OSCAR: Offline Spectral Covariance-Aware Rotation for 2-bit KV Cache Quantization

Zhongzhu Zhou[*,1,2] **Donglin Zhuang**[*,1,2] **Jisen Li**[1,3] **Ziyan Chen**[2]
Shuaiwen Leon Song[1] Ben Athiwaratkun[1] Xiaoxia Wu[†,1]

[1]Together AI [2]University of Sydney [3]University of Illinois Urbana-Champaign

[*]Equal contribution. [†]Corresponding author.

Code | **Website**
RotationZoo

Abstract

INT2 KV-cache quantization is attractive for long-context LLM serving, but it remains difficult to make both accurate and deployable. Simple rotations such as Hadamard transforms reduce outliers, but still degrade at INT2 because they are not aligned with downstream attention. We propose OSCAR, an Ultra-low-bit KV Cache quantization method that estimates *attention-aware covariance* structures offline and uses them to derive fixed rotations and clipping thresholds for quantization. In this way, it aligns KV quantization with the covariance structures that attention actually consumes. More importantly, we not only provide theoretical justification but also develop a fully deployable OSCAR system with a custom INT2 attention kernel that remains compatible with paged KV-cache serving and fused kernel pipelines, enabling seamless integration into modern LLM serving frameworks such as SGLang and vLLM.

We evaluate our methods on recent reasoning models with reasoning traces of up to 32k tokens across 5 tasks. On Qwen3-4B-Thinking-2507 and Qwen3-8B, OSCAR reduces the BF16 accuracy gap to 3.78 and 1.42 points, respectively, while naive rotation INT2 collapses to nearly zero. We further scale OSCAR to Qwen3-32B and GLM-4.7 (358B params), where it remains effectively on par with BF16. On long context - RULER-NIAH up to 128K, OSCAR remains robust on both Qwen3 models, while naive rotation INT2 collapses. System-wise, OSCAR reduces KV-cache memory by approximately 8×, improves throughput by up to 7× at large batch sizes under the same memory budget, and accelerates batch-size-1 decoding by up to 3× over BF16 due to reduced memory bandwidth overhead.

1 Introduction

Long-context inference has made the key–value (KV) cache one of the main costs of serving large language models. During autoregressive decoding, the cache grows with context length, batch size, and model depth, and every new token must read a large fraction of it from GPU memory. Compressing the KV cache is therefore a direct way to increase batch size and reduce memory traffic (Zhang et al., 2023; Sharma et al., 2024; Ge et al., 2023; Hooper et al., 2024; Yue et al., 2024; Xia et al., 2025; Su et al., 2025; LI et al., 2025). Among the available design points, INT2 quantization is especially attractive: it promises a large memory reduction while retaining a hardware-friendly fixed-width representation.

Aggressively compressing KV caches to ultra-low precision (e.g., INT2) remains challenging because KV activations contain severe channel-wise outliers: a small subset of channels often exhibit extremely large magnitudes, while most channels remain relatively well-behaved (Hooper et al., 2024). Under low-bit quantization, these outliers dominate the quantization scale, compressing most normal values into only a few effective quantization levels and substantially degrading attention quality. Rotation-based quantization addresses this issue by applying a fixed orthogonal transform, such as a Hadamard rotation, that redistributes a few extreme activation values across many channels, producing a more uniform activation distribution that is easier to quantize (Ashkboos et al., 2024; Tseng et al., 2024; Liu et al., 2024a; Jia et al., 2026). In addition, rotation preserves tensor dimensionality and applies a fixed linear transform without intro-

ducing per-channel routing or irregular sparse metadata. This makes it naturally compatible with paged KV-cache layouts (Kwon et al., 2023), and FlashAttention-style fused decode kernels (Dao et al., 2022; Dao, 2023; Shah et al., 2024; Zadouri et al., 2026): each KV vector is simply moved into a better-conditioned basis before quantization and moved back when used by page attention (Jia et al., 2026).

However, a random rotation is still data-oblivious. It can smooth activation ranges, but it does not know which directions are important to attention. At INT2, this distinction matters: only four quantization levels are available, so the error should be pushed into directions that the model reads less strongly. Attention operates on the correlations and score-weighted interactions induced by keys and values, rather than on their raw Euclidean representations. This suggests the optimal rotation target should be derived from attention statistics themselves. Based on this observation, we propose OSCAR, an INT2 KV quantizer that estimates attention-aware covariance structures through a lightweight calibration pass and uses them to derive separate rotations for keys and values, along with per-layer clipping thresholds. Figure 1 (left) illustrates that while data-oblivious rotations can partially smooth outliers, they remain insufficient for INT2 quantization. In contrast, covariance-aware rotations produce substantially smoother activation distributions, enabling effective quantization.

Our contributions are summarized as follows

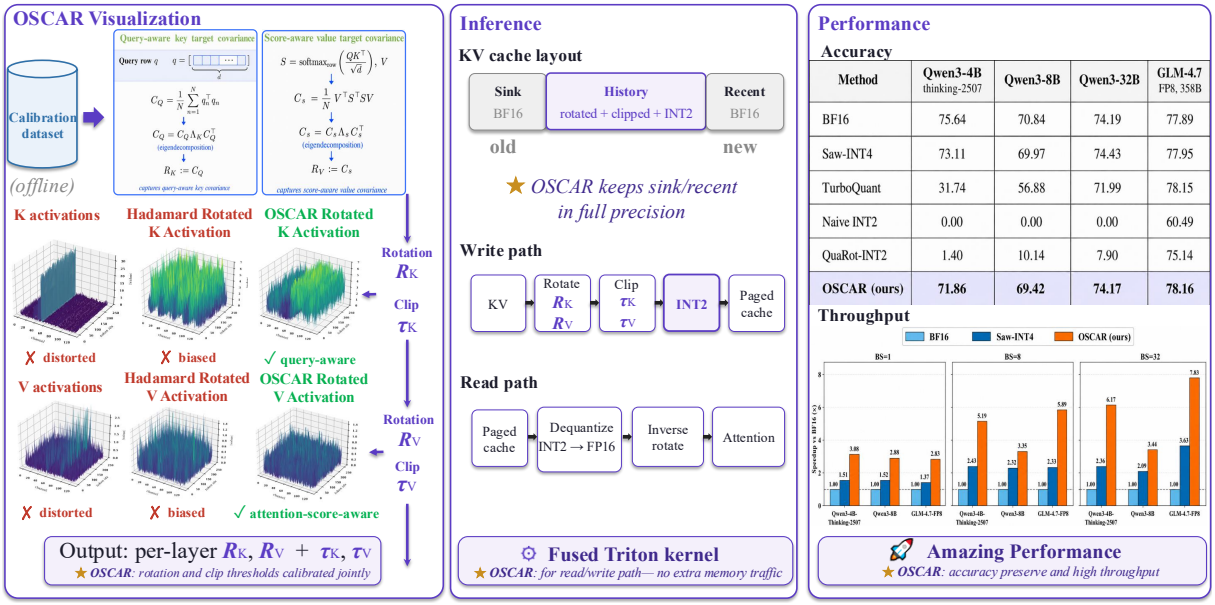
- **We identify the missing target in INT2 rotation-based KV.** Generic rotations mainly scatter activation outliers, but INT2 accuracy depends on the errors in attention scores and layer outputs; the rotation should be induced by downstream attention not by raw cache reconstruction alone.
- **We propose OSCAR, an attention-aware calibration framework for ultra-low-bit KV-cache quantization.** OSCAR uses a lightweight calibration set to obtain attention-aware rotations for keys and values, enabling the quantized cache to better preserve downstream attention computation. A theoretical analysis is provided to show that the resulting covariance-target rotations are optimal under a natural frozen-error surrogate. Empirically, we evaluated across a wide range of state-of-the-art models from 4B to 400B, and retains near-BF16 accuracy at only 2.28 bits per KV element across multiple LLM families, including on a challenging code benchmark (LiveCodeBench).
- **We develop a production-ready INT2 KV-cache serving system.** OSCAR preserves compatibility with paged and prefix KV-cache serving by keeping the bulk KV cache in a dense rotated INT2 representation. The system integrates into production SGLang decoding pipelines with customized Triton decoding kernels, such that one can fully utilize the prefix cache techniques (Zheng et al., 2024). Our system delivers up to 6.2× higher throughput at 100k length and achieves roughly 2× gains in both per-user speed and per-GPU throughput under full-cache workloads. This shows it is both user-friendly (lower latency) and system-efficient (higher GPU utilization).

Due to space constraints, we provide a full discussion in Appendix C, D on how our ideas are inspired by and connected to prior work in the research community.

2 Preliminaries and Motivation

Attention and KV cache. We use *row-vector* notation throughout. For simplicity, we define a single-head attention (Vaswani et al., 2017) as follows. Given a sequence T of hidden states $\{x_t\}_{t=1}^{T}$ with $x_t \in \mathbb{R}^{1\times d}$ and the weights $W_Q, W_K, W_V, \in \mathbb{R}^{d\times d}$, we formulate the query, key, and value as $Q = [q_1; \ldots; q_T] \in \mathbb{R}^{T\times d}, K = [k_1; \ldots; k_T] \in \mathbb{R}^{T\times d}, V = [v_1; \ldots; v_T] \in \mathbb{R}^{T\times d}$, where d is the head dimension; $q_t = x_t W_Q \in \mathbb{R}^{1\times d}$, $k_t = x_t W_K \in \mathbb{R}^{1\times d}$, and $v_t = x_t W_V \in \mathbb{R}^{1\times d}$.The attention scores is defined as $S = \mathrm{softmax}_{\mathrm{row}}(QK^\top/\sqrt{d}) \in \mathbb{R}^{T\times T}$, and the attention output is $O = SV \in \mathbb{R}^{T\times d}$, equivalently $o_i = \sum_{t=1}^{T} S_{i,t} v_t$. During autoregressive inference, $K_{1:t} = [k_1; \ldots; k_t]$ and $V_{1:t} = [v_1; \ldots; v_t]$ are stored in cache – this is the so-called KV cache. Covariance and more details are in Appendices A, B.

Quantization notation. A b-bit quantizer consists of a quantization map $\mathcal{Q}^+ : \mathbb{R} \to \mathcal{C}_b$ and a dequantization map $\mathcal{Q}^- : \mathcal{C}_b \to \mathbb{R}$, where $\mathcal{C}_b$ is a discrete set of 2^b representable codes. Their composition gives the quantize–



Method	Qwen3-4B thinking-2507	Qwen3-8B	Qwen3-32B	GLM-4.7 FP8, 358B
BF16	75.64	70.84	74.19	77.89
Saw-INT4	73.11	69.97	74.43	77.95
TurboQuant	31.74	56.88	71.99	78.15
Naive INT2	0.00	0.00	0.00	60.49
QuaRot-INT2	1.40	10.14	7.90	75.14
OSCAR (ours)	**71.86**	**69.42**	**74.17**	**78.16**

Figure 1: **OSCAR pipeline overview.** Offline, OSCAR estimates attention-aware key/value covariance rotation and shows how the resulting rotation makes KV activations more uniform: Hadamard mixing flattens raw peaks, while the OSCAR rotation separates directions that matter more or less to attention. Online, the serving path keeps sink and recent tokens in BF16 while applying the fixed rotate–clip–INT2 path to history KV tokens inside the paged SGLang cache. The right panels summarize that OSCAR preserves accuracy across models while achieving throughput close to the QuaRot(INT2) serving path and higher throughput than BF16.

dequantize map $\mathcal{Q}(x) = \mathcal{Q}^{-}(\mathcal{Q}^{+}(x))$. A symmetric uniform map can be written as $\mathcal{Q}(x) = s \cdot \mathrm{clip}(\lfloor x/s \rceil, -\tau, \tau)$, where $s > 0$ is the quantization scale and τ is the clipping limit. For a matrix X consisting of row vectors, we apply quantization element-wise to each row.[1]. Given rotation matrix R (Ashkboos et al., 2024), we denote the reconstruction of quantized K and V as $\widehat{K} = \mathcal{Q}(KR)R^{\top}$ and $\widehat{V} = \mathcal{Q}(VR)R^{\top}$.

Why raw-cache reconstruction is not enough. Rotation is effective for KV-cache quantization because it spreads large channel values into a basis with a more uniform dynamic range (Ashkboos et al., 2024). However, simple data-oblivious rotations such as Hadamard or random orthogonal transforms are often insufficient: they smooth the cached tensors, but do not identify which directions are most important for downstream attention. A standard tensor-reconstruction view minimizes $\|K - \mathcal{Q}(K)\|_F^2$ and $\|V - \mathcal{Q}(V)\|_F^2$, yet attention does not consume K and V through their Euclidean reconstruction errors. Keys are used through logits, and values are used through the attention-weighted aggregation. For keys K, the downstream logit distortion is

$$\|QK^{\top} - Q\widehat{K}^{\top}\|_F^2 = \mathrm{tr}\Big((K - \widehat{K})Q^{\top}Q(K - \widehat{K})^{\top}\Big), \tag{1}$$

which is controlled by the query covariance $Q^{\top}Q$ rather than by $K^{\top}K$ alone. For values V with quantized $\widehat{V} = \mathcal{Q}(V)$, the downstream output distortion is

$$\|SV - S\widehat{V}\|_F^2 = \mathrm{tr}\Big((V - \widehat{V})^{\top}S^{\top}S(V - \widehat{V})\Big), \tag{2}$$

which depends on how attention scores weight the value rows. Thus, if the goal is to reduce empirical attention distortion rather than raw-cache reconstruction error, the rotation should be estimated from target covariance induced by the attention computation itself.

Figure 2 illustrates this gap. Naive INT2, Hadamard-only rotation, and clipping alone still leave substantial attention-score divergence and output error. In contrast, OSCAR uses attention-aware calibration target covariance before quantization, reducing the error in four subfigures. We provide a detailed intuition for why each factor is needed and why they appear in this order in App. A.4.

[1]Row-wise is used for our theory. Empirically, quantization is applied to the head dimension with block-size 128, 64 or 32.

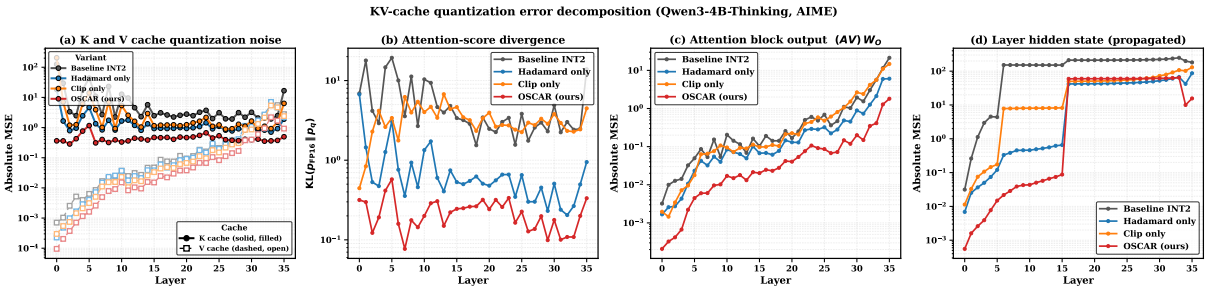


Figure 2: Quantization error. OSCAR limits it at every stage (Qwen3-4B-Thinking-2507, AIME). **(a)** Relative MSE in quantized K and V caches (keys solid, values dashed). **(b)** $\mathrm{KL}(p_{\mathrm{FP16}} \,\|\, p_q)$ between FP16 and quantized attention score distributions. **(c)** Relative MSE at the attention-block output (post-W_O). **(d)** Relative MSE in the propagated hidden states across layers. Curves compare naive INT2, Hadamard-only preprocessing, clip-only outlier control, and full OSCAR.

3 Algorithm Design: Offline Calibration and Justification

The method has two phases: an offline phase that estimates covariance , constructs layer-wise rotation matrices, and fits per-token clipping thresholds; and an online phase that applies the resulting fixed transforms based on a mixed-precision cache layout during serving. All covariance and rotations are estimated offline from a small calibration dataset. For each layer and attention head, we construct attention-aware target covariance from the calibration activations to determine the base rotations.

Query-aware key target covariance. For a query row q, with the attention-aware key target covariance $Q^\top Q$ introduced in equation (1), its empirical estimator is $C_Q = \frac{1}{N}\sum_{n=1}^{N} q_n^\top q_n$; we apply eigen decomposition as in App. A.2, $C_Q = U_Q \Lambda_Q U_Q^\top$, and define the rotation as $R_k := U_Q$.

Score-aware value target covariance. For the attention matrix $S = \mathrm{softmax}_{\mathrm{row}}(QK^\top/\sqrt{d})$, we heuristically define the target covariance in equation (2): $C_S = \frac{1}{N} V^\top S^\top S V$. We then compute the eigendecomposition as in App. A.2, $C_S = U_S \Lambda_S U_S^\top$, and define the raw rotation as $R_v := U_S$.

OSCAR rotations. Following rotation-based low-bit quantization methods (Ashkboos et al., 2024; Chee et al., 2023), we compose the base rotation with a Hadamard transform H_{Had} (App. A.1) and a bit-reversal permutation P_{br} to form the final rotations for keys and values:

$$R_K = U_Q\, H_{\mathrm{Had}}\, P_{\mathrm{br}}, \qquad R_V = U_S\, H_{\mathrm{Had}}\, P_{\mathrm{br}}. \tag{3}$$

Here R_k and R_v are the covariance base rotations, while H_{Had} further improves quantization geometry by redistributing channel energy, and P_{br} (Cooley and Tukey, 1965) interleaves large- and small-variance channels so that adjacent channels have similar dynamic range.

Scale determination and per-token clipping. Our quantization backend follows standard post-training quantization practice. We use affine asymmetric INT2 quantization with scale and zero point for both keys and values, together with percentile-based clipping to control outliers. These mechanisms are common in low-bit activation and KV-cache quantization pipelines (Liu et al., 2024b; Hooper et al., 2024; Xiao et al., 2023; Frantar et al., 2023), with technical details shown in App. A.5.

Optimality of targets under ambient error assumption. Below we propose that under the diagonal residual assumption, our heuristic targets $C_Q = \frac{1}{N}\sum_{n=1}^{N} q_n^\top q_n$ and $C_S = \frac{1}{N} V^\top S^\top S V$ achieve the lowest frozen-error surrogate. The proof of the theorem and the justification of surrogate objectives mentioned in the theorem are shown in App. A.6 and App. A.7 respectively.

