DUAL-QK: SHARP QUERIES AND FLAT KEYS FOR PRUNABLE 2-BIT KV CACHES

Anonymous authors
Paper under double-blind review

ABSTRACT

Long inputs and extended generation increase the storage and access costs of the key–value (KV) cache. Low-bit quantization reduces storage and memory traffic, while query-channel pruning can further reduce key-cache reads. Rotation-based quantization redistributes the energy of key outliers across channels. To maintain computational invariance, the same orthogonal transform must be applied to queries, preserving query–key dot products. However, this rotation can disperse query energy, weakening the separation between a few large components to retain and many small ones to prune. We introduce Dual-QK, which uses paired non-orthogonal query and key transforms to address this conflict. Using calibrated query and key statistics, Dual-QK combines partial key whitening with a query-aligned basis to balance key scales for INT2 quantization and concentrate query energy for dynamic channel pruning. Channel-0 protection and bucket-relative RoPE support low-bit accuracy over long contexts. Experiments on four models across five generative benchmarks and long-context retrieval tasks show improved accuracy over OSCAR on most tasks at 40% query-channel sparsity. At a 128K context, Dual-QK provides $6.8\times$ KV-cache compression and an estimated $8.3\times$ reduction in KV read volume relative to unpruned BF16. Under the evaluated configurations, our SGLang implementation achieves up to $3.75\times$ the decoding throughput of unpruned BF16.

1 INTRODUCTION

Large language models are expanding from understanding long documents and code repositories to multistep reasoning and tool use. Long inputs require substantial context before generation begins, while reasoning traces and accumulated tool outputs extend the history during generation. Autoregressive decoding stores this history in the key–value (KV) cache, and dense attention revisits accumulated keys and values for every token. As the context grows, storing and repeatedly accessing this cache places increasing demands on memory capacity and bandwidth. Efficient long-context inference therefore requires reducing both cache storage and access costs during decoding.

KV quantization reduces both storage and memory traffic by lowering the number of bits per element (Liu et al., 2024; Hooper et al., 2024; Zhou et al., 2026). Dynamic sparse attention can further reduce reads by selecting tokens or channels for each query while retaining the cached history (Tang et al., 2024; Ribar et al., 2024). Each query–key (QK) dot product is a sum of channel contributions, so omitting channels with small contributions reduces the key components needed to compute attention scores. Combining query-channel sparsity with low-precision keys allows fewer channels to be read at fewer bits, preserving the storage benefits of quantization while further reducing key accesses. Realizing both benefits requires a representation that supports accurate key quantization and effective channel selection.

However, key quantization and query-channel pruning favor different channel distributions, as illustrated in Figure 1 (a) and (b). For key quantization, outliers widen the shared quantization range and reduce the resolution available to other values, favoring a flatter distribution. Query-channel pruning instead benefits from a sharp distribution that separates a few important channels from many small ones. Rotation-based quantization redistributes the energy of key outliers across channels (Ashkboos et al., 2024; Liu et al., 2025; Zhou et al., 2026). To maintain computational invariance, the same orthogonal transform is applied to queries, preserving query–key dot products

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

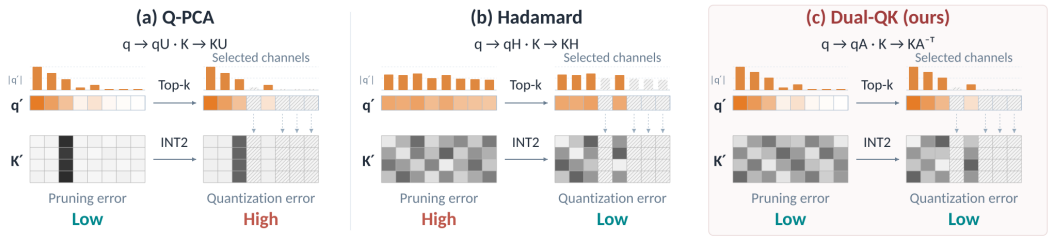

Figure 1: Conceptual comparison at the same bit width and retained-channel budget. (a) Q-PCA concentrates query energy, whereas (b) Hadamard rotation mitigates key outliers. (c) Dual-QK targets both properties using reciprocal Q/K transforms. Hatched channels are omitted from the dot product.

before quantization. However, this shared transform can also disperse query energy, weakening the concentration exploited by pruning. Conversely, query principal component analysis (Q-PCA) uses the eigenvectors of query second moments as a shared rotation basis for queries and keys. This concentrates query energy in a few channels but does not explicitly control key outliers that affect quantization error. These contrasting requirements motivate a joint design of query and key representations.

We propose **Dual-QK**, which uses paired non-orthogonal query and key transforms to address this conflict. Pairing an invertible query transform with its inverse transpose on keys preserves query–key dot products before quantization and pruning. This extends the shared orthogonal rotations used in prior work, allowing the query and key distributions to be shaped differently. Dual-QK uses this freedom to balance key scales while concentrating query energy in a few channels. These transforms support 2-bit key quantization and dynamic channel selection based on current query magnitudes.

Specifically, Dual-QK combines partial key whitening with a query-aligned basis derived from calibrated query and key statistics to balance key scales and concentrate query energy. Large query magnitudes make attention scores sensitive to quantization error in the leading key channel (channel-0). We therefore retain this channel in BF16 and exclude it from range estimation to narrow the quantization range of the remaining INT2 channels. Bucket-relative RoPE represents each stored key relative to its bucket anchor, keeping stored-key positional offsets within the calibration window. The corresponding query adjustment preserves attention scores before quantization and pruning.

Our contributions are as follows:

- We analyze the conflicting distributional requirements of key quantization and query-channel pruning and introduce Dual-QK, a paired transformation that constructs flat keys and sharp queries while preserving attention scores before compression.
- We exploit these distributions to combine dynamic query-channel selection with INT2 key quantization, using leading-channel protection and bucket-relative RoPE to maintain accuracy across context positions.
- We evaluate Dual-QK on four models for generative accuracy, long-context retrieval, and decoding throughput. At 40% query-channel sparsity, Dual-QK improves accuracy over OSCAR on most tasks. At a 128K context, it provides $6.8\times$ KV-cache compression and an estimated $8.3\times$ reduction in KV read volume relative to unpruned BF16.

2 Related Work

2.1 KV Cache Quantization

KV cache quantization reduces storage and memory traffic by storing keys and values at lower precision. Quantization accuracy depends on how values share a quantization range and how outliers are handled. Mixed precision, fine-grained grouping, and channel rescaling address these issues in LLM quantization (Dettmers et al., 2022; Zhao et al., 2024; Lin et al., 2025b). For KV caches, KIVI (Liu et al., 2024) quantizes keys per channel and values per token. KVQuant (Hooper et al., 2024) combines per-channel key quantization before rotary position embedding (RoPE) with non-

uniform quantization and separate outlier storage. Other approaches improve low-bit KV compression through channel reordering and group-wise clipping (Duanmu et al., 2024), or estimate query–key inner products using sign-quantized random projections (Zandieh et al., 2025).

Rotation-based methods have achieved state-of-the-art results in 4-bit LLM quantization by redistributing outliers through orthogonal transforms (Ashkboos et al., 2024; Liu et al., 2025), with outlier-aware variants targeting KV caches (Su et al., 2025). TurboQuant (Zandieh et al., 2026) combines random rotations with non-uniform scalar quantization for KV-cache compression. OSCAR (Zhou et al., 2026) combines calibrated spectral bases with Hadamard mixing for INT2 KV quantization. Serving-oriented methods integrate KV quantization with efficient attention kernels (Lin et al., 2025b; Jia et al., 2026).

2.2 Sparse Attention

KV cache eviction reduces both storage and read traffic but may discard information needed by future queries. Query-aware sparse attention methods retain the full cache and select relevant entries for each query, reducing read traffic while keeping unselected information available for later queries (Ribar et al., 2024; Tang et al., 2024). This flexibility can better preserve accuracy as token importance changes across queries (Tang et al., 2024).

Token-level sparsity. Token-based methods retain or evict cached tokens using attention statistics and positional heuristics (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024; Cai et al., 2025). Query-aware access selects KV pages using key statistics (Tang et al., 2024) or predicts upcoming attention to prefetch relevant entries from host memory (Lee et al., 2024). LServe (Yang et al., 2025b) combines KV quantization with static streaming attention and dynamic page selection.

Channel-level sparsity. Channel-based methods select dimensions of query–key dot products. ThinK (Xu et al., 2025) uses prompt query–key statistics to remove low-importance channels from older keys and reuses the mask during decoding. SparQ Attention, Double Sparsity, and Loki use selected channels or low-dimensional key representations to approximate token importance before attending to retrieved tokens (Ribar et al., 2024; Yang et al., 2024; Singhania et al., 2024). Related approaches reduce dimensionality through low-rank projections (Saxena et al., 2024; Chang et al., 2024; Lin et al., 2025a), which can be combined with quantization (Chang et al., 2024; Lin et al., 2025a). Dual-QK jointly designs Q/K coordinates for low-bit key quantization while keeping all transformed key channels available for dynamic selection based on current query magnitudes.

