CONSISTENCY OVER ACCURACY: OUTLIER-BYPASS LAYER-ADAPTIVE QUANTIZATION FOR CODEC-BASED KV CACHE COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reusing key–value (KV) caches can eliminate repeated prefill computation in long-context large language model serving, but practical reuse requires the cache to be compressed without significantly altering the model response. Practical lossy KV cache compression raises two challenges: how to measure reuse fidelity and how to mitigate quantization distortion. For the former, aggregate task accuracy is insufficient to reliably characterize answer consistency before and after compression, as compression-induced perturbations may spuriously turn an incorrect answer into a correct one. To address this issue, we propose replacing accuracy with answer consistency as the metric for evaluating compression-induced distortion. For the latter, existing codec-based systems compress the dense representation effectively, yet sparse outliers are not adequately handled. To be specific, a few extreme values can consequently dominate the shared group scale, reducing the effective resolution of normal values and compromising subsequent related allocations. We refer to this effect as outlier-scale pollution. Accordingly, we propose outlier-bypass layer-adaptive quantization. The method detects group-wise extremes by gap prominence, stores them exactly in a separate sparse bypass. Then, the remaining values are quantized using layer-specific level where the corresponding thresholds are customized for each model through offline cumulative-distribution fitting (CDF) calibration. By the calibrated CDF, we derive two presets corresponding to distinct consistency–compression operating points. Across five models and three long-context benchmarks, the high-fidelity preset (Ours-HF) attains 82.33% consistency at $7.41\times$ compression, simultaneously outperforming KVCodec and CacheGen by $+1.96$ percentage points in consistency with $+5.26\%$ and $+26.24\%$ in compression ratio, respectively. For workloads that prioritize storage efficiency, the high-compression preset (Ours-HC) further increases the compression ratio to $8.35\times$, a gain of 18.61% over KVCodec and 42.25% over CacheGen, at a modest consistency cost of 1.45 percentage points.

1 INTRODUCTION

Modern long-context large language model (LLM) applications increasingly reuse shared prefixes and precomputed document contexts across requests, thereby avoiding repeated prefill computation in multi-call and retrieval-augmented workloads (Zheng et al., 2024; Yao et al., 2025). However, the FP16 key–value (KV) cache grows linearly with context length. Extending cache reuse beyond GPU memory to distributed memory and storage therefore introduces substantial storage and communication overhead (Qin et al., 2025; Chen et al., 2025). Effective compression is consequently essential for practical KV cache reuse. Recent systems reduce this overhead through hierarchical entropy coding or codec-friendly tensor layouts (Liu et al., 2024a; Mi et al., 2026; Xu et al., 2024; 2025). Although these systems produce compact bitstreams, their compression quality is generally assessed using aggregate task accuracy scores. Such scores do not directly measure reuse fidelity because compression-induced answer changes can occur in both directions and cancel at the benchmark level. A compressed cache may preserve aggregate accuracy while substantially changing the model responses on individual inputs. For lossy KV cache reuse, fidelity should instead reflect agreement between compressed inference and the FP16 reference output on the same input.

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.

Quantization is required before entropy or video coding, but sparse extreme values make this frontend particularly sensitive. Prior KV cache quantization methods address extreme values through asymmetric quantization, salient-channel preservation, rotation, or residual correction (Liu et al., 2024b; Hooper et al., 2024; Su et al., 2025; Kang et al., 2024). We identify a complementary failure mode in which a few input-dependent extremes determine the shared group scale. The resulting outlier-scale pollution reduces the effective resolution of normal values and contaminates the scale statistics used for layer-wise precision allocation. When the allocation signals are derived from contaminated scales, they may conflate sparse scale inflation with intrinsic quantization sensitivity.

We address these problems with outlier-bypass layer-adaptive quantization for codec-based KV cache compression. Group outlier bypass detects sparse extremes from gaps in the upper order statistics, stores the detected values exactly in a sparse FP16 bypass, and quantizes the remaining values using a scale determined by the bulk distribution. Layer-wise level-bound allocation then derives layer statistics from the cleaned group scales and assigns precision along the coding axis. Separate policies account for the different characteristics of Key and Value. Offline cumulative-distribution fitting maps preset tier quotas to thresholds adapted to each model and benchmark, avoiding manual threshold selection for individual model–benchmark pairs. The resulting presets provide controllable consistency–compression operating points within a unified quantization framework.

Contributions.

- **Consistency over accuracy as the primary fidelity metric for lossy KV cache reuse.** We show that aggregate task accuracy can conceal bidirectional answer changes and evaluate compressed inference by its agreement with the FP16 reference output on the same input.
- **Group outlier bypass based on gap prominence.** We detect sparse group-wise extremes from upper-order-statistic gaps and preserve them exactly outside the dense quantization path, preventing them from degrading the resolution of normal values.
- **Codec-aligned layer-wise level-bound allocation.** To accommodate the video codec backend, we allocate layer precision using cleaned scale distributions and calibrate the corresponding thresholds to each model and benchmark through offline CDF fitting.

2 Related work

KV cache compression. Existing approaches reduce KV cache cost either by shortening the retained sequence or by compressing its numerical representation. Token-selection methods preserve salient entries or fixed attention windows, reducing the sequence dimension before subsequent processing (Zhang et al., 2023; Li et al., 2024; Xiao et al., 2024). Codec-based systems instead retain the full cache. CacheGen combines hierarchical quantization with arithmetic coding for cache streaming (Liu et al., 2024a). KVCodec reorganizes KV tensors for GPU-native video encoding, while VcLLM/LLM.265 extend standard video codecs to general tensor representations (Mi et al., 2026; Xu et al., 2024; 2025). These systems establish efficient coding and transport backends but expose quantization as a distinct source of distortion. Our work is complementary to sequence reduction and improves the quantization frontend that supplies the codec backend.

Outlier-aware KV quantization. KV quantization reduces the numerical footprint before coding. KIVI exploits the distinct distributions of Key and Value through per-channel Key and per-token Value quantization, while Atom jointly optimizes low-bit quantization and serving kernels (Liu et al., 2024b; Zhao et al., 2024b). Outlier-aware weight and activation quantization isolates or redistributes extreme channels through matrix decomposition or smoothing (Dettmers et al., 2022; Xiao et al., 2023), but its objectives and granularity differ from per-input KV group quantization. KV-specific methods employ several strategies. KVQuant preserves persistently salient channels and applies non-uniform quantization to the remaining values (Hooper et al., 2024). RotateKV redistributes Key outlier energy through rotations (Su et al., 2025), whereas GEAR compensates post-quantization error with sparse and low-rank residuals (Kang et al., 2024). These approaches operate through channel-level preservation, distribution transformation, or reconstruction correction. Our method instead targets scale estimation within each input-dependent quantization group. It removes detected extremes before estimating the dense scale and restores them exactly after decoding, preventing sparse values from reducing the effective resolution of the remaining distribution.

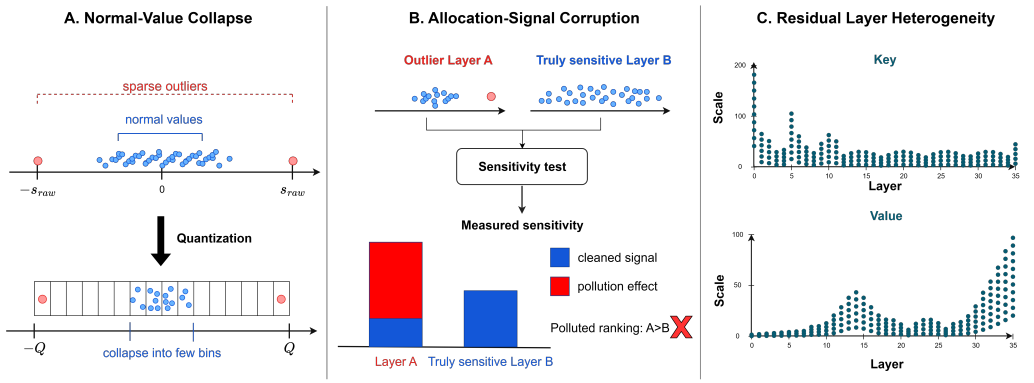

Figure 1: Motivation for outlier-bypass layer-adaptive quantization. **(A)** Normal-value collapse. **(B)** Allocation-signal corruption. **(C)** Residual layer heterogeneity.