Theorem 1 (Optimality of the simple spectral variants under ambient-basis diagonal residuals)**.** *Consider the frozen-error surrogate objectives for the key and value base rotations:* $\tilde{\mathcal{L}}_K(R_k) = \mathrm{tr}(R_k^\top C_Q R_k E_K), R_k^\top R_k = I_d$ *and* $\tilde{\mathcal{L}}_V(R_v) = \mathrm{tr}\big(R_v^\top C_S R_v E_V\big), R_v^\top R_v = I_d$. *Assume that the frozen (independent of input and rotation) residual covariances are diagonal in the ambient basis:*

$$E_K = \sum_{j=1}^{N}\big(\mathcal{Q}(k_j R_k) - k_j R_k\big)^\top \big(\mathcal{Q}(k_j R_k) - k_j R_k\big) = \mathrm{diag}(\mu_1, \ldots, \mu_d), \quad \mu_1 \le \cdots \le \mu_d,$$

$E_V = \sum_{j=1}^{N} \big(\mathcal{Q}(v_j R_v) - v_j R_v\big)^\top \big(\mathcal{Q}(v_j R_v) - v_j R_v\big) = \mathrm{diag}(\eta_1, \ldots, \eta_d), \quad \eta_1 \le \cdots \le \eta_d.$

Then, on the calibration dataset: $R_k = U_Q$ and $R_v = U_S$ are minimizers of $\tilde{\mathcal{L}}_K(R_k)$ and $\tilde{\mathcal{L}}_V(R_v)$.

4 System Design: Online Serving with 2-bit KV Cache

KV Cache Layout. We integrate OSCAR into the SGLang (Zheng et al., 2024) serving stack as an INT2 KV-cache mode with full compatibility with paged-attention (Kwon et al., 2023). The implementation preserves two short high-precision windows: the first S_0 tokens, which behave as attention sinks, and the most recent W tokens before the current position. The rest of the middle context is stored in INT2 after the fixed OSCAR rotation. Thus, at decoding position t, the logical cache consists of

$$\underbrace{[1, S_0]}_{\text{bf16 sink}} \;\|\; \underbrace{[S_0+1, t-W]}_{\text{int2 history}} \;\|\; \underbrace{[t-W+1, t]}_{\text{bf16 recent}}$$

KV Cache Update. During prefill, the runtime writes cache rows through a fully fused Triton kernel (Tillet et al., 2019). For a token with BF16 rows k_t, v_t and a clip value of $\tau_t^{(K)}, \tau_t^{(V)}$, the stored INT2 are

$$k_t^+ = \mathcal{Q}_2^+\Big(\mathrm{clip}(k_t R_K, \tau_t^{(K)})\Big),$$
$$v_t^+ = \mathcal{Q}_2^+\Big(\mathrm{clip}(v_t R_V, \tau_t^{(V)})\Big). \tag{4}$$

with four 2-bit values packed per byte.

New decode tokens will be first written to the window recent window as $k_t R_K$ and $v_t R_V$. As decoding advances, the oldest recent token will be demoted into the INT2 middle region by a fused Triton kernel that applies the same clip–quantize operation. OSCAR then optimize value rotation by absorbing R_V into the model's projection weights, achieving compute saving and latency reduction.

Decoding Attention Kernel. During decoding, OSCAR partitions each request's cache indices into BF16 (sink + recent) and INT2 segments on the GPU. The INT2 kernel unpacks bytes, applies the stored scale/zero parameters, and accumulates them in floating point. Existing decoding attention kernels typically consist of two kernel launches (Dao et al., 2023), one for parallel processing on KV cache segments along the sequence dimension. The second one is used to merge partial attention results from segments with online softmax (Dao et al., 2022). OSCAR introduces an additional kernel for BF16 KV cache attention, then reuses the second merge kernel to piggyback on the merge of high-precision partial results. Since the BF16 (sink + recent) segment has orders of magnitude fewer elements than the INT2 segment, the overhead is negligible.

Table 1: **AIME25 at 32K generation.**

Method	BPE	Qwen3-8B	Qwen3-32B
Original BF16	16.00	66.00±7.33	72.59±7.41
KIVI-KV2 (Liu et al., 2024b)	2.25	52.33±9.00	57.41±9.26
KIVI-KV2* (Liu et al., 2024b)	2.26	57.67±9.00	59.05±12.38
Kitty (Xia et al., 2025)	2.39	59.67±10.33	69.26±9.26
OSCAR	2.38	66.67±3.33	74.00±5.48

5 Experiments

5.1 Experimental Setup

Models & Benchmarks. We evaluate OSCAR on four model configurations: **Qwen3-4B-Thinking-2507**, **Qwen3-8B**, **Qwen3-32B**, and **GLM-4.7-FP8** (Qwen Team, 2025; GLM Team, 2024). These models cover a small reasoning model, a mid-sized dense model, and a frontier-scale model, allowing us to test OSCAR across different levels of INT2 robustness. We evaluate accuracy on five reasoning and coding benchmarks: AIME25 (Mathematical Association of America, 2025),GPQA-Diamond (Rein et al., 2024), HumanEval (Chen et al., 2021a), LiveCodeBench v6 (Jain et al., 2024), and MATH500 (Hendrycks et al., 2021).

We also evaluate long-context retrieval with **RULER** (Hsieh et al., 2024) to test the long-sequence robustness. Particularly, RULER-NIAH is the cleanest stress test: the answer is explicitly present in the prompt, and the main question is whether quantized history tokens can be attended to in a long context.

Hardware and Framework. Qwen3-4B-Thinking and Qwen3-8B are served on a single NVIDIA H100 (80 GB); Qwen3-32B and GLM-4.7-FP8 are served on 2×H100 and 8×H100 with tensor parallelism, respectively. All system-level runs use our SGLang (Zheng et al., 2024) implementation.

Generation Protocol & Calibration. We use temperature T=0.6 (T=1.0), top-p=0.95, and top-k=20 for Qwen3-family (GLM-

4.7), with thinking mode enabled for reasoning benchmarks. Unless otherwise noted, each Qwen configuration is run with **5 independent seeds** and GLM-4.7-FP8 with **3 runs**; we report mean ± standard deviation. All accuracy evaluations use a maximum generation length of **32768 tokens** and run end-to-end inside SGLang, using the same execution path as our system measurements. All OSCAR parameters are estimated once from a small MMLU-style calibration set. For each model, we run one calibration pass and dump per-layer Q, K, V activations (8878 tokens × number of layers), from which we compute the key/value rotations and per-layer clipping thresholds, then reuse the same parameters for all benchmarks. No task-specific calibration is used.

Baselines. We use two baseline groups. (1) *Group A* contains channel-wise KV methods such as KIVI and Kitty. These methods require residual buffers, channel-wise scales, promoted channels, or custom page layouts, and we do not have paged/fused kernels for them at 32K generation length.

For the only shared 32K accuracy setting, we show the reported Qwen3-8B and Qwen3-32B AIME25 results in the Table 1; The BPE values include INT payload, BF16 scale/zero metadata with group size 128, and BF16 initial tokens where applicable; the Kitty row is the 12.5% key-channel boost variant. (2) *Group B* contains rotation-based methods: FP16/BF16, naive INT2/INT4, QuaRot-style Hadamard rotation, block-diagonal Hadamard (Saw-INT4) (Jia et al., 2026), and TurboQuant (vibhavagarwal5, 2026). For TurboQuant, we use the official vLLM implementation from PR #38479 (vibhavagarwal5, 2026) and run the same 32K-generation evaluation; for fairness, we disable its mixed precision setup and quantize all layers.

For OSCAR we always pair rotation with sink (S=64) and recent-window (R=256) BF16 protection, calibration-derived per-layer clip thresholds (whose typical values are $c_K \approx 0.96$ and $c_V \approx 0.92$, see the calibration paragraph above), and per-channel asymmetric INT2 quantization. In Table 2, BPE counts INT payload plus BF16 scale/zero metadata with block size 128. For OSCAR, we report effective BPE at 128K context length, including the BF16 sink/recent tokens (S=64, R=256). LiveCodeBench v6 is evaluated with the same 32K generation cap as the other tasks; this truncates some long code-generation outputs, so the reported LCB numbers are lower than they would be under a longer 128K generation budget.

5.2 Main Results

Table 2 reports the main accuracy comparison on Qwen3-4B-Thinking-2507, Qwen3-8B, Qwen3-32B, and GLM-4.7-FP8.

For OSCAR we use the configuration selected by the ablation studies in Appendix E.2.2: sink=64, recent=256, calibration-derived clip thresholds, attention-aware key/value covariance, and asymmetric INT2 quantization. The key observation is that OSCAR is the only near-2-bit method that remains close to the BF16 accuracy frontier under the 32K-generation evaluation. On Qwen3-4B-Thinking-2507 and Qwen3-8B, OSCAR reduces the BF16 gap to 3.78 and 1.42 points, respectively, while TurboQuant drops 43.90 and 13.96 points and rotation-only INT2 baselines largely collapse. On Qwen3-32B and GLM-4.7-FP8, OSCAR is essentially tied with BF16 at 2.28 BPE, while Saw-INT4 uses 4.25 BPE.

Long-context robustness are reported in Appendix E.2. The ablation studies that select the default OSCAR configuration are also moved to Appendix E.2.2.

5.3 End-to-End Serving Throughput

Kernel-Level Profiling. Per-step decode latency details are reported in Appendix E.5. The profiling isolates the runtime cost of OSCAR and shows that OSCAR reduces attention time by shrinking KV traffic while adding only a small fused quantization cost.

Representative Serving Throughput. Table 3 reports throughput under 32 concurrent requests with 8192-token inputs and 1024-token outputs, giving the per-user and per-GPU throughput used to contextualize the speedup trends below.

Pure Decoding Speed. We first study how effective our kernel is in pure decoding speed. We isolated the prefill time or time to first token (TTFT) by setting a full prefix-cache-hit setting with a batch size 1, where all methods fully reuse the cached KV states

Table 2: INT2 KV-cache quantization methods compared on four model configurations and five benchmarks. Entries are $\mu \pm \sigma$ over 5 seeds, except GLM-4.7-FP8 (3 runs)*. BPE denotes effective bits per KV element at 128K context length; "no MP" means mixed precision is disabled. "Drop" is the average-column gap to the BF16 upper bound; smaller is better.

Model	Method	BPE	GPQA	HumanE	LCB v6	AIME25	MATH500	Mean	Drop
Qwen3-4B-Thinking-2507	BF16	16.00	67.27±1.80	94.05±0.54	48.66±2.20	74.67±1.83	93.55±0.33	75.64	–
	Saw-INT4 (Jia et al., 2026)	4.25	66.37±2.19	89.78±0.80	46.20±1.94	70.00±4.08	93.19±0.51	73.11	-2.53
	TurboQuant (no MP) (Zandieh et al., 2025)	3.25	41.41	31.83	0.58	16.67	68.20	31.74	-43.90
	QuaRot-INT2 (Ashkboos et al., 2024)	2.25	0.34±0.48	0.98±0.10	0.00±0.00	0.00±0.00	5.67±0.19	1.40	-74.24
	Naive INT2	2.25	0.00	0.00	0.00	0.00	0.00	0.00	-75.64
	OSCAR (ours)	2.28	64.95±1.16	92.24±1.02	45.38±1.97	64.00±3.65	92.75±0.39	71.864	-3.78
Qwen3-8B	BF16	16.00	56.67±2.30	85.95±1.01	49.01±2.13	70.00±3.33	92.59±0.62	70.84	–
	Saw-INT4 (Jia et al., 2026)	4.25	54.85±3.17	86.44±0.42	47.95±2.15	68.00±2.98	92.63±0.27	69.97	-0.87
	TurboQuant (no MP) (Zandieh et al., 2025)	3.25	55.05	74.63	21.05	46.67	87.00	56.88	-13.96
	QuaRot-INT2 (Ashkboos et al., 2024)	2.25	14.98±0.63	9.80±0.76	0.58±0.58	2.22±1.57	23.13±1.88	10.14	-60.70
	Naive INT2	2.25	0.00	0.00	0.00	0.00	0.00	0.00	-70.84
	OSCAR (ours)	2.28	55.05±1.47	87.88±0.44	46.32±1.91	66.67±3.33	92.22±0.83	69.416	-1.42
Qwen3-32B	BF16	16.00	58.49±1.21	91.19±1.11	59.06±1.80	68.67±5.58	93.55±0.41	74.19	–
	Saw-INT4 (Jia et al., 2026)	4.25	59.29±0.76	90.85±0.47	56.02±2.41	72.67±4.90	93.31±0.10	74.43	+0.24
	TurboQuant (no MP) (Zandieh et al., 2025)	3.25	58.69	88.41	55.56	66.67	90.60	71.99	-2.20
	QuaRot-INT2 (Ashkboos et al., 2024)	2.25	19.70±1.09	1.83±0.33	0.39±0.55	0.00±0.00	17.60±1.66	7.90	-66.29
	Naive INT2	2.25	0.00	0.00	0.00	0.00	0.00	0.00	-74.19
	OSCAR (ours)	2.28	60.40±3.47	90.12±0.71	53.57±0.67	74.00±5.48	92.75±0.46	74.17	-0.02
GLM-4.7-FP8, 358B	BF16	16.00	73.23±1.33	91.46±0.65	49.12±0.59	80.00±3.33	95.66±0.61	77.89	–
	Saw-INT4 (Jia et al., 2026)	4.25	73.74±1.01	91.30±0.92	50.49±1.47	78.89±1.92	95.32±0.12	77.95	+0.06
	TurboQuant (no MP) (Zandieh et al., 2025)	3.25	66.67	90.24	58.48	80.00	95.39	78.15	+0.26
	QuaRot-INT2 (Ashkboos et al., 2024)	2.25	68.01±3.79	89.87±1.78	48.54±4.79	78.89±1.92	90.40±0.35	75.14	-2.75
	Naive INT2	2.25	54.55±3.54	86.50±1.43	37.03±6.41	37.78±1.92	86.60±0.80	60.49	-17.40
	OSCAR (ours)	2.28	73.57±0.58	91.06±0.25	52.63±1.01	78.89±1.92	94.66±0.76	78.16	+0.27

*TurboQuant entries are single-run results as its vLLM path is too slow for repeated 32K evaluations under our time budget.

Table 3: Representative end-to-end serving throughput under 32 concurrent requests with 8192-token inputs and 1024-token outputs. Qwen3-4B-Thinking and Qwen3-8B are evaluated on 1×H100 (80GB), while GLM-4.7-FP8 is evaluated on 8×H100 (TP=8). U = tokens/s/user, G = tokens/s/GPU. Acc. is averaged over the five benchmarks in Table 2.

	Qwen3-4B-Thinking-2507			Qwen3-8B		
Method	U	G	Acc.	U	G	Acc.
BF16	41.1	1187.9	75.64	35.8	999.9	70.84
Naive INT2	66.4	1797.4	0.00	54.2	1415.1	0.00
QuaRot-INT2 (Ashkboos et al., 2024)	66.3	1795.8	1.40	54.1	1411.9	10.14
Saw-INT4 (Jia et al., 2026)	50.6	1444.1	73.11	43.0	1183.2	69.97
OSCAR (ours)	63.3	1723.9	71.86	52.5	1374.0	69.42

throughout decoding. In this regime, throughput differences directly reflect decode efficiency rather than cache residency or scheduling effects. Figure 10 in Appendix E.6 (left) shows decode throughput normalized to BF16 across three models under varying context lengths. It shows that **(i)** OSCAR consistently outperforms Saw-INT4 (Jia et al., 2026) by up to 2x when the sequence length is 100k, reaching the theoretical bound, verifying the efficiency of our kernel design; **(ii)** The speedup of OSCAR increases with context length, from 1.98× at 30k, 2.52× at 60k, to 3.08× at 100k on Qwen3-4B. This trend scales up to 358B. As such, decoding speed becomes increasingly KV-bandwidth-bound at long contexts. By reducing KV-cache memory by a factor of 8 (BF16 to INT2), OSCAR achieves substantially larger gains as the context length grows, while the online rotation overhead remains effectively hidden within the decode kernels.

Effect of Batch Size. We next study how serving throughput scales with batch size under long-context decoding. Figure 10 in Appendix E.6 (right) reports job-level throughput with prefix warm up at a fixed input length (100k tokens) with batch sizes BS $\in \{1, 8, 32\}$. Prefix warm up is used to simulate popular agentic serving workload with long prefix reuse. Three observations stand out. **(i)** OSCAR consistently outperform BF16 across all batch sizes. On GLM-4.7-FP8, OSCAR achieves a 2.83× speedup over BF16 at BS=1, which further increases to 7.83× at BS=32. This behavior highlights that the reduced KV-cache footprint of INT2 methods becomes increasingly beneficial under highly concurrent long-context serving workloads. **(ii)** OSCAR consistently outperforms Saw-INT4 across all models and batch sizes in Figure 10, showing that the fused on-

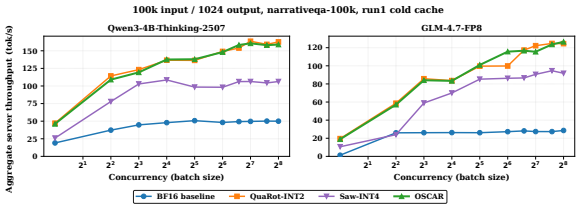


Figure 3: Long-context serving stress test with 100k-token inputs. OSCAR's uniform INT2 KV-cache scales up to 2^8 requests with continued throughput gains.

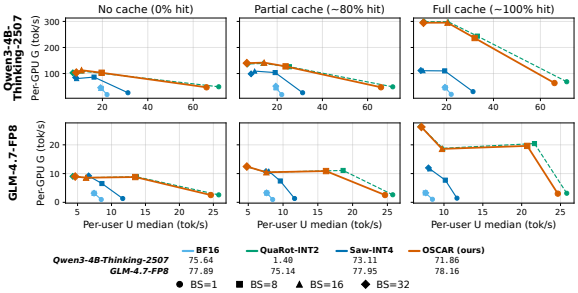


Figure 4: Effect of prefix-cache hit ratio on end-to-end serving throughput (100k ISL, 1K OSL). Each subplot shows median per-user throughput U (x-axis, tok/s) versus mean per-GPU throughput G (y-axis, tok/s), with markers denoting batch sizes BS $\in \{1, 8, 16, 32\}$. Rows correspond to Qwen3-4B-Thinking-2507 and GLM-4.7-FP8. Columns correspond to radix cache disabled, radix cache enabled during normal execution, and immediate replay after warmup (near-100% hit ratio).

line rotation does not introduce an observable throughput penalty. **(iii)** Throughput improves substantially with larger batch sizes as OSCAR can accommodate more concurrent requests with less memory pressure. OSCAR's design can also benefit prefix cache (Zheng et al., 2024) thus improve cache hit rate with better job-level cache reuse.