3 Method

3.1 Dual Transforms

We use row-vector notation and denote queries and keys after RoPE by Q and K, with head dimension d. Transforming keys changes their dot products unless queries receive a compensating transform. For any invertible matrix A, choosing $B = A^{-\mathsf{T}}$ gives

$$(QA)(KB)^{\mathsf{T}} = QK^{\mathsf{T}}. \tag{1}$$

When A is orthogonal, $B = A$, recovering a shared rotation for queries and keys. Dual-QK allows the two transforms to differ, so key scales can be adjusted without forcing the same adjustment onto queries. This identity preserves the original scores before quantization and pruning.

For each layer and KV head, we estimate the uncentered second moments $C_Q = \mathbb{E}[q^{\mathsf{T}}q]$ and $C_K = \mathbb{E}[k^{\mathsf{T}}k]$ from calibration data. Query statistics are pooled over heads sharing the KV head. We first reduce scale differences across key directions using partial whitening. With $G = C_K + \epsilon I$ and $\epsilon > 0$, keys are multiplied by $G^{-\alpha/2}$ and queries by its inverse, $G^{\alpha/2}$. The parameter $\alpha \in [0, 1]$ controls whitening strength; regularization limits amplification along directions with little calibration energy.

Whitening alone does not arrange query energy for pruning. We therefore rotate the compensated queries into their principal directions:

$$G^{\alpha/2} C_Q G^{\alpha/2} = U \Lambda U^{\mathsf{T}}, \tag{2}$$



Figure 2: Query (top) and key (bottom) magnitudes from Llama-3.1-8B-Instruct (layer 29, KV group 6, query head 26) in the original, Q-PCA, Hadamard, and Dual-QK coordinates. Q-PCA concentrates queries but leaves key imbalance. Hadamard spreads both, whereas Dual-QK concentrates queries while reducing key imbalance. Magnitude axes are scaled independently across panels.

where the diagonal entries of Λ are ordered from largest to smallest. Applying the same orthogonal matrix U to queries and keys gives the base transforms

$$A_0 = G^{\alpha/2}U, \qquad B_0 = G^{-\alpha/2}U. \tag{3}$$

The transformed query moment is $A_0^\top C_Q A_0 = \Lambda$: leading channels capture the dominant calibrated query energy. Meanwhile, partial whitening compresses the scale differences in the regularized key moment. At $\alpha = 0$, the pair reduces to Q-PCA; at $\alpha = 1$, the base transform whitens the regularized key moment. Increasing whitening strength also changes the query spectrum through the reciprocal scaling. Partial whitening controls how much the two distributions are reshaped, balancing key scale reduction with query-energy concentration. Figure 2 illustrates the resulting distributions. These moment properties describe average channel energy on the calibration distribution; dynamic selection accounts for variation across individual queries.

We choose α once per model and reuse the transforms across benchmarks and context lengths. The matrix decompositions are performed offline. During inference, each layer and KV head uses a fixed transform pair, while the channel mask adapts to the current queries. The moment identities and calibration procedure are given in Appendices C.1 and B.2.

3.2 Dynamic Query-Channel Pruning

The calibrated ordering identifies channels that are important on average, but their importance changes with each query. We therefore select channels dynamically. Query magnitude is a useful proxy when key channels have comparable scales: otherwise, a small query component can still multiply a large key component. Key flattening makes this proxy more appropriate, although it does not make query energy an exact measure of attention-score error.

In grouped-query attention (GQA) (Ainslie et al., 2023), g query heads share one KV head. Let $q'_h = q_h A_0$ be the base transformed query of head h. For each channel j, we compute the L2 norm across the group:

$$e_j = \sqrt{\sum_{h=1}^{g} (q'_{h,j})^2}. \tag{4}$$

We retain the m channels with the largest e_j and apply the same mask to all query heads in the group. This maximizes the retained query energy for the chosen channel budget. Multi-head attention (MHA) is the case $g = 1$, where selection reduces to query magnitude.

Attention dot products use the retained channels and keep the original normalization by $\sqrt{d}$. All key channels remain stored, allowing a channel skipped by one query to be used by the next. With the

channel-major layout in Section 3.3, the shared mask lets the kernel load only selected key channels. Quantization reduces bits per element, while pruning reduces the key elements read. Values and metadata are still read.

3.3 INT2 KV QUANTIZATION

In the base coordinates, a key error $\delta k'_j$ on a retained channel contributes $q'_{h,j}\delta k'_j/\sqrt{d}$ to head h's attention logit. The leading base query coordinate has the largest calibrated expected energy, making errors in its corresponding key coordinate especially consequential for attention scores. We therefore retain key channel 0 in BF16 and exclude it from the quantization range of the remaining channels. Those channels use one asymmetric INT2 grid per token and KV head. Removing channel 0 from range estimation also prevents its magnitude from reducing the resolution available to the other channels. Precision protection does not force channel 0 into every dot product; it remains subject to the dynamic mask.

Second moments do not fully describe the ranges used by a low-bit quantizer. We further refine channel scales with a positive diagonal matrix D, using $A = A_0D$ and $B = B_0D^{-1}$. Inspired by reciprocal channel scaling (Lin et al., 2025b), we calibrate D to reduce the attention-score error caused by key quantization and fix its channel-0 entry to one. The two scale factors cancel in each channel. Thus, rescaling preserves both the full dot product and the contribution of any fixed channel set in the base dual coordinates before quantization. The refinement is folded into the transforms before inference. Channel selection still uses the base queries q_hA_0; retained dot products use the final scaled queries and keys.

For values, we follow OSCAR's calibrated rotation, clipping, and INT2 quantization recipe (Zhou et al., 2026), reversing the value rotation after aggregation. We retain the first 64 and most recent 256 tokens in BF16 for both keys and values. These tokens use the same query-channel mask. Quantization details appear in appendix.

Within each 128-token page, INT2 keys are stored channel-major for each KV head, with one channel occupying a contiguous 32-byte row. The decode kernel issues asynchronous copies only for rows selected by the shared query mask. Values are stored token-major, with separate buffers for quantization metadata and BF16-protected entries.

3.4 BUCKET-RELATIVE ROPE

The dual basis is calibrated over a finite positional window. At longer positions, RoPE changes the channel statistics presented to this fixed basis. We address this mismatch by expressing stored keys relative to local bucket anchors. This changes their positional frame without recalibrating the transforms.

We divide positions into buckets of width w, with bucket b starting at $a_b = bw$. Let T_b undo the RoPE rotation at this anchor. A key k_n in the bucket and a current query q are represented as

$$z_n = k_nT_bB, \qquad \widetilde{q}_b = qT_bA. \tag{5}$$

Because T_b is orthogonal and the dual transforms cancel, their uncompressed dot product is unchanged:

$$\widetilde{q}_b z_n^{\mathsf{T}} = qk_n^{\mathsf{T}}. \tag{6}$$

The stored key's residual position is $n - a_b \in [0, w)$. We use $w = 2048$, matching the calibration window, and reuse the same dual pair for every bucket. This bounds the positional offsets of stored keys; the query's relative position can still exceed the window. The anchor adjustment precedes the dual transform so that the calibrated basis acts on bucket-relative coordinates. General channel transforms do not commute with RoPE, so reversing this order changes the representation used for quantization and channel selection.

During decoding, each new key is transformed in its bucket frame and stored. For each bucket, the current query heads receive the corresponding anchor adjustment, and Equation 4 is applied to their base coordinates $q_hT_bA_0$. Thus, channel masks can differ across buckets. The selected channels produce attention scores using the final scaled coordinates. Scores from all buckets and precision tiers share one global softmax, followed by value aggregation and the inverse value rotation.

Table 1: Generative accuracy (%) with 40% query-channel sparsity for quantized methods. BF16 is unpruned. Scores report mean ± standard deviation over five seeds (three for Ministral-3). Capacity and bandwidth are effective bits per KV element at 128K context. **Bold** marks the highest mean among quantized methods within each model.