Precision allocation. Mixed-precision methods allocate additional precision according to token importance or diversity (Yang et al., 2024; Zhang et al., 2026; Liu et al., 2026b; He et al., 2024). LampQ and LLM-PQ incorporate sensitivity or deployment cost in broader mixed-precision settings (Kim et al., 2026; Zhao et al., 2024a). CacheGen instead follows a fixed depth prior. These strategies derive their decisions from token importance, quantization sensitivity, system cost, or layer position. When range-based signals are used, sparse extremes can inflate the observed range and alter the resulting layer order. We derive codec-aligned layer bounds from group-scale distributions after outlier removal.

Evaluation of lossy KV cache reuse. Prior KV compression studies primarily report aggregate task accuracy, perplexity, or generation quality (Liu et al., 2024a; Mi et al., 2026; Liu et al., 2024b; Hooper et al., 2024). These metrics characterize benchmark-level utility but do not directly reveal whether compressed inference reproduces the FP16 output on each input. We therefore use consistency as the primary fidelity objective for lossy cache reuse and analyse its distinction from aggregate task accuracy scores.

3 Motivation

3.1 Consistency as a reuse-fidelity objective

In KV cache reuse, the compressed cache replaces the FP16 cache to avoid repeated prefill while preserving the model behavior associated with the reused context. The relevant fidelity objective is therefore agreement between compressed-cache inference and the FP16 reference on the same input. Consistency measures this sample-level preservation directly, whereas aggregate task accuracy reports only the net utility over a benchmark. Compression-induced changes can improve some predictions and degrade others, allowing their effects to cancel. For Qwen3-8B on LongBench-v2, uniform 4-bit quantization improves prediction correctness on 9.5% of samples and degrades it on 10.0%, while changing aggregate task accuracy by only -0.5 percentage points (pp). Therefore, we use consistency as the primary measure of reuse fidelity, since aggregate accuracy cannot isolate compression-induced output divergence.

3.2 Outlier-scale pollution and layer heterogeneity

Impact 1 (normal-value collapse). When a sparse extreme determines the group scale, the quantization range expands around that value and compresses normal values into a small number of bins (Fig. 1A). Shuffling outlier positions increases cross-entropy by 13.6 for Key but only 0.04 for Value, confirming a strong positional asymmetry. Zeroing or clipping Key outliers still increases cross-entropy by 3.2 or 0.6. These results indicate that neither suppression nor truncation preserves the reference behavior, motivating exact FP16 bypass and the component-specific policies.

Impact 2 (allocation-signal corruption). Layer-wise allocation requires a reliable ordering of quantization difficulty, whether it is estimated from scale statistics or approximated by a structural

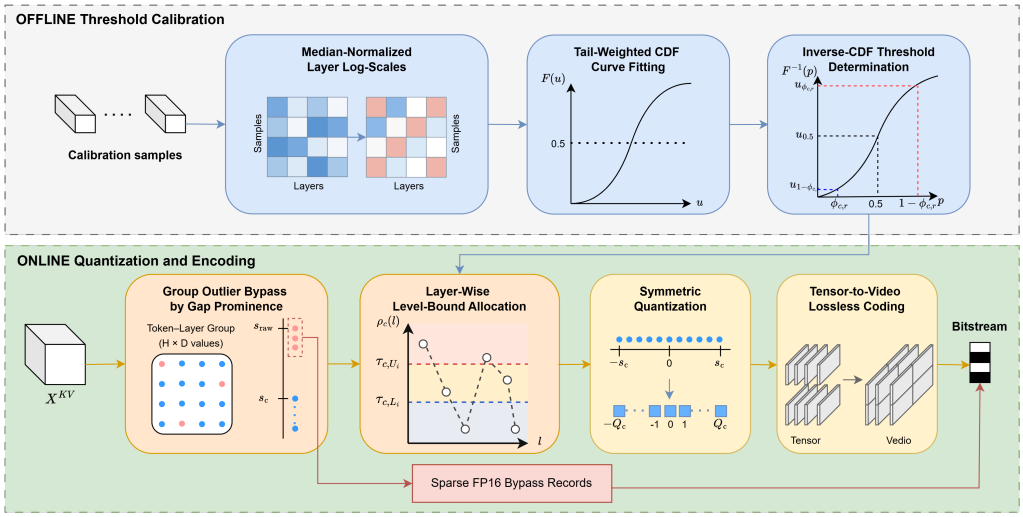

Figure 2: Overview of the proposed quantization pipeline. The offline stage maps preset tier quotas to model- and benchmark-specific ratio thresholds. During online encoding, sparse group outliers enter an exact FP16 bypass before the cleaned scales guide layer-wise level-bound allocation. The dense values are quantized and encoded by the video backend, while the bypass records are included as side information for exact restoration.

prior (Liu et al., 2024a). Outlier-inflated scales can cause a contaminated layer to rank above a genuinely difficult layer, as illustrated in Fig. 1B. After group outlier bypass, sensitivity decreases for 75.8% of layer–configuration pairs, the top-20% sensitive-layer sets before and after bypass have a Jaccard overlap of only 0.739, and the maximum rank shift is 16 positions. Precision allocation should therefore be derived from the cleaned distribution.

Residual layer heterogeneity. Group outlier bypass restores each group scale to its bulk range but does not eliminate genuine layer heterogeneity (Fig. 1C). Some layers retain broadly distributed large values and therefore require wider level bounds, whereas sparse scale inflation is better handled by exact bypass.

4 Outlier-Bypass Layer-Adaptive Quantization

4.1 Problem Formulation and Method Overview

Let $X^{KV} = (X^K, X^V)$ denote the KV cache, where $X^c \in \mathbb{R}^{T\times L\times H\times D}$ for $c \in \{K, V\}$. The four axes correspond to tokens, layers, attention heads, and head dimensions. Our objective is to reduce the complete coded representation while preserving the inference behavior of the original FP16 cache. Figure 2 separates threshold calibration from per-cache encoding. The offline stage maps preset tier quotas to model- and benchmark-specific thresholds (Sec. 4.4). During online encoding, group outlier bypass removes sparse extremes before scale estimation (Sec. 4.2), and the resulting cleaned scales determine codec-aligned layer bounds (Sec. 4.3). This ordering prevents outlier-contaminated scales from guiding precision allocation. The dense symbols are passed to the KVCodec video backend (Mi et al., 2026), while bypassed values and their indices are stored as sparse side information and restored exactly during reconstruction.

4.2 Group Outlier Bypass by Gap Prominence

For a fixed component $c \in \{K, V\}$, a group $g = (t, l)$ contains the $N = H \times D$ values $x_{g,i}$ from one token–layer slice and shares a common scale $s_c(g) \equiv s_c(t, l)$. The token–layer granularity isolates each group from other layers and distant token positions, while retaining enough elements along the head and head-dimension axes to expose a sparse order-statistic gap.

Conventional symmetric group quantization sets the scale to the group maximum:

$$s_{\mathrm{raw},c}(g) = \max_i |x_{g,i}|. \tag{1}$$

Sparse extremes inflate this scale and reduce the effective resolution of the remaining values. Unlike a fixed percentile rule, we detect a distinct sparse tail from gaps between upper order statistics. Let $v_1 \geq \cdots \geq v_{K+1}$ be the top-$(K+1)$ absolute values in g. The implementation uses $K = 8$, which yields eight adjacent ratios $\gamma_i = v_i / v_{i+1}$. We identify the largest ratio and normalize it by the mean of the remaining ratios:

$$i^* = \arg\max_{1 \leq i \leq K} \gamma_i, \qquad \gamma_g^* = \gamma_{i^*}, \qquad \pi_g = \frac{\gamma_g^*}{(K-1)^{-1} \sum_{j \neq i^*} \gamma_j}, \qquad m_g = i^*. \tag{2}$$

Here, $m_g = i^*$ is the number of bypassed values. The prominence π_g measures the separation of this breakpoint relative to the other gaps in the same group. Because both γ_i and π_g are ratios, the decision is invariant to a common multiplicative rescaling of the group. The detector therefore responds to a discontinuity in the upper tail rather than to the absolute magnitude of the layer.