Serving Throughput with Prefix Cache from 0 to 100%. Figure 4 evaluates end-to-end serving throughput under concurrent multi-user workloads by sweeping prefix-cache hit ratio from cache disabled to near-100% cache hits.

The results show that **(i)** increasing the prefix-cache hit ratio consistently expands the throughput frontier by reducing prefill recomputation, with the largest gains appearing at higher batch sizes; **(ii)** OSCAR remains on or near the efficiency frontier across all cache regimes, closely matching aggressive INT2 baselines while avoiding extra decode-time memory traffic or indirection; **(iii)** compared with BF16, low-bit KV quantization enables substantially higher per-GPU throughput because it improves decode efficiency and reduces KV-cache memory footprint, allowing larger concurrent batches under the same memory budget; and **(iv)** OSCAR achieves the best accuracy among INT2 methods while preserving the standard paged KV-cache abstraction and integrating cleanly into fused SGLang decode kernels, whereas QuaRot-INT2 can approach the throughput ceiling but suffers significant accuracy degradation as shown in Table 2.

Memory and Extreme Batch Scalability. Finally, we study memory usage and batch scalability under long-context serving. We evaluate on a single H100 (80 GB) with 100k-token inputs while scaling the concurrent batch size; detailed scaling curves are provided in the appendix. OSCAR reduces the KV-cache footprint by approximately 8× relative to BF16, enabling substantially larger feasible batch sizes under the same memory budget.

As shown in Figure 6, this stress test demonstrates that OSCAR can scale to very large concurrent batch sizes with inputs under 100k-tokens. Under 100k-token inputs, OSCAR's uniform INT2 KV-cache pages enable substantially larger feasible batch sizes, scaling up to 2^8 concurrent requests on a single H100 while throughput continues to improve. In contrast, BF16 and INT4 baselines either run out of memory at smaller batch sizes or quickly plateau as concurrency increases. This shows that OSCAR's design is not only memory-efficient but also practically scalable to extremely long-context serving regimes. More ablation runs and discussion are in Appendix E.4 and D.

6 Conclusion

We introduced **OSCAR**, an INT2 KV-cache quantization method that chooses rotations from attention-aware target covariance rather than raw cache reconstruction. With one offline calibration pass, OSCAR fixes key/value rotations and clip thresholds, then uses a fused SGLang path to quantize history KV tokens while preserving sink and recent tokens. This simple target-and-kernel co-design recovers much of the BF16 accuracy lost by naive INT2 while keeping the memory and serving advantages of 2-bit KV caches.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pages 100213–100240.

Saleh Ashkboos, Mahdi Nikdan, Soroush Tabesh, Roberto L. Castro, Torsten Hoefler, and Dan Alistarh. 2025. HALO: Hadamard-assisted lower-precision optimization for LLMs. *arXiv preprint arXiv:2501.02625*.

Zefan Cai and 1 others. 2024. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.

Chi-Chih Chang, Chien-Yu Lin, Yash Akhauri, Wei-Cheng Lin, Kai-Chiang Wu, Luis Ceze, and Mohamed S. Abdelfattah. 2025. xKV: Cross-layer SVD for KV-cache compression. *arXiv preprint arXiv:2503.18893*.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. 2024. PALU: Compressing KV-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M. De Sa. 2023. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, and 1 others. 2021a. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Patrick H. Chen, Hsiang-Fu Yu, Inderjit S. Dhillon, and Cho-Jui Hsieh. 2021b. DRONE: Data-aware low-rank compression for large NLP models. *Advances in Neural Information Processing Systems*, 34:29321–29334.

James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. 2023. Flash-decoding for long-context inference. https://pytorch.org/blog/flash-decoding/. PyTorch Blog. Accessed: 2026-05-06.

Haojie Duanmu, Zhihang Yuan, Xiuhong Li, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. 2024. SKVQ: Sliding-window key and value cache quantization for large language models. In *Conference on Language Modeling*.

Vage Egiazarian, Roberto L. Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Noll Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2026. Bridging the gap between promise and performance for microscaling FP4 quantization. In *The Fourteenth International Conference on Learning Representations*.

Ky Fan. 1951. Maximum properties and inequalities for the eigenvalues of completely continuous operators. *Proceedings of the National Academy of Sciences of the United States of America*, 37(11):760–766.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2023. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*.

GLM Team. 2024. ChatGLM: A family of large language models from GLM-130B to GLM-4 all tools. *arXiv preprint arXiv:2406.12793*.

Mohsen Hariri, Alan Luo, Weicong Chen, Shaochen Zhong, Tianyi Zhang, Qifan Wang, Xia Hu, Xiaotian Han, and Vipin Chaudhary. 2025. Quantize what counts: More for keys, less for values. *arXiv preprint arXiv:2502.15075*.

Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. 2024. ZipCache: Accurate and efficient KV cache quantization with salient token identification. In *Advances in Neural Information Processing Systems*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *NeurIPS Datasets and Benchmarks Track*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In

Advances in Neural Information Processing Systems.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, and 1 others. 2024. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Jinda Jia, Jisen Li, Zhongzhu Zhou, Jung Hwan Heo, Jue Wang, Tri Dao, Shuaiwen Leon Song, Ben Athiwaratkun, Chenfeng Xu, Tianyi Zhang, and 1 others. 2026. SAW-INT4: System-aware 4-bit KV-cache quantization for real-world LLM serving. *arXiv preprint arXiv:2604.19157*.

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

Seonggon Kim, Taehyeon Kim, and Eunhyeok Park. 2025a. HaPPI: Efficient KV cache compression with hadamard PCA-based power iteration. OpenReview. Submitted to ICLR 2026.

Seonggon Kim, Juncheol Shin, Seung-taek Woo, and Eunhyeok Park. 2025b. HOT: Hadamard-based optimized training. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4787–4796.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*.

Haoyang LI, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole HU, Wei Dong, Li Qing, and Lei Chen. 2025. A survey on large language model acceleration based on KV cache management. *Transactions on Machine Learning Research*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. SnapKV: LLM knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, Tianqi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. 2024a. MatryoshkaKV: Adaptive KV compression via trainable orthogonal projection. *arXiv preprint arXiv:2410.14731*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024b. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*.

Tengxuan Liu, Shiyao Li, Jiayi Yang, Tianchen Zhao, Feng Zhou, Xiaohui Song, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. 2025. PM-KVQ: Progressive mixed-precision KV cache quantization for long-CoT LLMs. *arXiv preprint arXiv:2505.18610*.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024a. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*.

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

Mathematical Association of America. 2025. AIME 2025: American invitational mathematics examination. https://maa.org/math-competitions/aime.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Akshat Ramachandran, Marina Neseem, Charbel Sakr, Rangharajan Venkatesan, Brucek Khailany, and Tushar Krishna. 2025. ThinKV: Thought-adaptive KV cache compression for efficient reasoning models. *arXiv preprint arXiv:2510.01290*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof Q&A benchmark. *Conference on Language Modeling*.

Utkarsh Saxena and Kaushik Roy. 2025. KVLinC: KV cache quantization with hadamard rotation and linear correction. *arXiv preprint arXiv:2510.05373*.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2023.

OmniQuant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*.

Akshat Sharma, Hangliang Ding, Jianping Li, Neel Dani, and Minjia Zhang. 2024. Minikv: Pushing the limits of llm inference via 2-bit layer-discriminative kv cache. *arXiv preprint arXiv:2411.18077*.

Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Linge Li, Huangqi Yu, and Kehong Yuan. 2025. RotateKV: Accurate and robust 2-bit KV cache quantization for LLMs via outlier-aware adaptive rotations. *arXiv preprint arXiv:2501.16383*.

Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235:48630.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

vibhavagarwal5. 2026. TurboQuant: 2-bit KV cache compression with 4x capacity. https://github.com/vllm-project/vllm/pull/38479. VLLM pull request #38479.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. 2025a. SVD-LLM v2: Optimizing singular value truncation for large language model compression. In *Proceedings of the Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2024. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*.

Yixuan Wang, Haoyu Qiao, Lujun Li, Qingfu Zhu, and Wanxiang Che. 2025b. CommonKV: Compressing KV cache with cross-layer parameter sharing. *arXiv preprint arXiv:2508.16134*.

Haojun Xia, Xiaoxia Wu, Jisen Li, Robert Wu, Junxiong Wang, Jue Wang, Chenxi Li, Aman Singhal, Alay Dilipbhai Shah, Alpay Ariyak, Donglin Zhuang, Zhongzhu Zhou, Ben Athiwaratkun, Zhen Zheng, and Shuaiwen Leon Song. 2025. Kitty: Accurate and efficient 2-bit KV cache quantization with dynamic channel-wise precision boost. *arXiv preprint arXiv:2511.18643*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*.

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

Xianglong Yan, Zhiteng Li, Tianao Zhang, Linghe Kong, Yulun Zhang, and Xiaokang Yang. 2025. RecalKV: Low-rank KV cache compression via head reordering and offline calibration. *arXiv preprint arXiv:2505.24357*.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Leon Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. 2024. CorDA: Context-oriented decomposition adaptation of large language models. *arXiv preprint arXiv:2406.05223*.

Zhihang Yuan, Yuzhang Shang, Yang Song, Qiang Wu, Yan Yan, and Guangyu Sun. 2023. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*.

Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. 2024. WKVQuant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*.

Ted Zadouri, Markus Hoehnerbach, Jay Shah, Timmy Liu, Vijay Thakkar, and Tri Dao. 2026. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling. *arXiv preprint arXiv:2603.05451*.

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

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. 2023. H2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*.

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. 2024. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*.

Zhongzhu Zhou, Fengxiang Bie, Ziyan Chen, Zhenyu Zhang, Yibo Yang, Junxiong Wang, Ben Athiwaratkun, Xiaoxia Wu, and Shuaiwen Leon Song. 2026. CARE: Covariance-aware and rank-enhanced decomposition for enabling multi-head latent attention. In *International Conference on Learning Representations*.

A Additional Details and Theoretical Analysis

A.1 Hadamard Transform

The Hadamard transform is an orthogonal mixing transform with entries of equal magnitude. We use the normalized Walsh-Hadamard matrix H_{Had}, which satisfies $H_{\text{Had}}^\top H_{\text{Had}} = I$. Its dimension is inferred from the key or value vector being transformed. For power-of-two dimensions, it can be defined recursively by $H_1 = [1]$ and

$$H_{2m} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_m & H_m \\ H_m & -H_m \end{bmatrix}.$$

In low-bit LLM quantization, Hadamard transforms are commonly used to spread outlier energy across channels before rounding (Ashkboos et al., 2024). In OSCAR, H_{Had} is composed after the attention-aware covariance rotation to further smooth the rotated coordinates while keeping the transform fixed and efficient.

A.2 Principal Component Analysis

Given a symmetric positive semidefinite matrix $A \in \mathbb{R}^{d\times d}$, principal component analysis (PCA) diagonalizes it via

$$\begin{aligned} A &= V\Lambda V^\top, \\ \Lambda &= \operatorname{diag}(\lambda_1, \dots, \lambda_r, \dots), \\ \lambda_1 &\geq \lambda_2 \geq \cdots \geq 0. \end{aligned}$$

Below we recall the important proposition of PCA (Fan, 1951).

Proposition 1 (Indication of top-r eigenvectors)**.** *For any $1 \leq r \leq d$, let*

$$V_r = [v_1, \dots, v_r] \in \mathbb{R}^{d\times r}.$$

Then

$$V_r = \operatorname{argmax}_{U^\top U = I_r} \operatorname{tr}(U^\top A U),$$

and the optimal value is

$$\max_{U^\top U = I_r} \operatorname{tr}(U^\top A U) = \sum_{i=1}^{r} \lambda_i.$$

Note that in our setting, when A is chosen as a target covariance induced by the attention computation, PCA provides an orthogonal basis aligned with the directions that are most important to preserve under KV compression.

Proof. Let $U \in \mathbb{R}^{d\times r}$ satisfy $U^\top U = I_r$. Since $A = V\Lambda V^\top$, we have

$$\operatorname{tr}(U^\top A U) = \operatorname{tr}(U^\top V \Lambda V^\top U).$$

Define

$$B = V^\top U \in \mathbb{R}^{d\times r}.$$

Because V is orthogonal,

$$B^\top B = U^\top V V^\top U = U^\top U = I_r.$$

Hence the columns of B are orthonormal. Using cyclicity of trace,

$$\operatorname{tr}(U^\top A U) = \operatorname{tr}(B^\top \Lambda B) = \operatorname{tr}(\Lambda B B^\top).$$

Write

$$P = BB^\top \in \mathbb{R}^{d\times d}.$$

Then P is an orthogonal projection matrix of rank r, so

$$0 \leq P_{ii} \leq 1, \qquad \sum_{i=1}^{d} P_{ii} = \operatorname{tr}(P) = r.$$

Therefore

$$\operatorname{tr}(U^\top A U) = \operatorname{tr}(\Lambda P) = \sum_{i=1}^{d} \lambda_i P_{ii}.$$

Set

$$w_i = P_{ii}, \qquad i = 1, \dots, d.$$

Then

$$0 \leq w_i \leq 1, \qquad \sum_{i=1}^{d} w_i = r,$$

and thus

$$\operatorname{tr}(U^\top A U) = \sum_{i=1}^{d} \lambda_i w_i.$$

Since $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$, the quantity $\sum_{i=1}^{d} \lambda_i w_i$ is maximized by assigning full weight to the largest r eigenvalues, namely

$$w_1 = \cdots = w_r = 1, \qquad w_{r+1} = \cdots = w_d = 0.$$

Hence

$$\operatorname{tr}(U^\top A U) \leq \sum_{i=1}^{r} \lambda_i.$$

Now choose $U = V_r$. Then

$$V_r^\top A V_r = V_r^\top V \Lambda V^\top V_r = \operatorname{diag}(\lambda_1, \dots, \lambda_r),$$

so

$$\operatorname{tr}(V_r^\top A V_r) = \sum_{i=1}^{r} \lambda_i.$$

Thus V_r attains the maximum, and therefore

$$V_r \in \operatorname{argmax}_{U^\top U = I_r} \operatorname{tr}(U^\top A U).$$

□

A.3 Target Covariance

Given a matrix $X = [x_1; \dots; x_N] \in \mathbb{R}^{N\times d}$ whose rows are feature vectors, its empirical covariance, assuming the rows have been centered, is

$$C_X = \frac{1}{N} X^\top X = \frac{1}{N} \sum_{i=1}^{N} x_i^\top x_i \in \mathbb{R}^{d\times d}.$$

In this paper, we use the uncentered second moment in the same algebraic form and refer to it as a covariance target. For any direction $u \in \mathbb{R}^d$ with $\|u\|_2 = 1$, the scalar

$$u^\top C_X u = \frac{1}{N} \sum_{i=1}^{N} (x_i u)^2$$

measures the average squared magnitude of the rows of X along direction u. Thus, the eigenvectors of C_X identify directions where the row activations have large energy, and PCA uses these eigenvectors as a basis aligned with the dominant second-order structure of X.

In OSCAR, the covariance target is not chosen only from the raw cached tensors. The reason is that the rotation should reduce the downstream attention error, not merely the Euclidean reconstruction error of K and V. For keys, the empirical logit distortion satisfies

$$\|QK^\top - Q\widehat{K}^\top\|_F^2 = \mathrm{tr}\Big((K - \widehat{K})Q^\top Q (K - \widehat{K})^\top\Big).$$

This identity shows that key errors are weighted by the query covariance $Q^\top Q$. Heuristically, directions with large eigenvalues of $Q^\top Q$ are directions in which the queries have large energy, so quantization error in those directions causes larger perturbations to the attention logits. Therefore, we use $Q^\top Q$ as the key-side target covariance for determining the raw key rotation.

For values, the downstream output distortion satisfies

$$\|SV - S\widehat{V}\|_F^2 = \mathrm{tr}\Big((V - \widehat{V})^\top S^\top S (V - \widehat{V})\Big).$$

Equivalently, the matrix

$$V^\top S^\top SV = (SV)^\top (SV)$$

is the target covariance of the attention-weighted value activations SV. It measures which value feature directions remain large after aggregation by the attention scores. Hence, $V^\top S^\top SV$ is an attention-induced value target covariance, while $V^\top V$ is only the raw-cache covariance of values.

Therefore, $Q^\top Q$ and $V^\top S^\top SV$ are covariance targets in our context: they are second-moment matrices induced by the downstream attention computation. This is why OSCAR estimates rotations from these attention-aware covariance targets rather than from raw-cache targets such as $K^\top K$ and $V^\top V$.

A.4 Intuition of the combination $R_K = U_Q H_{\mathrm{Had}} P_K$

The rotation applied to keys before quantization in OSCAR is the product of three orthogonal factors,

$$R_K \;=\; U_Q \, H_{\mathrm{Had}} \, P_K,$$

where U_Q is the eigenbasis of the key target covariance $C_Q = Q^\top Q$ (Sec. A.3), H_{Had} is the normalized Walsh–Hadamard matrix (Sec. A.1), and P_K is a permutation matrix that we describe below. The same construction is used for R_V with $C_S = V^\top S^\top SV$ in place of C_Q. This subsection explains, before proceeding to the formal proof in Sec. A.6, why each factor is needed and why they appear in this order. Each factor addresses a distinct failure mode of per-group low-bit quantization.

Two effects of R_K. Recall from Sec. A.3 that the rotated cache $\tilde{K} = KR_K$ is what the quantizer sees, and the resulting attention-logit distortion is

$$\|QK^\top - Q\widehat{K}^\top\|_F^2 \;=\; \mathrm{tr}\Big(R_K^\top C_Q R_K \cdot E_K\Big),$$

where E_K is the residual covariance of $\tilde{K} - \mathcal{Q}(\tilde{K})$. The rotation enters in two distinct places:

(i) it shapes $R_K^\top C_Q R_K$, the query-importance metric in the rotated coordinates;

(ii) it shapes E_K, because the quantizer's per-token, per-group min–max scale (Sec. A.5) depends on the channelwise distribution of $\tilde{K}$.

The three factors U_Q, H_{Had}, P_K each address one of these effects in turn.

