ParisKV: Fast and Drift-Robust KV-Cache Retrieval for Long-Context LLMs

Yanlin Qi[1] Xinhang Chen[2] Huiqiang Jiang[3] Qitong Wang[4] Botao Peng[5] Themis Palpanas[1]
[1]Université Paris Cité | [2]Xi'an Jiaotong University | [3]Qwen Team, Alibaba
[4]Harvard University | [5]Chinese Academy of Sciences

ICML International Conference On Machine Learning ICML 2026

Contributions

1. **Drift-robust retrieval**: normalize–rotate keys and use analytic centroids to avoid stale prefill-only clusters.
2. **GPU-native Top-k**: collision-count candidate generation followed by 4-bit calibrated inner-product reranking.
3. **CPU offload**: keep the full KV cache in CPU memory and fetch only selected KV pairs via UVA.
4. **Long-context scalability**: preserve quality while reducing decode cost at long cache lengths.

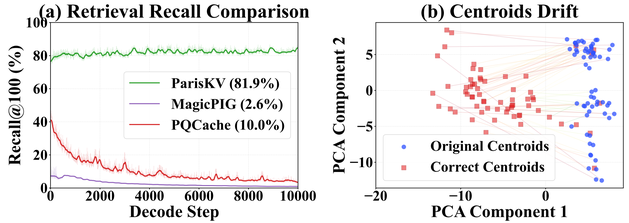
Motivation and problem setup

✓ Long-context decoding is memory-bound: every new token attends to a growing KV cache.
✓ Retrieval keeps the complete history, unlike eviction methods that permanently drop tokens.
× Existing ANN/PQ retrieval degrades under **decode-time distribution drift**.

$$\mathbf{o}_t = \mathrm{softmax}\left(\frac{\mathbf{q}_t K_t^\top}{\sqrt{d}}\right) V_t \quad \Rightarrow \quad \tilde{\mathbf{o}}_t = \mathrm{softmax}\left(\frac{\mathbf{q}_t K_\mathcal{C}^\top}{\sqrt{d}}\right) V_\mathcal{C}, \ |\mathcal{C}| \ll t.$$

Goal: candidate set $\mathcal{C}(\mathbf{q}_t)$ should cover the true attention Top-k without scanning the full cache.

Decode-time drift hurts prior retrieval



Observation.
ParisKV keeps Recall@100 high over 10K decode steps; PQCache/MagicPIG degrade because decode keys drift away from prefill-only centroids.

Long-generation quality

Table 2

Methods	GPQA_dia (pass@1)	MATH500 (pass@1)	AIME_2025 (pass@8)
Qwen-3-4B	64.14	88.60	**86.67**
PQCache	38.38	58.80	3.33
MagicPIG	32.32	46.40	6.67
ParisKV (Ours)	**72.22**	**92.80**	**80.00**
DS-R1-Llam-8B	49.49	80.40	50.00
PQCache	23.81	66.75	13.30
MagicPIG	27.78	49.00	13.30
ParisKV (Ours)	**57.07**	**84.40**	**53.30**
Qwen-3-8B	54.54	87.40	**83.33**
PQCache	47.85	69.21	16.67
MagicPIG	33.84	45.80	10.00
ParisKV (Ours)	**55.05**	**93.00**	**73.33**

Table 2. Accuracy on long-generation tasks: GPQA_diamond (pass@1), MATH500 and AIME_2025 (pass@8).

Table 3

Methods	Overall	Short		Medium		Long	
		Easy	Hard	Easy	Hard	Easy	Hard
Qwen3-4B	**25.84**	27.12	16.53	36.36	25.20	26.67	28.57
PQCache	17.91	16.95	19.00	13.60	19.00	20.00	19.05
MagicPIG	16.70	18.64	10.74	14.77	20.47	28.89	12.70
ParisKV (Ours)	24.60	35.59	19.49	26.14	22.05	28.89	23.81
Qwen-3-8B	**33.59**	50.85	34.71	32.95	25.98	37.78	28.57
PQCache	25.50	23.70	31.60	28.20	24.00	25.00	20.00
MagicPIG	10.34	8.47	15.70	7.95	7.87	13.33	7.94
ParisKV (Ours)	33.07	52.54	34.71	34.09	26.77	37.21	19.67
DS-R1-Llam-8B	13.12	18.64	15.70	12.50	8.66	11.11	14.29
PQCache	19.90	18.60	21.50	21.60	22.20	15.60	14.30
MagicPIG	13.92	15.25	11.57	11.36	14.17	17.78	17.46
ParisKV (Ours)	**28.43**	37.29	25.62	28.41	25.20	31.11	30.16

Table 3. LongBench_V2 accuracy breakdown by context length (Short/Medium/Long) and difficulty (Easy/Hard).

Sliding-window KV-cache update

Figure 5. Sliding-window KV-cache update.