A group is selected only if

$$1 \leq m_g \leq 3, \qquad \gamma_g^* \geq 1.5, \qquad \pi_g \geq \pi_{\min}^c, \tag{3}$$

where $\pi_{\min}^K = 2.5$ and $\pi_{\min}^V = 3.0$. The cap $m_g \leq 3$ restricts the bypass to sparse events and bounds its metadata cost. The component-specific prominence thresholds reflect the greater sensitivity to Key perturbations. The threshold asymmetry consequently places a stricter storage criterion on Value without changing the detection rule.

For a selected group, the dense quantization scale is set to the first value below the breakpoint. Otherwise, the original maximum is retained.

$$s_c(g) = \begin{cases} v_{i^*+1}, & \text{if } g \text{ satisfies the split criteria,} \\ s_{\mathrm{raw},c}(g), & \text{otherwise.} \end{cases} \tag{4}$$

The m_g values above the breakpoint and their local indices are stored in FP16 and removed from the dense path. This operation preserves the extremes exactly while preventing them from determining the quantization step of the remaining $N - m_g$ values.

4.3 Layer-wise level-bound allocation

Layer-wise level-bound allocation assigns a single level bound $Q_c(l)$ to all groups within the same layer, ensuring that every token–layer group in layer l uses the same quantization granularity. This per-layer uniformity is compatible with the KVCodec backend, which processes each layer as an independent coding unit.

After group outlier bypass, genuine differences in the retained layer scales remain. We summarize them by

$$S_c(l) = \max_t s_c(t, l), \tag{5}$$

where $S_c(l)$ is the largest cleaned group scale in layer l and serves as the signal for level-bound allocation.

Let $\widetilde{S}_c$ be the median of $\{S_c(l)\}$. We measure the upper- and lower-tail deviations of each layer as

$$\rho_c^+(l) = \frac{S_c(l)}{\widetilde{S}_c}, \qquad \rho_c^-(l) = \frac{\widetilde{S}_c}{S_c(l)}. \tag{6}$$

A layer enters an upper or lower tier when the corresponding ratio exceeds its calibrated threshold and remains in the centre tier otherwise. Median normalization makes this decision invariant to a common rescaling of the layer statistics. The six-tier, component-specific level-bound policies are given in Appendix A.1.

For each component and preset, $\phi_{c,r}$ specifies an upper bound on the fraction of layers assigned to non-centre tier r. Using the calibrated ratio threshold $\tau_{c,r}$, each layer is provisionally assigned to the outermost tier for which it qualifies. Tiers are then processed from outermost to innermost and capped at $\lceil L\phi_{c,r} \rceil$ layers. If a tier exceeds its capacity, the least extreme layers, measured by proximity to the median scale, are demoted to the adjacent inner tier. A quota may remain unfilled when fewer layers satisfy its threshold.

Key–Value asymmetry. Key and Value use the same tier-assignment rule but different level-bound ladders and quotas. Key perturbations alter the attention logits through $QK^\top$ and can redistribute attention weights across cached positions, whereas Value perturbations directly affect the vectors aggregated under those weights. As shown in Table 5, Key disables the two lowest tiers (L_2 and L_3) while retaining the highest tier U_2. Value permits lower level bounds but disables U_2, providing a more compression-oriented policy.

Given the layer-specific level bound $Q_c(l)$, dense values are quantized symmetrically as

$$q_{g,i} = \mathrm{clip}\left(\mathrm{round}\left(x_{g,i}\frac{Q_c(l)}{s_c(g)}\right), -Q_c(l), Q_c(l)\right). \tag{7}$$

The signed symbols are shifted to $u_{g,i} = q_{g,i} + Q_c(l)$ for uint8 storage and reconstructed by $\hat{x}_{g,i} = (u_{g,i} - Q_c(l))s_c(g)/Q_c(l)$. Finally, the sparse FP16 values overwrite their reconstructed positions. For a non-bypassed value that is not clipped, the rounding error is bounded by $s_c(g)/(2Q_c(l))$. Group outlier bypass reduces this bound by decreasing $s_c(g)$. Layer-wise allocation controls the second factor by increasing $Q_c(l)$ for upper tiers and decreasing it for lower tiers. The two mechanisms therefore separate sparse-extreme preservation from the layer-wise rate–distortion trade-off.

4.4 CDF-CALIBRATED THRESHOLD DETERMINATION

Using fixed ratio thresholds for the layer-wise tier assignment described above would implicitly assume that normalized layer-scale distributions are comparable across models and benchmarks. This assumption is generally invalid because the distribution of layer scales varies across model architectures and KV components. Rather than specifying the thresholds directly, we define preset tier quotas that represent the target fractions of layers in the upper and lower distribution tails, and derive the corresponding thresholds from fitted cumulative distribution functions.

Each benchmark contributes 10 fixed calibration samples, all of which are excluded from evaluation. Let $\mathcal{T}_s$ denote the fixed set of token positions selected from calibration sample s. The raw calibration statistic of component c at layer l is defined as

$$M_{s,c,l} = \max_{t\in\mathcal{T}_s,h,d} |X^c_{s,t,l,h,d}|, \qquad u_{s,c,l} = \log \frac{M_{s,c,l}}{\mathrm{median}_{l'}(M_{s,c,l'})}. \tag{8}$$

Here $M_{s,c,l}$ is the maximum magnitude over the sampled token–layer groups. Median normalization removes sequence-level scale differences, while the logarithm places the two tails on a common additive axis. This raw statistic is used only to calibrate relative thresholds for each model and benchmark. Online tier assignment applies these thresholds to the cleaned statistic $S_c(l)$ defined in Sec. 4.3, while the tier quotas bound the number of layers assigned to each non-centre tier.

For a fixed model–benchmark pair and component c, let U_c denote a normalized log-scale drawn from the pooled calibration observations $\{u_{s,c,l}\}_{s,l}$. We approximate its distribution with a parametric CDF $F_c(u;\theta_c) \approx \Pr(U_c \le u)$ and define the fitted quantile function as $F_c^{-1}(p) = \inf\{u : F_c(u;\theta_c) \ge p\}$. Appendix A.1 specifies the CDF families, fitting objective, numerical solver, and post-processing rules.

For an enabled tier with quota $\phi_{c,r}$, the target probability is $p_{c,r} = 1 - \phi_{c,r}$ in the upper tail and $p_{c,r} = \phi_{c,r}$ in the lower tail. With $u_p = F_c^{-1}(p_{c,r})$ and $u_{0.5} = F_c^{-1}(0.5)$, the corresponding ratio threshold is

$$\tau^{\mathrm{raw}}_{c,r} = \begin{cases} \exp(u_p - u_{0.5}), & r \text{ is an upper tier,} \\ \exp(u_{0.5} - u_p), & r \text{ is a lower tier.} \end{cases} \tag{9}$$

Subtracting the fitted median converts a log-scale quantile into a multiplicative deviation from the centre scale. This construction therefore replaces manually selected absolute thresholds with relative decision boundaries that adapt to each model and benchmark.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Models. We evaluate five instruction-tuned models from three families. The models are Qwen3-4B, Qwen3-8B, Qwen3-14B (Yang et al., 2025), Llama3.1-8B (Grattafiori et al., 2024), and Ministral3-

Table 1: Task accuracy scores, score-flip rates, and output consistency for Qwen3-8B on three benchmarks. FP16 denotes inference with the FP16 reference cache. [Unit:%]

Method	LongChat				LVEval				LongBench-v2			
	Score	Acc↑	Acc↓	Cons.	Score	Acc↑	Acc↓	Cons.	Score	Acc↑	Acc↓	Cons.
FP16	92.50	-	-	100.00	36.36	-	-	100.00	37.00	-	-	100.00
Baselines	90.50	1.50	3.50	90.50	29.08	13.00	13.00	57.26	40.50	9.50	6.00	80.00
Ours-HF	92.50	3.00	3.00	91.00	29.02	11.00	11.00	57.02	38.00	8.50	7.50	79.00
Ours-HC	88.00	1.50	6.00	88.00	29.07	12.50	12.50	52.73	35.50	8.00	9.50	76.50

