED CODE SPLIT

A direct compander on $y \in [0, 1]$ would place its finest resolution near the group minimum. This is not the sensitive point identified in Section 3.2. We instead anchor the compander at the real value $x = 0$. Let

$$z = \mathrm{clip}\left(\frac{-m}{s}, 0, 1\right), \qquad \ell_0 = \mathrm{round}(Lz), \tag{6}$$

be the normalized zero position and its integer level. Both are deterministic functions of the stored (m, s) and bit width, so zero anchoring adds no per-group metadata. The negative and positive sides receive

$$L_- = \max(\ell_0, 1), \qquad L_+ = \max(L - \ell_0, 1) \tag{7}$$

levels, preserving the asymmetric side extents of min–max quantization. With $e_- = \max(z, \epsilon)$ and $e_+ = \max(1 - z, \epsilon)$, the side-normalized magnitudes are

$$u_i^- = \frac{z - y_i}{e_-} \quad (y_i < z), \qquad u_i^+ = \frac{y_i - z}{e_+} \quad (y_i \geq z). \tag{8}$$

Thus both signs share the same compander shape, consistent with the shared normalized tail shape observed in Figure 1(c), but keep different side extents.

4.3 QUANTIZATION RULE

The integer code is obtained by companding and rounding the appropriate side:

$$q_i = \begin{cases} \ell_0 + \mathrm{round}\big(L_+ \Phi_\alpha(u_i^+)\big), & y_i \geq z, \\ \ell_0 - \mathrm{round}\big(L_- \Phi_\alpha(u_i^-)\big), & y_i < z, \end{cases} \tag{9}$$

then clipping to $[0, L]$ and packing into bytes. Since $L = 2^b - 1$, the same rounding rule applies to any bit width b, with no learned codebook or extra per-group metadata. Reconstruction applies the inverse:

$$\hat{y}_i = \begin{cases} z + e_+ \Phi_\alpha^{-1}((q_i - \ell_0)/L_+), & q_i \geq \ell_0, \\ z - e_- \Phi_\alpha^{-1}((\ell_0 - q_i)/L_-), & q_i < \ell_0, \end{cases} \qquad \hat{x}_i = m + s\hat{y}_i. \tag{10}$$

The code ℓ_0 reconstructs $x=0$ from the stored (m, s). More importantly, it is also the point of highest code density: for the positive side, the near-zero step size scales as

$$\Delta_+(0) \approx \frac{e_+ \log(1+\alpha)}{L_+ \alpha}, \tag{11}$$

so the zero-neighborhood gains a factor $\alpha/\log(1+\alpha)$ in resolution relative to a uniform grid with the same side budget. Appendix C.4 derives the corresponding tail tradeoff, and Appendix C.5 turns this local step-size argument into deterministic reconstruction and attention-logit error bounds.

4.4 Layer-wise adaptive α

α is the only parameter in μ-Zero: it decides how much tail fidelity to exchange for accuracy near the zero anchor. We use a conservative prefill-time rule to set α once per layer and cache type. The rule starts at $\alpha=2$ and selects from $\{2, 4, 16\}$ using scale-free statistics of the first quantized block: tail ratio, concentration ratio, and mean zero-anchor offset. Promotion thresholds depend on the bit width and distinguish keys from values; the default 2- and 4-bit configurations retain $\alpha=2$. The selected value is reused throughout decoding and stored only per layer and cache type, leaving the per-group format unchanged. Appendix C.7 gives the statistical formulation.

5 Experiments

5.1 Experimental setup

Tasks and models. We evaluate reasoning and code generation on Llama-3.1-8B-Instruct and Qwen3-8B using GSM8K-CoT and HumanEval, and long-context understanding on four Llama/Qwen models spanning 0.6B–8B using LongBench-v1. Appendix E extends the evaluation to long-output reasoning, models up to 70B, and selected MQA/MLA architectures.

Baselines. We compare against representative state-of-the-art KV cache quantizers spanning the main design families. Scalar quantizers KIVI (Liu et al., 2024b) and HQQ (Badri & Shaji, 2023), transform/sketch-based TurboQuant (Zandieh et al., 2025), and mixed-precision KVTuner (Li et al., 2025b), with BF16 as the uncompressed quality reference.

Protocol. Methods share model weights, prompts, and decoding settings within each quality comparison. Unless otherwise specified, μ-Zero uses channel-wise keys, token-wise values, group size $G=64$, and residual window size $W=128$. Main-text experiments use an NVIDIA RTX PRO 6000 Blackwell GPU with 96 GB memory. Appendix D details benchmark protocols.

5.2 Reasoning and code generation

Table 1 reports GSM8K and HumanEval results on the two 8B models. These autoregressive tasks are particularly sensitive to KV-cache errors:a perturbation that changes an intermediate reasoning step or generated code token can propagate through all subsequent decoding. The 2-bit regime is therefore the most diagnostic. Compared with the strongest scalar baseline, KIVI, μ-Zero improves HumanEval pass@1 by 8.17 percentage points on Qwen3-8B and 8.29 points on Llama-3.1-8B-Instruct, with consistent gains in pass@5 and GSM8K EM. TurboQuant degrades more sharply on Qwen3-8B, suggesting that preserving compressed-domain inner products alone is not sufficient when attention logits depend strongly on signed cancellations. At 3–4 bits, the comparison shifts from error recovery to quality saturation: μ-Zero reaches BF16-level GSM8K at 3 bits, matches or exceeds BF16 on five of six 4-bit metrics, and outperforms KVTuner's offline 4-bit allocation on every reported metric. Additional AIME'25 and 70B-model evaluations likewise yield higher point estimates than KIVI at both 2 and 4 bits (Appendices E.3–E.4). Overall, these results show that μ-Zero is not merely a low-bit rescue mechanism; its zero-centered scalar grid preserves autoregressive generation quality across bit widths until quantization error approaches the resolution of the benchmarks.

5.3 Long-context evaluation

Table 2 summarizes LongBench performance across model families and scales. The aggregate scores reveal a clear precision transition. At 2 bits, μ-Zero is the best quantized method on all four models,

Table 1: **Autoregressive reasoning and code generation on 8B models.** GSM8K uses 8-shot CoT prompting; HumanEval reports pass rates. Abbreviations: μ-Z for μ-Zero, TQ for TurboQuant, KVT for KVTuner, EM for Exact Match. TQ-2.5/3/4 correspond to TurboQuant's K3V2/K4V2/K4V4 key–value bit modes; see Appendix D.2 for details. **Bold** marks the best quantized result per group.

Model	Task	Metric	Raw	2-bit			3-bit		4-bit			
			BF16	KIVI	TQ-2.5	μ-Z	TQ	μ-Z	KVT	KIVI	TQ	μ-Z
Qwen3-8B	GSM8K	EM	0.8863	0.8120	0.5747	**0.8575**	0.8393	**0.8886**	0.8757	0.8825	0.8370	**0.8825**
	HumanEval	Pass@1	0.5439	0.4250	0.2963	**0.5067**	0.3811	**0.5384**	0.5116	0.5421	0.3963	**0.5500**
		Pass@5	0.8035	0.6927	0.5569	**0.7577**	0.7325	**0.7905**	0.7980	0.7983	0.7268	**0.8039**
Llama-3.1-8B-Inst.	GSM8K	EM	0.7551	0.6861	0.5315	**0.7111**	0.6975	**0.7665**	0.7415	0.7559	0.7089	**0.7589**
	HumanEval	Pass@1	0.5409	0.4415	0.3817	**0.5244**	0.4091	**0.5537**	0.4988	0.5311	0.4067	**0.5566**
		Pass@5	0.7705	0.6825	0.6772	**0.7368**	0.6931	**0.7722**	0.7356	0.7627	0.6985	**0.7747**

Table 2: **Average LongBench-v1 score over all 21 tasks.** Abbreviations: μ-Z for μ-Zero, TQ for TurboQuant, KVT for KVTuner. TQ-2.5/3/4 correspond to TurboQuant's K3V2/K4V2/K4V4 key–value bit modes; see Appendix D.2 for details. *KVTuner uses offline mixed-precision allocation under a nominal 4-bit budget. Appendix E.1 gives per-category breakdowns.

Model	Raw	2-bit				3-bit		4-bit					8-bit	
	BF16	HQQ	KIVI	TQ-2.5	μ-Z	TQ	μ-Z	HQQ	KIVI	KVT*	TQ	μ-Z	HQQ	μ-Z
Qwen3-0.6B	**24.20**	8.93	18.88	8.99	**20.73**	10.21	**23.65**	16.51	23.67	23.84	10.94	**24.11**	**24.20**	**24.20**
Llama-3.2-3B	29.20	14.38	27.36	26.40	**27.50**	28.70	**29.13**	28.85	28.98	28.84	28.80	**29.08**	29.17	**29.22**
Llama-3.1-8B-Inst.	39.21	23.54	37.57	36.66	**38.62**	38.49	**38.94**	39.08	39.13	39.07	38.79	**39.19**	39.06	**39.25**
Qwen3-8B	37.70	19.22	35.62	33.85	**36.04**	36.70	**37.00**	37.33	37.61	37.05	37.14	**37.75**	37.69	**37.73**

with the largest gains over KIVI on Qwen3-0.6B and Llama-3.1-8B-Instruct. At 3 bits, it closes most of the BF16 gap, while TurboQuant remains highly model dependent, especially on Qwen3-0.6B. At 4 bits, the strong baselines nearly saturate the benchmark, yet μ-ZERO still gives the best model averages. Appendix E.1 shows that these averages do not come from uniform dominance over every task family. Rather, μ-ZERO avoids severe category collapses at low precision and remains balanced when different categories favor different baselines. RULER further confirms the long-context gains over KIVI at 2 and 4 bits across 4k–32k contexts (Appendix E.2). Complementary task-level and softmax-KL evaluations support transfer to MQA and native-latent MLA caches (Appendix E.5). This pattern is consistent with Section 3: zero-centered resolution is most valuable under tight budgets, and at moderate precision its main benefit is robustness across architectures and task types.

5.4 ABLATION STUDIES

We isolate the design choices behind μ-ZERO using softmax KL to the BF16 attention distribution. The reference oracle in Figure 3 performs layer-wise grid search over α under the attention-weighted objective of Appendix C.6. Algorithm 1 replaces this search with a conservative rule that starts from the robust mild-companding default $\alpha = 2$ and promotes only when prefill statistics indicate that additional zero-neighborhood resolution is worth the tail cost.

Alpha selection. Figure 3(a) normalizes quantized policies by the layer-wise attention oracle and includes BF16 as the full-precision reference. The fixed-$\alpha = 2$ baseline is already competitive, matching our observation that the common channel-wise key setting usually lies in the mild-companding regime. Under token-wise key grouping, where stronger companding is more often useful, the conservative promotion rule closes most of the remaining gap to the layer-wise oracle.

Zero anchoring. Figure 3(b) disables the zero-centered split while keeping the same nonuniform compander. The resulting degradation shows that logarithmic spacing alone is not sufficient. The highest-resolution region must be tied to the real value $x = 0$, where Section 3.2 shows attention cancellation is most fragile.