U_Q: align channels with query-importance directions. Eigendecompose

$$C_Q = U_Q \Lambda_Q U_Q^\top,$$
$$\Lambda_Q = \mathrm{diag}(\lambda_1, \ldots, \lambda_d), \qquad \lambda_1 \geq \cdots \geq \lambda_d \geq 0.$$

Setting $R_K = U_Q$ diagonalizes the importance metric:

$$R_K^\top C_Q R_K \;=\; \Lambda_Q.$$

Geometrically, the j-th channel of $\tilde{K} = KU_Q$ is the projection of each key onto the j-th query principal direction u_j, and $\lambda_j = \|Qu_j\|_2^2$ measures how much total query energy lies along u_j. Quantization noise on channel j then propagates to the attention logits with weight λ_j: large-λ channels are exactly those whose error is most amplified, while small-λ channels can absorb error almost for free. Sec. A.6 formalizes this via a rearrangement-inequality argument on the surrogate $\tilde{\mathcal{L}}_K(R_k) = \mathrm{tr}(R_k^\top C_Q R_k E_K)$: under the frozen-residual assumption, $R_K = U_Q$ is the orthogonal rotation that minimizes the importance-weighted error.

This is the right first step but it leaves a per-group quantization problem. The eigenspectrum Λ_Q of C_Q in real LLM activations is sharply anisotropic: a few eigenvalues carry most of $\mathrm{tr}(C_Q)$. After $R_K = U_Q$, the rotated key channels inherit this imbalance, with a few channels having much larger variance than the rest. A per-group min–max quantizer with a single scale per group then sets its scale based on the largest channel in each group, and the smaller channels in the same group are quantized at a granularity far coarser than their dynamic range. Hence E_K has a few very large diagonal entries and many tiny ones, and $\mathrm{tr}(E_K)$ is dominated by the worst groups.

H_{Had}: equalize on the importance metric, suppress outliers on K. The U_Q step is chosen for the surrogate, which is a Q-side criterion ($C_Q = Q^\top Q$). The Hadamard composition serves two complementary purposes: on the surrogate side, it equalizes the rotated importance metric across channels; on the actual K distribution, it mixes outlier energy across channels so that the per-group min–max quantizer (Sec. A.5) sees comparable per-channel magnitudes. The first claim is exact; the second is what the standard outlier-suppression literature relies on (Ashkboos et al., 2024) and only requires that the Hadamard column weights all have equal magnitude.

The exact statement on the importance metric is the following.

Lemma 1 (Diagonal equalization under Hadamard conjugation)**.** *Let $\Lambda \in \mathbb{R}^{d\times d}$ be diagonal and H_{Had} the normalized Walsh–Hadamard matrix. Then*

$$\left(H_{\mathrm{Had}}^\top \Lambda\, H_{\mathrm{Had}}\right)_{ii} \;=\; \frac{1}{d}\mathrm{tr}(\Lambda), \qquad i = 1, \ldots, d.$$

Proof. Every entry of H_{Had} has magnitude $1/\sqrt{d}$. Then $(H_{\mathrm{Had}}^\top \Lambda H_{\mathrm{Had}})_{ii} = \sum_j H_{\mathrm{Had}}[j,i]^2\, \Lambda_{jj} = \frac{1}{d}\sum_j \Lambda_{jj} = \frac{1}{d}\mathrm{tr}(\Lambda)$, independent of i. □

Applying the lemma to $\Lambda = \Lambda_Q$ (which is diagonal because of the U_Q step) gives

$$\begin{aligned}\left(R_K^\top C_Q R_K\right)_{ii} &= \left(H_{\mathrm{Had}}^\top \Lambda_Q H_{\mathrm{Had}}\right)_{ii} \\ &= \frac{1}{d}\mathrm{tr}(C_Q), \qquad \forall\, i.\end{aligned}$$

The peaky eigenspectrum that U_Q exposed has been replaced by a uniform diagonal: every channel now carries the same importance weight $\mathrm{tr}(C_Q)/d$ in the rotated frame.

The K-side effect is heuristic but practically important. After $R_K = U_Q H_{\mathrm{Had}}$, each row of $\tilde{K} = KR_K$ is a Walsh-signed sum of the principal-direction projections KU_Q, with every entry of weight $\pm 1/\sqrt{d}$. A per-token outlier on any single principal direction is thus redistributed in magnitude $1/\sqrt{d}$ across all d channels, shrinking the per-token max by roughly the same factor and bringing the per-group min–max scale into a much tighter range. The frozen residual E_K becomes much smaller in trace and far more uniform across i. The price paid is that the importance metric is no longer diagonal: $R_K^\top C_Q R_K = H_{\mathrm{Had}}^\top \Lambda_Q H_{\mathrm{Had}}$ has off-diagonals that couple channels. In the regime where per-group quantization noise dominates $\mathrm{tr}(E_K)$, the reduction in $\mathrm{tr}(E_K)$ wins over the loss of importance alignment.

Empirical aside: U_Q does *not* accidentally diagonalize Σ_K. One might hope that, because W_Q and W_K are trained on the same activation distribution, $C_Q = Q^\top Q$ and $\Sigma_K = K^\top K$ would share an eigenbasis, so that U_Q would simultaneously serve both the Q-side (importance) and K-side (variance) objectives.

Empirically, this is not the case. On Qwen3-8B with 2000 calibration tokens per layer, the top-8 eigenvector self-alignment $\sum_{k=1}^{8} |U_Q^{Q\top} U_Q^K|/8$ ranges from 0.05 to 0.15 across the 36 layers — essentially indistinguishable from the random alignment $1/\sqrt{d} \approx 0.09$ expected at $d = 128$. Rotating Σ_K by U_Q typically *reduces* the fraction of Σ_K's energy concentrated on the diagonal (e.g., layer 1: 0.90 raw vs. 0.09 after U_Q rotation; layer 16: 0.16 vs. 0.07). Figure 5 shows a representative layer.

The construction is nevertheless robust because the two factors target different things. Lemma 1 guarantees exact diagonal equalization of the importance metric C_Q regardless of the eigenbasis of Σ_K, since the lemma only relies on Λ_Q being diagonal in the post-U_Q frame. The K-side outlier-suppression effect of H_{Had} described above also does not require alignment between U_Q and the eigenbasis of Σ_K: the Walsh-signed mixing shrinks per-token outliers by a factor of $1/\sqrt{d}$ no matter which orthogonal basis K was placed in beforehand. The Q-side (U_Q) and K-side (H_{Had}) factors thus address genuinely independent objectives and compose without interference, rather than relying on an approximate $W_Q \approx W_K$ similarity that does not hold in practice.

P_K: balance importance across groups. Hadamard mixing equalizes *marginal* channel variance, but the per-group quantizer in Sec. A.5 acts on contiguous blocks of G_K channels at a time. What matters at the group level, beyond marginal variance, is whether each group sees a balanced sample of the underlying importance directions: a group whose G_K channels are dominated by the same handful of large-λ_j eigenvectors is statistically harder to quantize than a group whose channels span the full range of λ_j.

OSCAR uses a *permuted bit-reversal* (PBR) ordering. Let σ be the descending-eigenvalue permutation, so that $\lambda_{\sigma(0)} \geq \lambda_{\sigma(1)} \geq \cdots$, and let β be the bit-reversal permutation on $\{0, 1, \ldots, d-1\}$. The permutation matrix P_K places the eigenvector with the k-th largest eigenvalue at position $\beta(k)$, i.e. $(P_K)_{:,\beta(k)} = e_{\sigma(k)}$. For $d = 128$, this yields the placement

$$\begin{aligned} &\text{top-1} \mapsto 0, && \text{top-2} \mapsto 64, \\ &\text{top-3} \mapsto 32, && \text{top-4} \mapsto 96, \\ &\text{top-5..8} \mapsto 16, 80, 48, 112, \ldots \end{aligned}$$

The defining property of bit-reversal is that for any power-of-two group size $G \mid d$, the top-d/G eigenvectors land in d/G *distinct* groups, exactly one per group; the same recursive balance holds at every coarser binary level. Composed with the Hadamard, this means each per-group quantization block sees one representative from each level of the importance hierarchy, regardless of which group size G_K is chosen at deployment.

Empirical aside: errors separate after attention. Figure 6 compares layer-wise errors on Qwen3-4B-Thinking-2507, measured on raw KV tensors and on the downstream quantities consumed by attention. OSCAR is not designed to minimize plain Euclidean reconstruction error of K or V, so its raw K-MSE and V-MSE are not always dramatically smaller than rotation-only baselines. The gap becomes much clearer after the attention computation: the errors in $QK^\top$ and SV are substantially lower, and the advantage further propagates to attention-output errors. This is exactly what the covariance targets optimize: the key rotation is aligned with query-side covariance, while the value rotation is aligned with the score-weighted value covariance. The trend is consistent with the U_K intuition above: aligning channels with query-importance directions need not minimize raw K error, but it directly lowers the error after multiplication by Q; analogously, the value target is chosen so that the advantage appears after aggregation by S.

Worked example: Qwen3-4B-Thinking, layer 10, $d = 128$. We illustrate the three factors on real activations rather than a stylized spectrum. Calibration sample: layer 10 of Qwen3-4B-Thinking-2507 (32 query heads, 8 key/value heads, GQA ratio = 4), $T = 8000$ tokens of GPQA-diamond context. We use the production importance metric — Q-side query covariance, averaged within each GQA group

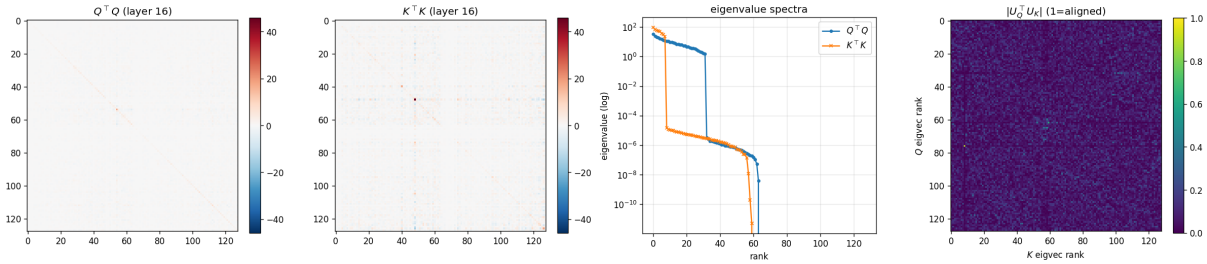


Figure 5: Layer 16 of Qwen3-8B. From left: heatmap of $Q^\top Q$; heatmap of $K^\top K$ (same colour scale); eigenvalue spectra; absolute eigenvector alignment $|U_Q^\top U_Q|$ with both bases sorted by descending eigenvalue. A diagonal-bright alignment matrix would indicate a shared eigenbasis; the observed uniform-texture pattern shows the eigenbases are nearly uncorrelated.

of 4 query heads and then across all 8 kv-heads,

$$C_Q = \frac{1}{H_{\text{kv}}} \sum_{h=1}^{H_{\text{kv}}} \frac{1}{T \cdot g} \sum_{i \in \mathcal{G}_h} Q_i^\top Q_i,$$

The decode-time per-group min–max quantizer uses INT2 with $G_K = 64$ (two groups per head); we deliberately pick $G_K < d$ here so the PBR step has a non-trivial effect (see Step 3 for the $G_K = d$ degenerate case). The K activations we display are from kv-head 0; the shared rotation $R_K = U_Q H_{\text{Had}} P_K$ is the one produced by `compute_qqt`.

Spectrum at layer 10. Eigendecomposing $C_Q = U_Q \Lambda_Q U_Q^\top$ yields an anisotropic but not pathologically peaky spectrum (sorted descending),

$$\begin{aligned} \lambda_{1\ldots4} &\approx (162.5,\ 85.5,\ 35.6,\ 17.4), \\ \lambda_{5\ldots8} &\approx (11.6,\ 7.5,\ 6.1,\ 5.0), \\ \lambda_{121\ldots124} &\approx (0.24,\ 0.22,\ 0.22,\ 0.20), \\ \lambda_{125\ldots128} &\approx (0.18,\ 0.05,\ 0.027,\ 0.009). \end{aligned}$$

The trace is $\text{tr}(C_Q) \approx 443$. The top eigendirection λ_1 alone accounts for 36.6% of the trace and the top eight carry 74.7%: $\lambda_1/\bar{\lambda} \approx 46.9$ where $\bar{\lambda} = \text{tr}(C_Q)/d = 3.46$. Importance is concentrated in a low-rank subspace, but not as extremely as one might guess from K's own variance — instead, covariance $Q^\top Q$ reflects what the attention layer *actually queries*, which is a softer concentration than where K's variance happens to land.

One token, all 128 channels, split into two $G_K = 64$ groups. The INT2 per-group min–max quantizer fits its scale to $\big(\max_{i \in g} \tilde{K}_{t,i} - \min_{i \in g} \tilde{K}_{t,i}\big)/3$ per token t per group g. We visualize what the quantizer sees by printing all 128 channels for a single typical non-sink token ($t = 5$, kv-head 0), reshaped into an 8×16 grid (channel j at grid position $(\lfloor j/16 \rfloor,\ j \bmod 16)$). The horizontal rule splits the grid into the two $G_K = 64$ groups: rows 0–3 are group 0 (channels 0–63), rows 4–7 are group 1 (channels 64–127). Each matrix is annotated with the per-group max–min.

The raw token in the channel basis:

The two groups are wildly unbalanced: group 0 holds both outliers (-30.62 at $j = 50$ and 14.19 at $j = 42$); group 1 is benign. Group 0's INT2 scale must fit a range of 44.81, wasting two bits on a few extreme values while the other ~ 60 channels of that group quantize at granularity an order of magnitude coarser than their actual dynamic range.

Step 1: $\tilde{K} = KU_Q$ (eigenbasis rotation). Projecting K onto the $Q^\top Q$ eigendirections gives:

The raw outlier hasn't been destroyed; it has been *redistributed* into the $Q^\top Q$ eigenbasis. Because U_Q is built from $Q^\top Q$, not $K^\top K$, the K-spike at $j = 50$ does not project onto a single U_Q direction — it spreads across the lower-eigenvalue tail with its largest projection landing in the bottom-rank eigenchannels (the matrix's bottom-right entries: $29.77, 15.28, -8.07, \ldots$). The diagonal of $U_Q^\top C_Q U_Q$ is exactly Λ_Q, so the per-channel *importance* is now exactly the eigenvalue, but the K-variance has been pushed into low-importance channels with no improvement in per-group dynamic range — in fact the imbalance between groups has flipped and grown: group 1's range jumps from 7.19 (raw) to 37.85, while group 0's drops from 44.81 to 7.16. Per-group INT2 is now harder, not easier.

Step 2: $\tilde{K} = KU_Q H_{\text{Had}}$ (Walsh-Hadamard

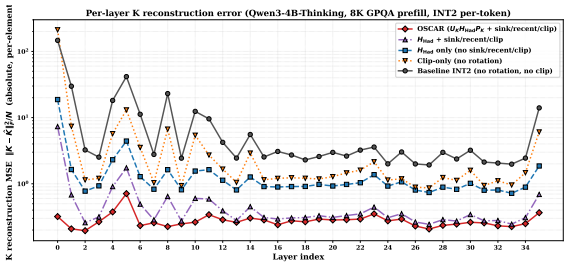


Figure 6a: K MSE

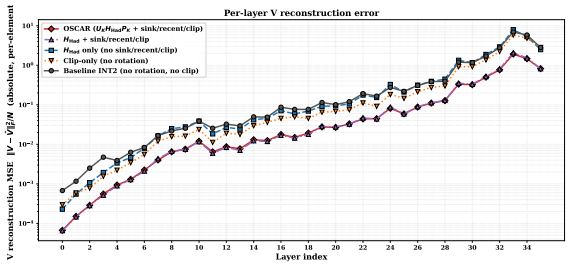


Figure 6b: V MSE

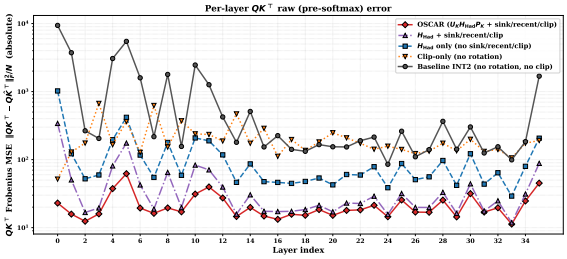


Figure 6c: $QK^\top$ MSE

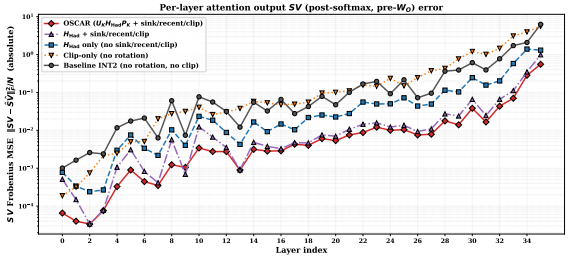


Figure 6d: SV MSE

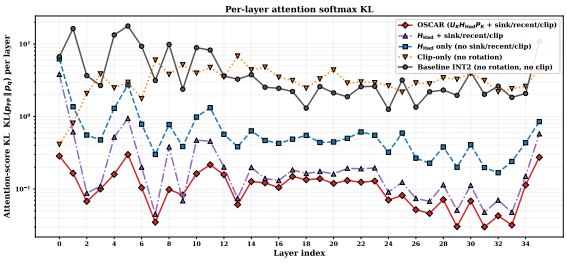


Figure 6e: Attention KL

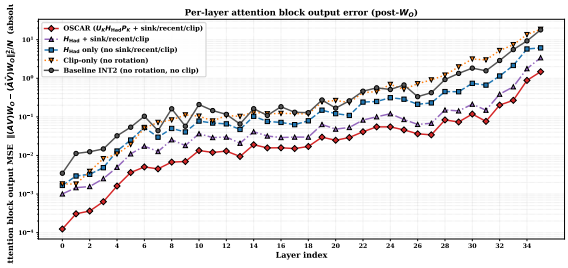


Figure 6f: Attention-output MSE

Figure 6: **OSCAR's advantage appears on attention-consumed quantities on Qwen3-4B-Thinking-2507.** Raw K/V reconstruction MSE alone does not fully explain downstream quality, because OSCAR does not optimize raw-cache reconstruction. Instead, it uses query- and score-aware covariance targets, which more directly reduce $QK^\top$, SV, and attention-output errors.

mixing). Multiplying by the orthonormal Walsh-Hadamard $H_{\text{Had}} \in \mathbb{R}^{128\times128}$ rewrites each output channel as a ±1-weighted sum of all 128 eigenchannels (each scaled by $1/\sqrt{128}$). The single-channel spike of 29.77 is distributed across all 128 outputs with mixed signs:

Every one of the 128 entries now sits in a tight band $[-6.79, 7.03]$, the two groups are within 0.5% of each other in dynamic range, and the absolute range collapses from 37.85 (Step 1) to 13.58 — a 2.8× reduction. The importance metric also equalizes exactly: Lemma 1 applied to Λ_Q predicts every diagonal entry of $H_{\text{Had}}^\top \Lambda_Q H_{\text{Had}}$ collapses to the constant $\text{tr}(\Lambda_Q)/d = 443/128 \approx 3.46$, regardless of how peaky Λ_Q was. Table 4 confirms $\max_i (R_K^\top C_Q R_K)_{ii}/\text{mean}_i(\cdot) = 1.00$ exactly: *the* 46.9× *peak in the* $Q^\top Q$ *importance is compressed to* 1.00×. This is the key invariant Step 2 produces; the per-group dynamic range alone is not yet better than pure Hadamard (compare to Step 5 below).