8B (Liu et al., 2026a). All models use 8 KV heads per layer and a head dimension of 128, and are configured for a 128K-token context window. **Benchmarks.** We evaluate on three long-context benchmarks. *LongBench-v2* (Bai et al., 2025) is a bilingual multi-task benchmark covering single- and multi-document QA. *LongChat* (Li et al., 2023) targets long-context dialogue and instruction following across contexts. *LVEval* (Yuan et al., 2025) is a balanced long-context benchmark for long-sequence comprehension, covering five task categories. **Baselines.** We compare against two codec-based systems: CacheGen (Liu et al., 2024a) and KVCodec (Mi et al., 2026). Both baselines use the same layered quantization. They differ in the coding backend, with CacheGen using lossless arithmetic coding and KVCodec using lossless video coding. **Presets.** Our method provides two presets corresponding to distinct operating points. Ours-HF (High-Fidelity) prioritizes high consistency and Ours-HC (High-Compression) prioritizes high compression ratios. Table 6 lists the tier quotas for each preset. **Metrics.** We use consistency (Cons.) as the primary fidelity metric. It measures the similarity between the outputs produced with the compressed cache and the FP16 reference cache on the same input, using F1 for LongChat and LVEval and exact match for LongBench-v2. We measure storage efficiency by the compression ratio (CR), defined as the size of the original FP16 KV cache divided by that of the complete compressed bitstream.

5.2 Consistency as an evaluation objective

We next justify the use of consistency as the primary fidelity metric. Alongside consistency (Cons.), Table 1 reports three diagnostic quantities. Baselines denotes CacheGen and KVCodec, which share the same quantization and produce identical output statistics. Score measures agreement with the benchmark reference, using F1 for LongChat and LVEval and exact-match accuracy for LongBench-v2. Acc↑ and Acc↓ denote the percentages of samples whose task accuracy score increases or decreases, respectively, when the compressed cache replaces the FP16 cache. Because these opposing changes can offset each other, aggregate Score may remain stable even when individual outputs change; it also cannot distinguish different outputs that receive the same task accuracy score.

The Qwen3-8B results make this limitation explicit. On LongChat, Ours-HF matches the FP16 Score of 92.50%, although Acc↑ and Acc↓ are both 3.00% and Cons. is only 91.00%. On LongBench-v2, Baselines increase Score from 37.00% to 40.50% while reaching only 80.00% consistency. Thus, an unchanged, near-unchanged, or improved Score does not imply preservation of FP16 behavior, whereas Cons. directly reveals the remaining output divergence.

5.3 Main results

Cross-benchmark consistency–compression performance. Figure 3 shows the consistency–compression operating points across all benchmark aggregates. Ours-HC and Ours-HF form a trade-off between consistency and compression ratio from HC to HF: CR decreases from $8.35\times$ to $7.41\times$ while Cons. increases from 78.92% to 82.33%. This ordering holds in all three benchmark aggregates.

Against KVCodec, Ours-HF simultaneously improves Cons. by 1.96 pp and CR by 5.26% overall, dominating the baseline on all three benchmark aggregates. The improvements are consistent across benchmarks: $+0.20$–$+4.10$ pp in Cons. and $+5.01\%$–$+5.36\%$ in CR. These comparisons show that our adaptive quantization frontend achieves higher consistency with the FP16 reference at a comparable storage budget. Ours-HC prioritizes storage efficiency, increasing CR by 18.61% at a Cons. reduction of 1.45 pp. The deficit is concentrated on LVEval (-4.48 pp). This concentration is consistent with the greater output sensitivity of heterogeneous free-form long-sequence tasks. Small

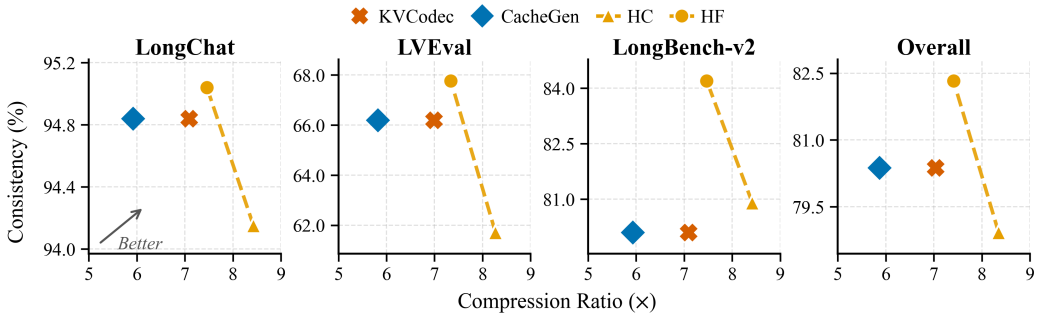

Figure 3: Consistency–compression operating points. The panels show LongChat, LVEval, LongBench-v2, and the overall aggregate. Each point macro-averages Cons. over five models and computes CR from pooled byte totals. Moving upward and right is better.

Table 2: Results for each model averaged across the three benchmarks. Cons. is macro-averaged over benchmarks. CR is computed from pooled byte totals over the evaluation samples. Higher is better for both metrics.

Method	Qwen3-4B		Qwen3-8B		Qwen3-14B		Llama3.1-8B		Ministral3-8B	
	Cons.	CR	Cons.	CR	Cons.	CR	Cons.	CR	Cons.	CR
CacheGen	72.85	6.14	**75.92**	6.09	80.92	6.04	85.61	5.78	86.59	5.37
KVCodec	72.85	7.28	**75.92**	7.22	80.92	7.20	85.61	7.05	86.59	6.51
Ours-HC	69.06	**8.73**	72.41	**8.71**	80.68	**8.66**	84.64	**8.23**	87.82	**7.54**
Ours-HF	**75.88**	7.63	75.67	7.59	**84.17**	7.57	**87.16**	7.45	**88.79**	6.86

cache perturbations may change selected content or surface realization, both of which reduce F1 agreement with the FP16 reference output. On LongBench-v2, Ours-HC improves both metrics simultaneously (+0.80 pp Cons., +18.76% CR).

Against CacheGen, Ours-HF improves Cons. by 1.96 pp and CR by 26.24%. It obtains the highest model-averaged Cons. on all three benchmarks: 95.04% on LongChat, 67.77% on LVEval, and 84.20% on LongBench-v2. CacheGen has the lowest CR in every benchmark aggregate, ranging from 5.82× to 5.93×. Since KVCodec and CacheGen share the same quantization frontend and differ only in the coding backend, the CR gap between them (+19.93%) isolates the compression advantage of the video-coding backend. Our adaptive quantization frontend then adds a further 5.26% CR gain alongside the 1.96 pp consistency improvement of Ours-HF.

Cross-model consistency–compression performance. Table 2 tests whether the aggregate trend is preserved across model families and scales. The Cons. columns of CacheGen and KVCodec are identical for all five models, while KVCodec achieves consistently higher CR (18.56%–21.97% above CacheGen), confirming that the backend advantage is robust across architectures. Ours-HC attains the highest CR on all five models, with gains of 15.82%–20.64% over KVCodec. Ours-HF dominates KVCodec on four of the five models, improving both Cons. (+1.55 to +3.25 pp) and CR (+4.81% to +5.67%) simultaneously. The single exception is Qwen3-8B, where Ours-HF is 0.25 pp below KVCodec in Cons. while still delivering 5.12% higher CR. It nonetheless achieves 24.63% higher CR than CacheGen. On Ministral3-8B, Ours-HC and Ours-HF outperform KVCodec in Cons. by 1.23 pp and 2.20 pp, respectively, while achieving 15.82% and 5.38% higher CR. Such variation is consistent with CacheGen's fixed depth heuristic, which favors shallow layers but cannot respond to model- and benchmark-dependent layer statistics. Our method replaces the fixed depth prior with layer bounds derived from the outlier-cleaned scale distribution for each model and benchmark. This data-dependent allocation is consistent with the stable cross-architecture results.