Grouping and allocation. Figure 3(c) compares the common channel-wise-key/token-wise-value setting with an alternative that groups keys token-wise. The common setting gives lower attention

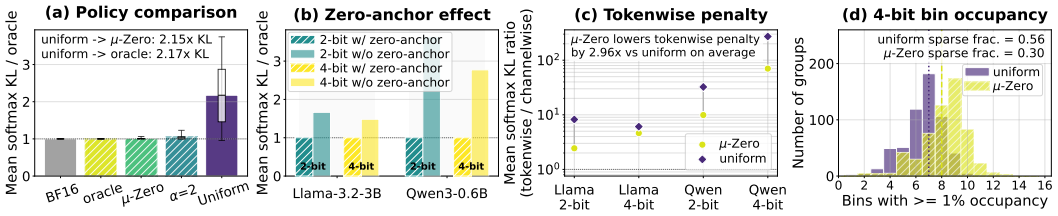

Figure 3: **Ablation of alpha selection and zero-centered allocation.** Lower softmax KL is better; BF16 is the full-precision reference and oracle is layer-wise grid search over α. (a) The μ-Zero promotion rule approaches the layer-wise attention oracle while retaining the strong fixed-$\alpha = 2$ default. (b) Removing the zero anchor hurts both 2-bit and 4-bit settings. (c) Token-wise key grouping increases attention error, but μ-Zero reduces the penalty versus uniform quantization. (d) Uniform 4-bit leaves many bins sparse, while μ-Zero allocates codes near the populated zero-centered region.

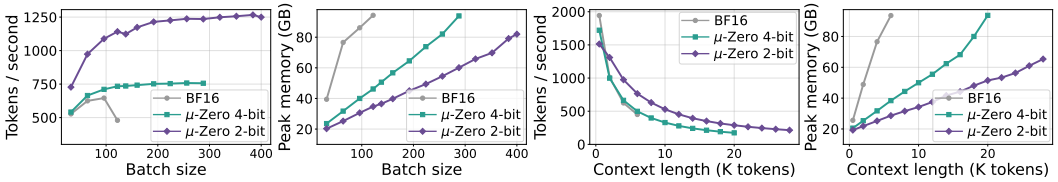

Figure 4: **End-to-end decoding efficiency.** The left two panels sweep batch size at context length 4096, and the right panels sweep context length at batch size 64; each run generates 512 new tokens. Lines terminate when a method runs out of memory.

error and is therefore our default. When keys are grouped token-wise, attention error increases for both quantizers, but μ-ZERO preserves more quality than uniform min–max quantization. This indicates that zero-centered companding adapts to the changed key distribution rather than relying on a single grouping choice. The advantage also persists across group sizes: at 2 bits, μ-ZERO achieves higher accuracy point estimates than KIVI for $G \in \{32, 64, 128\}$, with larger gains on Qwen3-8B under coarser grouping (Appendix G.1). Finally, Figure 3(d) visualizes the mechanism: at 4 bits, uniform quantization leaves many bins effectively unused, whereas μ-ZERO redistributes codes toward populated regions around zero. Together, the panels link these gains to attention-aligned alpha selection, an exact zero anchor, and distribution-adaptive nonuniform code allocation.

5.5 PRACTICAL OVERHEAD

μ-ZERO changes the scalar reconstruction rule but not the cache layout: it stores packed integer codes and the same per-group affine metadata (m, s) as min–max quantization, derives the zero level from this metadata, and keeps α once per layer and cache type. We implement μ-ZERO on top of the HuggingFace Transformers codebase. Appendix F.2 reports effective bit rates, measured cache footprints, and compression ratios. Figure 4 shows improved decoding capacity and throughput relative to BF16. At context 4096, μ-ZERO increases the largest feasible batch from 122 to 288 at 4 bits and at least 400 at 2 bits, with up to $1.53\times$ and $2.38\times$ throughput on shared batch settings; Appendix F.3 gives the full breakdown. Across three additional deployment configurations, μ-ZERO achieves 1.26–$1.46\times$ the throughput of official KIVI at 4 bits and 1.03–$1.07\times$ at 2 bits, with lower peak reserved memory (Appendix F.4). Appendix F.5 compares kernel timings and LUT versus per-element exponential reconstruction.

6 CONCLUSION

We introduced μ-ZERO, a zero-anchored log-companded quantizer that reallocates scalar levels toward the attention-sensitive zero neighborhood while preserving the standard per-group KV cache interface. Across reasoning, code generation, long-context, and system-level evaluations, the results show that improving the local scalar quantizer can increase both low-bit accuracy and decoding efficiency without changing the attention interface.

AI USE STATEMENT

During manuscript preparation and revision, we used generative AI tools for drafting and editing text, organizing supplementary results and assisting with LaTeX formatting and consistency checks. The authors retain responsibility for the final text, claims, experimental results, and artifacts, including content prepared with AI assistance.

ETHICS STATEMENT

This work studies inference-time KV cache quantization using existing pretrained models and benchmarks. Improved efficiency can broaden access to language models while also facilitating misuse of the underlying models. Quantization may alter model behavior, and the reported accuracy and efficiency results do not establish safety or fairness in downstream applications. Appendix H discusses these broader implications.

REPRODUCIBILITY STATEMENT

Section 4 specifies the quantizer, and Appendix C provides theoretical derivations and the prefill-time selection rule. Appendix D documents benchmark protocols, baseline configurations, and hardware; Appendices E–G provide extended results and experiment-specific settings, including memory accounting and runtime measurements. The implementation is available through the code repository linked in the abstract.

REFERENCES

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

Hicham Badri and Appu Shaji. Half-quadratic quantization of large machine learning models, November 2023. URL https://mobiusml.github.io/hqq_blog/.

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

Adithya Bhaskar, Alexander Wettig, Tianyu Gao, Yihe Dong, and Danqi Chen. Cache me if you can: How many kvs do you need for effective long-context lms? *arXiv preprint arXiv:2506.17121*, 2025.

Han Chen, Zicong Jiang, Zining Zhang, Bingsheng He, Pingyi Luo, Mian Lu, and Yuqiang Chen. Logquant: Log-distributed 2-bit quantization of kv cache with superior accuracy preservation. *arXiv preprint arXiv:2503.19950*, 2025.

Peilin Chen and Xiaoxuan Yang. Titanus: Enabling kv cache pruning and quantization on-the-fly for llm acceleration, 2025. URL https://arxiv.org/abs/2505.17787.

Renze Chen, Zhuofeng Wang, Beiquan Cao, Tong Wu, Size Zheng, Xiuhong Li, Xuechao Wei, Shengen Yan, Meng Li, and Yun Liang. Arkvale: Efficient generative llm inference with recallable key-value eviction. *Advances in Neural Information Processing Systems*, 37:113134–113155, 2024.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *arXiv preprint arXiv:2407.11550*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model, 2024. URL https://arxiv.org/abs/2307.06945.

Jiawei Guan, Feng Zhang, Jiesong Liu, Hsin-Hsuan Sung, Ruofan Wu, Xiaoyong Du, and Xipeng Shen. Trec: Transient redundancy elimination-based convolution. *Advances in Neural Information Processing Systems*, 35:26578–26589, 2022.

Mohsen Hariri, Alan Luo, Weicong Chen, Shaochen Zhong, Tianyi Zhang, Qifan Wang, Xia Hu, Xiaotian Han, and Vipin Chaudhary. Quantize what counts: More for keys, less for values, 2025. URL https://arxiv.org/abs/2502.15075.

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.

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?, 2024. URL https://arxiv.org/abs/2404.06654.

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*, pp. 611–626, 2023.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. {InfiniGen}: Efficient generative inference of large language models with dynamic {KV} cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 155–172, 2024.

Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. A survey on large language model acceleration based on kv cache management. *arXiv preprint arXiv:2412.19442*, 2024a.

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, 2025a. URL https://arxiv.org/abs/2506.18879.

Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Wulong Liu, Yiwu Yao, Sinno Jialin Pan, and Mingxuan Yuan. Kvtuner: Sensitivity-aware layer-wise mixed-precision kv cache quantization for efficient and nearly lossless llm inference. *arXiv preprint arXiv:2502.04420*, 2025b.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction, 2021. URL https://arxiv.org/abs/2102.05426.

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.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *Proceedings of Machine Learning and Systems*, 7, 2025.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Tengxuan Liu, Shiyao Li, Jiayi Yang, Tianchen Zhao, Feng Zhou, Xiaohui Song, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. Pm-kvq: Progressive mixed-precision kv cache quantization for long-cot llms. *arXiv preprint arXiv:2505.18610*, 2025.

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. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024b.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. Towards efficient generative large language model serving: A survey from algorithms to systems. *ACM Computing Surveys*, 58(1):1–37, 2025.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization, 2020. URL https://arxiv.org/abs/2004.10568.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pp. 31094–31116. PMLR, 2023.

Donghyun Son, Euntae Choi, and Sungjoo Yoo. Nsnquant: A double normalization approach for calibration-free low-bit vector quantization of kv cache, 2025. URL https://arxiv.org/abs/2505.18231.

Yi Su, Yuechi Zhou, Quantong Qiu, Juntao Li, Qingrong Xia, Ping Li, Xinyu Duan, Zhefeng Wang, and Min Zhang. Accurate kv cache quantization with outlier tokens tracing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12895–12915, 2025a.

Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Linge Li, Huangqi Yu, and Kehong Yuan. Rotatekv: Accurate and robust 2-bit kv cache quantization for llms via outlier-aware adaptive rotations, 2025b. URL https://arxiv.org/abs/2501.16383.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Xindi Wang, Mahsa Salmani, Parsa Omidi, Xiangyu Ren, Mehdi Rezagholizadeh, and Armaghan Eshaghi. Beyond the limits: A survey of techniques to extend the context length in large language models. *arXiv preprint arXiv:2402.02244*, 2024.

Zongwu Wang, Peng Xu, Fangxin Liu, Yiwei Hu, Qingxiao Sun, Gezi Li, Cheng Li, Xuan Wang, Li Jiang, and Haibing Guan. Million: Mastering long-context llm inference via outlier-immunized kv product quantization, 2025. URL https://arxiv.org/abs/2504.03661.

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. *Advances in neural information processing systems*, 37:119638–119661, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023a.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023b.

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

Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. Wkvquant: Quantizing weight and key/value cache for large language models gains more, 2024. URL https://arxiv.org/abs/2402.12065.

Amir Zandieh, Majid Daliri, and Insu Han. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead, 2024. URL https://arxiv.org/abs/2406.03482.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate. *arXiv preprint arXiv:2504.19874*, 2025.

Jiebin Zhang, Dawei Zhu, Yifan Song, Wenhao Wu, Chuqiao Kuang, Xiaoguang Li, Lifeng Shang, Qun Liu, and Sujian Li. More tokens, lower precision: Towards the optimal token-precision trade-off in kv cache compression, 2025a. URL https://arxiv.org/abs/2412.12706.

Tao Zhang, Ziqian Zeng, Hao Peng, Huiping Zhuang, and Cen Chen. Mixkvq: Query-aware mixed-precision kv cache quantization for long-context reasoning, 2025b. URL https://arxiv.org/abs/2512.19206.

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.

A Additional Cache Statistics

Figure 1 protocol. The cache statistics in Figure 1 are computed from a forward pass of Qwen3-8B on a held-out LongBench slice. The grouping follows the quantization layout: key cache groups are formed along the channel dimension, while value cache groups are formed along the token dimension, both with group size $G = 64$. For each group, we record empirical samples, absolute-value quantiles, sign-conditioned tail curves, and the occupancy of a 4-bit uniform min–max quantizer.