Model	Method	KV bits per element		Benchmark score (%)				
		Capacity	Bandwidth	GPQA-Diamond	HumanEval	LCB v6	MATH-500	AIME25
Llama-3.1-8B -Instruct	BF16	16	16	26.2 ± 1.5	66.7 ± 2.5	16.5 ± 1.4	49.4 ± 1.5	1.3 ± 1.8
	TurboQuant	3.13	2.53	17.7 ± 2.4	24.3 ± 1.4	3.8 ± 0.5	13.8 ± 1.1	0.7 ± 1.5
	KIVI	3.01	2.41	19.2 ± 2.7	55.2 ± 4.3	9.8 ± 2.6	28.9 ± 1.4	0.7 ± 1.5
	OSCAR	2.28	1.88	16.4 ± 3.0	28.2 ± 2.2	3.2 ± 1.0	16.4 ± 0.4	**1.3 ± 1.8**
	Dual-QK (ours)	2.34	1.93	**23.2 ± 1.9**	**64.3 ± 2.9**	**14.2 ± 1.3**	**43.0 ± 2.0**	0.7 ± 1.5
Qwen3-4B -Thinking-2507	BF16	16	16	65.6 ± 1.9	96.2 ± 0.7	54.0 ± 1.6	98.0 ± 0.3	74.7 ± 3.8
	TurboQuant	3.13	2.53	3.5 ± 1.6	0.0 ± 0.0	0.0 ± 0.0	0.2 ± 0.2	0.0 ± 0.0
	KIVI	3.01	2.41	54.3 ± 2.8	88.4 ± 1.8	31.6 ± 2.1	92.5 ± 1.0	42.0 ± 6.9
	OSCAR	2.28	1.88	23.1 ± 2.1	33.9 ± 1.7	2.1 ± 1.1	51.8 ± 1.6	6.0 ± 1.5
	Dual-QK (ours)	2.34	1.93	**60.8 ± 2.2**	**95.0 ± 0.8**	**38.8 ± 2.3**	**94.3 ± 0.4**	**48.0 ± 5.6**
Qwen3-8B	BF16	16	16	56.1 ± 2.2	90.1 ± 1.3	50.2 ± 2.9	96.4 ± 0.3	65.3 ± 3.0
	TurboQuant	3.13	2.53	19.8 ± 2.9	16.0 ± 1.2	2.4 ± 0.8	31.7 ± 1.7	0.0 ± 0.0
	KIVI	3.01	2.41	49.0 ± 3.2	87.3 ± 1.8	32.4 ± 2.9	92.0 ± 1.0	47.3 ± 4.9
	OSCAR	2.28	1.88	38.1 ± 3.2	66.1 ± 4.3	14.8 ± 2.4	80.3 ± 0.8	13.3 ± 4.1
	Dual-QK (ours)	2.34	1.93	**55.7 ± 1.9**	**90.6 ± 2.0**	**36.2 ± 1.2**	**95.0 ± 0.3**	**53.3 ± 2.4**
Ministral-3-14B -Reasoning-2512	BF16	16	16	53.7 ± 1.1	96.3 ± 1.2	55.7 ± 0.8	95.4 ± 0.7	43.3 ± 5.8
	TurboQuant	3.13	2.53	44.8 ± 4.9	93.3 ± 2.7	40.7 ± 2.2	85.4 ± 0.9	23.3 ± 0.0
	KIVI	3.01	2.41	45.1 ± 0.8	92.3 ± 1.8	41.5 ± 1.2	86.4 ± 0.2	27.8 ± 6.9
	OSCAR	2.28	1.88	37.4 ± 2.5	91.5 ± 0.6	37.2 ± 1.2	85.5 ± 0.8	25.6 ± 1.9
	Dual-QK (ours)	2.34	1.93	**50.2 ± 3.2**	**96.3 ± 0.6**	**47.6 ± 3.6**	**92.7 ± 0.4**	**37.8 ± 1.9**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Models and benchmarks. We evaluate Dual-QK on Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-4B-Thinking-2507, Qwen3-8B (Yang et al., 2025a), and Ministral-3-14B-Reasoning-2512 (Liu et al., 2026). Generative benchmarks cover knowledge reasoning with GPQA-Diamond, code generation with HumanEval and LiveCodeBench v6, and mathematical reasoning with MATH-500 and AIME25 (Rein et al., 2023; Chen et al., 2021; Jain et al., 2025; Hendrycks et al., 2021; Zhang & Math-AI, 2025). We report pass@1 for code generation and accuracy for reasoning tasks. Long-context retrieval is evaluated on the needle-in-a-haystack (NIAH) tasks of RULER (Hsieh et al., 2024), with context lengths from 4K to 128K. All methods use the same prompts, sampling parameters, generation limits, and scoring procedure within each model–benchmark pair. Repetition counts are specified in the corresponding table and figure captions.

Baselines and comparison protocol. We compare Dual-QK with KIVI (Liu et al., 2024) and two rotation-based quantization methods, TurboQuant (Zandieh et al., 2026) and OSCAR (Zhou et al., 2026). All model weights remain in BF16. For the main accuracy comparison, we apply the same query-channel pruning policy at 40% sparsity to all quantized methods and use unpruned BF16 as the reference. Following OSCAR, Dual-QK retains the KV entries of the first 64 and most recent 256 tokens in BF16. At 128K context, we report format-based estimates of KV storage and read volume as effective bits per KV element (BPE). Capacity BPE describes the stored cache, while bandwidth BPE estimates the KV data read per decoding step with channel selection. Both account for quantization metadata and high-precision entries, including the protected sink and recent tokens.

Calibration and implementation. Dual-QK is calibrated on 128 sequences of 2,048 tokens from the WikiText-2 training set (Merity et al., 2016) and we select α using separate 8K and 32K inputs. We collect post-RoPE query and key second moments for each layer and KV head, pooling query statistics across heads that share the same KV head. The transforms are constructed offline without retraining model weights and reused across benchmarks and context lengths. We implement Dual-QK in SGLang (Zheng et al., 2024). We evaluate accuracy on NVIDIA H200 GPUs and decoding throughput on a single NVIDIA RTX PRO 6000 Blackwell GPU with 96 GB of memory.

Table 2: RULER NIAH accuracy (%) across context lengths with 40% query-channel sparsity for quantized methods. BF16 is the unpruned reference. Entries report mean ± sample standard deviation across three NIAH data draws, each containing 25 examples per subtask across eight subtasks. Capacity and bandwidth are effective bits per KV element at a 128K context. **Bold** denotes the highest mean among quantized methods.

Model	Method	KV bits per element		Context length					
		Capacity	Bandwidth	4K	8K	16K	32K	64K	128K
Llama-3.1-8B-Instruct	BF16	16	16	100.0 ± 0.1	99.9 ± 0.0	100.0 ± 0.1	100.0 ± 0.1	99.3 ± 0.3	94.8 ± 0.4
	OSCAR	2.28	1.88	65.0 ± 1.0	53.6 ± 1.6	51.3 ± 0.4	42.9 ± 1.7	39.5 ± 1.8	25.2 ± 0.6
	Dual-QK (ours)	2.34	1.93	**92.4 ± 0.4**	**92.5 ± 0.8**	**89.0 ± 0.2**	**83.7 ± 0.7**	**78.9 ± 0.6**	**58.9 ± 1.5**
Qwen3-4B-Thinking-2507	BF16	16	16	100.0 ± 0.0	99.0 ± 0.1	98.3 ± 0.4	96.8 ± 0.4	96.7 ± 0.6	94.0 ± 0.1
	OSCAR	2.28	1.88	78.4 ± 0.8	72.0 ± 0.4	61.2 ± 0.4	55.0 ± 0.8	32.7 ± 1.6	2.0 ± 0.0
	Dual-QK (ours)	2.34	1.93	**94.0 ± 0.3**	**92.3 ± 0.1**	**88.8 ± 0.2**	**84.8 ± 0.4**	**81.5 ± 0.7**	**64.4 ± 0.2**
Qwen3-8B	BF16	16	16	99.9 ± 0.1	99.8 ± 0.0	100.0 ± 0.1	99.0 ± 0.1	86.7 ± 0.2	76.5 ± 1.3
	OSCAR	2.28	1.88	87.1 ± 0.2	80.0 ± 0.9	74.0 ± 0.9	55.1 ± 1.6	43.0 ± 0.9	10.8 ± 0.5
	Dual-QK (ours)	2.34	1.93	**95.6 ± 0.3**	**95.4 ± 0.1**	**91.9 ± 0.1**	**83.5 ± 0.7**	**68.4 ± 0.7**	**57.0 ± 1.4**
Ministral-3-14B-Reasoning-2512	BF16	16	16	99.7 ± 0.1	97.8 ± 0.1	98.6 ± 0.2	91.3 ± 0.6	92.1 ± 0.7	82.9 ± 0.9
	OSCAR	2.28	1.88	93.3 ± 0.5	89.8 ± 0.5	**85.8 ± 0.3**	76.3 ± 1.0	73.1 ± 0.3	65.6 ± 1.2
	Dual-QK (ours)	2.34	1.93	**95.2 ± 0.3**	**90.4 ± 0.5**	83.2 ± 0.3	**78.7 ± 0.4**	**77.8 ± 1.0**	**71.1 ± 1.2**

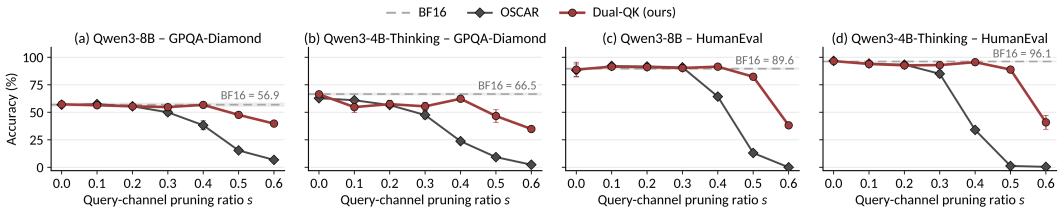

Figure 3: Accuracy on GPQA-Diamond and HumanEval as query-channel pruning increases for Qwen3-8B and Qwen3-4B-Thinking-2507. Dashed lines indicate the unpruned BF16 references. Solid curves show OSCAR and Dual-QK with quantized KV caches, including at zero pruning. Error bars indicate standard deviation over three seeds.

4.2 Main Results

Generative benchmarks. Table 1 shows that Dual-QK maintains higher generative accuracy under 40% query-channel pruning at a storage cost comparable to OSCAR. Dual-QK uses 2.34 capacity BPE, compared with 2.28 for OSCAR, while improving accuracy on most knowledge, coding, and mathematical reasoning tasks. On Qwen3-4B-Thinking-2507, HumanEval pass@1 reaches 95.0%, compared with 33.9% for OSCAR and 96.2% for unpruned BF16. Dual-QK also achieves higher mean accuracy than KIVI on most tasks while using 2.34 rather than 3.01 capacity BPE. Among the evaluated quantized methods, Dual-QK generally remains closest to the BF16 reference.