Composition of the compressed bitstream. Table 3 decomposes the complete compressed representation for each preset. Although Key and Value have equal FP16 footprints in the evaluated caches, Value video occupies 58.0609%–59.9150% of the stored representation, compared with 39.3457%–41.1055% for Key video. The consistently larger Value payload indicates lower compressibility under the same video backend. The combined video payload accounts for 99.1664%–

Table 3: Composition of the stored representation. K video and V video denote the coded Key and Value payloads. Scale includes Key and Value quantization scales. Bypass includes FP16 outlier values and their indices. Other contains all remaining reconstruction metadata. [Unit:%]

Method	K video	V video	Scale	Bypass	Other
Ours-HC	41.1055	58.0609	0.8156	0.0005	0.0175
Ours-HF	39.3457	59.9150	0.7234	0.0004	0.0155

Table 4: Controlled ablations averaged over five models. The two blocks use different reference configurations and isolate group outlier bypass and layer placement, respectively. [Unit:%]

Factor	Configuration	LongChat	LVEval	LongBench-v2	Average
Outlier bypass	Uniform 4-bit	94.16	63.57	80.10	79.28
	Uniform 4-bit + bypass	**94.26**	**63.80**	**81.10**	**79.72**
Layer allocation	Shuffled HC	92.86	58.08	79.10	76.68
	Calibrated HC	**94.15**	**61.71**	**80.90**	**78.92**

99.2607% of the final representation. Scales and all remaining metadata together occupy less than 0.84%. The FP16 bypass, including sparse values and their indices, contributes only 0.0004%–0.0005%. Exact restoration therefore adds negligible storage relative to the dense stream.

5.4 Ablation study

Table 4 uses paired controls because removing a component from the complete pipeline can also change the scale distribution and induced rate. To isolate group outlier bypass, we keep the dense path at a uniform 4-bit precision and change only whether detected extremes are stored separately. Adding group outlier bypass to uniform 4-bit quantization increases average Cons. from 79.28% to 79.72%, a gain of 0.44 pp with only 0.0004%–0.0005% storage overhead. All three benchmark aggregates improve, with the largest gain of 1.00 pp on LongBench-v2. Therefore, group outlier bypass provides a consistent fidelity improvement at negligible storage cost without increasing the precision of the dense path. To isolate layer placement, we compare the calibrated HC allocation with a fixed random permutation using seed 0. The permutation preserves the number of layers in every precision tier separately for Key and Value, and therefore retains the same nominal layer-bit budget. Calibrated HC improves average Cons. from 76.68% to 78.92% relative to the shuffled allocation. The gains are 1.30, 3.63, and 1.80 pp on LongChat, LVEval, and LongBench-v2, respectively. These results show that the benefit of mixed precision depends on assigning the available precision tiers to appropriate layers rather than only selecting their aggregate proportions.

6 Conclusion

This work addresses two distinct challenges in lossy codec-based KV cache reuse: measuring behavioral fidelity and mitigating quantization distortion caused by sparse outliers. First, aggregate task accuracy can conceal compression-induced output changes at the sample level. Consistency with the FP16 reference provides a more direct measure of behavioral fidelity. At the quantization level, we identify outlier-scale pollution, in which sparse extremes dominate a shared group scale, reducing the effective resolution of normal values and compromising subsequent related allocations. To mitigate this problem, we propose outlier-bypass layer-adaptive quantization, which separates sparse group-wise extremes before scale estimation and restores the detected values exactly through a sparse bypass, while the remaining dense values are quantized using layer-specific level derived from the cleaned scale distributions. Component-specific Key and Value policies reflect their different roles in attention, and offline CDF calibration converts preset tier quotas into model- and benchmark-specific thresholds. Across five models and three long-context benchmarks, Ours-HC achieves 78.92% consistency at 8.35× compression, while Ours-HF achieves 82.33% consistency at 7.41× compression.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, organizational refinement, and the drafting and refactoring of code for data analysis and plotting. They were not used to generate experimental observations or replace the authors' scientific judgment. All AI-assisted text and code were reviewed by the authors. Analysis scripts were inspected and executed on the original experimental outputs, and the reported numerical results were checked against those outputs. The authors determined the experimental design, evaluated the evidence, revised the resulting text and artifacts, and take full responsibility for the final content of this work.

REFERENCES

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 3639–3664, 2025.

Weijian Chen, Shuibing He, Haoyang Qu, Ruidong Zhang, Siling Yang, Ping Chen, Yi Zheng, Baoxing Huai, and Gang Chen. IMPRESS: An importance-informed multi-tier prefix KV storage system for large language model inference. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*, pp. 187–201, 2025.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 30318–30332, 2022.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct.

Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. ZipCache: Accurate and efficient KV cache quantization with salient token identification. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 68287–68307, 2024.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 1270–1303, 2024.

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

Minjun Kim, Jaeri Lee, Jongjin Kim, Jeongin Yun, Yongmo Kwon, and U Kang. LampQ: Towards accurate layer-wise mixed precision quantization for vision transformers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(7):5691–5699, 2026.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can open-source LLMs truly promise on context length? https://lmsys.org/blog/2023-06-29-longchat, June 2023.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 22947 – 22970, 2024.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026a. URL https://huggingface.co/mistralai/Ministral-3-8B-Instruct-2512.

Xuyang Liu, Xiyan Gui, Yuchao Zhang, and Linfeng Zhang. Mixing importance with diversity: Joint optimization for KV cache compression in large vision-language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026b.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. CacheGen: KV cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 38–56, 2024a.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pp. 32332–32344, 2024b.

Liang Mi, Weijun Wang, Jinghan Chen, Ting Cao, Haipeng Dai, and Yunxin Liu. Efficient remote KV cache reuse with GPU-native video codec. In *Proceedings of the ACM SIGCOMM 2026 Conference*, pp. 30–45, 2026.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation—a KVCache-Centric architecture for serving LLM chatbot. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*, pp. 155–170, 2025.

Zunhai Su, Hanyu Wei, Zhe Chen, Wang Shen, Linge Li, Huangqi Yu, and Kehong Yuan. RotateKV: Accurate and robust 2-bit KV cache quantization for LLMs via outlier-aware adaptive rotations. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 6200–6208, 2025.

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 *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 38087–38099, 2023.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Ceyu Xu, Yongji Wu, Xinyu Yang, Beidi Chen, Matthew Lentz, Danyang Zhuo, and Lisa Wu Wills. VcLLM: Video codecs are secretly tensor codecs. *arXiv preprint arXiv:2407.00467*, 2024.

Ceyu Xu, Yongji Wu, Xinyu Yang, Beidi Chen, Matthew Lentz, Danyang Zhuo, and Lisa Wu Wills. LLM.265: Video codecs are secretly tensor codecs. In *Proceedings of the 58th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 445–460, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL `https://huggingface.co/collections/Qwen/qwen3`.

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. *arXiv preprint arXiv:2402.18096*, 2024.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 94–109, 2025.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. LV-Eval: A balanced long-context benchmark with 5 length levels up to 256k. In *Second Conference on Language Modeling*, 2025.

Tao Zhang, Ziqian Zeng, Hao Peng, Huiping Zhuang, and Cen Chen. MixKVQ: Query-aware mixed-precision KV cache quantization for long-context reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 7189–7204, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 34661–34710, 2023.

Juntao Zhao, Borui Wan, Yanghua Peng, Haibin Lin, and Chuan Wu. LLM-PQ: Serving LLM on heterogeneous clusters with phase-aware partition and adaptive quantization. In *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pp. 460–462, 2024a.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. ATOM: Low-bit quantization for efficient and accurate LLM serving. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 6, 2024b.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Table 5: Six-tier level-bound policies. Q is the largest signed integer level. A dash denotes a tier unavailable for that KV component in all presets.