Panel definitions. Panel (a) plots normalized histograms for three representative groups. These groups are selected by sorting all groups by layer index, concentration, and tail ratio, then taking evenly spaced positions in the sorted list; this gives shallow, middle, and deep examples without hand-picking individual heads. Panel (b) summarizes every valid group by two scale-free statistics:

$$r_{\text{tail}} = \frac{p_{99}(|X|)}{p_{95}(|X|)}, \qquad r_{\text{conc}} = \frac{p_{95}(|X|)}{p_{50}(|X|)}.$$

The dotted horizontal references are the concentration ratios of a half-normal distribution (2.91) and an exponential absolute-value distribution, equivalently the positive half of a Laplace law (4.33). Panel (c) compares the sign-conditioned survival curves

$$S^{+}(t) = \Pr(x > t \mid x > 0), \qquad S^{-}(t) = \Pr(-x > t \mid x < 0).$$

Before aggregation, each group's threshold is divided by its own $p_{95}(|x|)$; hence $t/p_{95} = 1$ denotes that group's 95th absolute-percentile scale. Curves are interpolated in log-survival space onto a common threshold grid. The heavy curves are medians across groups, the darker and lighter bands are the 25–75% and 5–95% ranges, and faint background curves show a random subsample of groups. Panel (d) applies per-group uniform 4-bit min–max quantization and counts how many of the 16 bins contain at least 1% of that group's samples. The sparse-bin fraction is the fraction of bins below this 1% occupancy threshold.

Key cache observations. For the key cache in Figure 1, 99% of groups lie above the half-normal concentration reference, and a visible fraction also exceed the half-Laplace reference. The tail-ratio axis spans more than an order of magnitude, indicating that groups differ in distributional shape and not merely in range. The conditional survival curves are approximately linear on the log-survival axis over the high-mass tail, supporting a progressively coarser allocation away from the dense center. The positive and negative median curves nearly overlap after p_{95} normalization; the fitted log-slope difference over $t/p_{95} \in [1, 3]$ rounds to 0.00 decades per unit threshold in the plotted statistic. Uniform 4-bit binning has median occupancy $7/16$ and median sparse-bin fraction 0.56, showing that the nominal bit budget overstates the number of bins effectively used by typical groups.

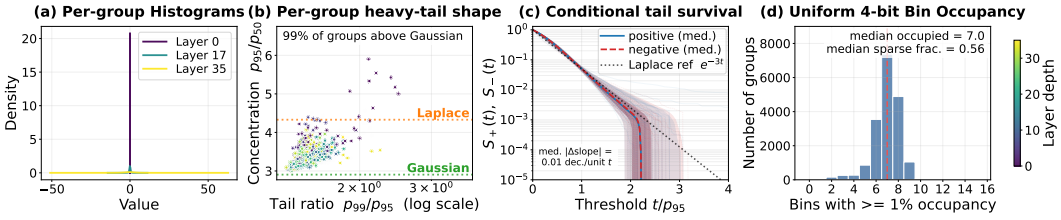

Figure 5: **Per-group distributional statistics of value cache in Qwen3-8B.** Token-dimension value groups show the same near-zero concentration, log-linear tail decay, shared sign-conditioned tail shape, and sparse uniform-bin usage observed for keys. Panel definitions are given in Appendix A.

Value cache observations. Figure 5 repeats the same analysis for value cache groups, using token-dimension grouping with $G = 64$. The qualitative conclusions match the key cache case. The representative histograms show an even sharper concentration near zero, while the scatter plot again places 99% of groups above the half-normal reference. Compared with the key cache, the visible tail-ratio range is narrower, but the concentration ratios still vary substantially across groups, so a fixed allocation remains mismatched to parts of the distribution. The conditional positive and negative tails again nearly coincide after p_{95} normalization; the plotted median-curve log-slope difference is 0.01 decades per unit threshold. Uniform 4-bit bin usage is also essentially unchanged, with median

occupancy $7/16$ and median sparse-bin fraction 0.56. Thus, the value cache supports the same design conclusion as the key cache: the codebook should be nonuniform and density-aware, with a shared tail-shape rule across signs.

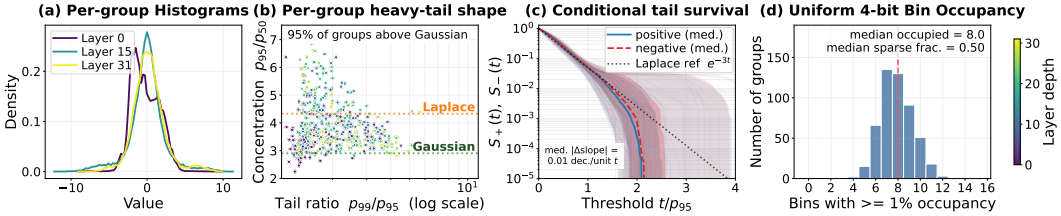

Figure 6: **Per-group distributional statistics of key cache in Llama-3.1-8B-Instruct.** Channel-dimension key groups reproduce the main Qwen3 pattern: near-zero concentration, heavy-tailed group statistics, approximately log-linear conditional survival, and sparse use of uniform 4-bit bins.

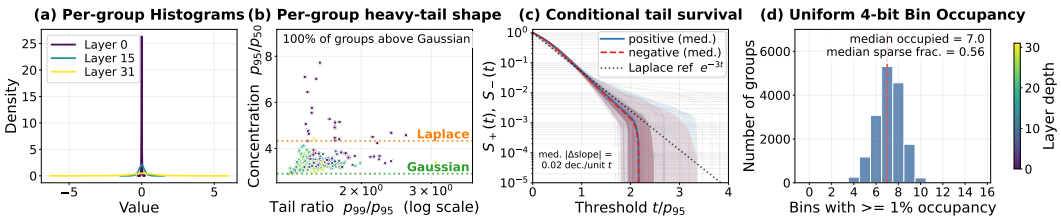

Figure 7: **Per-group distributional statistics of value cache in Llama-3.1-8B-Instruct.** Token-dimension value groups show the same distributional structure, with especially sharp near-zero concentration and inefficient uniform-bin usage.

Cross-model observations. Figures 6 and 7 repeat the analysis on Llama-3.1-8B-Instruct. The results are qualitatively consistent with Qwen3-8B, but the strength differs by cache type. For key groups, 95% of groups lie above the half-normal concentration reference, the positive and negative conditional tails have nearly identical normalized decay, and uniform 4-bit quantization uses a median of $8/16$ bins with median sparse-bin fraction 0.50; this is still inefficient, though slightly less sparse than Qwen3 key groups ($7/16$ bins and 0.56 sparse fraction). For value groups, the agreement is stronger: 100% of groups lie above the half-normal reference, the representative histograms exhibit a very sharp mode near zero, and uniform 4-bit quantization again uses only $7/16$ bins at the median with sparse-bin fraction 0.56. The fitted positive/negative median-curve log-slope differences are small for both caches (0.01 for keys and 0.02 for values). Thus the observations motivating μ-Zero are not model-specific: after applying the common key-channel/value-token grouping, KV cache groups in both model families remain near-zero concentrated, heavy-tailed, and poorly matched to uniform code spacing, with the closest numerical match on value groups and a milder but consistent pattern on Llama key groups.

B Attention-Level Sensitivity Analysis

Figure 2 protocol. Figure 2 is computed from Qwen3-8B on the same held-out LongBench slice. For sampled query-key pairs, we record the BF16 raw logit, the absolute sum of per-dimension products, the cancellation ratio, and the logit produced after per-group uniform 4-bit key quantization. The plotted error in panels (b) and (c) is relative logit error under uniform quantization unless otherwise specified. The sign-flip indicator records whether quantization changes the sign of the pre-softmax logit.

Panel definitions. Panel (a) plots the empirical distribution of the cancellation ratio

$$C(q,k) = \frac{\left|\sum_j q_j k_j\right|}{\sum_j |q_j k_j| + \varepsilon}.$$

This is a diagnostic metric defined for this analysis. It compares the magnitude of the final dot product with the sum of magnitudes of its individual terms: $C \approx 1$ means the terms mostly agree in sign, whereas $C \approx 0$ means the logit is produced by strong cancellation between positive and negative contributions. The dashed and dotted vertical lines mark $C = 0.5$ and $C = 0.3$, respectively.

Panel (b) plots relative logit error against $C(q, k)$. The blue hexbin is a two-dimensional histogram: darker cells contain more sampled query-key pairs with that cancellation ratio and error. To summarize the trend, we partition the horizontal axis into bins of C; the red curve reports the median relative error within each bin, and the shaded red band reports the interquartile range. The green dashed curve uses the same bins and reports the fraction of pairs whose logit sign flips after uniform key quantization. Panel (c) groups pairs by $|\text{raw logit}|$ and plots median and 90th-percentile relative error for uniform quantization and μ-Zero. The inset reports the median-error ratio of uniform over μ-Zero. Panel (d) is a one-group intervention: one cache group is quantized while the rest of the cache remains full precision, and the local output MSE is averaged over all sampled groups as well as over the bottom and top quartiles of near-zero fraction.

Findings. Cancellation is common: the median score has $C = 0.18$, with 91.9% of sampled scores below 0.5 and 72.2% below 0.3. Uniform key quantization is most fragile in this low-C regime, where small perturbations alter a logit that is already the residual of larger positive and negative contributions. The sign-flip curve confirms the same mechanism: sign errors concentrate at low cancellation ratios and rapidly vanish as C increases. Across logit scales, μ-Zero lowers both median and high-percentile relative error relative to uniform quantization. In the one-group intervention, μ-Zero reduces local output MSE by 38% on average, with reductions of 40% in the bottom near-zero-fraction quartile and 32% in the top quartile. These results support the operator-level claim used in the main text: the near-zero region is not only dense, but also important for preserving cancellation and sign structure in attention.

Cross-model diagnostic. Applying the same cancellation diagnostic across model families reveals a useful difference behind the benchmark trends. For Llama-3.1-8B-Instruct, the median score has $C = 0.41$, with 77.6% of scores below 0.5 and 21.7% below 0.3. Qwen3-8B is substantially more cancellation-dominated: its median is $C = 0.18$, with 91.9% below 0.5 and 72.2% below 0.3. Thus Qwen3 attention logits are more often small residuals of larger signed products. This makes a fixed absolute perturbation in the key cache produce a larger relative logit error and a higher risk of sign changes. The observation provides a mechanistic explanation for why Qwen3 is a harder stress test for KV cache quantization, and why methods that mainly preserve global range, average reconstruction, or compressed-domain inner products can degrade sharply on Qwen3 in Table 2. It also strengthens the case for μ-Zero's design: the quantizer should spend resolution where cancellation makes attention most fragile, rather than only where the cache range is largest.

C THEORY FOR μ-ZERO

This appendix gives a compact theoretical account of the design choices in μ-Zero. The goal is not to prove global optimality for attention, but to connect the deployed rule to a tractable approximation of the objective that attention actually uses. The empirical facts established in Section 3 are that cache groups are nonuniform, approximately log-tailed after normalization, asymmetric in side extent, and most fragile near zero under attention cancellation. The theory proceeds from local companding error and attention sensitivity to the distributional, zero-anchor, and oracle arguments behind the deployed selector.