Long-context retrieval. Table 2 reports NIAH retrieval accuracy from 4K to 128K context lengths. Dual-QK improves over OSCAR at most lengths across all four models, with particularly large gains on Llama and the two Qwen models. At 64K, Llama-3.1-8B-Instruct and Qwen3-4B-Thinking-2507 reach 78.9% and 81.5%, compared with 39.5% and 32.7% for OSCAR. The advantage extends to 128K, where Dual-QK retains 57.0∼64.4% accuracy across Llama and the two Qwen models, compared with 2.0–25.2% for OSCAR. Ministral shows smaller improvements at most lengths, although OSCAR remains higher at 16K. These gains use the same 40% query-channel sparsity and comparable KV storage and read budgets. The selected transforms remain fixed throughout the context-length sweep.

4.3 Ablation Studies

Sensitivity to channel pruning. Figure 3 varies the query-channel pruning ratio from 0% to 60% on GPQA-Diamond and HumanEval for Qwen3-8B and Qwen3-4B-Thinking-2507. Dual-QK and OSCAR perform similarly at low sparsity, but the accuracy gap widens as more channels are removed. At 40%, Dual-QK retains most of its accuracy without pruning on both tasks, while OSCAR declines substantially. Dual-QK remains ahead on both tasks and models at 50%. At 60%, its HumanEval

Table 3: Component ablation of Dual-QK on Qwen3-8B with INT2 KV quantization and 40% query-channel sparsity. BF16 is unpruned. Entries report mean ± standard deviation. BF16 and Full Dual-QK reuse the five-seed results from Table 1. The remaining configurations use three seeds.

Configuration	GPQA-D	HumanEval	LCB v6	MATH-500	AIME25
BF16 reference	56.1 ± 2.2	90.1 ± 1.3	50.2 ± 2.9	96.4 ± 0.3	65.3 ± 3.0
Full Dual-QK ($\alpha = 0.5$)	55.7 ± 1.9	90.6 ± 2.0	36.2 ± 1.2	95.0 ± 0.3	53.3 ± 2.4
w/o channel-0 protection	46.5 ± 4.5	71.5 ± 1.9	16.5 ± 1.2	87.3 ± 1.0	24.4 ± 1.9
w/o bucket-relative RoPE	33.2 ± 0.3	51.4 ± 2.5	8.4 ± 0.8	51.1 ± 1.1	2.2 ± 1.9
w/o diagonal refinement ($D = I$)	52.5 ± 4.8	87.6 ± 2.8	35.4 ± 1.9	89.5 ± 5.4	51.1 ± 9.6
Q-PCA base ($\alpha = 0$)	45.3 ± 3.8	75.2 ± 3.9	15.3 ± 1.3	78.7 ± 11.6	17.8 ± 1.9
Hadamard base (H)	7.9 ± 1.1	4.5 ± 2.1	0.3 ± 0.4	9.8 ± 2.6	0.0 ± 0.0

Table 4: Generative accuracy (%) **without query-channel pruning**. Benchmark entries report mean ± standard deviation over five seeds. BF16 is the unquantized reference. **Bold** denotes the highest score among quantized methods within each model.

Model	Method	GPQA-D	HumanEval	LCB v6	MATH-500	AIME25	Mean
Llama-3.1-8B-Instruct	BF16	26.2 ± 1.5	66.7 ± 2.5	16.5 ± 1.4	49.4 ± 1.5	1.3 ± 1.8	32.0
	TurboQuant	**26.8 ± 3.1**	59.3 ± 2.9	12.5 ± 1.3	39.9 ± 1.1	0.0 ± 0.0	27.7
	OSCAR	24.3 ± 1.7	65.6 ± 3.0	15.3 ± 0.9	50.8 ± 0.5	**2.0 ± 1.8**	31.6
	Dual-QK (ours)	26.4 ± 1.9	**66.0 ± 1.7**	**16.5 ± 0.7**	**50.9 ± 0.6**	1.3 ± 3.0	**32.2**
Qwen3-8B	BF16	56.1 ± 2.2	90.1 ± 1.3	50.2 ± 2.9	96.4 ± 0.3	65.3 ± 3.0	71.6
	TurboQuant	47.1 ± 2.2	75.7 ± 1.6	24.7 ± 1.7	92.4 ± 0.6	50.7 ± 2.8	58.1
	OSCAR	**56.6 ± 1.2**	**92.2 ± 1.1**	**46.1 ± 1.6**	**95.8 ± 0.5**	**61.3 ± 1.8**	**70.4**
	Dual-QK (ours)	55.9 ± 2.2	92.0 ± 0.9	43.2 ± 2.4	95.2 ± 0.4	**61.3 ± 3.0**	69.5

accuracy also drops sharply. These results show that accuracy under quantization alone does not guarantee robustness to channel pruning.

Component ablation. Table 3 compares component removals and alternative Q/K bases on Qwen3-8B at 40% query-channel sparsity. Q-PCA ($\alpha = 0$) lowers LiveCodeBench accuracy from 36.2% to 15.3% and AIME25 from 53.3% to 17.8%. These results indicate that concentrating query energy alone is insufficient for joint quantization and pruning. The Hadamard base yields larger accuracy losses. Removing channel-0 protection or bucket-relative RoPE lowers HumanEval accuracy from 90.6% to 71.5% and 51.4%, respectively. Removing diagonal refinement lowers the mean accuracy on all five benchmarks, although variability across seeds is substantial on MATH-500 and AIME25.

Accuracy without channel pruning. Table 4 evaluates quantization accuracy without pruning. Dual-QK achieves a mean accuracy across the five benchmarks of 32.2% on Llama-3.1-8B-Instruct and 69.5% on Qwen3-8B, compared with 31.6% and 70.4% for OSCAR, respectively. The means differ by less than one percentage point on each model. OSCAR achieves a higher mean on Qwen3-8B, with the largest gap on LiveCodeBench. Together with Figure 3, these results show that Dual-QK maintains competitive accuracy without pruning while supporting higher channel pruning ratios.

Calibration-domain sensitivity. Table 5 compares WikiText-2, MMLU, and GPQA-Diamond calibration on Qwen3-4B-Thinking-2507 at 0% and 40% query-channel sparsity. At 40%, the spread across calibration datasets is 0.5 percentage points on MATH-500, 2.0 on HumanEval, and 2.1 on LiveCodeBench. GPQA-Diamond calibration does not yield a higher GPQA-Diamond mean than WikiText-2 at either sparsity. AIME25 shows a larger spread of 12.2 points without pruning and 5.6 points at 40%. WikiText-2 calibration is competitive on most tasks, while AIME25 is more sensitive to the calibration source.

4.4 THROUGHPUT

Decoding Speedup. Figure 4 compares BF16, OSCAR, and Dual-QK across all four models. At batch size 1, Dual-QK achieves higher throughput than both baselines at every tested input length under the evaluated configurations. Its relative advantage over OSCAR grows with input length as accessing the KV cache becomes more costly. On Qwen3-4B-Thinking-2507 at 90K, Dual-QK reaches 90 tokens/s compared with 24 tokens/s for BF16, giving a 3.75× speedup. These measurements include transformations and cache operations during decoding.

Table 5: Calibration-domain ablation of Dual-QK on Qwen3-4B-Thinking-2507. Results are reported as mean ± standard deviation over three seeds. Sparsity denotes query-channel sparsity. BF16 is the unpruned reference. **Bold** highlights WikiText, our default calibration dataset, and its results.

Method	Sparsity	Calibration	GPQA-D	HumanEval	LCB v6	MATH-500	AIME25
BF16	—	—	63.0 ± 2.4	95.5 ± 0.9	51.7 ± 0.4	97.5 ± 0.3	72.2 ± 3.8
Dual-QK (ours)	0%	**WikiText**	**62.3 ± 1.6**	**94.9 ± 0.4**	**44.5 ± 0.9**	**96.1 ± 0.2**	**47.8 ± 1.9**
		MMLU	61.3 ± 2.8	94.7 ± 0.7	45.3 ± 2.9	95.7 ± 0.4	57.8 ± 1.9
		GPQA-D	62.3 ± 1.8	94.9 ± 0.9	45.0 ± 1.5	96.7 ± 0.8	60.0 ± 5.8
	40%	**WikiText**	**58.1 ± 1.3**	**92.9 ± 1.8**	**38.9 ± 2.3**	**93.9 ± 0.8**	**51.1 ± 1.9**
		MMLU	56.6 ± 1.5	94.7 ± 0.4	39.4 ± 1.6	93.7 ± 1.0	50.0 ± 0.0
		GPQA-D	57.7 ± 1.1	94.9 ± 2.1	41.0 ± 1.6	94.2 ± 0.4	55.6 ± 1.9

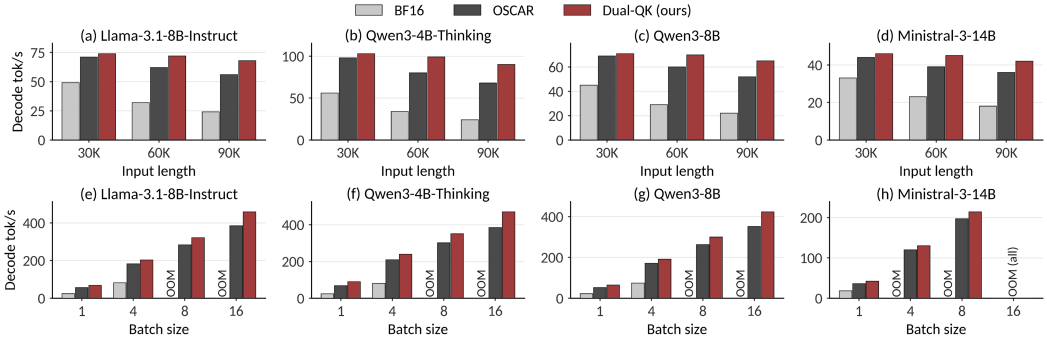

Figure 4: Decoding throughput excluding prefill. Top: throughput versus input length @ batch size = 1. Bottom: throughput versus batch size @ input length = 90K. BF16/OSCAR are unpruned; Dual-QK uses 40% query-channel sparsity. OOM indicates insufficient GPU memory.