Tier	Direction	Key Q	Value Q
L_3	lower	—	1
L_2	lower	—	3
L_1	lower	5	5
C	—	7	7
U_1	upper	11	11
U_2	upper	15	—

Table 6: Tier-quota fractions for the two presets. ϕ_{K} and ϕ_{V} denote the tier-quota vectors $[\phi_{\mathrm{L}_3}, \phi_{\mathrm{L}_2}, \phi_{\mathrm{L}_1}, \phi_{\mathrm{U}_1}, \phi_{\mathrm{U}_2}]$ for Key and Value, respectively.

Preset	Key					Value				
	ϕ_{L_3}	ϕ_{L_2}	ϕ_{L_1}	ϕ_{U_1}	ϕ_{U_2}	ϕ_{L_3}	ϕ_{L_2}	ϕ_{L_1}	ϕ_{U_1}	ϕ_{U_2}
HC	—	—	0.05	0.10	0.05	0.10	0.20	0.50	0.05	—
HF	—	—	0.00	0.25	0.15	0.00	0.00	0.30	0.15	—

A Appendix

A.1 Layer-allocation presets and CDF calibration details

Layer-allocation presets. The layer-allocation mechanism uses the component-specific level-bound ladder in Table 5. Key disables the two lowest tiers to avoid the most aggressive reduction in quantization resolution, whereas Value permits lower level bounds but disables the highest tier. The same ladder is used for all models and benchmarks. Table 6 gives the quota vectors used by the two operating presets. HC permits the most aggressive lower-tier allocation and prioritizes compression. HF removes the lowest active Value tier and increases the Key upper-tier capacities, thereby prioritizing consistency.

CDF calibration details. Let $u_{(1)} \le \cdots \le u_{(n)}$ denote the ordered normalized log-scale observations defined in Sec. 4.4, with empirical plotting positions $\widehat{F}_i \equiv \widehat{F}(u_{(i)}) = (i - 0.5)/n$. We use a shifted-exponential CDF for Qwen Key, a two-component logistic mixture for Qwen Value, and a logistic CDF for Llama and Ministral Key and Value:

$$F_{\exp}(u) = 1 - e^{-b(u-u_0)}, \tag{10}$$

$$F_{\log}(u) = \big(1 + e^{-b(u-c)}\big)^{-1}, \tag{11}$$

$$F_{\mathrm{mix}}(u) = w\big(1 + e^{-b_1(u-c_1)}\big)^{-1} + (1-w)\big(1 + e^{-b_2(u-c_2)}\big)^{-1}. \tag{12}$$

Tier assignment depends on tail quantiles. To prevent the more numerous central observations from dominating the fit, we assign

$$w_i = \frac{\min\{100, [\widehat{F}_i(1-\widehat{F}_i)]^{-1}\}}{n^{-1}\sum_j \min\{100, [\widehat{F}_j(1-\widehat{F}_j)]^{-1}\}} \tag{13}$$

and minimize $\sum_i w_i[F(u_{(i)};\theta) - \widehat{F}_i]^2$ by trust-region reflective least squares with tolerance 10^{-12}. The CDF parameters are estimated separately for each model, benchmark, and KV component and are shared across the two presets. The inverse of the mixture CDF is computed with Brent's method. After obtaining the raw ratio threshold in Eq. 9, the implementation applies the configured penalty and ratio floor. A ratio floor of 8.0 is applied only to the Value L_3 tier, while all other floors are zero. The resulting thresholds are rounded to the nearest multiple of 0.2, after which collisions are resolved to preserve strict ordering within each tail.

Table 7: Per-component CDF goodness-of-fit (R^2) by model and benchmark. Each entry is the tail-weighted least-squares R^2 of the fitted CDF on the per-layer log-scale distribution.

Model	KV	LongChat	LVEval	LongBench-v2
Qwen3-4B	Key	0.993	0.994	0.993
	Value	0.996	0.996	0.996
Qwen3-8B	Key	0.990	0.993	0.992
	Value	0.996	0.997	0.996
Qwen3-14B	Key	0.994	0.995	0.994
	Value	0.998	0.997	0.998
Llama3.1-8B	Key	0.995	0.992	0.993
	Value	0.985	0.990	0.994
Ministral3-8B	Key	0.987	0.992	0.990
	Value	0.989	0.985	0.983

Table 8: Within-calibration stability of the empirical CDFs. Pairwise KS is averaged over all 45 pairs of the ten sample-level CDFs and then over the three benchmarks. LOO KS is the maximum distance between the pooled ten-sample CDF and any leave-one-sample-out CDF. Lower is better.

Model	KV	Mean pairwise KS	Max LOO KS
Qwen3-4B	Key	0.1381	0.0198
	Value	0.1014	0.0117
Qwen3-8B	Key	0.1321	0.0160
	Value	0.1113	0.0170
Qwen3-14B	Key	0.1298	0.0172
	Value	0.1217	0.0150
Llama3.1-8B	Key	0.1301	0.0167
	Value	0.1382	0.0184
Ministral3-8B	Key	0.1632	0.0222
	Value	0.1503	0.0176
Overall	Key	0.1387	0.0222
	Value	0.1246	0.0184

A.2 CDF FIT QUALITY AND CALIBRATION STABILITY

CDF fit quality. As described in Sec. 4.4, the per-tier ratio thresholds are calibrated by fitting a parametric cumulative distribution function to the empirical distribution of the per-layer log-scale, with tail-weighted least squares. Table 7 reports the goodness-of-fit statistic R^2 for every (model, KV component, benchmark) triple. All 30 configurations exceed the 0.98 threshold, confirming that the chosen CDF families adequately characterize the per-layer log-scale distribution across model families and benchmarks.

CDF calibration stability. In addition, we evaluate within-calibration stability using the per-sample statistics already collected by the calibration pipeline. For each model, benchmark, and KV component, we construct one empirical CDF per sample from the median-normalized per-layer log-scales. The pairwise Kolmogorov–Smirnov (KS) distance measures agreement among the ten sample-level curves. We additionally compare the pooled ten-sample empirical CDF with each of its ten leave-one-sample-out (LOO) variants, which tests whether the aggregate curve is dominated by an individual sample. Table 8 shows that the sample-level distributions are similar despite spanning heterogeneous long-context inputs. Averaged over all models and benchmarks, the pairwise KS distance is 0.1387 for Key and 0.1246 for Value. More importantly, removing any one of the ten samples changes the pooled empirical CDF by at most 0.0222 for Key and 0.0184 for Value. No individual calibration sample dominates the pooled empirical distribution. Together with the uniformly high fit quality in Table 7, these results show that the fitted CDFs both approximate the

Table 9: Ranges of the data-dependent HC ratio thresholds over the three benchmarks. A single value indicates that threshold rounding produces the same value on all benchmarks.

Model	$\tau^K_{\mathrm{L}_1}$	$\tau^K_{\mathrm{U}_1}$	$\tau^K_{\mathrm{U}_2}$	$\tau^V_{\mathrm{L}_3}$	$\tau^V_{\mathrm{L}_2}$	$\tau^V_{\mathrm{U}_1}$
Qwen3-4B	1.4	2.2	3.0–3.2	8.0	4.0–4.6	6.8–8.0
Qwen3-8B	1.4	2.2	3.0	10.6–11.0	5.8–6.0	6.6–7.4
Qwen3-14B	1.4	2.2	3.0–3.2	8.0	4.0–4.4	9.0–10.2
Llama3.1-8B	1.2	1.2	1.4	8.0	1.4	1.8–2.0
Ministral3-8B	1.2	1.2	1.4	8.0	1.4–1.6	2.2–2.6

Table 10: Cross-model task accuracy scores, directional score-flip rates, and output consistency. Results cover five models, three benchmarks. FP16 denotes the reference output obtained from the original FP16 cache. Metric names follow Sec. 5.2. [Unit:%]