ParisKV algorithm-system pipeline

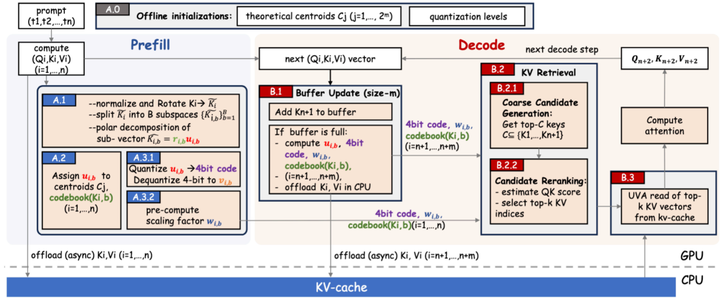


Figure 2. ParisKV pipeline. Offline, we construct an analytic centroid codebook and a quantization configuration. During prefill, we materialize the KV cache and build GPU-resident key summaries (centroid IDs for Stage-I vote-based filtering, and low-bit codes with lightweight weights for Stage-II reranking), while asynchronously offloading full-precision KV to CPU memory. During decoding, summaries for newly generated keys are incrementally updated; the GPU performs coarse-to-fine retrieval (voting → reranking) using only these summaries, then fetches the selected Top-k KV pairs from CPU via UVA for attention.

Rotation-based assignment + two-stage GPU retrieval

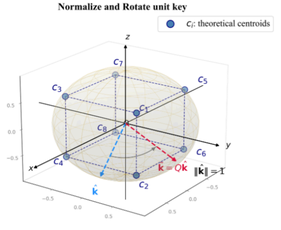


Figure 3. Illustration of rotation-based codebook assignment on the unit sphere.

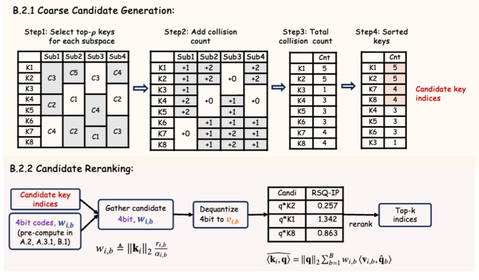


Figure 4. ParisKV Retrieval algorithm (shown as B.2 in Fig. 2)

Stable summaries and calibrated scores

- Normalize + rotate builds a stable angular space.
- Analytic centroids avoid online clustering and remain stable during long generation.
- Coarse-to-fine retrieval: cheap voting first, then calibrated 4-bit reranking.

$$\widehat{\langle \mathbf{k}_i, \mathbf{q} \rangle} = \|\mathbf{q}\|_2 \sum_{b=1}^{B} w_{i,b} \langle \mathbf{v}_{i,b}, \hat{\mathbf{q}}_b \rangle, \quad w_{i,b} = \|\mathbf{k}_i\|_2 \frac{r_{i,b}}{\alpha_{i,b}}.$$

Efficiency and scaling

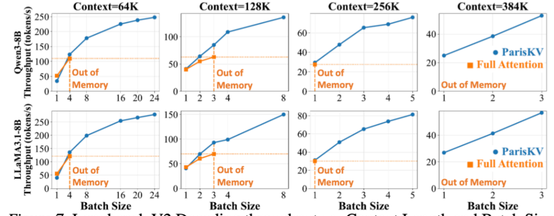


Figure 7. Longbench V2 Decoding throughput vs. Context Length and Batch Size

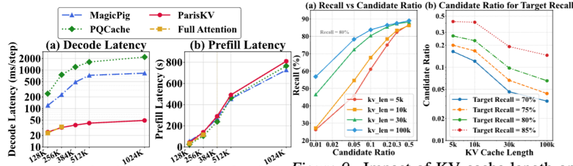


Figure 8. Prefill and Decode Time.

Figure 9. Impact of KV cache length on candidate-ratio requirements.

Key message: ParisKV increases batch-size scalability and keeps decode latency close to full attention while remaining runnable when full attention runs out of memory.

Why analytic centroids work (Prop. 4.1)

For a Haar-random rotation $\mathbf{R}$, write $\tilde{\mathbf{k}} = \mathbf{R}\hat{\mathbf{k}}$ and split it into B subspaces of dimension m ($D = Bm$). The per-subspace energy and unit-direction coordinates have fixed priors:

$$z_b = \|\tilde{\mathbf{k}}_b\|_2^2 \sim \mathrm{Beta}\left(\frac{m}{2}, \frac{D-m}{2}\right),$$
$$(\mathbf{u}_b)_j^2 \sim \mathrm{Beta}\left(\frac{1}{2}, \frac{m-1}{2}\right).$$

Implication: ParisKV places radius and level centroids analytically, without prefill-dependent clustering, so summaries do not drift during decoding.

Resources

Code
Code: github.com/amy-77/ParisKV

Paper
Paper: icml.cc/virtual/2026/poster/60751

Contact
Contact: amy-77.github.io

WeChat