Effect of Batch Size. The batch sweep shows how cache compression increases serving capacity within the same GPU memory budget. At 90K, BF16 runs out of memory at batch sizes 8 and 16 on Llama and both Qwen models, while OSCAR and Dual-QK support batch size 16. On Ministral, BF16 runs out of memory starting at the tested batch size of 4, while both quantized methods support batch size 8. All three methods run out of memory at batch size 16. Across the feasible batch sizes shown, Dual-QK maintains higher aggregate throughput than OSCAR.

5 Discussion

Practical use. KV-cache quantization enables larger batches on a single GPU. As weight accesses are amortized across more requests, KV-cache reads can become the bottleneck. Channel pruning can then provide additional decoding speedup.

Limitations. The additional speedup from pruning is limited at small batch sizes, where selection overhead can offset the savings from skipped key reads. Its benefit grows when KV-cache access accounts for more of the decode step, as in larger batches and longer contexts. Accuracy gaps to BF16 remain at long contexts. Bucket-relative RoPE bounds stored-key offsets, but query offsets relative to older buckets can exceed the 2K calibration window. We leave calibration over broader positional ranges to future work.

6 Conclusion

We introduce Dual-QK, a paired query–key transformation for combining low-bit KV quantization with dynamic query-channel pruning. Partial key whitening and query-energy ordering construct a representation that balances key scales and concentrates query energy while preserving attention scores before compression. Experiments show improved accuracy under INT2 quantization and channel pruning, together with higher decoding throughput in the evaluated decoding configurations. The results demonstrate the value of designing query and key coordinates for quantization and channel selection together.

AI USE STATEMENT

In this work, we used generative AI tools for translation, feedback on research methodology and experimental design, interpretation of experimental results, and code implementation. We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, or proposing or refining hypotheses, and dataset cleaning and reformatting and qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for identifying related work and potential baselines, polishing our writing, and drafting portions of the manuscript. We have reviewed all AI-assisted work. We checked suggested references against the original sources and reviewed AI-assisted text and interpretations for technical accuracy. LLM-generated code was verified and tested for correctness by 3 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For reproducibility, we describe the models, benchmarks, and experimental setup in Section 4.1. Further details on calibration, quantization configurations, and evaluation protocols are provided in Appendix B. We plan to release our code and calibration data upon paper acceptance, subject to internal approval.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4895–4901, 2023.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240. Curran Associates, Inc., 2024. doi: 10.52202/079017-3180. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b5b939436789f76f08b9d0da5e81af7c-Paper-Conference.pdf.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Priyansh Bhatnagar, Ashkan Moradifirouzabadi, Se-Hyun Yang, SeungJae Lee, Jungwook Choi, and Mingu Kang. STAR-KV: Low-rank KV cache compression via soft thresholding for adaptive rank control. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=lJjH1q6RwY.

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

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Haojie Duanmu, Zhihang Yuan, Xiuhong Li, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. Skvq: Sliding-window key and value cache quantization for large language models. *arXiv preprint arXiv:2405.06219*, 2024.

Samuel Fernández-Menduiña, Amir Ziashahabi, Eduardo Pavez, Antonio Ortega, and Salman Avestimehr. Spend bits where queries look: Kv cache vector quantization with attention-preserving transforms. *arXiv preprint arXiv:2608.04074*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

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

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Simon Jegou and Maximilian Jeblick. Kvzap: Fast, adaptive, and faithful kv cache pruning. *arXiv preprint arXiv:2601.07891*, 2026.

Jinda Jia, Jisen Li, Zhongzhu Zhou, Jung Hwan Heo, Jue Wang, Tri Dao, Shuaiwen Leon Song, Ben Athiwaratkun, Chenfeng Xu, Tianyi Zhang, et al. Saw-int4: System-aware 4-bit kv-cache quantization for real-world llm serving. *arXiv preprint arXiv:2604.19157*, 2026.

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

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pp. 155–172, 2024.

Junyan Li, Yang Zhang, Muhammad Yusuf Hassan, Talha Chafekar, Tianle Cai, Zhile Ren, Pengsheng Guo, Foroozan Karimzadeh, Colorado Reed, Chong Wang, and Chuang Gan. CommVQ: Commutative vector quantization for KV cache compression. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=sbbyCB39HN.

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

Huanxuan Liao, Yixing Xu, Shizhu He, Guanchen Li, Xuanwu Yin, Dong Li, Emad Barsoum, Jun Zhao, and Kang Liu. Spark: Query-aware unstructured sparsity with recoverable kv cache channel pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31961–31969, 2026.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, Tianqi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. Matryoshkakv: Adaptive kv compression via trainable orthogonal projection. In *International Conference on Learning Representations*, volume 2025, pp. 86669–86690, 2025a.

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

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: LLM quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ogO6DGE6FZ.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=L057s2Rq8O.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.

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

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for kv cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15332–15344, 2024.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. *Advances in Neural Information Processing Systems*, 37: 16692–16723, 2024.

Zunhai Su, Hanyu Wei, Zhe Chen, Wang Shen, Linge Li, Huangqi Yu, and Kehong Yuan. Rotatekv: Accurate and robust 2-bit kv cache quantization for llms via outlier-aware adaptive rotations. In James Kwok (ed.), *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 6200–6208. International Joint Conferences on Artificial Intelligence Organization, 8 2025. doi: 10.24963/ijcai.2025/690. URL https://doi.org/10.24963/ijcai.2025/690. Main Track.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. Think: Thinner key cache by query-driven pruning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=n0OtGl6VGb.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Shang Yang, Junxian Guo, Haotian Tang, Qinghao Hu, Guangxuan Xiao, Jiaming Tang, Yujun Lin, Zhijian Liu, Yao Lu, and Song Han. Lserve: Efficient long-sequence llm serving with unified sparse attention. *Proceedings of Machine Learning and Systems*, 7, 2025b.

Shuo Yang, Ying Sheng, Joseph E Gonzalez, Ion Stoica, and Lianmin Zheng. Post-training sparse attention with double sparsity. *arXiv preprint arXiv:2408.07092*, 2024.

Amir Zandieh, Majid Daliri, and Insu Han. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25805–25813, 2025.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tO3ASKZlok.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Yike Zhang, Zhiyuan He, Huiqiang Jiang, Chengruidong Zhang, Yuqing Yang, Jianyong Wang, and Lili Qiu. Leank: Learnable k cache channel pruning for efficient decoding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31122–31137, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209, 2024.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

Zhongzhu Zhou, Donglin Zhuang, Jisen Li, Ziyan Chen, Shuaiwen Leon Song, Ben Athiwaratkun, and Xiaoxia Wu. Oscar: Offline spectral covariance-aware rotation for 2-bit kv cache quantization. *arXiv preprint arXiv:2605.17757*, 2026.

Table 6: Models and generation settings. Max. tokens is the generation limit per problem. A dash indicates no top-p or top-k filtering.

Model	α	Temp.	Top-p	Top-k	Max. tokens
Llama-3.1-8B-Instruct	0.75	0.6	0.9	–	16,384
Qwen3-4B-Thinking-2507	0.5	0.6	0.95	20	32,768
Qwen3-8B	0.5	0.6	0.95	20	32,768
Ministral-3-14B-Reasoning-2512	0.375	1.0	–	–	65,536

Table 7: Evaluation datasets. LiveCodeBench v6 uses the February∼May 2025 subset. LongBench includes eight tasks.

Benchmark	Examples	Metric
GPQA-Diamond	198	Accuracy
HumanEval	164	Pass@1
LiveCodeBench v6	131	Pass@1
MATH-500	500	Accuracy
AIME25	30	Accuracy
RULER NIAH	200 per length	Substring match
LongBench	200∼500 per task	Task-specific

A Additional Related Work

Vector quantization. CommVQ (Li et al., 2025) uses additive vector quantization with codebooks designed to commute with RoPE, allowing reconstruction to be integrated into attention. NOVA-KV (Fernández-Menduiña et al., 2026) derives non-orthogonal key transforms under an attention-weighted distortion criterion. Its transform maps query-weighted key error to Euclidean error, and it applies vector quantization to groups of transformed coordinates. Both NOVA-KV and Dual-QK use query statistics to guide key compression. Dual-QK designs the coordinates for scalar INT2 quantization and dynamic channel selection, with query energy concentrated in a small subset of channels.