Model	Method	LongChat				LVEval				LongBench-v2			
		Score	Acc↑	Acc↓	Cons.	Score	Acc↑	Acc↓	Cons.	Score	Acc↑	Acc↓	Cons.
Qwen3-4B	FP16	89.18	-	-	100.00	29.12	-	-	100.00	29.50	-	-	100.00
	Baselines	84.60	2.00	7.50	84.78	20.66	14.50	14.50	54.76	30.00	7.00	6.50	79.00
	Ours-HC	84.59	1.50	7.00	84.27	18.74	9.00	9.00	43.90	28.00	5.50	7.00	79.00
	Ours-HF	85.50	1.50	5.00	86.18	21.20	12.00	12.00	57.45	27.00	4.00	6.50	84.00
Qwen3-8B	FP16	92.50	-	-	100.00	36.36	-	-	100.00	37.00	-	-	100.00
	Baselines	90.50	1.50	3.50	90.50	29.08	13.00	13.00	57.26	40.50	9.50	6.00	80.00
	Ours-HC	88.00	1.50	6.00	88.00	29.07	12.50	12.50	52.73	35.50	8.00	9.50	76.50
	Ours-HF	92.50	3.00	3.00	91.00	29.02	11.00	11.00	57.02	38.00	8.50	7.50	79.00
Qwen3-14B	FP16	100.00	-	-	100.00	44.54	-	-	100.00	43.00	-	-	100.00
	Baselines	97.40	0.00	0.50	98.90	39.56	15.00	15.00	64.35	40.00	6.00	9.00	79.50
	Ours-HC	97.50	0.50	0.50	98.50	39.49	18.00	18.00	60.04	42.00	5.50	6.50	83.50
	Ours-HF	97.00	0.00	0.50	98.50	39.98	17.00	17.00	67.50	42.50	4.50	5.00	86.50
Llama3.1-8B	FP16	100.00	-	-	100.00	37.59	-	-	100.00	33.50	-	-	100.00
	Baselines	100.00	0.00	0.00	100.00	34.31	7.50	7.50	78.82	30.00	5.50	9.00	78.00
	Ours-HC	100.00	0.00	0.00	100.00	35.50	9.00	9.00	75.41	29.50	4.00	8.00	78.50
	Ours-HF	99.50	0.00	0.50	99.50	36.93	9.00	9.00	78.49	29.00	3.00	7.50	83.50
Ministral3-8B	FP16	100.00	-	-	100.00	34.95	-	-	100.00	37.00	-	-	100.00
	Baselines	100.00	0.00	0.00	100.00	33.64	12.50	12.50	75.76	37.50	5.50	5.00	84.00
	Ours-HC	100.00	0.00	0.00	100.00	33.43	12.50	12.50	76.47	40.00	5.50	2.50	87.00
	Ours-HF	100.00	0.00	0.00	100.00	33.73	11.50	11.50	78.38	37.50	4.50	4.00	88.00

observed scale distributions and capture structure shared across the selected calibration samples. This analysis establishes the internal stability of the chosen ten-sample calibration set.

Cross-domain threshold adaptation. Table 9 reports the data-dependent HC thresholds produced by the same preset quotas and calibration procedure. Each threshold entry gives its range over LongChat, LVEval, and LongBench-v2. HC is reported because it activates the largest set of data-dependent tiers. The Value threshold for L_1 is fixed at 1.0. The calibrated thresholds vary more across model families than across benchmarks of the same model. For Key, the Qwen models use thresholds of approximately $(1.4, 2.2, 3.0)$ for $(\mathrm{L}_1, \mathrm{U}_1, \mathrm{U}_2)$, whereas Llama3.1-8B and Ministral3-8B use $(1.2, 1.2, 1.4)$. The model dependence is more pronounced for Value. Its U_1 threshold lies between 1.8 and 2.6 for Llama and Ministral but between 6.6 and 10.2 for the Qwen models. These differences remain after median normalization and reflect architecture-dependent shapes of the layer-scale distribution. A single absolute threshold would induce different allocation behavior across model families. Thresholds also show smaller but nonzero benchmark dependence, primarily for Value. These differences support model- and benchmark-specific calibration rather than a single absolute threshold shared across all settings.

A.3 Accuracy flip analysis

Table 10 extends the diagnostic analysis in Sec. 5.2 from Qwen3-8B to all five models. Across all 45 compressed method–model–benchmark configurations, the median absolute Score change is only 2.50 pp, whereas the median fraction of samples with a score flip is 6.00% and the median output inconsistency is 20.00%. Small aggregate changes conceal substantially larger sample-level behavioral changes.

Table 11: Complete per-model and aggregate results. Cons. denotes consistency in percent and CR denotes compression ratio in $\times$. Higher is better for both metrics. For each model, Average contains the average consistency over the three benchmarks and the compression ratio computed from pooled byte totals over all samples. In the Benchmark Avg. block, the three benchmark groups average over five models, while Average gives the overall aggregate over all 15 model–benchmark pairs.

Model	Method	LongChat		LVEval		LongBench-v2		Average	
		Cons.	CR	Cons.	CR	Cons.	CR	Cons.	CR
Qwen3-4B	CacheGen	84.78	6.20	54.76	6.08	79.00	6.20	72.85	6.14
	KVCodec	84.78	7.35	54.76	7.21	79.00	7.35	72.85	7.28
	Ours-HC	84.27	8.87	43.90	8.62	79.00	8.81	69.06	8.73
	Ours-HF	86.18	7.70	57.45	7.55	84.00	7.71	75.88	7.63
Qwen3-8B	CacheGen	90.50	6.14	57.26	6.03	80.00	6.14	75.92	6.09
	KVCodec	90.50	7.28	57.26	7.16	80.00	7.28	75.92	7.22
	Ours-HC	88.00	8.79	52.73	8.62	76.50	8.79	72.41	8.71
	Ours-HF	91.00	7.66	57.02	7.53	79.00	7.65	75.67	7.59
Qwen3-14B	CacheGen	98.90	6.09	64.35	5.98	79.50	6.08	80.92	6.04
	KVCodec	98.90	7.26	64.35	7.14	79.50	7.24	80.92	7.20
	Ours-HC	98.50	8.77	60.04	8.58	83.50	8.70	80.68	8.66
	Ours-HF	98.50	7.66	67.50	7.49	86.50	7.63	84.17	7.57
Llama3.1-8B	CacheGen	100.00	5.83	78.82	5.75	78.00	5.81	85.61	5.78
	KVCodec	100.00	7.09	78.82	7.02	78.00	7.06	85.61	7.05
	Ours-HC	100.00	8.29	75.41	8.20	78.50	8.25	84.64	8.23
	Ours-HF	99.50	7.46	78.49	7.43	83.50	7.47	87.16	7.45
Ministral3-8B	CacheGen	100.00	5.38	75.76	5.33	84.00	5.43	86.59	5.37
	KVCodec	100.00	6.52	75.76	6.47	84.00	6.58	86.59	6.51
	Ours-HC	100.00	7.53	76.47	7.49	87.00	7.63	87.82	7.54
	Ours-HF	100.00	6.87	78.38	6.80	88.00	6.95	88.79	6.86
Benchmark Avg.	CacheGen	94.84	5.92	66.19	5.82	80.10	5.93	80.37	5.87
	KVCodec	94.84	7.09	66.19	6.99	80.10	7.09	80.37	7.04
	Ours-HC	94.15	8.43	61.71	8.27	80.90	8.42	78.92	8.35
	Ours-HF	95.04	7.46	67.77	7.34	84.20	7.47	82.33	7.41

The cross-model results further show that this is a structural limitation of Score rather than an isolated cancellation. On LVEval, Acc$\uparrow$ equals Acc$\downarrow$ in all 15 compressed configurations, while Cons. ranges from 43.90% to 78.82%. Cons. thus quantifies the severity of output divergence even when the directional flip counts are perfectly balanced. Moreover, on LongBench-v2 with Ministral3-8B, Baselines and Ours-HF obtain the same Score of 37.50%, but their Cons. values are 84.00% and 88.00%, respectively. The aggregate task metric treats these configurations as equivalent, whereas consistency distinguishes how faithfully they reproduce FP16 behavior. This additional discriminative power motivates using Cons. for fidelity comparisons and retaining Score and flip rates only as diagnostics.

A.4 Complete per-model results

Table 11 reports all method–model–benchmark results together with the two complementary aggregation views used in the main text. The Average columns in each model block report the model average across three benchmarks, whereas the Benchmark Avg. block reports the five-model average for each benchmark and the overall aggregate.