C.1 HIGH-RATE DISTORTION OF COMPANDED QUANTIZATION

Let $u \in [0, 1]$ denote a side-normalized magnitude, with density $p(u)$. A scalar compander quantizes $v = \Phi(u)$ uniformly and reconstructs with Φ^{-1}, where Φ is monotone, $\Phi(0) = 0$, and $\Phi(1) = 1$. At high rate, the Bennett approximation gives the mean squared reconstruction error

$$D(\Phi; p) \;\approx\; \frac{1}{12L^2} \int_0^1 p(u)\,[\Phi'(u)]^{-2}\,du, \tag{12}$$

where L is the number of quantization intervals on that side. For the μ-law family

$$\Phi_\alpha(u) = \frac{\log(1+\alpha u)}{\log(1+\alpha)}, \qquad \Phi'_\alpha(u) = \frac{\alpha}{(1+\alpha u)\log(1+\alpha)}, \tag{13}$$

Eq. (12) becomes

$$D(\alpha; p) \;\approx\; \frac{\log^2(1+\alpha)}{12L^2\alpha^2} \int_0^1 p(u)(1+\alpha u)^2 \, du. \tag{14}$$

If minimized by itself, Eq. (14) would define a reconstruction-distortion oracle. We do not use that oracle as the deployment objective: attention is nonlinear in keys and values, and a group with large reconstruction error may have little effect if it is not attended to or if its logit perturbation is softmax-invariant. Instead, Eq. (14) is used as a local error model: it shows how increasing α reduces near-zero error by increasing $\Phi'_\alpha(0)$, while increasing the error assigned to the tail.

C.2 Cancellation amplifies reconstruction error

Let $\ell = q^\top k$ be a full-precision attention logit and let $\hat{k} = k + \delta$ be the quantized key. The logit error is

$$\Delta\ell = q^\top \delta = \sum_j q_j \delta_j. \tag{15}$$

Define the cancellation ratio

$$C(q,k) = \frac{|q^\top k|}{\sum_j |q_j k_j| + \varepsilon}. \tag{16}$$

Then

$$\frac{|\Delta\ell|}{|\ell| + \varepsilon} \le \frac{\sum_j |q_j||\delta_j|}{|q^\top k| + \varepsilon} \approx \frac{1}{C(q,k)} \cdot \frac{\sum_j |q_j||\delta_j|}{\sum_j |q_j k_j| + \varepsilon}. \tag{17}$$

Equation (17) formalizes the mechanism in Figure 2: for the same weighted reconstruction error, relative logit error grows as $C(q,k)$ decreases. Since Figure 2(a) shows that low-cancellation logits are common, reducing reconstruction error near the dense region is not only a data-side objective; it directly controls the relative perturbation of attention logits in the regime where sign flips are most likely.

The deterministic bound below instantiates $|\delta_j|$ with the local μ-Zero reconstruction bound. Because the zero-centered compander reduces the maximum local error near zero, it lowers the numerator of Eq. (17) precisely when the denominator is small. This explains why a density-matched quantizer can improve attention behavior even when the full-range extrema are already preserved.

C.3 Exponential tails imply log-like companding

Figure 1(c) shows that the sign-conditioned tail survival is close to linear on a log scale after p_{95} normalization. A simple model for a side magnitude is therefore a truncated exponential density

$$p_\lambda(u) = \frac{\lambda e^{-\lambda u}}{1 - e^{-\lambda}}, \qquad u \in [0,1], \tag{18}$$

where larger λ corresponds to stronger concentration near zero. Substituting Eq. (18) into Eq. (14) gives

$$D(\alpha; \lambda) \;\approx\; \frac{\log^2(1+\alpha)}{12L^2\alpha^2} \left(1 + 2\alpha\mu_1(\lambda) + \alpha^2\mu_2(\lambda)\right), \tag{19}$$

where

$$\mu_1(\lambda) = \mathbb{E}_{p_\lambda}[u] = \frac{1}{\lambda} - \frac{1}{e^\lambda - 1}, \tag{20}$$

and

$$\mu_2(\lambda) = \mathbb{E}_{p_\lambda}[u^2] = \frac{2}{\lambda^2} - \frac{1 + 2/\lambda}{e^\lambda - 1}. \tag{21}$$

Equation (19) makes the role of α explicit. Increasing α improves resolution near zero but increases the penalty on larger u. Thus the useful α depends on concentration: highly concentrated groups can benefit from a stronger compander, while groups with more mass in the tail require a milder shape. The statistics used by μ-Zero approximate this dependence without fitting λ: $r_{\mathrm{conc}} = p_{95}/p_{50}$ increases with near-zero concentration, $r_{\mathrm{tail}} = p_{99}/p_{95}$ tracks tail spread, and r_{anch} captures side imbalance in the observed range.

Concentration and the alpha direction. The heuristic selection rule in Section 4.4 uses larger α when the layer statistics indicate stronger zero concentration. The density-only proxy in Eq. (19) explains this direction.

Write Eq. (19) as $D(\alpha;\lambda) = g(\alpha)\cdot h(\alpha,\lambda)$, where

$$g(\alpha) = \frac{\log^2(1+\alpha)}{\alpha^2}, \qquad h(\alpha,\lambda) = \frac{1+2\alpha\mu_1(\lambda)+\alpha^2\mu_2(\lambda)}{12L^2}. \tag{22}$$

Since $\mu_1(\lambda)$ and $\mu_2(\lambda)$ are strictly decreasing in λ (standard calculation from the truncated exponential CDF), $h(\alpha,\lambda)$ decreases in λ for any $\alpha > 0$. Moreover, this decrease is larger for stronger companders: at fixed λ, $\partial_\lambda h(\alpha,\lambda) = 2\alpha\mu_1'(\lambda) + \alpha^2\mu_2'(\lambda)$, whose magnitude grows with α since $\mu_1', \mu_2' < 0$. Thus, as λ increases, larger α gains a comparatively larger reduction in h.

Consider the boundary cases. At $\lambda = 0$ the distribution is uniform, $\mu_1 = \frac{1}{2}$, $\mu_2 = \frac{1}{3}$, and $D(\alpha;0) \propto g(\alpha)(1+\alpha+\frac{\alpha^2}{3})$, which is *increasing* in α over the candidate set $\mathcal{A}$ (uniform quantization is optimal for uniform data; numerically $D(2;0) \approx 1.31 < D(4;0) \approx 1.67 < D(16;0) \approx 3.21$). In contrast, as $\lambda \to \infty$ the moments vanish, so $D(\alpha;\lambda) \to g(\alpha)/(12L^2)$. Since $g(\alpha)$ is strictly *decreasing* in α—verified by differentiating and using $\frac{\alpha}{1+\alpha} < \log(1+\alpha)$ for all $\alpha > 0$—larger α strictly dominates in the concentrated limit.

For the density-only proxy, continuity gives a crossover between any two candidates $\alpha_1 < \alpha_2$:

$$D(\alpha_1;\lambda^\star_{\alpha_1,\alpha_2}) = D(\alpha_2;\lambda^\star_{\alpha_1,\alpha_2}). \tag{23}$$

Below this crossover the milder compander has lower density-only distortion; above it the stronger compander has lower density-only distortion:

$$\arg\min_{\alpha\in\{\alpha_1,\alpha_2\}} D(\alpha;\lambda) = \begin{cases}\alpha_1 & \lambda < \lambda^\star_{\alpha_1,\alpha_2},\\ \alpha_2 & \lambda > \lambda^\star_{\alpha_1,\alpha_2}.\end{cases} \tag{24}$$

For $\alpha_1 = 2$ and $\alpha_2 = 4$, numerical evaluation of Eq. (19) places the density-only crossover near $\lambda^\star \approx 4.5$. Because $r_{\mathrm{conc}} = p_{95}/p_{50}$ is monotone increasing in λ under the truncated exponential model (it ranges from 1.9 at $\lambda = 0$ to 4.3 at $\lambda \to \infty$), a threshold on r_{conc} is a cheap observable proxy for which side of the tradeoff a layer occupies.

The same interpretation applies to r_{tail}: under the truncated exponential, $r_{\mathrm{tail}} = p_{99}/p_{95}$ is also monotone in λ (ranging from ≈ 1.04 to ≈ 1.54), so it provides a complementary signal about the zero-concentrated, heavy-tail shape. The r_{anch} term is not directly covered by this single-side exponential model; it is included because a large anchor offset concentrates most of the finite side budget on one side, making the zero-neighborhood allocation more consequential.

Attention weighting can change the exact choice: a layer with the same marginal p_λ can prefer a different α if its high-attention tokens or high-query-energy coordinates occupy different parts of the group. This is why the selector is motivated by the attention oracle defined in Appendix C.6 rather than by a reconstruction oracle. The density model supplies the direction and the cheap statistics; the attention-weighted objective supplies the deployment-relevant design target.

C.4 Zero-anchor resolution gain and tail cost

Consider the positive side of a group after zero anchoring. Let e_+ be its normalized side extent and L_+ its number of intervals. The local quantization step in the original normalized coordinate is approximately

$$\Delta_+(u) \;\approx\; \frac{e_+}{L_+\Phi'_\alpha(u)}. \tag{25}$$

At zero, $u = 0$, this gives

$$\Delta_+^{\mu\text{-Zero}}(0) = \frac{e_+\log(1+\alpha)}{L_+\alpha}. \tag{26}$$

Uniform quantization with the same side extent and side budget has constant step size

$$\Delta_+^{\mathrm{uni}} = \frac{e_+}{L_+}. \tag{27}$$

Therefore the near-zero resolution gain of μ-Zero over uniform quantization is

$$\rho_0(\alpha) = \frac{\Delta_+^{\text{uni}}}{\Delta_+^{\mu\text{-Zero}}(0)} = \frac{\alpha}{\log(1+\alpha)}. \tag{28}$$

For the candidate values used in our low-bit implementation, $\rho_0(2) \approx 1.82$, $\rho_0(4) \approx 2.49$, and $\rho_0(16) \approx 5.65$. This gives a direct interpretation of the discrete α set: it provides a small set of near-zero resolution gains.

The same calculation also exposes the tradeoff. At the tail $u = 1$,

$$\Delta_+^{\mu\text{-Zero}}(1) = \frac{e_+(1+\alpha)\log(1+\alpha)}{L_+\alpha}, \tag{29}$$

so the tail step relative to uniform is

$$\tau_1(\alpha) = \frac{\Delta_+^{\mu\text{-Zero}}(1)}{\Delta_+^{\text{uni}}} = \frac{(1+\alpha)\log(1+\alpha)}{\alpha}. \tag{30}$$

Larger α therefore buys finer near-zero resolution at the price of coarser tail coverage. This is why a fixed global α is not desirable: groups with different concentration and tail statistics should occupy different points on the tradeoff curve.

C.5 Deterministic reconstruction error bound

The preceding arguments describe average distortion and step-size tradeoffs. We also record a deterministic bound that explains where the method reduces worst-case local error. Consider one side $\sigma \in \{+, -\}$ of a group, with normalized side extent e_σ, side budget L_σ, and side-normalized magnitude $u \in [0, 1]$. Let

$$\hat{u} = \Phi_\alpha^{-1}\left(\frac{\text{round}(L_\sigma \Phi_\alpha(u))}{L_\sigma}\right) \tag{31}$$

be the reconstructed side magnitude before rescaling by e_σ and s. Rounding in the companded coordinate gives

$$\left|\frac{\text{round}(L_\sigma \Phi_\alpha(u))}{L_\sigma} - \Phi_\alpha(u)\right| \le \frac{1}{2L_\sigma}. \tag{32}$$

By the mean value theorem,

$$|\hat{x} - x| \le \frac{s e_\sigma}{2L_\sigma} \max_{w \in I(u)} \frac{1}{\Phi'_\alpha(w)}, \tag{33}$$

where $I(u)$ is the interval between u and $\hat{u}$. For the μ-law map,

$$\frac{1}{\Phi'_\alpha(w)} = \frac{(1+\alpha w)\log(1+\alpha)}{\alpha}. \tag{34}$$

Thus the local error bound is smallest near the zero anchor and increases smoothly toward the tail. In particular, for values reconstructed in the first side bin,

$$|\hat{x} - x| \approx \frac{s e_\sigma \log(1+\alpha)}{2L_\sigma \alpha}, \tag{35}$$

whereas a uniform side quantizer with the same e_σ and L_σ has local error at most $s e_\sigma/(2L_\sigma)$. The near-zero worst-case bound is therefore reduced by the same factor $\alpha/\log(1+\alpha)$ as the step-size gain in Eq. (28). Globally, the bound becomes

$$|\hat{x} - x| \le \frac{s e_\sigma (1+\alpha)\log(1+\alpha)}{2L_\sigma \alpha}, \tag{36}$$

which exposes the corresponding tail cost. This formalizes the design tradeoff: α moves worst-case error budget from the dense zero neighborhood toward sparse extremes.