Quantization and low-rank compression. GEAR (Kang et al., 2024) combines low-bit quantization with low-rank correction of quantization residuals and sparse storage of outliers. STAR-KV (Bhatnagar et al., 2026) learns ranks at attention-head and block levels through differentiable soft thresholding, and combines low-rank representations with mixed-precision quantization. Dual-QK retains the full transformed channel dimension in storage. Its channel mask controls which key coordinates are read for the current query, so unselected coordinates remain available at later steps.

Channel pruning. LeanK (Zhang et al., 2025) learns static key-channel masks in two stages, first estimating channel importance and then fitting masks under sparsity and hardware-alignment constraints. SparK (Liao et al., 2026) uses query-aware unstructured channel pruning and approximates omitted key entries during attention using statistics collected at prefill. These approaches reduce the stored key representation. Dual-QK instead stores every transformed key channel at low precision and computes a shared read mask from the current GQA queries for each bucket.

Learned token eviction. KVzap (Jegou & Jeblick, 2026) trains lightweight predictors on hidden states to estimate per-head token importance. It uses score thresholds and a protected recent window to prune KV entries during prefill and decoding. This adapts the stored token budget to the input. Dual-QK keeps all token positions and reduces the precision and number of key channels read at each step. Combining it with token eviction would additionally require deciding which positions to retain.

B Experimental Details

B.1 Models and Evaluation

Tables 6 and 7 summarize the models, generation settings, and evaluation datasets. Each generative run samples one response per problem. Main generative results report mean ± standard deviation over five runs for Llama and Qwen3, and three for Ministral-3.

We evaluate eight RULER NIAH subtasks (Hsieh et al., 2024) through lm-evaluation-harness, with 25 examples each at context lengths from 4K to 128K. We use greedy decoding with a 128-token

Table 8: Relative hidden-state error (%) for selecting α, averaged equally over 0% and 40% sparsity. Lower is better. Bold marks the selected value for each model.

α	Llama-3.1-8B	Qwen3-4B	Qwen3-8B	Ministral-3-14B
0	12.35	23.71	46.10	4.07
0.125	9.80	12.76	30.70	3.84
0.25	9.62	13.98	19.11	3.71
0.375	9.84	11.57	11.12	**3.64**
0.5	10.44	**7.68**	**8.66**	3.89
0.625	9.32	10.80	13.46	4.10
0.75	**9.27**	15.22	16.42	4.41
0.875	10.13	15.55	27.24	4.65
1	10.07	21.31	47.81	5.79

limit and average scores equally across subtasks. Results report mean $\pm$ standard deviation over three independently generated datasets. LongBench uses greedy decoding and one run per configuration, following Appendix E.2.

B.2 Calibration

Data and statistics. We use the first 128 non-overlapping 2,048-token windows from WikiText-2 training data (Merity et al., 2016), totaling 262,144 tokens. Data are tokenized separately for each model, with the two Qwen3 models sharing a token stream. BF16 forward passes collect post-RoPE queries and keys, after QK normalization for Qwen3. We accumulate second moments in FP64 per layer and KV head, pooling query heads within each GQA group. Transforms are computed in FP32 with $\epsilon = 10^{-5}\,\mathrm{tr}(C_K)/d$.

Diagonal refinement. We fit D per layer and KV head to minimize the query-weighted key error in Equation 9, keeping $\delta_0 = 1$. In base coordinates, we sample 2,048 fitting keys and 2,048 validation keys from a Gaussian with the calibrated key mean and covariance. We first optimize the logarithm of the uniform-noise approximation for 400 Adam steps at learning rate 0.06. We then optimize sampled INT2 error for 300 steps at learning rate 0.02, using a straight-through estimator for rounding. The free scales are normalized to unit geometric mean. We retain D only if it reduces validation error relative to $D = I$, then fold the final scales into A and B.

Whitening strength. We select α from $\{0, 1/8, \ldots, 1\}$ once per model, fitting D for each candidate. Selection uses 64 inputs not used for calibration. At each of 8K and 32K tokens, these comprise 16 WikiText-2 windows and 16 synthetic documents containing four passkeys. We minimize relative squared error of the final decoder hidden states against BF16, averaged over inputs and equally over 0% and 40% sparsity. Table 8 reports the criterion. The selected transforms are reused across benchmarks and context lengths.

Value quantization. We use OSCAR's calibrated value rotation composed with a Hadamard transform (Zhou et al., 2026), fitted on the first 32 calibration windows. Values use per-token asymmetric INT2 with a clipping ratio of 0.92. Dual-QK keys are not clipped.

Calibration-domain ablation. For Table 5, we refit all calibrated components with $\alpha = 0.5$ and 2,048-token windows. MMLU uses 1,531 validation prompts across all subjects, forming 98 windows. For GPQA-Diamond, each prompt contains a question and its four answer choices. The 198 prompts form 25 windows. These reuse the GPQA evaluation questions, whereas the default calibration uses WikiText-2. The ablation and its BF16 reference use three runs on one RTX PRO 6000 Blackwell Server Edition GPU.

B.3 Cache Configurations and Bit Accounting

Table 9 summarizes cache configurations and BPE. At nominal 40% query-channel sparsity, each KV group selects 77 of 128 channels for key reads while storing all channels. KIVI selects channels in model coordinates, and TurboQuant and OSCAR in their rotated coordinates. Dual-QK selects channels in base dual coordinates before D, separately for each 2,048-token bucket.

KIVI quantizes keys per channel in 32-token groups and values per token in 32-channel groups. TurboQuant uses random orthogonal rotations and 3-bit Lloyd–Max codebooks for keys and values. OSCAR uses per-token INT2 after RHP_{br}, with clipping ratios of 0.96 for keys and 0.92 for values.

Table 9: Cache configurations and effective KV bits per element at 128K context. Quantized methods use 40% query-channel sparsity. BF16 is unpruned. Quantization bits exclude metadata and BF16 entries. Dual-QK keeps key channel 0 in BF16.

Method	Quant. bits		Pruning basis	BF16 tokens	KV bits per element	
	K	V			Capacity	Bandwidth
BF16	16	16	Unpruned	All	16	16
TurboQuant	3	3	Rotated	None	3.13	2.53
KIVI	2	2	Model	Up to 128 recent	3.01	2.41
OSCAR	2	2	Rotated	64 sink, 256 recent	2.28	1.88
Dual-QK (ours)	2	2	Base dual	64 sink, 256 recent	2.34	1.93

Dual-QK quantizes keys per token after bucket anchoring and B, excludes channel 0 from the INT2 range, and uses the same value format as OSCAR.

Capacity. Capacity BPE averages stored bits over key and value elements, including metadata and BF16 entries. A quantized Dual-QK key occupies 302 bits, comprising 254 INT2 bits, 16 bits for channel 0, and 32 bits for the BF16 scale and zero point. Its value occupies 288 bits. Including 64 sink and 256 recent BF16 tokens at a context length of 131,072 gives 2.34 BPE overall.

Bandwidth. Bandwidth BPE counts selected key channels, all value channels, and metadata. A 77-channel mask containing channel 0 reads 200 key bits, comprising 152 INT2 bits, 16 BF16 bits, and 32 metadata bits. Applying the same mask to BF16 tokens gives 1.93 BPE overall. Masks omitting channel 0 read fewer bits. Both BPE measures exclude fixed transforms. Bandwidth BPE also excludes transaction alignment and kernel rereads, with measured traffic reported in Appendix D.1.

B.4 Implementation

Software and execution. Accuracy evaluations use SGLang 0.5.10.post1, PyTorch 2.9.1, CUDA 12.8, Transformers 5.3.0, FlashInfer 0.6.7.post3, and lm-evaluation-harness 0.4.9.2. Model weights remain in BF16. All methods use quantize–dequantize hooks with the same Triton attention backend and scheduler. Prefill uses exact keys and values before cache quantization, with 8,192-token chunks. CUDA graphs and the radix cache are disabled.

Hardware. Generative benchmark and RULER evaluations use one H200 per run. Throughput uses packed caches on one RTX PRO 6000 Blackwell Server Edition.

C Additional Analysis

We analyze the channel statistics and quantization errors in Figure 5. Moment statistics use the calibration data. Channel-error measurements use four 2,048-token WikiText-2 test windows and evaluate each layer separately, without BF16 sink or recent tokens and without propagating errors across layers.

C.1 Dual Transforms

For the base pair in Equation 3, the transformed second moments satisfy

$$A_0^\mathsf{T} C_Q A_0 = \Lambda, \qquad B_0^\mathsf{T} C_K B_0 = U^\mathsf{T}\big(G^{1-\alpha} - \epsilon G^{-\alpha}\big)U. \tag{7}$$

The first identity follows from the eigendecomposition defining U. The second uses $C_K = G - \epsilon I$. A key-moment eigenvalue μ_i therefore becomes $\mu_i(\mu_i + \epsilon)^{-\alpha}$, which approaches $\mu_i^{1-\alpha}$ when $\mu_i \gg \epsilon$. Partial whitening reduces the spread of key energies, while U orders the compensated query energy. These identities use uncentered moments and do not require zero-mean queries.