Step 3: $\tilde{K} = KU_Q H_{\text{Had}} P_K$ *(permuted bit-reversal).* P_K is a permutation matrix that reorders the 128 output channels. After $U_Q H_{\text{Had}}$ each Walsh output is a ±1-weighted sum of all eigenchannels, but the variances of those sums are not identical (driven by which Walsh row a high-eigenvalue eigvec activates most strongly in). PBR places the k-th largest eigenvector at Walsh output index $\beta(k)$ (the bit-reversal of k). At $G_K = 64$ this guarantees the top-2 eigenvectors split into the two groups (one each), the top-4 distribute one-per-quartile, and so on for any power-of-two grouping, so no single group inherits a disproportionate share of the high-energy eigenchannels.

The *set* of 128 values is identical to Step 2 — only column-index positions are shuffled — but P_K has gath-

$$K_{t,\cdot} = \left[\begin{array}{rrrrrrrrrrrrrrrr}
3.42 & -0.61 & 1.59 & 0.27 & -3.14 & -0.05 & 3.55 & 1.09 & 2.05 & -1.05 & 1.86 & -0.03 & -0.42 & -0.89 & -1.54 & 2.06 \\
-1.74 & -0.15 & -0.39 & -0.89 & -0.02 & 2.55 & 0.11 & 1.79 & -0.28 & 0.25 & -1.40 & -6.09 & -1.04 & -0.12 & 0.64 & -0.95 \\
9.06 & 1.17 & 0.08 & -1.49 & 0.29 & 1.13 & 0.47 & 0.17 & 1.22 & 0.55 & 14.19 & -0.26 & -0.09 & 0.20 & 2.39 & 0.40 \\
1.09 & -0.06 & \mathbf{-30.62} & 0.00 & 1.03 & 0.57 & -1.30 & 2.47 & -0.11 & -0.08 & 0.56 & -0.47 & 0.45 & -0.88 & 2.31 & -1.05 \\
\hline
0.50 & 0.44 & 0.10 & -0.35 & -0.78 & 1.46 & -1.16 & 2.56 & 1.64 & -0.09 & 0.90 & 2.33 & 1.09 & 2.72 & 0.95 & 0.20 \\
1.30 & 1.82 & 0.54 & 1.27 & -1.05 & 0.50 & -4.47 & 1.20 & 0.32 & 1.36 & -0.81 & -0.04 & 2.27 & -0.77 & 0.64 & 1.27 \\
0.03 & -0.20 & 0.41 & 0.58 & 0.79 & 1.05 & 0.46 & -0.09 & 0.49 & 0.29 & -0.02 & 0.20 & 1.36 & 0.77 & -1.01 & -2.33 \\
1.19 & -1.77 & -0.01 & -0.49 & 0.47 & 0.91 & 0.42 & 0.17 & 1.38 & -0.39 & -0.00 & 0.43 & -0.98 & -0.50 & 0.17 & -0.50
\end{array}\right]$$

group 0: $\max - \min = \mathbf{44.81}$ | group 1: $\max - \min = \mathbf{7.19}$.

$$(KU_Q)_{t,\cdot} = \left[\begin{array}{rrrrrrrrrrrrrrrr}
0.16 & -1.75 & 1.86 & -2.42 & 1.11 & 1.05 & -0.69 & 0.26 & 1.11 & -0.55 & 3.39 & -1.80 & 1.78 & 1.65 & -1.49 & -1.32 \\
0.05 & -0.18 & -2.00 & -1.27 & 3.20 & -1.47 & 0.31 & -1.07 & 0.87 & -1.54 & -3.78 & -0.89 & 1.60 & 1.03 & 0.50 & 0.55 \\
0.49 & 0.55 & -0.90 & -0.05 & 0.17 & 0.20 & 0.78 & -1.21 & -1.06 & -1.70 & 0.63 & 0.12 & -0.28 & 2.99 & 2.88 & -0.97 \\
-1.42 & 2.12 & -0.94 & -2.88 & 1.30 & -0.42 & 1.37 & 0.58 & -1.46 & -2.04 & 0.28 & -0.28 & -2.63 & -0.20 & -1.77 & -0.83 \\
\hline
0.50 & 0.12 & -1.19 & 0.81 & -1.90 & 0.40 & -1.31 & -0.15 & -2.31 & 1.52 & 3.20 & -0.46 & -0.77 & -0.09 & -1.34 & 1.72 \\
0.40 & -0.72 & -0.57 & 0.40 & 0.84 & -0.72 & -1.94 & -0.11 & 1.26 & 0.07 & 0.09 & -0.06 & 0.87 & -0.92 & -0.63 & 1.19 \\
-2.31 & 1.56 & 1.90 & 1.26 & 1.16 & -0.72 & -0.56 & -0.48 & 1.21 & 0.20 & -0.26 & -1.78 & 1.71 & -1.28 & 1.07 & 1.27 \\
-3.32 & 0.47 & -0.43 & -1.77 & -1.42 & 1.42 & 4.99 & -1.91 & -0.25 & -1.02 & 0.51 & -1.65 & -1.11 & -8.07 & 15.28 & \mathbf{29.77}
\end{array}\right]$$

group 0: $\max - \min = \mathbf{7.16}$ | group 1: $\max - \min = \mathbf{37.85}$.

ered most of the high-magnitude entries ($\{7.03, -6.54, -6.79, -6.06, 6.89, 6.73, \ldots\}$) into group 0 (rows 0–3) and left group 1 (rows 4–7) with a much tighter $[-5.15, 4.21]$ range. On this particular token PBR has increased one group's dynamic range slightly ($13.52 \to 13.82$) while shrinking the other's ($13.58 \to 9.36$); the residual on the tighter group drops as $(9.36/13.58)^2 \approx 0.48$ while the looser group is essentially unchanged. Averaged over all $T = 8000$ tokens, this gathering effect reduces the mean per-group max–min from 13.40 ($U_Q H_{\text{Had}}$) to 11.58 (OSCAR), translating to a 1.23× reduction in $\text{tr}(E_K)$ ($208 \to 169$). At this layer, *PBR carries most of OSCAR's quantization-residual improvement* over $U_Q H_{\text{Had}}$ alone; the role of $U_Q H_{\text{Had}}$ is to make Lemma 1 fire so that the importance metric is equalized exactly ($\max/\text{mean} = 1.00$).

Degenerate case ($G_K = d$). pure Hadamard, QuaRot-style. A natural simpler alternative is to drop U_Q entirely and rotate by H_{Had} alone:

Hadamard mixing on its own balances the two groups ($13.12 \approx 14.01$) and on this token gives a per-group dynamic range comparable to $U_Q H_{\text{Had}}$ (Step 2). The crucial structural difference is in the importance metric: C_Q is not diagonal in the channel basis, so Lemma 1 does not apply to $H_{\text{Had}}^\top C_Q H_{\text{Had}}$. We measure $\max/\text{mean} = 1.72$ rather than 1.00 — the importance *floor* is not equalized.

The two "what the quantizer sees" columns make the OSCAR advantage on quantization residual concrete. Pure Hadamard cuts the per-group max–min from 25.7 (raw) to 13.3 (1.9×); OSCAR cuts it further to 11.6 (2.2× over raw). Since INT2 residual scales as the square of dynamic range, the $\text{tr}(E_K)$ ratios should be $(25.7/11.6)^2 \approx 4.9$ and $(13.3/11.6)^2 \approx 1.3$, matching empirical $233/169 = 1.4$ and $206/169 = 1.2$ (the empirical ratios are smaller than the squared dynamic range because the per-token correlation structure of the residual matters; the relation is only proportional, not exact). PBR alone contributes the bulk of OSCAR's advantage over $U_Q H_{\text{Had}}$ at this layer ($\text{tr}(E_K)$: $208 \to 169$, 1.23×). The importance column tells the orthogonal story: only the $U_Q H_{\text{Had}}$ and OSCAR rows achieve $\max/\text{mean} = 1.00$, the exact diagonal equalization predicted by Lemma 1; pure Hadamard alone leaves it at 1.72, and raw / U_Q alone leave it at 43–47.

Summary. The three factors are not interchangeable, and each provides a guarantee the others cannot:

- U_Q exposes the importance spectrum by diagonalizing C_Q, so per-channel importance becomes exactly λ_j. By itself it makes per-group dynamic range *worse* on real activations because the K-variance gets pushed into low-importance directions, not flattened.
- H_{Had}, composed with U_Q, equalizes the importance-metric diagonal to a constant $\text{tr}(C_Q)/d$ via Lemma 1 ($\max/\text{mean} = 1.00$ exactly). Without the U_Q pre-step (pure Hadamard alone), the importance metric is not equalized ($\max/\text{mean} = 1.72$ here), so quantization error in any single rotated channel can land disproportionately on a high-importance direction in C_Q depending on

$$
(KU_QH_{\text{Had}})_{t,\cdot} = \left[\begin{array}{rrrrrrrrrrrrrrrr}
2.24 & 1.08 & -3.73 & 0.26 & -5.76 & 0.92 & 4.08 & -3.39 & -3.72 & 0.29 & 5.55 & -2.67 & 3.19 & -3.70 & -4.16 & 1.80 \\
-0.87 & 0.64 & 4.12 & -3.18 & 5.12 & 1.16 & -6.79 & 0.97 & 1.91 & -2.08 & -4.91 & 4.21 & -1.67 & 1.97 & 4.47 & -0.63 \\
-2.86 & 0.66 & 6.73 & 1.83 & 4.37 & 1.49 & -5.77 & 0.72 & 1.52 & -0.05 & -4.94 & 3.97 & -3.09 & 2.28 & 4.34 & -0.37 \\
1.90 & -1.30 & -6.06 & -1.90 & -2.58 & 3.27 & 4.18 & -3.10 & -2.31 & 0.88 & 5.04 & -2.61 & -0.07 & -1.10 & -2.17 & 4.09 \\
\hline
-3.71 & 2.31 & 6.89 & -1.79 & 1.12 & -0.12 & -5.61 & 3.44 & 4.10 & 0.85 & -3.94 & 3.17 & -3.51 & 0.62 & \mathbf{7.03} & -5.15 \\
4.11 & 1.23 & -4.84 & 0.19 & -3.60 & 2.16 & 3.36 & -0.01 & -4.31 & 0.17 & 3.97 & -2.02 & 2.49 & -0.04 & -4.11 & 2.57 \\
3.72 & 2.20 & -3.51 & 1.67 & -5.30 & -0.01 & 4.53 & -2.49 & -3.30 & 0.32 & 2.06 & -4.40 & 3.14 & -1.10 & \mathbf{-6.54} & 1.40 \\
-2.85 & 1.72 & 5.21 & -0.32 & 4.97 & 1.00 & -5.68 & -1.24 & 3.87 & -1.45 & -5.40 & 1.67 & -4.30 & 1.20 & 3.73 & -3.39
\end{array}\right]
$$

group 0: $\max - \min = \mathbf{13.52}$ | group 1: $\max - \min = \mathbf{13.58}$.

$$
(KU_QH_{\text{Had}}P_K)_{t,\cdot} = \left[\begin{array}{rrrrrrrrrrrrrrrr}
2.24 & -3.71 & -2.86 & 3.72 & -0.87 & 4.11 & 1.90 & -2.85 & -3.72 & 4.10 & 1.52 & -3.30 & 1.91 & -4.31 & -2.31 & 3.87 \\
-5.76 & 1.12 & 4.37 & -5.30 & 5.12 & -3.60 & -2.58 & 4.97 & 3.19 & -3.51 & -3.09 & 3.14 & -1.67 & 2.49 & -0.07 & -4.30 \\
-3.73 & 6.89 & 6.73 & -3.51 & 4.12 & -4.84 & -6.06 & 5.21 & 5.55 & -3.94 & -4.94 & 2.06 & -4.91 & 3.97 & 5.04 & -5.40 \\
4.08 & -5.61 & -5.77 & 4.53 & -6.79 & 3.36 & 4.18 & -5.68 & -4.16 & \mathbf{7.03} & 4.34 & \mathbf{-6.54} & 4.47 & -4.11 & -2.17 & 3.73 \\
\hline
1.08 & 2.31 & 0.66 & 2.20 & 0.64 & 1.23 & -1.30 & 1.72 & 0.29 & 0.85 & -0.05 & 0.32 & -2.08 & 0.17 & 0.88 & -1.45 \\
0.92 & -0.12 & 1.49 & -0.01 & 1.16 & 2.16 & 3.27 & 1.00 & -3.70 & 0.62 & 2.28 & -1.10 & 1.97 & -0.04 & -1.10 & 1.20 \\
0.26 & -1.79 & 1.83 & 1.67 & -3.18 & 0.19 & -1.90 & -0.32 & -2.67 & 3.17 & 3.97 & -4.40 & 4.21 & -2.02 & -2.61 & 1.67 \\
-3.39 & 3.44 & 0.72 & -2.49 & 0.97 & -0.01 & -3.10 & -1.24 & 1.80 & -5.15 & -0.37 & 1.40 & -0.63 & 2.57 & 4.09 & -3.39
\end{array}\right]
$$

group 0: $\max - \min = \mathbf{13.82}$ | group 1: $\max - \min = \mathbf{9.36}$.

the layer/token distribution.

- P_K ensures that for any power-of-two group size $G_K < d$, the importance hierarchy is recursively balanced across groups. This is where most of OSCAR's empirical $\text{tr}(E_K)$ reduction over U_QH_{Had} comes from on real activations. When $G_K = d$ it is a column permutation within a single group and has no effect; the PBR step is strictly beneficial only for finer grouping.

The same three-factor construction defines $R_V = U_SH_{\text{Had}}P_V$ on the value side, with $C_S = V^\top S^\top SV$ (the attention-score-weighted value covariance) in place of C_Q.

A.5 Scale Determination and Per-token Clipping Details

Quantization is applied after the composed rotations in Eq. (3). At decoding step t, the key and value rows are first mapped into the rotated coordinates

$$\widetilde{k}_t = k_tR_K, \qquad \widetilde{v}_t = v_tR_V.$$

Before quantization, the implementation applies token-wise percentile clipping separately to keys and values. Given clip ratios $\rho_K, \rho_V \in (0, 1]$, define

$$
\begin{aligned}
\tau_t^{(K)} &= \text{quantile}_{\rho_K}\left(\{|\widetilde{k}_{t,c}|\}_{c=1}^d\right), \\
\tau_t^{(V)} &= \text{quantile}_{\rho_V}\left(\{|\widetilde{v}_{t,c}|\}_{c=1}^d\right).
\end{aligned}
$$

The clipped rows are

$$
\begin{aligned}
\widetilde{k}_t^{\text{clip}} &= \text{clip}\left(\widetilde{k}_t, -\tau_t^{(K)}, \tau_t^{(K)}\right), \\
\widetilde{v}_t^{\text{clip}} &= \text{clip}\left(\widetilde{v}_t, -\tau_t^{(V)}, \tau_t^{(V)}\right).
\end{aligned}
$$

In the current implementation, the default clip ratios are supplied as fixed runtime/calibration hyperparameters, with typical values $\rho_K = 0.96$ and $\rho_V = 0.92$.

For keys, the default implementation uses asymmetric INT2 quantization, optionally with groups along the channel dimension. Let $b = 2$ and $q_{\max} = 2^b - 1 = 3$. For a group g of a clipped rotated key row, write

$$\widetilde{k}_{t,g}^{\text{clip}} \in \mathbb{R}^{1\times G_K},$$

where G_K is the key quantization group size. The implementation computes a dynamic min–max scale and zero point for each token and group:

$$
a_{t,g}^{(K)} = \min_c \widetilde{k}_{t,g,c}^{\text{clip}}, \qquad b_{t,g}^{(K)} = \max_c \widetilde{k}_{t,g,c}^{\text{clip}},
$$

$$
s_{t,g}^{(K)} = \frac{b_{t,g}^{(K)} - a_{t,g}^{(K)}}{q_{\max}}, \qquad z_{t,g}^{(K)} = -\frac{a_{t,g}^{(K)}}{s_{t,g}^{(K)}}.
$$

The quantized and dequantized key group is then

$$
\mathcal{Q}^+\left(\widetilde{k}_{t,g}^{\text{clip}}\right) = \text{clip}\left(\left\lfloor \frac{\widetilde{k}_{t,g}^{\text{clip}}}{s_{t,g}^{(K)}} + z_{t,g}^{(K)} \right\rceil, 0, q_{\max}\right),
$$

and

$$
\begin{aligned}
\mathcal{Q}\left(\widetilde{k}_{t,g}^{\text{clip}}\right) &= \mathcal{Q}^-\left(\mathcal{Q}^+\left(\widetilde{k}_{t,g}^{\text{clip}}\right)\right) \\
&= s_{t,g}^{(K)}\left(\mathcal{Q}^+\left(\widetilde{k}_{t,g}^{\text{clip}}\right) - z_{t,g}^{(K)}\right).
\end{aligned}
$$

Thus, unlike a fixed Frobenius-norm scale, the key scale is computed dynamically from the clipped rotated key row, independently for each token and each group. In our default configuration, $G_K = 64$.

$$
(KH_{\text{Had}})_{t,\cdot} = \left[\begin{array}{rrrrrrrrrrrrrrrr}
2.20 & -0.20 & 3.44 & 2.25 & -1.33 & 2.33 & 3.83 & 2.71 & -2.34 & -5.96 & 4.20 & 5.20 & -3.46 & -0.77 & 4.42 & 4.48 \\
\mathbf{7.15} & 5.23 & -4.17 & -3.54 & 5.85 & 5.28 & -3.61 & -1.41 & 1.05 & 3.94 & -1.22 & -2.85 & 1.48 & 1.03 & 2.05 & -0.46 \\
1.02 & -1.19 & -2.61 & -2.01 & 0.93 & 3.00 & -1.01 & -3.19 & 3.09 & 2.20 & -5.85 & -5.58 & 4.90 & 3.06 & -4.39 & -3.55 \\
-3.07 & -3.65 & 0.48 & 4.83 & -3.59 & -2.92 & 3.38 & 5.09 & -2.81 & -0.89 & 1.22 & 0.46 & -3.49 & -2.21 & 2.88 & 0.84 \\
\hline
-1.53 & 1.03 & 0.94 & 0.19 & -2.30 & 0.20 & 5.16 & 1.89 & -1.37 & -3.52 & 4.74 & 4.87 & -3.51 & -4.25 & 4.14 & 5.80 \\
5.48 & 5.35 & -4.44 & -1.88 & 7.12 & 5.46 & -1.44 & -2.71 & 1.37 & 3.77 & 0.45 & -2.59 & 3.21 & 2.03 & -0.67 & 0.81 \\
-1.55 & 2.51 & -2.80 & -3.34 & 0.29 & 1.81 & -2.03 & -0.88 & 5.68 & 4.70 & \mathbf{-6.89} & -5.69 & 1.38 & 1.07 & -4.79 & -2.27 \\
-3.94 & -4.43 & 2.41 & 5.23 & -2.12 & -5.19 & 2.62 & 2.32 & -1.78 & -2.21 & 0.40 & 0.43 & -0.79 & -1.48 & 1.86 & 2.20
\end{array}\right]
$$

group 0: $\max - \min = \mathbf{13.12}$ | group 1: $\max - \min = \mathbf{14.01}$.