C.6 ATTENTION-WEIGHTED ALPHA ORACLE

The preceding sections define the compander family, its gain–cost curve, and the local error bounds controlled by α. We now define the attention-weighted objective that an ideal α selector would minimize if query-dependent cache access were available. For one attention head and one query, let

$$s_j = \frac{q^\top k_j}{\sqrt{d}}, \qquad a_j = \mathrm{softmax}(s)_j, \qquad o = \sum_j a_j v_j. \tag{37}$$

The first-order effect of value quantization is

$$\delta o_V = \sum_j a_j \delta v_j. \tag{38}$$

Ignoring cross-coordinate covariance gives the diagonal sensitivity weight

$$w^V_{j,c} \propto a_j^2, \tag{39}$$

so value errors at highly attended tokens matter more than equally large errors at tokens with negligible attention mass.

For key quantization, $\delta s_j = q^\top \delta k_j / \sqrt{d}$, and the softmax Jacobian gives

$$\delta o_K = \sum_j a_j (v_j - o)\, \delta s_j = \sum_j a_j (v_j - o) \frac{q^\top \delta k_j}{\sqrt{d}}. \tag{40}$$

Again using a diagonal approximation across coordinates,

$$w^K_{j,c} \propto \frac{a_j^2}{d} \, \|v_j - o\|_2^2 \, q_c^2. \tag{41}$$

Thus key error is important when the token has nontrivial attention mass, the value vector differs from the current output, and the query has energy in the perturbed coordinate. This is the formal reason that unweighted group NMSE is not the right alpha objective.

Let $\Delta_{\alpha,g,i}$ denote the local quantization step, or the corresponding high-rate error bound, for entry i in group g under shape α. For cache type $T \in \{K, V\}$, the attention-weighted oracle for layer ℓ is

$$\alpha^{\mathrm{attn}}_{\ell,T} = \arg\min_{\alpha \in \mathcal{A}} \mathbb{E}_{q,K,V} \left[\sum_g \sum_{i \in g} w_i^T(q, K, V)\, \Delta^2_{\alpha,g,i} \right], \tag{42}$$

where w_i^T is given by Eq. (39) or Eq. (41), with indices matched to the grouping layout. Equation (42) is the ideal attention-weighted selector objective. It is not an online algorithm: computing it would require queries, attention probabilities, values, and a search over candidate α values. The deployed selector in Section 4.4 is a layer-wise, prefill-time approximation to this objective.

C.7 CONSERVATIVE ALPHA PROMOTION RULE

The attention oracle in Eq. (42) cannot be evaluated during inference, but it suggests which quantities a deployed selector should preserve. The local step-size term $\Delta_{\alpha,g,i}$ is controlled by the compander tradeoff in Eq. (28) and Eq. (30); the attention weights in Eq. (39) and Eq. (41) are layer-structured but query-dependent. We therefore use the first quantized prefill block to estimate the layer-level side of the objective, and rely on the empirical stability of layer attention patterns for reuse during decoding.

A useful empirical fact is that the common channel-wise key layout is usually in the mild-companding regime: layer-wise attention search frequently selects $\alpha = 2$. This is not a failure of adaptation; it means that stronger companding has asymmetric risk. It can improve the zero neighborhood, but it also pays the tail cost in Eq. (30). The prefill-time rule is therefore deliberately conservative: start from the mild prior $\alpha_0 = 2$, and promote to a stronger compander only when layer statistics give evidence of a regime shift. Token-wise key quantization and very low-bit regimes are examples where this promotion is more often useful.

Algorithm 1 Conservative layer-wise α promotion rule

Require: Prefill cache tensor $X_{\ell,T}$, cache type $T \in \{K, V\}$, bit width b
Compute $r_{\text{tail}}, r_{\text{conc}}, r_{\text{anch}}$ from Eq. (43)
$\alpha \leftarrow 2$ ▷ mild-companding prior
if aggressive-promotion boundary is crossed **then**
    $\alpha \leftarrow 16$
else if moderate-promotion boundary is crossed **then**
    $\alpha \leftarrow 4$
else
    keep $\alpha = 2$
end if
return α

For each layer and cache type, the rule computes

$$r_{\text{tail}} = \frac{p_{99}(|x|)}{p_{95}(|x|)}, \qquad r_{\text{conc}} = \frac{p_{95}(|x|)}{p_{50}(|x|)}, \qquad r_{\text{anch}} = \mathbb{E}_g \left| z_g - \frac{1}{2} \right|, \tag{43}$$

where the percentiles are computed on a bounded subsample of prefill cache entries, and z_g is the zero anchor of group g. These three statistics are the observable proxies for the terms in Eq. (42). r_{conc} estimates how much mass lies in the high-gain zero neighborhood, r_{tail} estimates how much the range is stretched by sparse extremes beyond the high-mass region, and r_{anch} captures side imbalance in the zero-centered split. Large values are evidence for promotion, but the absence of such evidence keeps the safer default $\alpha = 2$.

To see why a promotion rule is the natural low-cost approximation, compare two candidate strengths $\alpha_1 < \alpha_2$. Split each side-normalized group into a zero neighborhood $C = \{u \le \tau\}$ and a tail $R = \{u > \tau\}$. A two-region approximation to Eq. (42) gives

$$J_T(\alpha) \approx W_C^T \, \Delta_\alpha(0)^2 + W_R^T \, \Delta_\alpha(1)^2, \tag{44}$$

where $W_C^T = \mathbb{E} \sum_{i \in C} w_i^T$ and $W_R^T = \mathbb{E} \sum_{i \in R} w_i^T$ are attention-weighted central and tail masses. The stronger compander is preferred when

$$\frac{W_C^T}{W_R^T} > \frac{\Delta_{\alpha_2}(1)^2 - \Delta_{\alpha_1}(1)^2}{\Delta_{\alpha_1}(0)^2 - \Delta_{\alpha_2}(0)^2}. \tag{45}$$

The right-hand side is determined only by the candidate pair and bit budget: it is the tail-cost increase required to buy the zero-neighborhood gain. The promotion rule therefore only needs coarse evidence about the left-hand side. r_{conc} increases when more entries lie near zero, raising W_C/W_R. r_{tail} increases when the observed range is stretched by sparse extremes beyond the high-mass region; under the attention-weighted objective this usually means that uniform quantization is spending codes on entries with small aggregate weight, again favoring a stronger compander. r_{anch} increases when zero is far from the midpoint, which makes the finite side budget near the zero anchor more valuable. Algorithm 1 is thus a conservative classifier for Eq. (45): promote only when at least one observable statistic crosses a calibrated boundary.

The promotion boundaries are monotone decision rules in $(r_{\text{tail}}, r_{\text{conc}}, r_{\text{anch}})$ and are fixed once for each cache type and bit width. In our implementation, moderate promotion uses a disjunction of the form

$$r_{\text{tail}} > \theta_{\text{tail}}^{T,b} \quad \text{or} \quad r_{\text{conc}} > \theta_{\text{conc}}^{T,b} \quad \text{or} \quad r_{\text{anch}} > \theta_{\text{anch}}^{T,b}, \tag{46}$$

with a second, stricter boundary for aggressive promotion in the 1-bit key setting. The constants are not fitted density parameters; they are coarse boundaries chosen from a small held-out layer-wise attention search to minimize false promotion. This calibration reflects the asymmetric risk above: if the layer-wise oracle is already in the $\alpha = 2$ regime, promoting α can only add tail error. The released implementation records the exact thresholds used for all reported experiments.

The resulting rule is cheap enough to run once per layer and cache type, stores no per-group α, and keeps the deployed cache format identical to fixed-α μ-Zero. The ablation in Section 5.4 evaluates whether this conservative promotion rule recovers the attention-oracle benefit relative to fixed, model-wise, and finer-grained alternatives.

Why a power-of-two candidate set? The power-of-two values are not required by the compander algebra; their role is to provide a small logarithmic grid over the gain–cost curve. From Eq. (28), the candidate values $\alpha \in \{2, 4, 16\}$ correspond to near-zero resolution gains

$$\rho_0(2) \approx 1.82, \qquad \rho_0(4) \approx 2.49, \qquad \rho_0(16) \approx 5.65.$$

Thus each candidate represents a qualitatively different allocation regime: mild companding, moderate companding, and aggressive protection of the zero neighborhood. The corresponding tail costs from Eq. (30) are also ordered and separated,

$$\tau_1(2) \approx 1.65, \qquad \tau_1(4) \approx 2.01, \qquad \tau_1(16) \approx 3.01.$$

A denser continuous grid would require more calibration or search while offering little benefit if the attention-oracle optimum is stable only up to a coarse regime. A linear grid would waste candidates in parts of the curve with similar gain. The power-of-two set is therefore a compact logarithmic covering of the relevant operating points, not a hardware constraint. It also makes the ablation easy to interpret: fixed-α baselines test each allocation regime, while the prefill-time selector tests whether the layer statistics choose the right regime without search.

D Experimental Setup

Shared configuration. Main-text experiments run on Ubuntu 22.04.5 LTS with an NVIDIA RTX PRO 6000 Blackwell GPU (96 GB). Unless otherwise specified, μ-ZERO uses channel-wise key grouping, token-wise value grouping, group size $G = 64$, and residual window $W = 128$. Appendix D.1 details benchmark datasets, prompting, decoding, and evaluation metrics; Appendix D.2 specifies baseline configurations and implementations. Protocols for extended experiments accompany their results in Appendices E–G.

D.1 Benchmark settings

Table 3 summarises the evaluation protocol for the two autoregressive generation benchmarks reported in Section 5.2. Both tasks use a fixed 4096-token input cap and a 256-token output cap. GSM8K uses greedy decoding under an 8-shot chain-of-thought prompt; HumanEval uses nucleus sampling with temperature $T = 0.8$ and $\mathrm{top_p} = 0.95$ over 20 samples per task, from which we estimate pass@1 and pass@5.

Table 3: Evaluation settings for GSM8K and HumanEval.

Benchmark	Task type	Eval units	Prompt / sampling	Input cap	Output cap	Metric
GSM8K	Math word problem	1,319 problems	8-shot CoT, greedy	4096	256	Exact Match
HumanEval	Code generation	164 tasks, 20 samples/task	$T = 0.8$, top_p$= 0.95$	4096	256	pass@1, pass@5

We evaluate on LongBench-v1 (Bai et al., 2024), a bilingual long-context benchmark covering 21 tasks in 6 groups with 4,750 total samples. The reported score is the mean of per-dataset scores, where each dataset-level metric is multiplied by 100 following the official scorer. Table 4 lists the full task inventory, including dataset name, sample count, mean and maximum context length (from the official `length` field), maximum generation budget (`max_gen`), and evaluation metric.

