ParisKV: Fast and Drift-Robust KV-Cache Retrieval

Long-context LLM inference: a research paper discussion

Paper by Yanlin Qi, Xinhang Chen, Huiqiang Jiang,
Qitong Wang, Botao Peng, and Themis Palpanas

ICML 2026 arXiv:2602.07721v3
Reading: 26 October 2026

ParisKV combines stable routing with selective KV transfer

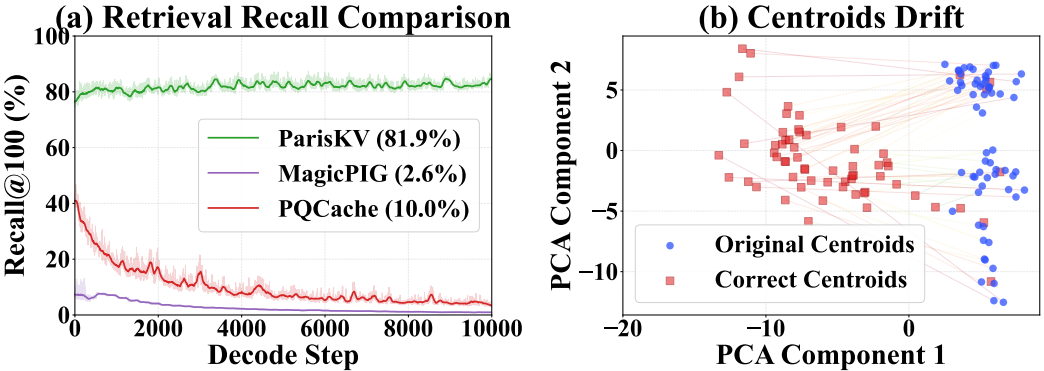
Core idea

Keep compact key summaries on GPU; use fixed spherical direction codes to shortlist keys, then estimate raw attention scores from 4-bit summaries.

- Full-precision KV resides in CPU memory; UVA fetches only final Top-100 pairs.
- On long-generation reasoning, accuracy meets or exceeds full attention in 7 of 9 model–task settings.
- At 1024K context, Llama3.1-8B decode latency is 49 ms/step versus 830 and 2179 ms/step for MagicPIG and PQCache.

Paper, Secs. 4–5, PDF pp. 3–8; Appendix D, PDF p. 20. Different methods use their recommended retrieval budgets.

Long decoding makes prefill-trained centroids stale



Recall@100 on AIME: ParisKV 81.9%; MagicPIG 2.6%; PQCache 10.0% in this example.

Paper, Fig. 1, PDF p. 1. Left: recall vs. decode step; right: prefill-only vs. prefill+decode reference centroids.

The target is high-score KV indices at each decode step

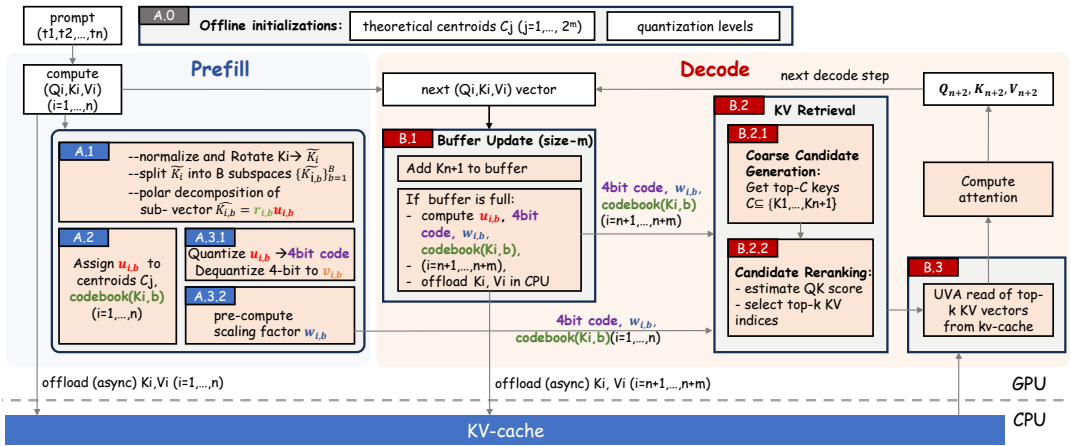
Full attention over n_t historical keys

$$s_i = \langle \mathbf{k}_i, \mathbf{q} \rangle, \qquad \mathbf{o} = \sum_{i=1}^{n_t} \mathrm{softmax}(s)_i \mathbf{v}_i.$$

- Retrieve a small candidate set $\mathcal{C}(\mathbf{q})$ that covers the true Top-k scores.
- Recompute softmax and value aggregation using the selected full-precision KV.
- The retrieval path must be cheaper than reading the entire cache every step.