Rotation R_K	*what the quantizer sees on $\tilde{K} = KR_K$*		*importance*	*INT2 residual*
	$\max\lvert\tilde{K}\rvert$	per-group max–min ($G_K = 64$)	$\frac{\max_i (R_K^\top C_Q R_K)_{ii}}{\text{mean}_i(\cdot)}$	$\text{tr}(E_K)$
I (no rotation)	36.3	25.7	43.0	233
H_{Had} (pure Hadamard, QuaRot-style)	11.1	13.3	1.72	206
U_Q (eigenbasis alone)	29.4	22.4	46.9	**1354**
$U_Q\, H_{\text{Had}}$	7.46	13.4	**1.00**	208
$U_Q\, H_{\text{Had}}\, P_K$ (OSCAR)	7.46	**11.6**	**1.00**	**169**

Table 4: Layer 10 of Qwen3-4B-Thinking, kv-head 0, $d = 128$, INT2 per-group ($G_K = 64$, two groups per head), $T = 8000$ tokens. $C_Q = Q^\top Q$ averaged over GQA groups (production setting). $\max|\tilde{K}|$ is the largest entry across all $T \times d$ positions; *per-group max–min* is $\mathbb{E}_t\left[\max_{i\in g} \tilde{K}_{t,i} - \min_{i\in g} \tilde{K}_{t,i}\right]$ averaged over the two $G_K = 64$ groups. The importance column measures how peaky the rotated importance metric's diagonal is (1.00 = all channels equally important; Lemma 1 forces this when C_Q is pre-diagonalized by U_Q). $\text{tr}(E_K)$ is the unweighted INT2 residual. At $G_K = d = 128$ (one group per head) the permutation P_K would be a no-op and OSCAR collapses to $U_Q H_{\text{Had}}$; we deliberately use $G_K = 64$ here so the PBR step has a non-trivial effect. Bold entries are extreme values referenced in the text.

Values use the same affine asymmetric INT2 quantization rule. For a group g of a clipped rotated value row, write

$$\widetilde{v}_{t,g}^{\text{clip}} \in \mathbb{R}^{1\times G_V},$$

where G_V is the value quantization group size. The implementation computes

$$a_{t,g}^{(V)} = \min_c \widetilde{v}_{t,g,c}^{\text{clip}}, \qquad b_{t,g}^{(V)} = \max_c \widetilde{v}_{t,g,c}^{\text{clip}},$$

and

$$s_{t,g}^{(V)} = \frac{b_{t,g}^{(V)} - a_{t,g}^{(V)}}{q_{\max}}, \qquad z_{t,g}^{(V)} = -\frac{a_{t,g}^{(V)}}{s_{t,g}^{(V)}}.$$

The quantized and dequantized value group is

$$\mathcal{Q}^+\left(\widetilde{v}_{t,g}^{\text{clip}}\right) := \text{clip}\left(\left\lfloor \frac{\widetilde{v}_{t,g}^{\text{clip}}}{s_{t,g}^{(V)}} + z_{t,g}^{(V)} \right\rceil, 0, q_{\max}\right),$$

and

$$\mathcal{Q}\left(\widetilde{v}_{t,g}^{\text{clip}}\right) := s_{t,g}^{(V)} \left(\mathcal{Q}^+\left(\widetilde{v}_{t,g}^{\text{clip}}\right) - z_{t,g}^{(V)}\right).$$

Finally, after quantization–dequantization in the rotated coordinates, the approximations are mapped back to the original coordinate system:

$$\widehat{k}_t = \mathcal{Q}\left(\widetilde{k}_t^{\text{clip}}\right) R_K^\top, \qquad \widehat{v}_t = \mathcal{Q}\left(\widetilde{v}_t^{\text{clip}}\right) R_V^\top.$$

A.6 Proof of Theorem. 1

We prove the key-side statement first. The row-vector rotation convention gives the frozen-error surrogate

$$\widetilde{\mathcal{L}}_K(R_k) = \text{tr}\left(R_k^\top C_Q R_k E_K\right), \qquad R_k^\top R_k = I_d,$$

where

$$\begin{aligned} C_Q &= U_Q \Lambda_Q U_Q^\top, \\ \Lambda_Q &= \text{diag}(\lambda_1, \ldots, \lambda_d), \\ \lambda_1 &\geq \cdots \geq \lambda_d \geq 0, \end{aligned}$$

and

$$E_K = \text{diag}(\mu_1, \ldots, \mu_d), \qquad \mu_1 \leq \cdots \leq \mu_d.$$

Here E_K is treated as a fixed frozen residual covariance matrix.

Let

$$R_k = U_Q Z, \qquad Z^\top Z = I_d.$$

Substituting into the objective gives

$$\begin{aligned} \widetilde{\mathcal{L}}_K(R_k) &= \text{tr}\left(Z^\top U_Q^\top C_Q U_Q Z E_K\right) \\ &= \text{tr}\left(Z^\top \Lambda_Q Z E_K\right). \end{aligned}$$

Writing $Z = (z_{ij})$, we have

$$\left(Z^\top \Lambda_Q Z\right)_{ii} = \sum_{j=1}^{d} \lambda_j z_{ji}^2.$$

Therefore,

$$\widetilde{\mathcal{L}}_K(R_k) = \sum_{i=1}^{d}\sum_{j=1}^{d} \mu_i \lambda_j z_{ji}^2. \tag{5}$$

Since Z is orthogonal, the matrix $W = (w_{ji})$ with $w_{ji} = z_{ji}^2$ is doubly stochastic. Hence Eq. (5) is a linear function over the Birkhoff polytope, and its minimum is attained at a permutation matrix.

Thus,

$$\min_{R_k^\top R_k = I_d} \widetilde{\mathcal{L}}_K(R_k) = \min_{\Pi \in \mathcal{P}_d} \operatorname{tr}\Big(\Pi^\top \Lambda_Q \Pi E_K\Big),$$

where $\mathcal{P}_d$ denotes the set of $d \times d$ permutation matrices. By the rearrangement inequality, since

$$\lambda_1 \geq \cdots \geq \lambda_d \qquad \text{and} \qquad \mu_1 \leq \cdots \leq \mu_d,$$

the minimum is achieved by pairing the largest λ_j with the smallest μ_i. This is achieved by the identity permutation. Therefore, one minimizer is

$$Z = I_d,$$

and hence

$$R_k = U_Q.$$

The value-side proof is identical. The value surrogate is

$$\widetilde{\mathcal{L}}_V(R_v) = \operatorname{tr}\Big(R_v^\top C_S R_v E_V\Big), \qquad R_v^\top R_v = I_d,$$

where

$$\begin{aligned} C_S &= U_S \Lambda_S U_S^\top, \\ \Lambda_S &= \operatorname{diag}(\nu_1, \ldots, \nu_d), \\ \nu_1 &\geq \cdots \geq \nu_d \geq 0, \end{aligned}$$

and

$$E_V = \operatorname{diag}(\eta_1, \ldots, \eta_d), \qquad \eta_1 \leq \cdots \leq \eta_d.$$

Again, E_V is treated as a fixed frozen residual covariance matrix.

Let

$$R_v = U_S Z_V, \qquad Z_V^\top Z_V = I_d.$$

Then

$$\begin{aligned} \widetilde{\mathcal{L}}_V(R_v) &= \operatorname{tr}\Big(Z_V^\top U_S^\top C_S U_S Z_V E_V\Big) \\ &= \operatorname{tr}\Big(Z_V^\top \Lambda_S Z_V E_V\Big). \end{aligned}$$

Writing $Z_V = (\zeta_{ij})$, we obtain

$$\widetilde{\mathcal{L}}_V(R_v) = \sum_{i=1}^{d}\sum_{j=1}^{d} \eta_i \nu_j \zeta_{ji}^2.$$

As before, (ζ_{ji}^2) is doubly stochastic. The minimum is therefore attained at a permutation matrix. Since

$$\nu_1 \geq \cdots \geq \nu_d \qquad \text{and} \qquad \eta_1 \leq \cdots \leq \eta_d,$$

the rearrangement inequality shows that the identity permutation is optimal. Thus one minimizer is

$$Z_V = I_d,$$

and therefore

$$R_v = U_S.$$

This proves both statements.

A.7 Justification of Surrogate Objectives in Theorem. 1

To justify the surrogate objectives in Theorem 1, we start from the downstream attention errors induced by quantizing keys and values. For the key branch, the empirical logit distortion on the calibration set is

$$\ell_K(R_k) = \sum_{i=1}^{N}\sum_{j=1}^{i} \big[q_i (k_j - \hat{k}_j)^\top\big]^2,$$

where $\hat{k}_j = \mathcal{Q}(k_j R_k) R_k^\top$. Writing the rotated quantization residual as

$$e_j^{(K)}(R_k) := \mathcal{Q}(k_j R_k) - k_j R_k,$$

we have $\hat{k}_j - k_j = e_j^{(K)}(R_k) R_k^\top$, so that

$$\ell_K(R_k) = \sum_{j=1}^{N} e_j^{(K)}(R_k)\, R_k^\top C_j R_k\, e_j^{(K)}(R_k)^\top,$$

$$C_j := \sum_{i=1}^{j} q_i^\top q_i.$$

To obtain a tractable objective, we approximate the position-dependent matrices C_j by the global query target covariance

$$C_Q = \frac{1}{N}\sum_{i=1}^{N} q_i^\top q_i.$$

Defining the frozen residual covariance (where R_k is fixed)

$$E_K := \sum_{j=1}^{N} e_j^{(K)}(R_k)^\top e_j^{(K)}(R_k),$$

the key-side frozen-error objective becomes, up to the constant normalization factor $\frac{1}{N}$,

$$\tilde{\mathcal{L}}_K(R_k) = \mathrm{tr}(R_k^\top C_Q R_k E_K).$$

Under a similar approximation, the value-side downstream error is heuristically, imitating that of key targets, represented by the frozen-error surrogate

$$\tilde{\mathcal{L}}_V(R_v) = \mathrm{tr}\big(R_v^\top C_S R_v E_V\big).$$

Note that the only difference between key and value targets is whether it is purely weighted by target covariance ($Q^\top Q$ for keys) or it is target covariance aware ($V^\top S^\top SV$). Therefore, it is reasonable that we use the same surrogate.

B Algorithm Flow and Serving Procedure

Algorithm 1 gives the concrete OSCAR procedure used in our experiments and SGLang implementation.

C Related Work

KV-cache compression. The KV cache has become a major target for post-training compression because its memory footprint grows with both context length and batch size. One line of work reduces or reprioritizes cached content through eviction, budgeted retention, attention-sink preservation, or saliency-aware precision assignment, as in H2O, SnapKV, PyramidKV, StreamingLLM, and ZipCache (Zhang et al., 2023; Li et al., 2024; Cai et al., 2024; Xiao et al., 2024; He et al., 2024). Other systems compress the cache through low-rank structure or cross-layer sharing, including GEAR, PALU, xKV, and MatryoshkaKV (Kang et al., 2024; Chang et al., 2024, 2025; Lin et al., 2024a). Another line keeps all tokens but lowers the precision of the cached tensors, which is the setting most closely related to OSCAR. KIVI, KVQuant, and WKVQuant study low-bit KV storage and identify channel-wise outliers, especially in keys, as a central obstacle to aggressive quantization (Liu et al., 2024b; Hooper et al., 2024; Yue et al., 2024). These methods show that KV quantization can be accurate, but they also leave open how far precision can be reduced without changing the serving layout.

2-bit KV quantization. Pushing KV cache quantization to 2 bits usually requires additional structure beyond naive rounding. SKVQ preserves a recent sliding window in higher precision and combines channel reordering with clipped dynamic quantization (Duanmu et al., 2024). Kitty promotes sensitive key-cache channels to higher precision and combines this with a residual cache (Xia et al., 2025), while PM-KVQ uses progressive mixed precision and block-wise memory allocation for long-CoT models (Liu et al., 2025). Related mixed-precision analyses also argue that keys should generally receive more precision than values under a fixed memory budget (Hariri et al., 2025). RotateKV uses outlier-aware adaptive rotations for 2-bit KV quantization (Su et al., 2025). These methods improve accuracy, but their use of channel-wise precision, residual buffers, progressive bit allocation, or adaptive layouts can complicate integration with paged KV-cache serving and fused decode kernels. OSCAR instead targets the same INT2 regime with fixed token-wise transforms and a uniform paged layout, so the quantization path can be fused into production serving.

Rotation-based quantization. Orthogonal transforms are an attractive way to make low-bit quantization more accurate without changing tensor shapes. QuaRot, HALO, and HOT show that rotations can reduce outlier concentration, redistribute activation energy, and improve low-bit rounding behavior (Ashkboos et al., 2024, 2025; Kim et al., 2025b; Egiazarian et al., 2026; Jia et al., 2026). In the KV-cache setting, TurboQuant develops a data-oblivious online vector-quantization method based on random rotations, per-coordinate Lloyd–Max quantization, and QJL-style residual coding (Zandieh et al., 2025). KVLinC combines Hadamard rotation with lightweight linear correction to reduce attention errors under aggressive KV quantization (Saxena and Roy, 2025). OSCAR follows the same broad rotation principle, but asks which rotation target is appropriate for INT2 KV cache quantization.

While (Liu et al., 2024a) effectively mitigates outliers at 4-bit using learned rotations, it collapses at extreme INT2 levels because its rotations fail to align with the actual covariance structures consumed by downstream attention. Furthermore, it focuses primarily on theoretical algorithmic gains, lacking the custom kernels and system-level integration necessary for real-world deployment in modern serving frameworks.

Covariance-aware calibration. A related family uses calibration data or covariance information to choose quantization or compression directions. For weight quantization, GPTQ and QuIP use Hessian information to guide low-bit rounding (Frantar et al., 2023; Chee et al., 2023); for low-rank compression, Drone, ASVD, SVD-LLM, SVD-LLMv2, and CorDA use calibration activations or data-aware statistics to guide factorization (Chen et al., 2021b; Yuan et al., 2023; Wang et al., 2024, 2025a; Yang et al., 2024). On the KV-cache side, HaPPI, CARE, RecalKV, and CommonKV apply covariance- or Hessian-aware factorization to keys and values for KV-side projection or compression (Kim et al., 2025a; Zhou et al., 2026; Yan et al., 2025; Wang et al., 2025b). Post-training quantization methods such as SmoothQuant, AWQ, GPTQ, and OmniQuant also use calibration data to estimate scales, equivalent transformations, clipping thresholds, or second-order surrogates before deployment (Xiao et al., 2023; Lin et al., 2024b; Frantar et al., 2023; Shao et al., 2023). OSCAR specializes this calibration paradigm to attention: it estimates targets covariance induced by downstream attention, rather than raw-cache statistics, and uses them to derive fixed key/value rotations and clip thresholds for serving-time INT2 KV quantization.

D Discussion

Limitations. Our theoretical analysis explains why the proposed covariance-target rotations are optimal under a frozen-error surrogate with explicit assumptions (Appendix A), but it is not yet a full proof of optimality for the complete autoregressive decoding process. Tightening this gap—for example, by proving when the calibration estimator is unbiased or optimal for the true attention objective—is an important direction for future work. We also focus on INT2 KV-cache quantization and do not explore other numerical formats or mixed-format designs, such as NVFP4, different weight precisions, or weight/KV precision combinations, which may create different accuracy–systems trade-offs. Similarly, OSCAR uses a fixed bit-width for quantized history tokens and does not study per-token bit allocation or thought-adaptive token precision, which recent work such as ThinKV suggests may be useful for long reasoning traces (Ramachandran et al., 2025). On the systems side, our implementation targets SGLang-style paged serving on H100 GPUs; additional tuning is needed to understand the best kernel design on other hardware such as B200. Finally, OSCAR is orthogonal to channel-wise or mixed-precision methods: applying attention-aware/Hadamard rotations inside channel-wise schemes may further improve accuracy, but combining these ideas while preserving an efficient serving layout remains open.

Broader impact. OSCAR reduces KV-cache memory and serving cost, which can make long-context LLM inference more accessible and energy-efficient. At the same time, cheaper long-context inference may also lower the cost of deploying powerful generative models, including applications with misuse risks; OSCAR does not introduce new model capabilities, but should be deployed with the same safety and usage controls as the underlying LLMs.

E Additional Experimental Results

E.1 Additional Motivation Figure

The additional motivation figures are shown in Fig 7 and Fig 8.