D.2 Baseline configurations and implementations

We report baselines using the bit-width labels exposed by their implementations, and use the same model weights, prompts, decoding settings, and residual-cache policy as in the main experiments unless otherwise noted. We use the official repositories for KIVI (`https://github.com/jy-yuan/KIVI`), TurboQuant (`https://github.com/0xSero/turboquant`), and KVTuner (`https://github.com/cmd2001/KVTuner`); HQQ uses the built-in HQQ quantized-cache backend in HuggingFace Transformers. Following KIVI, all quantizers keep the most recent 128 tokens in full precision as a residual window. Because several baselines include method-specific metadata or mixed-precision choices, these labels should be read as nominal operating points rather than identical physical memory footprints.

Table 4: LongBench-v1 task inventory. Mean/Max Length are from the official `length` field; Max Gen is the official per-dataset generation budget. Scores are multiplied by 100 by the official scorer. ZH denotes Chinese-language datasets evaluated with the corresponding localized metric.

Group	Dataset	# Samples	Mean Length	Max Length	Max Gen	Metric
Single-Doc QA	narrativeqa	200	18,405	36,418	128	QA F1
	qasper	200	3,619	8,460	128	QA F1
	multifieldqa_en	150	4,559	10,337	64	QA F1
	multifieldqa_zh	200	6,701	16,791	64	QA F1 (ZH)
Multi-Doc QA	hotpotqa	200	9,149	12,697	32	QA F1
	2wikimqa	200	4,885	11,958	32	QA F1
	musique	200	11,018	17,355	32	QA F1
	dureader	200	15,768	27,806	128	ROUGE-L (ZH)
Summarization	gov_report	200	8,169	40,508	512	ROUGE-L
	qmsum	200	10,546	24,585	512	ROUGE-L
	multi_news	200	2,114	10,933	512	ROUGE-L
	vcsum	200	15,147	42,073	512	ROUGE-L (ZH)
Few-Shot	trec	200	5,176	8,714	64	Classification Accuracy
	triviaqa	200	8,209	16,633	32	QA F1
	samsum	200	6,258	12,252	128	ROUGE-L
	lsht	200	22,333	44,221	64	Classification Accuracy (ZH)
Synthetic	passage_count	200	11,141	22,899	32	Count Accuracy
	passage_retrieval_en	200	9,288	11,516	32	Retrieval Accuracy
	passage_retrieval_zh	200	6,745	8,963	32	Retrieval Accuracy (ZH)
Code	lcc	500	1,235	10,029	64	Code Similarity
	repobench-p	500	4,206	18,754	64	Code Similarity
Total	21 datasets	**4,750**	–	–	–	Mean of per-dataset scores

KIVI. KIVI (Liu et al., 2024b) is the affine scalar-quantization baseline following the common KV cache layout: keys are quantized channel-wise, values token-wise, and the most recent cache tokens are kept in full precision as a residual window. We report KIVI at 2 and 4 bits.

HQQ. HQQ (Badri & Shaji, 2023) is evaluated through the Transformers HQQ quantized-cache backend. We use the official recommended cache axes, `axis-key`=1 and `axis-value`=1, for all HQQ runs, and report the 2-, 4-, and 8-bit settings. This baseline is calibration-free and quantizes the cache online, but it does not use the zero-anchored compander or layer-wise α selector introduced in μ-Zero.

TurboQuant. TurboQuant (Zandieh et al., 2025) represents the transform/sketch-domain family. In its released configuration, keys use the TurboQuant path: normalize, rotate, quantize rotated coordinates with precomputed scalar codebooks, and add a QJL residual sign sketch for inner-product correction. Values use per-group min–max affine quantization. We evaluate the three supported KV modes $K3V2$, $K4V2$, and $K4V4$, which correspond to the nominal labels 2.5, 3, and 4 bits per value, respectively. The implementation also gives the initial layers one extra key bit; in our accounting this increases the realized storage by about 0.74 bits per value relative to the nominal labels. We keep the nominal labels in the tables to match the baseline convention, and note that these settings are therefore slightly more expensive than their names suggest.

KVTuner. KVTuner (Li et al., 2025b) is an offline-searching baseline. Given a target budget, it searches for a per-layer key/value bit pair and then applies the selected mixed-precision allocation during inference. Its smallest candidate is $K2V2$, so a nominal 2-bit budget reduces to the KIVI-2 setting and is not reported separately. For the reported 4-bit KVTuner results, we use the allocation produced by offline search; the resulting average storage is model-dependent and ranges from 3.5938 to 3.8056 bits per value in our evaluated runs.

E Additional Quality Evaluation

E.1 Per-category LongBench results

Table 5 gives the full per-category breakdown of LongBench-v1 scores for Qwen3-0.6B. The six task groups follow the standard LongBench-v1 taxonomy above. The rightmost column repeats the 21-task average score from Table 2 for reference. **Bold** marks the best quantized result within each precision group and category.

Table 5: Per-category LongBench-v1 scores for **Qwen3-0.6B** under varying KV cache quantization methods and bit widths. **Bold** indicates the best result within each precision group per category. *KVTuner uses mixed-precision allocation.

Bits	Method	Single-Doc QA	Multi-Doc QA	Summarization	Few-Shot	Synthetic	Code	Average
1-bit	μ-Zero	**3.21**	**4.28**	**8.78**	**19.96**	**3.54**	**19.34**	**9.25**
2-bit	HQQ	2.88	3.50	8.88	19.44	4.19	18.12	8.93
	KIVI	8.02	7.64	14.23	49.06	8.11	**28.12**	18.88
	TurboQuant-2.5	4.41	5.27	6.08	20.12	2.95	18.20	8.99
	μ-Zero	**8.33**	**8.97**	**17.27**	**51.06**	**12.45**	27.79	**20.73**
3-bit	TurboQuant	5.46	5.77	5.59	22.96	6.14	18.44	10.21
	μ-Zero	**9.58**	**11.98**	**18.62**	**53.07**	**19.72**	**32.26**	**23.65**
4-bit	HQQ	6.75	7.50	12.04	40.75	9.65	24.85	16.51
	KIVI	9.33	12.47	18.70	53.15	18.96	32.84	23.67
	KVTuner*	**9.78**	12.55	**18.77**	**53.30**	18.73	**33.40**	23.84
	TurboQuant	6.02	5.99	6.01	24.08	7.10	20.02	10.94
	μ-Zero	9.42	**13.00**	18.57	53.23	**21.02**	33.14	**24.11**
8-bit	HQQ	**9.64**	12.96	**18.56**	**53.53**	**21.17**	33.07	**24.20**
	μ-Zero	9.52	**13.21**	18.40	53.38	21.10	**33.25**	**24.20**
Raw	BF16	9.50	13.08	18.52	53.46	**21.23**	33.13	**24.20**

Table 6: Per-category LongBench-v1 scores for **Qwen3-8B** under varying KV cache quantization methods and bit widths. **Bold** indicates the best result within each precision group per category. *KVTuner uses mixed-precision allocation. Dashes indicate runs not yet available.

Bits	Method	Single-Doc QA	Multi-Doc QA	Summarization	Few-Shot	Synthetic	Code	Score
1-bit	μ-Zero	5.40	5.06	14.16	36.74	17.42	31.59	**17.19**
2-bit	HQQ	6.65	6.93	9.92	38.74	29.17	33.55	19.22
	KIVI	13.97	12.35	22.39	61.15	**62.74**	**60.14**	35.62
	TurboQuant-2.5	12.89	12.21	22.20	58.44	59.03	55.41	33.85
	μ-Zero	**14.99**	**13.04**	**22.73**	**63.77**	60.32	58.85	**36.04**
3-bit	TurboQuant	**15.71**	**13.74**	**23.20**	61.10	**64.04**	61.83	36.70
	μ-Zero	15.25	13.57	21.99	**63.31**	63.06	**65.61**	**37.00**
4-bit	HQQ	15.53	13.77	22.11	63.17	64.90	65.44	37.33
	KIVI	15.41	13.89	22.02	**63.53**	66.13	**66.03**	37.61
	KVTuner*	15.12	13.56	22.18	63.06	64.71	64.10	37.05
	TurboQuant	**16.02**	**14.07**	**23.24**	62.21	64.13	62.70	37.14
	μ-Zero	15.56	13.82	21.89	**63.53**	**67.42**	65.61	**37.75**
8-bit	HQQ	15.61	13.99	21.93	**63.48**	**66.73**	65.67	37.69
	μ-Zero	**15.64**	**14.31**	**22.00**	63.42	66.50	**65.71**	**37.73**
Raw	BF16	15.68	13.94	22.17	63.51	66.46	65.58	37.70

Table 7: Per-category LongBench-v1 scores for **Llama-3.2-3B** under varying KV cache quantization methods and bit widths. **Bold** indicates the best result within each precision group per category. *KVTuner uses mixed-precision allocation. Dashes indicate runs not yet available.

Bits	Method	Single-Doc QA	Multi-Doc QA	Summarization	Few-Shot	Synthetic	Code	Score
1-bit	μ-Zero	**5.63**	**7.47**	**6.08**	**34.94**	**4.09**	**35.09**	**14.24**
2-bit	HQQ	7.18	7.62	6.82	36.91	3.56	28.55	14.38
	KIVI	18.53	**13.59**	**18.25**	58.91	3.87	**62.86**	27.36
	TurboQuant-2.5	18.79	13.33	16.54	58.10	3.78	58.04	26.40
	μ-Zero	**19.19**	13.30	18.23	**60.55**	**4.01**	60.20	**27.50**
3-bit	TurboQuant	**20.06**	**14.90**	20.58	59.66	**4.31**	64.44	28.70
	μ-Zero	19.31	14.45	**21.61**	**61.39**	4.13	**66.19**	**29.13**
4-bit	HQQ	19.97	14.31	20.83	60.98	3.55	65.40	28.85
	KIVI	19.66	14.51	**21.43**	60.12	4.23	66.53	28.98
	KVTuner*	**20.36**	**15.09**	20.02	60.10	3.63	66.28	28.84
	TurboQuant	19.90	14.25	21.04	59.79	**4.42**	65.88	28.80
	μ-Zero	19.55	14.56	20.95	**61.35**	3.94	**66.56**	**29.08**
8-bit	HQQ	19.75	14.06	21.03	61.02	4.11	66.59	29.17
	μ-Zero	**19.99**	**14.73**	**21.13**	**61.09**	**4.23**	**66.66**	**29.22**
Raw	BF16	20.02	14.70	20.99	61.06	4.23	66.70	29.20

Table 8: Per-category LongBench-v1 scores for **Llama-3.1-8B-Instruct** under varying KV cache quantization methods and bit widths. **Bold** indicates the best result within each precision group per category. *KVTuner uses mixed-precision allocation. Dashes indicate runs not yet available.

Bits	Method	Single-Doc QA	Multi-Doc QA	Summarization	Few-Shot	Synthetic	Code	Score
1-bit	μ-Zero	**6.46**	**8.82**	**9.03**	**34.54**	**23.14**	**28.25**	**17.20**
2-bit	HQQ	10.26	10.94	13.56	49.69	30.67	32.27	23.54
	KIVI	**22.27**	16.88	**26.29**	**63.16**	53.62	56.81	37.57
	TurboQuant-2.5	20.86	15.90	25.25	62.21	54.65	54.51	36.66
	μ-Zero	22.2	**18.05**	25.24	62.95	**60.22**	**58.3**	**38.62**
3-bit	TurboQuant	21.98	17.52	**25.38**	**63.63**	59.46	57.94	38.49
	μ-Zero	**22.5**	**18.62**	25.15	63.52	**59.72**	**59.66**	**38.94**
4-bit	HQQ	23.07	**18.68**	**25.47**	63.21	59.86	59.64	39.08
	KIVI	23.04	18.61	25.05	63.55	60.22	59.98	39.13
	KVTuner*	**23.27**	18.48	24.86	63.6	59.67	**60.3**	39.07
	TurboQuant	22.42	18.30	25.31	63.41	60.07	58.30	38.79
	μ-Zero	23.21	18.45	25.25	**63.66**	**60.33**	59.81	**39.19**
8-bit	HQQ	23	**18.55**	25.11	63.34	60.15	59.85	39.06
	μ-Zero	**23.11**	18.53	**25.25**	**63.62**	**60.75**	**60.03**	**39.25**
Raw	BF16	23.14	18.61	25.22	63.57	60.4	60.05	39.21

E.2 LONG-CONTEXT EVALUATION ON RULER

Protocol. We extend the LongBench evaluation in Section 5.3 with RULER (Hsieh et al., 2024), using its 13 official tasks with 100 examples per task at each context length. We evaluate Qwen3-8B and Llama-3.1-8B-Instruct at 4k, 16k, and 32k tokens with greedy decoding, and report the mean task score on a 0–100 scale. For each model and length, we compare BF16 with KIVI and μ-ZERO at 2 and 4 bits using the same prompts and decoding settings. Quantized configurations use group size $G = 64$ and residual window $W = 128$.