The disaggregated results confirm that the advantage of Ours-HF over KVCodec is broadly distributed. Ours-HF dominates KVCodec in 10 of the 15 model–benchmark pairs, simultaneously improving both Cons. and CR. The five exceptions are Qwen3-8B on LVEval (-0.24 pp Cons.) and LongBench-v2 (-1.00 pp Cons.), Qwen3-14B on LongChat (-0.40 pp Cons.), and Llama3.1-8B on LongChat (-0.50 pp Cons.) and LVEval (-0.33 pp Cons.). In each case the Cons. deficit is small: the Qwen3-14B/LongChat and Llama3.1-8B/LongChat pairs are near the performance ceil-

Table 12: Compressed-bitstream composition, aggregated separately by model and benchmark. Each model block pools the three benchmarks, and each benchmark block pools the five models. K video and V video denote the coded Key and Value payloads. [Unit:%]

Aggregate	Method	K video	V video	Scale	Bypass	Other
Qwen3-4B	Ours-HC	39.2705	59.8732	0.8525	0.0000	0.0038
	Ours-HF	36.7560	62.4954	0.7452	0.0000	0.0033
Qwen3-8B	Ours-HC	39.9347	59.2070	0.8501	0.0002	0.0080
	Ours-HF	37.3850	61.8664	0.7415	0.0001	0.0070
Qwen3-14B	Ours-HC	41.1769	57.9404	0.8452	0.0011	0.0365
	Ours-HF	38.7644	60.4638	0.7390	0.0009	0.0319
Llama3.1-8B	Ours-HC	41.7159	57.4546	0.8039	0.0007	0.0250
	Ours-HF	40.8801	58.3693	0.7273	0.0006	0.0226
Ministral3-8B	Ours-HC	43.1025	56.1478	0.7364	0.0003	0.0130
	Ours-HF	42.7157	56.6023	0.6698	0.0003	0.0119
LongChat	Ours-HC	40.7847	58.3726	0.8234	0.0005	0.0188
	Ours-HF	39.0474	60.2068	0.7288	0.0004	0.0166
LVEval	Ours-HC	41.2152	57.9598	0.8080	0.0004	0.0166
	Ours-HF	39.4397	59.8281	0.7171	0.0004	0.0147
LongBench-v2	Ours-HC	41.1644	57.9947	0.8225	0.0005	0.0179
	Ours-HF	39.4142	59.8397	0.7298	0.0004	0.0159

ing (98.50% and 99.50%, respectively), and the remaining three deficits do not exceed 1.00 pp. In all five pairs, Ours-HF still provides 5.1%–5.8% higher CR than KVCodec. Relative to CacheGen, Ours-HF provides higher CR in all 15 model–benchmark pairs and matches or exceeds its Cons. in 10. In the remaining five cases, the Cons. deficit is at most 1.00 pp, whereas the corresponding CR gain ranges from 24.6% to 29.2%. These comparisons complement the aggregate results by confirming that the improvements are not concentrated in a small subset of models or benchmarks.

The preset ordering is exact for storage but not for every individual consistency measurement. CR strictly decreases from Ours-HC to Ours-HF in all 15 model–benchmark pairs. Cons. increases in 14 of 15 pairs; the single reversal is 0.50 pp on Llama3.1-8B with LongChat, where Ours-HC already reaches 100.00% Cons. and both presets approach the benchmark ceiling. The presets change separate quota vectors for Key and Value and are not nested refinements of the same quantized cache. A less aggressive preset therefore does not impose monotonic reconstruction error on every layer or sample, and small reversals near the performance ceiling are expected.

The complete results also separate rate stability from task sensitivity. For every model and preset, the CR range across the three benchmarks is at most $0.25\times$, whereas the corresponding Cons. range reaches 40.37 pp. This contrast shows that the quota mechanism maintains a stable storage budget while output fidelity remains sensitive to the model and benchmark. LVEval accounts for the largest Ours-HC deficits against KVCodec, reaching 10.86 pp for Qwen3-4B and 4.53 pp for Qwen3-8B, consistent with the sensitivity of heterogeneous free-form long-sequence generation. Switching to Ours-HF removes most of these deficits: it exceeds KVCodec by 2.69 pp on Qwen3-4B/LVEval and 3.15 pp on Qwen3-14B/LVEval; the small residual deficit on Qwen3-8B/LVEval is only 0.24 pp. The detailed results therefore support selecting the preset according to workload sensitivity rather than assuming that one operating point is uniformly optimal.

A.5 Bitstream composition

Table 12 reports aggregates by model and benchmark. Value video is larger than Key video for every model, benchmark, and preset. Because the corresponding uncompressed tensors have equal byte counts, this ordering confirms that unequal input sizes do not cause the aggregate gap between Key and Value and that Value is consistently less compressible under the video representation. The magnitude varies across architectures, which is compatible with model-dependent Key and Value distributions and motivates treating the two components separately.

Table 13: Average TTFT stratified by context length over five models and three benchmarks. Cache-reuse measurements begin at transfer and exclude offline encoding. [Unit: s]

Method	< 32K	32–64K	64–96K	≥ 96K	Overall
Full prefill	1.7291	3.7234	7.1486	16.2089	3.7827
Lossless KV reuse	0.7714	1.9011	3.0951	5.4741	1.6573
CacheGen	0.7356	1.5739	2.4247	4.4258	1.4031
KVCodec	1.1633	1.1781	1.2707	1.3806	1.1905
Ours-HC	1.5552	1.2094	1.2402	1.3521	1.3919
Ours-HF	1.4017	1.1715	1.2232	1.3145	1.2984

Table 14: Average receiver-side TTFT breakdown of KVCodec and Ours. [Unit: s]

Method	Transfer	Video decode	Side info	Dequantize	KV rebuild	First token	Total
KVCodec	0.0441	0.9619	0.0161	0.0738	0.0154	0.0793	1.1905
Ours-HC	0.0372	1.0226	0.0106	0.0591	0.0147	0.2479	1.3919
Ours-HF	0.0419	0.9960	0.0106	0.0567	0.0146	0.1785	1.2984

The composition remains stable across the evaluated model–benchmark pairs. Within a preset, each of the Key and Value video fractions varies by less than 0.44 pp across the three benchmarks. Their combined fraction exceeds 99.11% for every model and preset. The largest bypass fraction is only 0.0011%, while all other side information remains below 0.89%.

A.6 Receiver-side TTFT and context-length scaling

We measure receiver-side time to first token (TTFT) from the start of cache transfer to the production of the first generated token. Sender-side encoding is excluded because the bitstream can be prepared before a reuse request. Full prefill is included as a reference and measures prompt-prefill TTFT without cache reuse. Lossless KV reuse pulls the raw KV cache remotely. Table 13 shows the TTFT for different methods at different context lengths. Below 32K tokens, the fixed decoder startup cost limits the benefit of cache compression, although compressed representations can still provide modest TTFT reductions. This cost is amortized as the context grows. From 32–64K tokens to at least 96K tokens, TTFT increases by $4.35\times$, $2.88\times$, and $2.81\times$ for full prefill, lossless KV reuse, and CacheGen, respectively, but by only approximately $1.12\times$ for both of our presets. At 96K tokens and above, Ours-HC–Ours-HF reduce TTFT by 91.7%–91.9% relative to full prefill, 75.3%–76.0% relative to lossless KV reuse, and 69.4%–70.3% relative to CacheGen. These results show that cache compression removes the dominant long-context transfer and prefill costs, while the video-coding backend provides substantially flatter scaling with context length. KVCodec and our method share the video-coding backend and consequently exhibit similar TTFT.

The stage decomposition in Table 14 shows that video decoding dominates receiver-side latency for both KVCodec and our method. Ours-HC lowers transfer time by 15.6% relative to KVCodec through its higher compression ratio. Ours-HF reduces transfer time by 5.0%. Relative to KVCodec, exact group outlier bypass and layer-specific level bounds introduce a more heterogeneous reconstruction path and are accompanied by modest increases in video-decoding and dequantization time. The proposed presets therefore trade a small amount of receiver-side processing for adaptive fidelity and rate control without changing the latency scale of the video-coding backend.