E.2 Long-Context Robustness and Ablation Studies

E.2.1 Long-Context Robustness

Long-Context Robustness. Table 5 reports RULER-NIAH accuracy from 4k to 128k tokens on the same serving-compatible baselines used in the main accuracy comparison. The expected pattern is simple: short contexts should remain close to BF16, while longer contexts expose accumulated attention-logit error. OSCAR should degrade more slowly than rotation-

only INT2 baselines because its rotation target is chosen from the covariance seen by attention, not from raw-cache reconstruction, and because long contexts amplify accumulated KV quantization error. Figure 9 provides a direct KL-based check before the retrieval results. Across both Qwen3-4B-Thinking-2507 and Qwen3-8B, OSCAR is more stable as sequence length increases: its attention distribution stays closer to FP16, while naive INT2 and Hadamard-only rotation drift more quickly. On Qwen3-4B-Thinking-2507, QuaRot-INT2 is already near zero at short contexts, whereas OSCAR stays close to BF16 through 16k and retains non-trivial retrieval at 64k and 128k. On Qwen3-8B, QuaRot-INT2 remains usable only at 4k–8k and collapses after 16k, while OSCAR keeps substantially higher retrieval accuracy throughout the sweep. The GLM-4.7-FP8 preliminary run shows a complementary large-model case: all three methods remain strong on this retrieval-only task, and OSCAR matches the BF16 curve up to 128k. Together, the RULER results and KL curves indicate that attention-aware covariance calibration mainly helps when long histories make small KV errors accumulate over many steps.

E.2.2 Ablation Studies

Rotation Analysis: Decomposing R and Comparing Rotation Targets. OSCAR's composed rotation is $R = U \cdot H_{\text{Had}} \cdot P_{\text{br}}$, the product of an attention-aware eigenbasis U, a Hadamard transform H_{Had}, and a bit-reversal permutation P_{br}. Table 6 removes each factor of R in turn (*top block*) and, on the same footing, changes the PCA target used to compute U (*bottom block*): OSCAR's attention-aware targets are replaced by raw-cache reconstruction targets ($K^\top K / V^\top V$), fixed Hadamard rotations, or no learned rotation.

Two observations emerge. First, both the attention-aware eigenbasis U and the Hadamard component H_{Had} contribute substantially; the bit-reversal permutation P_{br} does not change accuracy in floating-point math but improves quantization geometry by interleaving large and small eigenvalues so that per-group quantization sees a more uniform range. Second, none of the alternative rotation targets (random Hadamard, raw $K^\top K / V^\top V$ reconstruction targets, random orthogonal) matches the attention-aware $C_Q / C_{SQ} / C_S$ targets at the same INT2 budget; the score-weighted variant covariance C_{SQ} gives a further improvement over C_Q alone. This isolates the central claim of the paper: *which* covariance matrix one diagonalizes matters more than *whether* one diagonalizes at all.

Sink and Recent Window Sizes. We sweep $(S, R) \in \{(0,0), (32,128), (64,256), (128,512), (256,1024)\}$ on Qwen3-4B-Thinking-2507 and report accuracy with the additional BF16 KV memory that each protection window introduces (Table 7). A clear knee emerges at (S, R)=$(64, 256)$: smaller windows leave noticeable accuracy on the table, while larger windows provide negligible additional accuracy at the cost of substantially more BF16 KV memory. $(64, 256)$ is a sweet spot and thus used for the main results.

Clip Threshold. The clip thresholds are estimated per layer from rotated calibration activations rather than tuned on downstream tasks. Table 8 sweeps global clip ratios on Qwen3-4B-Thinking and shows that the calibration-derived point $(c_K, c_V) = (0.96, 0.92)$ is close to the best grid point.

Calibration Data Regime. We also test whether OSCAR is sensitive to calibration volume or domain. Table 9 shows that the default 8k GPQA-Diamond calibration is close to the best setting, supporting a small one-time calibration pass.

E.3 Full Table For Main Accuracy Run

We add more results from MiniMax-M2.7 toreval (Phase 9) here.

Table 11 reports the five independent runs for the Qwen3-4B-Thinking main OSCAR result. Table 12–14 report the five independent runs used to summarize Qwen3-8B OSCAR accuracy under three INT2 grouping configurations. Table 15–17 report the Qwen3-32B OSCAR runs. Table 18–20 report the GLM-4.7-FP8 detailed runs for Naive INT2, QuaRot/Hadamard rotation, and OSCAR.

E.4 Ablation Detail Runs

Table 21 reports the individual runs behind the rotation-decomposition study in Table 6.

Table 5: Long-context retrieval accuracy on **RULER-NIAH** (Hsieh et al., 2024) across context lengths from 4k to 128k tokens, on the Group A serving-compatible baselines. Each number is averaged over 3 random seeds, except GLM-4.7-FP8 preliminary results (one run, as GLM-4.7 is very stable).

Model	Method	4k	8k	16k	32k	64k	128k
Qwen3-4B (-Thinking-2507)	BF16	100.0±0.0	100.0±0.0	99.7±2.0	99.3±1.0	85.3±5.0	81.0±1.4
	QuaRot-INT2 (Ashkboos et al., 2024)	0.0±0.0	0.8±0.2	0.0±0.0	0.0±0.0	15.6±3.2	0.0±0.0
	OSCAR	99.7±0.1	100.0±0.0	97.8±0.2	87.6±0.1	61.9±1.3	39.5±1.0
Qwen3-8B	BF16	100.0±0.0	99.6±0.0	98.9±0.1	97.3±0.0	79.2±0.1	78.2±1.0
	QuaRot-INT2 (Ashkboos et al., 2024)	74.0±1.0	80.6±0.3	19.0±2.0	9.8±1.0	0.0±0.0	0.0±0.0
	OSCAR	99.5±0.2	97.8±0.3	93.9±0.7	86.3±0.6	61.9±1.8	45.0±0.0
GLM-4.7-FP8†	BF16	100.0	100.0	100.0	100.0	98.8	97.2
	QuaRot-INT2 (Ashkboos et al., 2024)	100.0	99.7	100.0	100.0	98.8	96.3
	OSCAR	100.0	100.0	100.0	100.0	98.8	97.2

Table 6: Decomposition of the composed OSCAR rotation $R = U \cdot H_{\text{Had}} \cdot P_{\text{br}}$ (*top block*) and comparison against alternative rotation-target choices used by prior work (*bottom block*), all at the same INT2 bit budget with sink/recent protection on Qwen3-8B. Entries are mean ± std over 3 runs.

Configuration	GPQA	HumanE	LCB v6	AIME25	MATH500	Mean
FP16 reference	57.07±0.94	84.71±0.99	48.07±2.20	66.00±2.79	88.28±0.46	68.83
Decomposition of OSCAR's composed rotation						
Full OSCAR: $U \cdot H_{\text{Had}} \cdot P_{\text{br}}$	55.89±1.27	87.97±0.19	46.00±1.88	67.78±1.92	92.38±0.20	70.01
OSCAR with $K^\top K/V^\top V$ PCA target	38.72±1.27	35.53±0.67	3.70±0.68	13.33±3.33	64.33±0.53	31.12
w/o P_{br}	52.86±1.62	86.71±0.84	45.22±0.34	63.33±8.82	91.85±1.45	68.00
w/o H_{Had} (only $U \cdot P_{\text{br}}$)	51.35±1.91	83.78±0.37	19.49±2.05	21.11±1.92	82.97±0.80	51.74
w/o U (QuaRot + Pbr: $H_{\text{Had}} \cdot P_{\text{br}}$)	40.74±3.44	38.54±1.18	4.48±0.68	15.56±1.92	64.80±1.29	32.82
no rotation (only clip + sink + recent)	18.52±3.29	0.00	0.39±0.68	2.22±1.92	0.00	4.23

Table 22 reports the individual runs behind the sink/recent-window study in Table 7.

Table 23 reports the per-task results behind the clip-threshold grid in Table 8.

Table 24 reports the individual runs behind the calibration-data regime study in Table 9.

E.5 Kernel Profiling Details

Table 25 reports the per-decode kernel profiling used to isolate OSCAR runtime overhead.

E.6 Throughput Scaling Curves

Figure 10 provides the detailed decode-length and batch-size throughput curves used in Section 5.3.

Table 7: Sink and recent window sizes on Qwen3-4B-Thinking-2507 (INT2 with OSCAR rotation and calibration-derived clip). (S, R) denote the number of BF16 sink tokens and recent-window tokens, respectively. "Extra BF16 KV" reports the protected BF16-token fraction, $(S + R)/128$K.

(S, R)	**GPQA**	**HumanE**	**LCB v6**	**AIME25**	**MATH500**	**Mean**	**Extra BF16 KV**
$(0, 0)$	0.00	0.00	0.00	0.00	0.00	0.00	0 %
$(32, 128)$	57.58±2.20	91.91±0.19	40.74±2.36	55.56±5.09	92.65±0.95	67.69	0.12 %
$(\mathbf{64, 256})$	**64.95±1.16**	**92.24±1.02**	**45.38±1.97**	**64.00±3.65**	**92.75±0.39**	**71.86**	**0.24 %**
$(128, 512)$	65.32±0.77	93.17±0.32	45.22±0.33	67.78±1.92	93.32±0.12	72.96	0.49 %
$(256, 1024)$	64.98±1.27	93.13±0.81	46.20±2.03	67.78±1.92	93.32±0.12	73.08	0.98 %

Table 8: Validation of the calibration-derived clip thresholds (mean accuracy on Qwen3-4B-Thinking-2507). Rows: c_K; columns: c_V. The cell highlighted in bold is the operating point produced by the calibration procedure described in Section 5.1; it is within 0.15 points of the best downstream-task grid point, confirming that OSCAR's clip thresholds do not need to be hand-tuned.

$c_K \setminus c_V$	0.88	0.92	0.96	0.98	1.00
0.88	67.20	68.27	68.10	67.56	68.25
0.92	69.15	70.00	70.46	70.74	70.33
0.96	68.77	**70.59**	70.70	68.17	69.14
0.98	65.22	65.76	64.12	64.79	67.31
1.00	60.35	59.19	58.15	58.63	56.02

Table 9: Calibration-data regime on Qwen3-4B-Thinking-2507 (INT2 with OSCAR rotation, calibration-derived clip, sink/recent protection). We sweep both the calibration *volume* (number of tokens) and the calibration *domain*. The default setting (8k GPQA-Diamond tokens) is highlighted.

Calibration domain	**Tokens**	**GPQA**	**HumanE**	**LCB v6**	**AIME25**	**MATH500**	**Mean**
MMLU prompts	2k	60.61±2.20	92.52±0.14	39.57±2.36	55.56±5.09	93.12±0.46	68.28
	8k	60.44±2.04	92.07±0.76	40.74±0.89	60.00±5.77	92.52±0.58	69.15
	16k	64.14±0.51	92.93±0.53	43.86±2.55	61.11±10.72	92.38±0.53	70.88
	32k	61.28±1.54	92.20±0.32	46.00±2.36	57.78±5.09	92.25±0.31	69.90
WikiText	8k	61.11±0.51	92.72±0.63	42.11±2.11	58.89±3.85	92.65±0.50	69.50
GPQA-Diamond	**8k (default)**	**62.96±1.27**	**92.56±0.00**	**44.64±1.47**	**62.22±1.92**	**92.65±0.31**	**71.01**

Table 10: MiniMax-M2.7 multi-benchmark results from tore-eval (Phase 9). INT2 uses LM_RATIO = 1.16. The two GPQA-Diamond rows report the simple_evals 32K run and an independent tore-eval cross-check, respectively. Δ is the INT2–BF16 difference in percentage points.

Benchmark	**BF16**	**INT2**	**Δ**
GPQA-Diamond (simple_evals, 32K)	0.7730	0.8131	+4.0 pp
GPQA-Diamond (tore-eval cross-check)	0.7828	0.7929	+1.0 pp
HumanEval	0.8817	0.8854	+0.4 pp
LiveCodeBench v6 (release_v6)	0.5614	0.6082	+4.7 pp
AIME 2025	0.7667	0.7667	0.0 pp
MATH500	0.9379	0.9279	-1.0 pp

Table 11: Full Qwen3-4B-Thinking results for the main OSCAR configuration (INT2 group size=128, sink=64, recent=256, $c_K = 0.96$, $c_V = 0.92$).

Task	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Mean**	**Std**
GPQA	66.16	63.64	64.14	64.65	66.16	64.95	1.16
HumanEval	93.17	91.59	92.44	90.85	93.17	92.24	1.02
LiveCodeBench v6	45.03	43.27	45.61	48.54	44.44	45.38	1.97
AIME25	63.33	63.33	63.33	60.00	70.00	64.00	3.65
MATH500	92.59	92.79	92.99	93.19	92.18	92.75	0.39

Algorithm 1: OSCAR calibration, prefill, and decode

Hyperparameters: sink length S_0, recent window W, block size G=128, bit-width b=2
State: BF16 sink/recent cache, rotated BF16 staging cache, packed INT2 history cache
procedure Calibrate($\mathcal{D}$)
for each layer and KV head **do**
  $Q, K, V \leftarrow \text{ForwardDump}(\mathcal{D})$
  $S \leftarrow \text{softmax}_{\text{row}}(QK^\top/\sqrt{d} + M)$
  $C_K \leftarrow \frac{1}{N}\sum_n q_n^\top q_n$
  $C_S \leftarrow \frac{1}{N}V^\top S^\top SV$
  $U_Q \leftarrow \text{EigVec}(C_K), \quad U_S \leftarrow \text{EigVec}(C_S)$
  $R_K \leftarrow U_Q H_{\text{Had}} P_{\text{br}}, \quad R_V \leftarrow U_S H_{\text{Had}} P_{\text{br}}$
  $c_K, C_S \leftarrow \text{CalibrateClip}(KR_K, VR_V)$
end for
return $\{R_K, R_V, c_K, C_S\}$
end procedure
procedure Prefill(X)
$Q, K, V \leftarrow \text{Forward}(X)$
$\widetilde{K} \leftarrow KR_K, \quad \widetilde{V} \leftarrow VR_V$ {rotate before prefill write}
$\text{Cache}_{\text{sink}} \leftarrow (K, V)_{[:S_0]}$
$\text{Cache}_{\text{recent}} \leftarrow (K, V)_{[-W:]}$
for $i = S_0$ **to** $|X| - W - 1$ **do**
  QuantizeAndWrite($\widetilde{K}_i, \widetilde{V}_i, i$)
end for
end procedure
procedure DecodeStep(x_t)
$q_t, k_t, v_t \leftarrow \text{Forward}(x_t)$
$\widetilde{k}_t \leftarrow k_t R_K, \quad \widetilde{v}_t \leftarrow v_t R_V$
Append (k_t, v_t) to $\text{Cache}_{\text{recent}}$; append $(\widetilde{k}_t, \widetilde{v}_t)$ to rotated staging
if $|\text{Cache}_{\text{recent}}| > W$ **then**
  $(\widetilde{k}_j, \widetilde{v}_j) \leftarrow \text{PopOldestRotatedStaging}()$
  QuantizeAndWrite($\widetilde{k}_j, \widetilde{v}_j, j$)
end if
$\widehat{K}_{\text{hist}}, \widehat{V}_{\text{hist}} \leftarrow \text{DequantHistory}(\text{Cache}_{\text{INT2}})$
$\widehat{K}_{\text{hist}} \leftarrow \widehat{K}_{\text{hist}} R_K^\top, \quad \widehat{V}_{\text{hist}} \leftarrow \widehat{V}_{\text{hist}} R_V^\top$
$K_{\text{all}}, V_{\text{all}} \leftarrow \text{Concat}(\text{Cache}_{\text{sink}}, \widehat{K}_{\text{hist}}, \text{Cache}_{\text{recent}})$
$o_t \leftarrow \text{softmax}(q_t K_{\text{all}}^\top/\sqrt{d} + M)V_{\text{all}}$
return o_t
end procedure
function QuantizeAndWrite($\widetilde{k}, \widetilde{v}, i$)
$\bar{k} \leftarrow \text{clip}(\widetilde{k}, c_K), \quad \bar{v} \leftarrow \text{clip}(\widetilde{v}, C_S)$
$K_i^+ \leftarrow \text{AffineINT2}(\bar{k}, G), \quad V_i^+ \leftarrow \text{AffineINT2}(\bar{v}, G)$
$\text{Cache}_{\text{INT2}}[i] \leftarrow (K_i^+, V_i^+, \text{scale}, \text{zero})$
end function

Table 12: Full Qwen3-8B results for INT2 group=128.

Task	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Mean**	**Std**
GPQA	53.03	54.04	54.55	57.58	55.05	54.85	1.70
HumanEval	86.59	88.54	88.90	87.56	87.68	87.85	0.91
LiveCodeBench v6	45.03	45.03	42.11	47.95	46.20	45.26	2.13
AIME25	66.67	63.33	56.67	63.33	63.33	62.67	3.65
MATH500	92.59	91.38	92.59	92.18	91.98	92.14	0.50

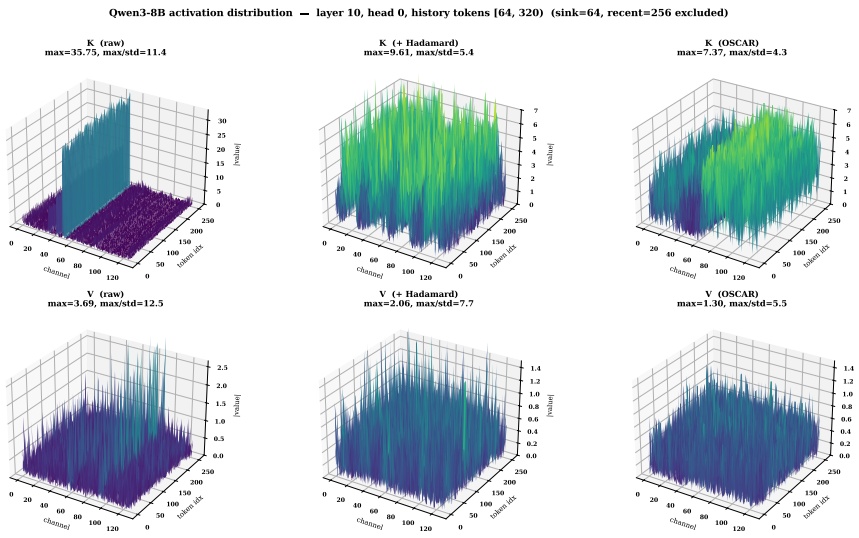


Figure 7: **Attention-aware rotations make history-token activations easier to quantize.** Hadamard mixing flattens raw activation peaks by spreading energy across channels. OSCAR goes further: the target covariance separates directions that matter more or less to attention, and the Hadamard transform then mixes each part into a more uniform range.