Table 9: Long-context quality on RULER. Each score averages 13 tasks with 100 examples per task and context length. Bold indicates the higher score within each matched-bit KIVI/μ-ZERO pair; it does not denote statistical significance.

Model	Context	BF16	KIVI-2	μ-Zero-2	KIVI-4	μ-Zero-4
Qwen3-8B	4k	82.61	64.45	**76.37**	79.85	**83.44**
	16k	84.92	61.45	**68.73**	83.81	**86.28**
	32k	82.42	59.22	**59.69**	81.20	**83.17**
Llama-3.1-8B-Instruct	4k	92.95	85.11	**86.62**	92.75	**93.06**
	16k	91.51	75.53	**80.97**	90.87	**91.70**
	32k	86.63	71.29	**74.21**	84.86	**85.92**

Results. Table 9 shows higher average scores for μ-ZERO than KIVI at both bit widths across all evaluated context lengths. At 2 bits, the gains range from 0.47 to 11.92 points on Qwen3-8B and from 1.51 to 5.44 points on Llama-3.1-8B-Instruct. The largest gain is on Qwen3-8B at 4k. Both 2-bit methods nevertheless lose substantial quality as context grows, particularly on Qwen3-8B, where their scores converge at 32k. Four-bit μ-ZERO remains near the BF16 reference across the tested lengths. These results extend the observed low-bit quality benefit to controlled long-input settings without implying that its magnitude grows with context length.

E.3 LONG-OUTPUT REASONING ON AIME'25

Protocol. AIME'25 complements RULER by testing extended reasoning generation. We evaluate Qwen3-8B in thinking mode and DeepSeek-R1-Distill-Llama-70B on all 30 problems, drawing four responses per problem for each of three seeds, for 360 responses per model and method. The generation budget is 16k tokens, rather than the 256-token output cap used for the GSM8K and HumanEval evaluations in Appendix D.1. We report pass@1 as a percentage, aggregated over the three seeds, with 95% confidence intervals. BF16, KIVI, and μ-ZERO share the prompts and sampling settings within each model; quantized runs use $G = 64$ and $W = 128$. Historical cache entries are quantized while the residual window remains in full precision throughout generation.

Table 10: AIME'25 long-output reasoning with a 16k-token generation budget. Results use 30 problems, four responses per problem, and three seeds for each method. DS-R1 denotes DeepSeek-R1-Distill-Llama-70B. Bold marks the higher point estimate within each matched-bit pair; the intervals are marginal 95% confidence intervals (CIs).

Model	Method	Pass@1 (%)	95% CI
Qwen3-8B (thinking)	BF16	52.22	[47.07, 57.33]
	KIVI-2	37.61	[32.76, 42.72]
	μ-Zero-2	**39.17**	[34.26, 44.30]
	KIVI-4	46.67	[41.58, 51.83]
	μ-Zero-4	**51.67**	[46.51, 56.78]
DS-R1-Distill-Llama-70B	BF16	50.00	[40.97, 59.03]
	KIVI-2	25.83	[21.69, 29.97]
	μ-Zero-2	**28.70**	[23.28, 35.64]
	KIVI-4	45.42	[29.56, 55.33]
	μ-Zero-4	**49.67**	[38.29, 61.28]

Results. In Table 10, μ-ZERO has higher point estimates than KIVI at both bit widths on both models. On Qwen3-8B, the improvements are 1.56 points at 2 bits and 5.00 points at 4 bits; the 4-bit score is 51.67 versus 52.22 for BF16. On the 70B model, the corresponding gains are 2.87 and 4.25 points, with 4-bit μ-ZERO scoring 49.67 versus 50.00 for BF16. Both 2-bit methods remain substantially below BF16. The reported intervals reflect considerable uncertainty on this 30-problem benchmark, so these comparisons describe point-estimate ordering rather than establish statistical superiority over KIVI or equivalence to BF16.

E.4 LARGER-MODEL EVALUATION: 70B GSM8K

Protocol. We evaluate DeepSeek-R1-Distill-Llama-70B, a GQA model based on Llama-3.3-70B-Instruct (DeepSeek-AI, 2025), following the GSM8K protocol in Appendix D.1. Quantized runs use $G = 64$ and $W = 128$. Table 11 reports exact match as a percentage.

Table 11: GSM8K reasoning on DeepSeek-R1-Distill-Llama-70B.

Method	BF16	KIVI-2	μ-Zero-2	KIVI-4	μ-Zero-4
GSM8K EM (%)	86.13	75.97	**78.62**	83.75	**85.60**

Results. μ-ZERO improves over the affine baseline by 2.65 percentage points at 2 bits and 1.85 points at 4 bits. The 4-bit result is 0.53 points below BF16, while the 2-bit setting retains a larger gap. Together with the completed AIME'25 evaluation in Appendix E.3, these results extend the observed quality benefit to a 70B model under both the standard GSM8K protocol and a long-output reasoning workload.

E.5 TRANSFER TO MQA AND MLA ARCHITECTURES

Protocol. We complement the grouped-query attention (GQA) evaluations with Gemma-2B-IT, which uses multi-query attention (MQA; one KV head), and DeepSeek-V2-Lite-Chat, which uses multi-head latent attention (MLA). For MLA, quantization operates on the native latent cache. We evaluate both models on all 503 LongBench-v2 examples with greedy decoding at 2 and 4 bits. HQQ serves as the affine baseline because the released KIVI kernels used in our evaluation do not support these two architectures. These compact models test architectural transfer.

Table 12: LongBench-v2 scores on MQA and MLA models (higher is better). All methods use the same 503 examples and greedy decoding. MLA quantization is applied to the native latent cache.

Model (attention)	BF16	HQQ-2	μ-Zero-2	HQQ-4	μ-Zero-4
Gemma-2B-IT (MQA)	17.1	16.7	**17.1**	17.0	**17.1**
DeepSeek-V2-Lite-Chat (MLA)	7.0	0.2	**6.4**	6.9	**7.0**

Task-level results. Table 12 shows that μ-ZERO attains the BF16 point score on Gemma at both bit widths and on DeepSeek-V2-Lite at 4 bits. The largest separation occurs at 2 bits on MLA: HQQ drops from the BF16 score of 7.0 to 0.2, whereas μ-ZERO retains 6.4. These results support transfer to the tested MQA and MLA models, although their low absolute benchmark scores limit the conclusions that can be drawn from task-level differences alone.

Attention-level diagnostic. We therefore also compare softmax KL divergence to BF16 on a held-out slice using the methodology of Figure 3. At 2 bits, the HQQ-to-μ-ZERO KL ratio is $4.6\times$ on Gemma and $7.8\times$ on DeepSeek-V2-Lite. This attention-level improvement is consistent with the ordering observed for GQA models in Figure 3, providing complementary evidence of transfer beyond downstream point scores.

F MEMORY AND RUNTIME EVALUATION

F.1 KV-CACHE STORAGE ACCOUNTING

Nominal code width does not include affine metadata, full-precision residual tokens, or implementation-specific state. For a scalar cache with code width b, group size G, two FP16 metadata values per group, and a full W-token residual window at cached length T, the effective storage is approximately

$$b_{\text{eff}}(T) = \left(1 - \frac{W}{T}\right)\left(b + \frac{32}{G}\right) + 16\frac{W}{T}. \tag{47}$$

This expression assumes complete groups and packed codes; exact byte accounting additionally depends on padding, actual residual occupancy, allocation capacity, and auxiliary state. With $G = 64$, $W = 128$, and $T = 4096$, it gives 2.922 and 4.859 bits per value at 2 and 4 bits. At $T = 32768$, the 4-bit estimate is 4.545 bits per value, or 78.08 GB for the batch/model geometry below; this is a projection, not a measured allocation.

Under matched grouping and residual occupancy, μ-ZERO and KIVI have the same packed-code and per-group metadata budget. The zero anchor is derived from (m, s), and α is retained only as layer-level state. The inverse-compander lookup tables are constructed transiently inside the fused kernels and are not persistent cache tensors.

F.2 MEASURED CACHE FOOTPRINTS AND QUALITY–STORAGE COMPARISON

Protocol and units. We evaluate Llama-3.1-8B-Instruct with batch size 64 after 4096 prefill tokens and eight decode-token cache updates. The model has 32 layers, 8 KV heads per layer, and head dimension 128. Scalar configurations use $G = 64$, residual-window parameter $W = 128$, and FP16 scale/offset metadata. TurboQuant uses its released $G = 32$ grouping, with cache-owned transform/sketch state included. The footprint probe uses FP16 tensors for compatibility with the KIVI kernels; FP16 and BF16 both require 16 bits per cached value.

Let M be the counted cache-tensor bytes and $N{=}2LBTH_{\mathrm{KV}}d$ the number of logical key and value elements. We report $8M/N$ bits per value, $M/10^9$ GB, and compression M_{16}/M relative to the full-precision cache at the same length. The full-precision reference occupies 34.43 GB at $T{=}4104$. These quantities describe cache tensor storage, excluding model weights, temporary attention workspaces, and allocator reservation; they differ from the peak reserved GPU memory in Appendix F.3.

Table 13: Cache footprints on Llama-3.1-8B-Instruct at batch 64 and 4104 cached tokens. LongBench-v1 scores are taken from Table 2, whose full-precision quality reference is BF16. KVTuner's measured row excludes retained dequantized copies; Table 14 distinguishes this storage-only scope from its full runtime state and packed-equivalent estimate.

Method	Nominal bits	Measured bits/value	Cache footprint (GB)	Compression	LongBench
FP16/BF16	16	16.00	34.43	1.00×	39.21
HQQ-2	2	2.53	5.44	6.33×	23.54
KIVI-2	2	2.93	6.31	5.45×	37.57
μ-Zero-2	2	2.93	6.31	5.45×	38.62
TurboQuant-2.5	2.5	3.59	7.72	4.46×	36.66
HQQ-4	4	4.52	9.73	3.54×	39.08
KIVI-4	4	4.87	10.48	3.29×	39.13
μ-Zero-4	4	4.87	10.48	3.29×	39.19
TurboQuant-4	4	5.53	11.89	2.90×	38.79
KVTuner (4-bit)†	4	8.52	18.32	1.88×	39.07

KVTuner accounting. The released vanilla backend of KVTuner stores codes as `int8`, and its runtime object also retains dequantized keys and values. Consequently, packed-equivalent accounting, measured storage excluding those dense copies, and the full retained tensor state answer different questions (Table 14). The nominal 4-bit label denotes the search budget; the selected per-layer allocation is mixed precision.