Figure 5(a∼h) shows the key and query profiles across models. Stronger whitening makes the key energies more uniform but also changes query scaling. We select α using the hidden-state error in Appendix B.2 to account for the combined effect of quantization and pruning.

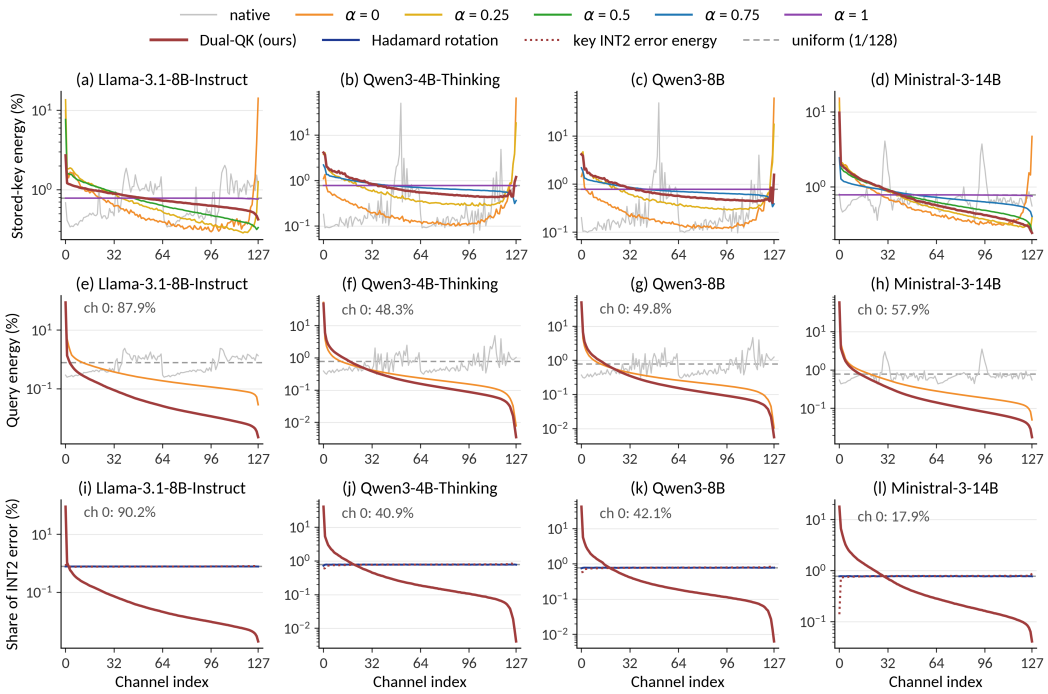

Figure 5: Channel profiles across four models. (a∼d) Base key energy at different α. (e∼h) Query energy in the original, Q-PCA, and deployed dual coordinates. (i∼l) Channel-wise shares of key reconstruction error (dotted) and query-weighted error terms (solid), before channel-0 protection, including a Hadamard control. The first two rows use calibration moments and the last uses WikiText-2 test activations. Error shares are pooled across layers and KV heads.

C.2 Channel Selection

The group-L2 rule retains the most query energy for a fixed channel budget. When the centered key covariance in the selection basis is $\sigma^2 I$, the variance of the omitted dot-product contribution for a fixed query is

$$\mathrm{Var}_k\left[\sum_{j\notin\mathcal{M}} q_j k_j\right] = \sigma^2 \sum_{j\notin\mathcal{M}} q_j^2, \tag{8}$$

where $\mathcal{M}$ is the retained channel set. Summing over the query heads in a GQA group gives the group-L2 criterion. Partial whitening of uncentered moments does not guarantee this covariance condition, so query energy serves as a proxy for pruning error in Dual-QK.

Selection uses qT_bA_0, before diagonal refinement. Scaling query channel j by δ_j and its key channel by $1/\delta_j$ leaves their product unchanged. Computing the mask in base coordinates therefore makes selection independent of the scaling used to refine quantization.

C.3 Key Quantization Error

Let ξ be key quantization error in the stored coordinates. Under the calibration model in which ξ is independent of the query, $A^\top C_Q A = D\Lambda D$ gives

$$\mathbb{E}\big[(qA\xi^\top)^2\big] = \sum_j \lambda_j \delta_j^2\, \mathbb{E}[\xi_j^2]. \tag{9}$$

Squared attention-logit error includes an additional factor of $1/d$. This expression motivates the calibration objective. It does not model error propagation during autoregressive decoding.

Figure 5(i∼l) compares reconstruction error with its query-weighted channel contributions. A channel can have modest reconstruction error but a large contribution to score error when its query energy is

high. Retaining key channel-0 in BF16 reduces this contribution. Excluding it from range estimation also narrows the INT2 range when it determines a row extremum. Diagonal refinement adjusts the remaining scales to reduce the same query-weighted error. The effect on downstream accuracy is evaluated in Table 3.

C.4 Bucket-Relative RoPE

For a key at position n in bucket b, removing the anchor rotation leaves position $n - a_b$. Applying the same adjustment to the current query preserves their relative position and uncompressed score (Equation 6). The adjustment precedes the dual transform because a general channel transform does not commute with RoPE.

With $w = 2048$, stored-key offsets remain within the calibration window. This bounds their positional offsets but does not guarantee matching activation statistics. Query offsets relative to older buckets can still exceed the window. Anchor adjustments use each checkpoint's rotary frequencies, including YaRN for Qwen3-8B beyond 32K and for Ministral. Table 3 evaluates the effect of removing bucket-relative RoPE.

D Implementation and Kernel Breakdown

D.1 Cache Writes and Attention Reads

Cache writes. Each incoming key is adjusted to its bucket anchor and transformed by B. Values receive the calibrated value rotation. The first 64 and most recent 256 tokens remain in BF16. When a token leaves the recent window, its key channels other than channel 0 and all value channels are quantized. Key channel 0 remains in BF16.

Attention reads. At each decoding step, a fused kernel forms bucket-relative queries and shared channel masks. The attention kernel reads the selected key channels and applies the original $1/\sqrt{d}$ normalization. Scores from all buckets and precision tiers are combined under one global softmax. Value aggregation uses all value channels, followed by the inverse value rotation. Cached keys are not retransformed during decoding.

Cache layout. In a token-major INT2 layout, 128 key channels occupy 32 bytes. Reading only some channels may therefore fetch the same memory sector. We store keys channel-major within 128-token pages, so one INT2 channel occupies a contiguous 32-byte row. The kernel requests only the selected rows. Values remain token-major, and metadata and BF16 entries use separate buffers.

Measured traffic. On Qwen3-8B, Nsight Compute reports 14∼15% fewer attention-kernel DRAM reads at 40% than at 0% sparsity over 30K∼90K inputs and batch sizes 1∼64. Values and metadata remain part of each read.

D.2 Decode-Time Breakdown

Table 10 reports per-step latency for Qwen3-8B at 30K context. We trace 40 decoding steps with Nsight Systems and group kernels by name. Each category's trace share is scaled by the unprofiled step latency from the same run. Overhead includes new-KV transformation, quantization and storage, query transformation, and channel selection. Other includes remaining kernels and idle time.

E Additional Results

E.1 Results without Channel Pruning

Table 11 reports the full results at 0% query-channel sparsity, extending Table 4 with KIVI-2 and Qwen3-4B-Thinking-2507. Dual-QK maintains competitive accuracy without pruning, with five-benchmark averages within one percentage point of OSCAR on Llama-3.1-8B-Instruct and Qwen3-8B. On Qwen3-4B-Thinking-2507, Dual-QK also remains close to BF16 on GPQA-Diamond and HumanEval, although larger gaps remain on LiveCodeBench and AIME25.

Table 10: Per-step decoding latency (ms) for Qwen3-8B at 30K context on one RTX PRO 6000 Blackwell Server Edition GPU. BF16 is unpruned and Dual-QK uses 40% query-channel sparsity. OOM denotes insufficient memory. Totals may differ from the sum of rounded entries.

Method	B	GEMM	Attn	Overhead	Other	Total
BF16	1	10.2	10.6	–	0.8	21.6
	8	11.0	23.2	–	0.6	34.8
	16∼64			OOM		
Dual-QK (ours)	1	10.3	1.7	0.4	1.5	13.8
	8	11.1	4.9	0.7	1.8	18.5
	16	11.4	8.9	1.0	2.0	23.3
	32	13.1	16.5	1.5	2.1	33.2
	64	14.1	31.7	2.8	2.3	50.8

Table 11: Generative accuracy (%) without query-channel pruning. Scores report mean ± standard deviation over five seeds. Capacity and bandwidth are effective bits per KV element at 128K context. Bold marks the highest mean among quantized methods within each model.