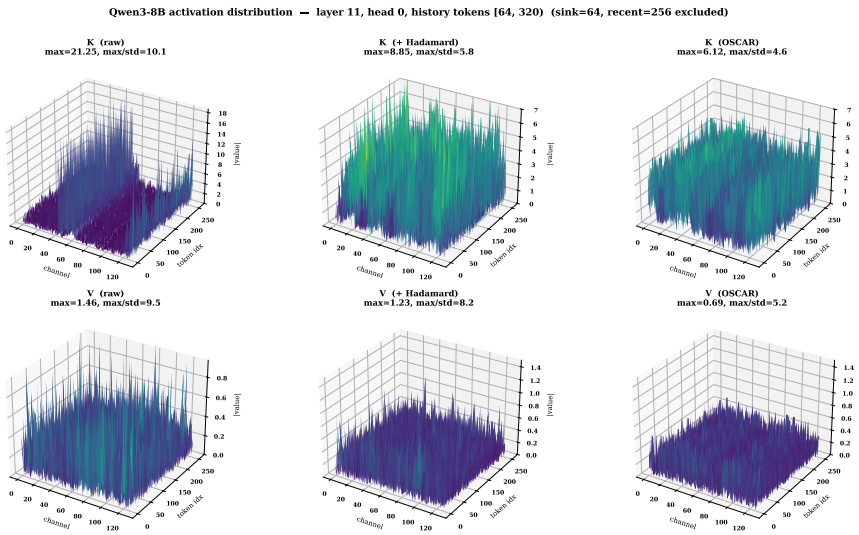


Figure 8: **Attention-aware rotations make history-token activations easier to quantize.** Hadamard mixing flattens raw activation peaks by spreading energy across channels. OSCAR goes further: the target covariance separates directions that matter more or less to attention, and the Hadamard transform then mixes each part into a more uniform range.

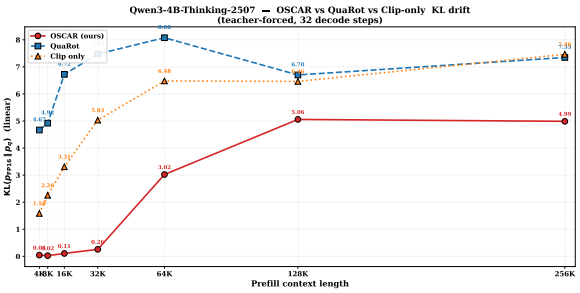


Figure 9a: Qwen3-4B-Thinking-2507

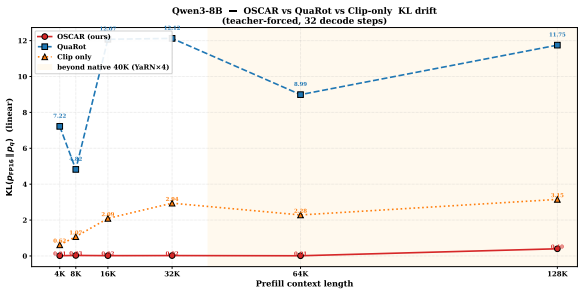


Figure 9b: Qwen3-8B

Figure 9: **OSCAR keeps attention distributions stable as context length grows.** KL divergence against the FP16 attention distribution across context lengths for naive INT2, Hadamard, and OSCAR.

Table 13: Full Qwen3-8B results for INT2 group=64.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	53.54	56.06	54.04	57.07	54.55	55.05	1.47
HumanEval	87.56	87.56	88.29	87.56	88.41	87.88	0.44
LiveCodeBench v6	47.37	44.44	49.12	45.03	45.61	46.32	1.91
AIME25	63.33	70.00	66.67	63.33	70.00	66.67	3.33
MATH500	92.38	92.79	91.58	91.18	93.19	92.22	0.83

Table 14: Full Qwen3-8B results for INT2 group=32.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	52.02	55.56	59.09	52.53	56.57	55.15	2.93
HumanEval	87.93	87.32	88.05	88.78	88.66	88.15	0.59
LiveCodeBench v6	45.61	47.37	43.86	43.86	45.61	45.26	1.47
AIME25	66.67	53.33	63.33	66.67	60.00	62.00	5.58
MATH500	91.78	92.38	92.59	92.18	92.38	92.26	0.30

Table 15: Full Qwen3-32B results for OSCAR INT2 group=128.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	59.60	59.09	57.58	60.10	–	59.09	1.09
HumanEval	90.37	89.76	89.27	89.76	89.27	89.68	0.45
LiveCodeBench v6	54.39	49.71	47.95	52.63	53.80	51.70	2.76
AIME25	63.33	66.67	76.67	76.67	70.00	70.67	5.96
MATH500	92.79	93.19	92.99	91.58	93.19	92.75	0.67

Table 16: Full Qwen3-32B results for OSCAR INT2 group=64.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	60.61	63.64	61.11	62.12	54.55	60.40	3.47
HumanEval	89.88	89.76	89.39	91.22	90.37	90.12	0.71
LiveCodeBench v6	53.80	52.63	53.80	54.39	53.22	53.57	0.67
AIME25	76.67	70.00	80.00	66.67	76.67	74.00	5.48
MATH500	93.19	92.18	93.19	92.38	92.79	92.75	0.46

Table 17: Full Qwen3-32B results for OSCAR INT2 group=32.

Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
GPQA	63.64	57.07	56.57	60.10	59.60	59.40	2.83
HumanEval	92.20	90.85	89.76	90.73	90.24	90.76	0.91
LiveCodeBench v6	58.48	54.39	54.39	50.29	55.56	54.62	2.94
AIME25	66.67	70.00	70.00	66.67	63.33	67.33	2.79
MATH500	93.39	92.99	92.79	93.59	93.19	93.19	0.32

Table 18: Full GLM-4.7-FP8 results for the Naive INT2 baseline.

Task	Run 1	Run 2	Run 3	Mean	Std
GPQA	58.08	51.01	54.55	54.55	3.54
HumanEval	85.61	88.15	85.73	86.50	1.43
LiveCodeBench v6	33.33	44.44	33.33	37.03	6.41
AIME25	40.00	36.67	36.67	37.78	1.92
MATH500	87.40	85.80	86.60	86.60	0.80

Table 19: Full GLM-4.7-FP8 results for the QuaRot/Hadamard rotation baseline.

Task	Run 1	Run 2	Run 3	Mean	Std
GPQA	70.20	63.64	70.20	68.01	3.79
HumanEval	91.71	89.76	88.15	89.87	1.78
LiveCodeBench v6	47.37	53.80	44.44	48.54	4.79
AIME25	80.00	80.00	76.67	78.89	1.92
MATH500	90.20	90.20	90.80	90.40	0.35

Table 20: Full GLM-4.7-FP8 results for the OSCAR rotation group size=128 configuration.

Task	Run 1	Run 2	Run 3	Mean	Std
GPQA	73.23	74.24	73.23	73.57	0.58
HumanEval	90.85	91.34	90.98	91.06	0.25
LiveCodeBench v6	53.80	52.05	52.05	52.63	1.01
AIME25	76.67	80.00	80.00	78.89	1.92
MATH500	93.79	94.99	95.19	94.66	0.76

Table 21: Full individual runs for the rotation-decomposition study on Qwen3-8B.

Configuration	Task	Run 1	Run 2	Run 3	Mean	Std
Full OSCAR	GPQA	56.06	54.55	57.07	55.89	1.27
	HumanEval	87.93	87.80	88.17	87.97	0.19
	LiveCodeBench v6	43.86	47.37	46.78	46.00	1.88
	AIME25	66.67	70.00	66.67	67.78	1.92
	MATH500	92.18	92.38	92.59	92.38	0.20
Tensor-recon. target	GPQA	37.37	39.90	38.89	38.72	1.27
	HumanEval	35.49	36.22	34.88	35.53	0.67
	LiveCodeBench v6	4.09	4.09	2.92	3.70	0.68
	AIME25	13.33	16.67	10.00	13.33	3.33
	MATH500	64.73	63.73	64.53	64.33	0.53
w/o P_{br}	GPQA	53.54	54.04	51.01	52.86	1.62
	HumanEval	87.20	85.73	87.20	86.71	0.84
	LiveCodeBench v6	45.03	45.03	45.61	45.22	0.34
	AIME25	73.33	60.00	56.67	63.33	8.82
	MATH500	92.79	90.18	92.59	91.85	1.45
w/o H_{Had}	GPQA	50.00	53.54	50.51	51.35	1.91
	HumanEval	84.15	83.41	83.78	83.78	0.37
	LiveCodeBench v6	19.30	17.54	21.64	19.49	2.05
	AIME25	23.33	20.00	20.00	21.11	1.92
	MATH500	83.77	82.97	82.16	82.97	0.80
w/o U	GPQA	41.92	43.43	36.87	40.74	3.44
	HumanEval	37.68	39.88	38.05	38.54	1.18
	LiveCodeBench v6	4.09	4.09	5.26	4.48	0.68
	AIME25	13.33	16.67	16.67	15.56	1.92
	MATH500	65.33	63.33	65.73	64.80	1.29
No rotation	GPQA	18.69	21.72	15.15	18.52	3.29
	HumanEval	0.00	0.00	0.00	0.00	0.00
	LiveCodeBench v6	0.00	1.17	0.00	0.39	0.68
	AIME25	3.33	0.00	3.33	2.22	1.92
	MATH500	0.00	0.00	0.00	0.00	0.00

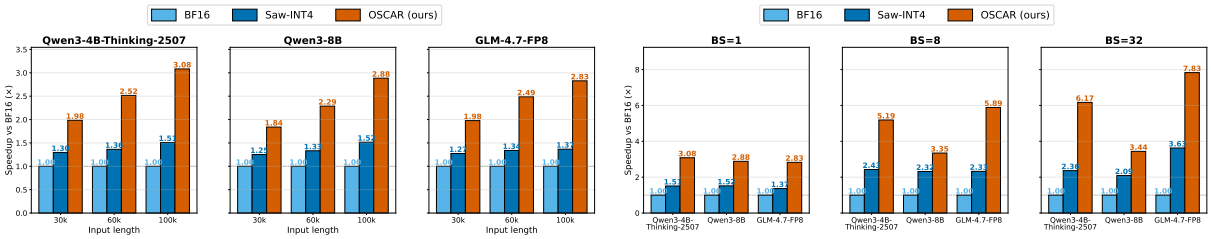


Figure 10: Left: Decode throughput speedup relative to BF16 at batch size 1 with a full prefix-cache hit with 1k our output sequence length. OSCAR achieves progressively larger speedups as the context increases. Right: Job-level throughput at 100k sequence, as a function of batch size for Qwen3-4B/8B and GLM-4.7-FP8. Each group of bars compares BF16, Saw-INT4, and ours at BS $\in \{1, 8, 32\}$.

Table 22: Full individual runs for the sink/recent-window study on Qwen3-4B-Thinking-2507.

(S, R)	Task	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Std
$(0, 0)$	GPQA	0.00	0.00	0.00	–	–	0.00	0.00
	HumanEval	0.00	0.00	0.00	–	–	0.00	0.00
	LiveCodeBench v6	0.00	0.00	0.00	–	–	0.00	0.00
	AIME25	0.00	0.00	0.00	–	–	0.00	0.00
	MATH500	0.00	0.00	0.00	–	–	0.00	0.00
$(32, 128)$	GPQA	60.10	56.57	56.06	–	–	57.58	2.20
	HumanEval	91.95	92.07	91.71	–	–	91.91	0.19
	LiveCodeBench v6	40.35	43.27	38.60	–	–	40.74	2.36
	AIME25	56.67	60.00	50.00	–	–	55.56	5.09
	MATH500	93.39	92.99	91.58	–	–	92.65	0.95
$(64, 256)$	GPQA	66.16	63.64	64.14	64.65	66.16	64.95	1.16
	HumanEval	93.17	91.59	92.44	90.85	93.17	92.24	1.02
	LiveCodeBench v6	45.03	43.27	45.61	48.54	44.44	45.38	1.97
	AIME25	63.33	63.33	63.33	60.00	70.00	64.00	3.65
	MATH500	92.59	92.79	92.99	93.19	92.18	92.75	0.39
$(128, 512)$	GPQA	65.15	66.16	64.65	–	–	65.32	0.77
	HumanEval	93.29	92.80	93.41	–	–	93.17	0.32
	LiveCodeBench v6	45.03	45.61	45.03	–	–	45.22	0.33
	AIME25	70.00	66.67	66.67	–	–	67.78	1.92
	MATH500	93.39	93.19	93.39	–	–	93.32	0.12
$(256, 1024)$	GPQA	66.16	65.15	63.64	–	–	64.98	1.27
	HumanEval	92.20	93.66	93.54	–	–	93.13	0.81
	LiveCodeBench v6	45.03	48.54	45.03	–	–	46.20	2.03
	AIME25	66.67	70.00	66.67	–	–	67.78	1.92
	MATH500	93.19	93.39	93.39	–	–	93.32	0.12

Table 23: Full per-task results for the clip-threshold sweep on Qwen3-4B-Thinking-2507. Each row is one (c_K, c_V) setting with one run per task.

c_K	c_V	GPQA	HumanE	LCB v6	AIME25	MATH500	Mean
0.88	0.88	59.09	91.83	39.18	53.33	92.59	67.20
0.88	0.92	62.12	91.22	39.77	56.67	91.58	68.27
0.88	0.96	57.07	91.10	42.69	56.67	92.99	68.10
0.88	0.98	62.12	91.95	37.43	53.33	92.99	67.56
0.88	1.00	57.58	91.59	46.78	53.33	91.98	68.25
0.92	0.88	61.62	93.17	41.52	56.67	92.79	69.15
0.92	0.92	62.12	91.22	43.86	60.00	92.79	70.00
0.92	0.96	64.14	92.32	46.20	56.67	92.99	70.46
0.92	0.98	65.15	93.90	48.54	53.33	92.79	70.74
0.92	1.00	64.65	93.90	46.78	53.33	92.99	70.33
0.96	0.88	61.62	92.80	39.77	56.67	92.99	68.77
0.96	**0.92**	**64.14**	**93.54**	**46.20**	**56.67**	**92.38**	**70.59**
0.96	0.96	63.13	91.59	43.27	63.33	92.18	70.70
0.96	0.98	62.12	92.68	40.35	53.33	92.38	68.17
0.96	1.00	62.12	91.59	42.11	56.67	93.19	69.14
0.98	0.88	55.56	91.71	37.43	50.00	91.38	65.22
0.98	0.92	55.05	91.59	36.84	53.33	91.98	65.76
0.98	0.96	48.48	90.37	36.84	53.33	91.58	64.12
0.98	0.98	55.56	91.71	37.43	46.67	92.59	64.79
0.98	1.00	56.57	91.34	39.77	56.67	92.18	67.31
1.00	0.88	48.48	89.88	30.99	40.00	92.38	60.35
1.00	0.92	44.95	90.98	32.16	36.67	91.18	59.19
1.00	0.96	45.45	90.37	30.41	33.33	91.18	58.15
1.00	0.98	43.43	90.73	32.16	36.67	90.18	58.63
1.00	1.00	47.47	90.73	30.41	23.33	88.18	56.02

Table 24: Full individual runs for the calibration-data regime study on Qwen3-4B-Thinking-2507.

Calibration setting	Task	Run 1	Run 2	Run 3	Mean	Std
MMLU 2k	GPQA	61.62	58.08	62.12	60.61	2.20
	HumanEval	92.68	92.44	92.44	92.52	0.14
	LiveCodeBench v6	37.43	39.18	42.11	39.57	2.36
	AIME25	50.00	56.67	60.00	55.56	5.09
	MATH500	93.39	93.39	92.59	93.12	0.46
MMLU 8k	GPQA	62.63	58.59	60.10	60.44	2.04
	HumanEval	91.22	92.68	92.32	92.07	0.76
	LiveCodeBench v6	40.94	41.52	39.77	40.74	0.89
	AIME25	66.67	56.67	56.67	60.00	5.77
	MATH500	92.18	93.19	92.18	92.52	0.58
MMLU 16k	GPQA	63.64	64.65	64.14	64.14	0.51
	HumanEval	93.29	92.32	93.17	92.93	0.53
	LiveCodeBench v6	46.78	42.69	42.11	43.86	2.55
	AIME25	73.33	53.33	56.67	61.11	10.72
	MATH500	92.18	92.99	91.98	92.38	0.53
MMLU 32k	GPQA	62.63	61.62	59.60	61.28	1.54
	HumanEval	92.07	92.56	91.95	92.20	0.32
	LiveCodeBench v6	48.54	45.61	43.86	46.00	2.36
	AIME25	53.33	63.33	56.67	57.78	5.09
	MATH500	92.59	91.98	92.18	92.25	0.31
WikiText 8k	GPQA	61.11	60.61	61.62	61.11	0.51
	HumanEval	93.41	92.20	92.56	92.72	0.63
	LiveCodeBench v6	39.77	43.86	42.69	42.11	2.11
	AIME25	56.67	56.67	63.33	58.89	3.85
	MATH500	93.19	92.59	92.18	92.65	0.50
GPQA-Diamond 8k	GPQA	63.13	64.14	61.62	62.96	1.27
	HumanEval	92.56	92.56	92.56	92.56	0.00
	LiveCodeBench v6	46.20	44.44	43.27	44.64	1.47
	AIME25	63.33	63.33	60.00	62.22	1.92
	MATH500	92.38	92.59	92.99	92.65	0.31

Table 25: Per-decode kernel profiling on Qwen3-8B, 1×H100, single decoding step latency in milliseconds.

B	BF16					OSCAR				
	GEMM	**Attn**	**Quant**	**Other**	**Total**	**GEMM**	**Attn**	**Quant**	**Other**	**Total**
1	5.6 (56.0%)	3.8 (37.9%)	–	0.6 (6.1%)	9.9	5.6 (69.9%)	1.3 (16.8%)	0.5 (5.9%)	0.6 (7.5%)	8.0
8	5.6 (53.0%)	4.4 (41.0%)	–	0.6 (6.0%)	10.7	5.7 (57.4%)	3.0 (30.1%)	0.5 (4.9%)	0.8 (7.7%)	9.9
16	5.7 (37.5%)	8.9 (58.1%)	–	0.7 (4.4%)	15.2	5.7 (49.7%)	4.6 (39.4%)	0.5 (4.2%)	0.8 (6.7%)	11.6
32	5.7 (24.3%)	17.0 (72.7%)	–	0.7 (3.0%)	23.4	5.7 (37.0%)	8.5 (54.6%)	0.5 (3.2%)	0.8 (5.2%)	15.5
64	capacity limited					5.8 (24.9%)	16.2 (69.1%)	0.5 (2.3%)	0.9 (3.7%)	23.4
128	capacity limited					5.9 (14.9%)	31.9 (81.1%)	0.6 (1.4%)	1.0 (2.6%)	39.3