Table 14: Accounting scopes for the same KVTuner 4-bit preset. The packed-equivalent row is an estimate; the remaining rows describe measured tensor storage. Compression uses the same 16-bit reference.

Accounting scope	Bits/value	Cache footprint (GB)	Compression
Packed-equivalent estimate	4.47	9.61	$3.58\times$
Measured, excluding dense copies[†]	8.52	18.32	$1.88\times$
Measured, including dense copies	24.52	52.75	$0.65\times$

Interpretation. The measured μ-ZERO footprint closely tracks packed-cache accounting: at 4 bits, 4.87 measured bits per value is close to the 4.86-bit estimate at 4096 tokens. The residual tail and implementation details explain why the effective rate exceeds the nominal code width. The quality gains therefore do not require extra per-group metadata. At 2 bits, μ-ZERO uses 6.31 GB versus TurboQuant-2.5's 7.72 GB while achieving LongBench scores of 38.62 versus 36.66; the corresponding GSM8K scores in Table 1 are 0.7111 versus 0.5315.

F.3 DECODING THROUGHPUT AND PEAK GPU MEMORY

Figure 4 reports end-to-end decoding throughput and peak reserved memory. The left panels sweep batch size at context length 4096, and the right panels sweep context length at batch size 64. Each run generates 512 new tokens. The implementation is built on the HuggingFace Transformers codebase.

Batch scaling. At context length 4096, BF16 is feasible through batch 122 and encounters OOM at batch 138. μ-ZERO-4bit reaches batch 288, a $2.4\times$ increase over the largest feasible BF16 batch, and μ-ZERO-2bit reaches at least batch 400, a $3.3\times$ lower bound. On the common feasible batches 32–122, 4-bit μ-ZERO is 1.03–$1.53\times$ faster than BF16 while using about half of the reserved memory on average. The 2-bit setting is 1.38–$2.38\times$ faster while using about 39% of BF16 reserved memory. These results show that the extra reconstruction arithmetic is outweighed by reduced KV cache traffic in the memory-bound regimes targeted by cache compression.

Context scaling. At batch size 64, BF16 succeeds up to context length 6144 and runs out of memory at 8192. In contrast, μ-ZERO-4bit reaches length 20480 before its first OOM at 22528, a $3.3\times$ increase over the largest feasible BF16 context, while μ-ZERO-2bit reaches at least 28672, a $4.7\times$ lower bound. At the shared feasible lengths 2048, 4096, and 6144, 4-bit μ-ZERO uses 52%, 41%, and 41% of BF16 reserved memory and achieves $0.99\times$, $1.06\times$, and $1.11\times$ BF16 throughput. The 2-bit setting reduces reserved memory to 45%, 33%, and 31% of BF16 and improves throughput to $1.30\times$, $1.56\times$, and $1.70\times$. At the shortest context length, 512, quantized decoding is slower, indicating that fixed quantization overhead dominates before cache memory traffic becomes large.

The current implementation quantizes the KV cache as an explicit cache-management step. Further fusing quantization with the preceding projection and cache-update operations would reduce intermediate traffic and launch overhead, and may substantially increase the observed speedups; we leave this system optimization to future work.

F.4 SYSTEM-LEVEL COMPARISON WITH OFFICIAL KIVI

Protocol. We compare the deployed implementations on Llama-3.1-8B-Instruct using an NVIDIA RTX PRO 6000 Blackwell Server Edition GPU, FP16 runtime, and 512 generated tokens. Table 15 reports decode-phase throughput and peak reserved GPU memory, normalized to the full-precision run at each setting. These measurements include model and runtime allocations, unlike the cache-only footprints in Appendix F.2. The full-precision and μ-ZERO paths use a decode-only harness with a static decode mask, 128-token chunked prefill, and memory-peak reset before decoding. KIVI uses its official fused kernels through a manual prefill/decode harness.

Table 15: Decode throughput and peak reserved GPU memory for the additional KIVI comparison. Each setting generates 512 tokens per sequence. Speedup and memory reduction are relative to that setting's FP16 reference; higher is better for both.

Setting	Method	Throughput (tokens/s)	Speedup vs. FP16	Peak reserved memory (GB)	Memory reduction (%)
Context 8k batch 16	FP16	310.99	1.000×	38.13	0.0
	KIVI-4	361.81	1.163×	28.45	25.4
	μ-Zero-4	467.68	1.504×	23.35	38.8
	KIVI-2	424.71	1.366×	22.54	40.9
	μ-Zero-2	440.04	1.415×	20.54	46.1
Context 32k batch 16	FP16	101.92	1.000×	101.19	0.0
	KIVI-4	143.54	1.408×	62.51	38.2
	μ-Zero-4	209.60	2.057×	40.50	60.0
	KIVI-2	186.57	1.831×	41.01	59.5
	μ-Zero-2	191.34	1.877×	29.66	70.7
Context 4k batch 120	FP16	726.97	1.000×	101.26	0.0
	KIVI-4	998.56	1.374×	64.13	36.7
	μ-Zero-4	1257.56	1.730×	48.39	52.2
	KIVI-2	1269.57	1.746×	43.35	57.2
	μ-Zero-2	1352.70	1.861×	37.34	63.1

Results. Across the three measured settings, μ-ZERO achieves 1.26–1.46× KIVI's throughput at 4 bits and 1.03–1.07× at 2 bits. Peak reserved memory is lower by 17.9%–35.2% and 8.9%–27.7%, respectively, relative to matched-bit KIVI. At context 32k and batch 16, μ-ZERO-4 reaches 209.60 tokens/s versus KIVI-4's 143.54, with 60.0% versus 38.2% memory reduction relative to FP16. These observations support the practical efficiency of the deployed fused path. They do not establish that cache storage is byte-identical in these runs or that layout-conversion temporaries alone explain the difference; attributing the gain requires matched profiling of the decode paths.

F.5 KERNEL BREAKDOWN AND INVERSE-COMPANDER IMPLEMENTATION

Protocol. We measure QK and AV kernels on the same GPU at batch 64, context 4096, 32 query heads, 8 KV heads, head dimension 128, 4-bit codes, and G=64. The deployed μ-ZERO path constructs a 16-entry reconstruction lookup table (LUT) per group and uses indexed loads, avoiding per-element exponential evaluation. Table 16 compares the deployed KIVI and μ-ZERO implementations and includes per-element exponential reconstruction under the same optimized μ-ZERO kernel schedule. QK+AV is the sum of the two kernel timings, not a full-model decode step.

Table 16: Attention-kernel latency and reconstruction comparison at batch 64/context 4096 with 4-bit codes. Latencies are in μs. The LUT row uses the deployed-path measurement; its QK+AV latency differs by only 0.06% from the separate reconstruction-ablation measurement.

Path	QK	AV	QK+AV
Official KIVI affine	775.0	451.0	1226.0
μ-Zero LUT	634.2	520.7	1154.9
μ-Zero per-element exponential	809.0	604.8	1413.8

Results. The deployed LUT path has 5.8% lower combined QK+AV latency than official KIVI at this shape (1154.9 versus 1226.0 μs), equivalent to a 1.06× kernel-throughput ratio. Its lower QK latency outweighs the higher AV latency. Within the optimized μ-ZERO schedule, LUT reconstruction has approximately 18.3% lower combined latency than per-element exponentials (1154.9 versus 1413.8 μs in Table 16). These results support the LUT over per-element exponential implementation and show competitive combined kernel latency against official KIVI at the measured shape.

G Additional Ablations

G.1 Sensitivity to group size

Protocol. Following the GSM8K protocol in Appendix D.1, we evaluate Llama-3.1-8B and Qwen3-8B at 2 bits with $G \in \{32, 64, 128\}$ and $W = 128$, reusing the main evaluation at $G = 64$. Paired 95% confidence intervals (CIs) for $\Delta = \mu$-Zero$-$KIVI use 10,000 bootstrap resamples of the same test examples.

Table 17: Group-size sensitivity at 2 bits on GSM8K. EM and paired differences are proportions; differences are computed before rounding. The packed rate is $2+32/G$ bits per quantized value, including two FP16 metadata values per group but excluding residual tokens, padding, and auxiliary state.

Model	G	Packed rate (bits/value)	KIVI	μ-Zero	Δ [95% CI]
Llama-3.1-8B	32	3.00	0.7218	0.7240	$+0.0023$ $[-0.0190, +0.0243]$
	64	2.50	0.6861	0.7111	$+0.0250$ $[+0.0000, +0.0500]$
	128	2.25	0.6482	0.6490	$+0.0008$ $[-0.0258, +0.0273]$
Qwen3-8B	32	3.00	0.8302	0.8688	$+0.0387$ $[+0.0205, +0.0569]$
	64	2.50	0.8120	0.8575	$+0.0455$ $[+0.0250, +0.0660]$
	128	2.25	0.7172	0.8196	$+0.1024$ $[+0.0773, +0.1266]$

Accuracy sensitivity. μ-Zero has a higher point estimate than KIVI at every tested group size (Table 17). On Qwen3-8B, its advantage increases from 3.87 to 10.24 percentage points as G grows from 32 to 128, with paired intervals excluding zero at all three sizes. Over this range, KIVI loses 11.30 points while μ-Zero loses 4.92 points. On Llama-3.1-8B, the methods degrade similarly (7.36 and 7.50 points, respectively); the largest observed gap is at $G = 64$, and the reported intervals include or touch zero. Thus, robustness to coarser grouping is model-dependent.

Accuracy–storage trade-off. With two FP16 metadata values per group, metadata costs $32/G$ bits per quantized value. On Qwen3-8B, μ-Zero at $G = 128$ has a higher EM point estimate than KIVI at $G = 64$ (0.8196 versus 0.8120), at a lower packed code-plus-metadata rate (2.25 versus 2.50 bits per value). This comparison concerns the quantized portion of the cache; total storage additionally includes the residual window and implementation state described in Appendix F.1. The within-group confidence intervals do not establish significance for this cross-group comparison.

G.2 Sensitivity to grouping layout

The default layout groups keys channel-wise and values token-wise. Figure 3(c) compares this configuration with token-wise key grouping using softmax KL to BF16. Token-wise keys increase attention error for both uniform quantization and μ-Zero, but μ-Zero reduces the penalty relative to uniform quantization. This operator-level ablation complements the downstream group-size sweep: the benefit persists under the alternative key layout, while channel-wise key grouping remains the default. Architecture-level transfer to GQA, MQA, and native-latent MLA caches is evaluated separately in Appendix E.5 and the preceding quality evaluations.

H Broader Impacts

μ-Zero is an inference-time KV cache quantization method for reducing the memory traffic and footprint of long-context language-model serving. Its primary positive impact is improved efficiency: lower memory use and higher feasible batch sizes or context lengths can reduce serving cost, make long-context inference more accessible on fixed hardware, and lower the energy required per generated token in memory-bound regimes. Because the method does not train or release a new model and does not alter model objectives, its direct societal effects are mediated through the language models and applications into which it is integrated.

The same efficiency improvements can also amplify existing risks of deployed language models. Cheaper and longer-context inference may make it easier to scale both beneficial uses, such as document analysis and code assistance, and harmful uses, such as automated persuasion, spam, surveillance workflows, or generation of misleading content. Quantization can also change model outputs in low-bit settings; although our evaluations measure task accuracy and attention error, application-specific safety or fairness behavior should be revalidated before deployment in high-stakes settings. We therefore view μ-ZERO as a systems primitive that should be combined with the same access controls, monitoring, safety evaluations, and usage policies appropriate for the underlying model and deployment context.