Model	Method	KV bits per element		Benchmark score (%)				
		Capacity	Bandwidth	GPQA-Diamond	HumanEval	LCB v6	MATH-500	AIME25
Llama-3.1-8B -Instruct	BF16	16	16	26.2 ± 1.5	66.7 ± 2.5	16.5 ± 1.4	49.4 ± 1.5	1.3 ± 1.8
	TurboQuant	3.13	3.13	**26.8 ± 3.1**	59.3 ± 2.9	12.5 ± 1.3	39.9 ± 1.1	0.0 ± 0.0
	KIVI-2	3.01	3.01	23.8 ± 2.8	61.7 ± 1.7	14.8 ± 1.8	43.6 ± 1.0	0.0 ± 0.0
	OSCAR	2.28	2.28	24.3 ± 1.7	65.6 ± 3.0	15.3 ± 0.9	50.8 ± 0.5	**2.0 ± 1.8**
	Dual-QK (ours)	2.34	2.34	26.4 ± 1.9	**66.0 ± 1.7**	**16.5 ± 0.7**	**50.9 ± 0.6**	1.3 ± 3.0
Qwen3-4B -Thinking-2507	BF16	16	16	65.6 ± 1.9	96.2 ± 0.7	54.0 ± 1.6	98.0 ± 0.3	74.7 ± 3.8
	TurboQuant	3.13	3.13	28.7 ± 1.9	15.9 ± 2.0	0.2 ± 0.3	36.2 ± 2.0	1.3 ± 1.8
	KIVI-2	3.01	3.01	62.5 ± 1.3	93.4 ± 1.2	**47.8 ± 1.8**	**96.8 ± 0.4**	**64.7 ± 5.1**
	OSCAR	2.28	2.28	63.5 ± 1.6	**96.1 ± 1.0**	46.7 ± 1.3	96.4 ± 0.5	**64.7 ± 3.8**
	Dual-QK (ours)	2.34	2.34	**65.8 ± 1.4**	95.9 ± 1.1	42.9 ± 1.0	95.8 ± 0.3	54.0 ± 4.9
Qwen3-8B	BF16	16	16	56.1 ± 2.2	90.1 ± 1.3	50.2 ± 2.9	96.4 ± 0.3	65.3 ± 3.0
	TurboQuant	3.13	3.13	47.1 ± 2.2	75.7 ± 1.6	24.7 ± 1.7	92.4 ± 0.6	50.7 ± 2.8
	KIVI-2	3.01	3.01	54.2 ± 1.9	91.8 ± 0.9	**46.6 ± 1.4**	**95.9 ± 0.3**	**64.0 ± 6.4**
	OSCAR	2.28	2.28	**56.6 ± 1.2**	**92.2 ± 1.1**	46.1 ± 1.6	95.8 ± 0.5	61.3 ± 1.8
	Dual-QK (ours)	2.34	2.34	55.9 ± 2.2	92.0 ± 0.9	43.2 ± 2.4	95.2 ± 0.4	61.3 ± 3.0

E.2 LongBench Results

We evaluate the full test sets of eight LongBench (Bai et al., 2024) tasks using greedy decoding and task-specific generation limits. Inputs exceeding 31.5K tokens are truncated in the middle. We apply chat templates to Qasper, QMSum, and MultiNews, and disable thinking for Qwen3-8B. For TREC, prompts end with the answer prefix and the first output line is scored following the official evaluator.

Tables 12 and 13 compare accuracy with and without query-channel pruning. At 40% sparsity, Dual-QK achieves mean scores of 37.68 and 39.41 on Llama-3.1-8B-Instruct and Qwen3-8B, exceeding OSCAR by 10.08 and 3.12 points, respectively. Increasing sparsity from 0% to 40% reduces the Dual-QK mean by only 1.54 and 1.22 points, compared with 11.32 and 4.09 for OSCAR. Without pruning, Dual-QK also achieves the highest mean among the quantized methods on both models.

E.3 Comparison with ThinK

We compare Dual-QK with KIVI-2 and OSCAR combined with ThinK (Xu et al., 2025) at 40% channel sparsity. ThinK removes key channels using a prompt-derived mask while keeping recent tokens unpruned. Dual-QK retains all key channels and selects the channels to read for each query. We report both capacity and bandwidth costs to account for this difference.

On LongBench (Table 14), Dual-QK exceeds OSCAR + ThinK by 5.56 and 6.58 mean-score points on the two models. Its means are within 0.53 points of KIVI-2 + ThinK, with lower bandwidth cost at 1.93 versus 2.41 bits per KV element. On the generative benchmarks (Table 15), Dual-QK achieves higher mean GPQA-Diamond and HumanEval accuracy than both ThinK combinations on both models.

Table 12: LongBench scores with 40% query-channel sparsity for quantized methods. BF16 is unpruned. Mean averages the eight task scores. Bold marks the highest score among quantized methods within each model.

Model	Method	Qasper	QMSum	Multi News	TREC	Trivia QA	SAMSum	LCC	Repo Bench-P	Mean
Llama-3.1-8B -Instruct	BF16	39.59	23.79	24.87	56.00	80.44	33.84	35.94	30.01	40.56
	TurboQuant	14.30	18.66	12.99	46.50	54.04	20.86	18.36	17.99	25.46
	OSCAR	18.03	18.29	15.45	49.50	57.49	24.26	18.13	19.62	27.60
	Dual-QK (ours)	**38.97**	**22.99**	**24.31**	**54.00**	**75.00**	**33.60**	**26.95**	**25.65**	**37.68**
Qwen3-8B	BF16	46.72	23.80	23.86	58.63	89.13	41.04	25.98	23.51	41.58
	TurboQuant	2.50	6.72	11.74	23.50	0.04	1.23	14.48	12.69	9.11
	OSCAR	34.15	20.28	22.76	**65.50**	81.36	36.11	15.03	15.15	36.29
	Dual-QK (ours)	**40.78**	**22.34**	**23.56**	61.00	**87.07**	**38.45**	**21.76**	**20.27**	**39.41**

Table 13: LongBench scores without query-channel pruning. Mean averages the eight task scores. Bold marks the highest score among quantized methods within each model.

Model	Method	Qasper	QMSum	Multi News	TREC	Trivia QA	SAMSum	LCC	Repo Bench-P	Mean
Llama-3.1-8B -Instruct	BF16	39.59	23.79	24.87	56.00	80.44	33.84	35.94	30.01	40.56
	TurboQuant	**39.21**	22.60	22.94	55.25	75.95	30.26	27.67	25.68	37.44
	OSCAR	36.91	**22.67**	23.99	53.50	76.53	**34.10**	**35.33**	28.31	38.92
	Dual-QK (ours)	36.06	22.65	**24.67**	**56.50**	**77.32**	33.87	34.28	**28.38**	**39.22**
Qwen3-8B	BF16	46.72	23.80	23.86	58.63	89.13	41.04	25.98	23.51	41.58
	TurboQuant	37.08	20.96	**23.99**	**66.50**	73.78	36.17	20.99	19.88	37.42
	OSCAR	**43.73**	23.07	23.53	55.43	**89.78**	40.23	24.80	**22.51**	40.38
	Dual-QK (ours)	41.34	**23.23**	23.71	61.70	88.77	**40.62**	**24.89**	20.79	**40.63**

Table 14: LongBench scores with ThinK and Dual-QK at 40% channel sparsity. BF16 is unpruned. Capacity and bandwidth are effective bits per KV element at 128K context. Mean averages the eight task scores. Bold marks the highest score among compressed methods within each model.

Model	Method	KV bits per element		LongBench score								
		Capacity	Bandwidth	Qasper	QMSum	Multi News	TREC	Trivia QA	SAMSum	LCC	Repo Bench-P	Mean
Llama-3.1-8B -Instruct	BF16	16	16	39.59	23.79	24.87	56.00	80.44	33.84	35.94	30.01	40.56
	KIVI-2 + ThinK	2.41	2.41	38.00	**23.58**	21.67	**54.54**	**78.83**	30.59	31.98	26.48	**38.21**
	OSCAR + ThinK	1.89	1.89	19.74	20.98	17.49	33.00	70.12	**34.68**	**32.64**	**28.32**	32.12
	Dual-QK (ours)	2.34	1.93	**38.97**	22.99	**24.31**	54.00	75.00	33.60	26.95	25.65	37.68
Qwen3-8B	BF16	16	16	46.72	23.80	23.86	58.63	89.13	41.04	25.98	23.51	41.58
	KIVI-2 + ThinK	2.41	2.41	**42.71**	**23.24**	23.54	54.82	**87.18**	**40.05**	24.43	**21.99**	**39.74**
	OSCAR + ThinK	1.89	1.89	26.82	20.33	16.46	30.75	84.87	37.25	**24.46**	21.72	32.83
	Dual-QK (ours)	2.34	1.93	40.78	22.34	**23.56**	**61.00**	87.07	38.45	21.76	20.27	39.41

Table 15: Generative accuracy (%) with ThinK and Dual-QK at 40% channel sparsity. BF16 is unpruned. Scores report mean ± standard deviation over three seeds. Bold marks the highest mean among compressed methods within each model.

Model	Method	Benchmark score (%)	
		GPQA-Diamond	HumanEval
Llama-3.1-8B -Instruct	BF16	26.8 ± 1.5	67.7 ± 2.4
	KIVI-2 + ThinK	22.2 ± 2.3	57.7 ± 1.8
	OSCAR + ThinK	14.5 ± 1.5	62.8 ± 0.6
	Dual-QK (ours)	$\mathbf{24.2 \pm 1.5}$	$\mathbf{65.2 \pm 3.4}$
Qwen3-8B	BF16	56.9 ± 1.9	89.2 ± 0.3
	KIVI-2 + ThinK	55.7 ± 2.0	88.2 ± 0.4
	OSCAR + ThinK	2.0 ± 0.9	35.6 ± 0.4
	Dual-QK (ours)	$\mathbf{56.7 \pm 1.8}$	$\mathbf{91.9 \pm 0.9}$