Paper, Sec. 3, Eqs. (1)–(3), PDF p. 3.

The pipeline separates GPU search from CPU storage



Paper, Fig. 2, PDF p. 4. Blue: prefill summaries. Red: decode-time shortlist, rerank, and UVA fetch.

Normalization and rotation enable a fixed codebook

$$\hat{\mathbf{k}} = \frac{\mathbf{k}}{\|\mathbf{k}\|_2}, \quad \hat{\mathbf{q}} = \frac{\mathbf{q}}{\|\mathbf{q}\|_2}, \quad \tilde{\mathbf{k}} = R\hat{\mathbf{k}}, \quad \tilde{\mathbf{q}} = R\hat{\mathbf{q}}.$$

- R is a shared orthogonal transform (implemented with SRHT); it preserves normalized inner products.
- Split D dimensions into B blocks of size $m = D/B$.
- Each block uses fixed sign-pattern centers:

$$\Omega = \{\pm 1/\sqrt{m}\}^m, \quad |\Omega| = 2^m, \quad c_{i,b} = \arg\max_{\omega \in \Omega} \mathbf{u}_{i,b}^\top \omega.$$

Paper, Sec. 4.1.1–4.1.2, Eqs. (5)–(6), PDF pp. 3–5. $\mathbf{u}_{i,b}$ is the rotated block's unit direction.

The codebook is data-independent; the quantizer uses a prior

- Fixed centers cover all sign patterns, so new decode keys can be indexed without retraining centroids.
- Under Haar-random rotation, block energy and squared direction coordinates obey Beta distributions:

$$r_b^2 \sim \mathrm{Beta}\left(\frac{m}{2}, \frac{D-m}{2}\right), \qquad (u_b)_j^2 \sim \mathrm{Beta}\left(\frac{1}{2}, \frac{m-1}{2}\right).$$

- Offline Lloyd–Max quantization gives shared 3-bit magnitude levels; a fourth bit stores the sign.
- The final reranker retains exact block radii inside stored weights.

Paper, Proposition 4.1, PDF p. 6; Appendix B.1, PDF p. 14. SRHT uses the Haar prior as an approximation.

Stage I uses multi-tier collision votes to prune keys

- In each block, score a key using its assigned center and keep the top-ρ fraction.
- Divide that fraction into six percentile tiers with bonuses $6, 5, 4, 3, 2, 1$.

$$S_i = \sum_{b=1}^{B} \mathbf{1}\{i \in T_b(\rho)\}\, w_{\ell(i,b)}.$$

- Choose the top-β keys by S_i; typically $\beta = 5$–10% and $\rho \geq \beta$.
- Small integer scores allow histogram and prefix-scan selection instead of a full sort.

Paper, Sec. 4.2.2, PDF p. 5; Appendix B.2.1, Eq. (15), PDF p. 15.

Stage II estimates the raw attention score from 4-bit codes

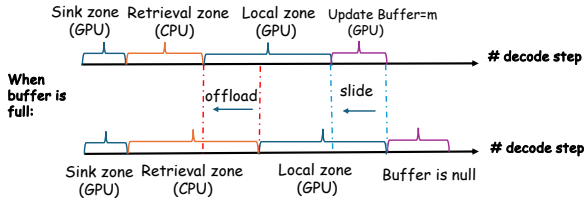
$$r_{i,b} = \|\tilde{\mathbf{k}}_{i,b}\|_2, \quad \mathbf{u}_{i,b} = \tilde{\mathbf{k}}_{i,b}/r_{i,b}, \quad \alpha_{i,b} = \langle \mathbf{v}_{i,b}, \mathbf{u}_{i,b} \rangle.$$

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

- $\mathbf{v}_{i,b}$ reconstructs the key block from one sign bit and three magnitude bits per coordinate.
- α corrects alignment loss; the division is an approximation, not an identity for every query.
- Key-only factors are precomputed, so GPU ranking needs no full-precision keys.

Paper, Eqs. (7)–(10), PDF p. 5; Appendix B.2.2, PDF pp. 15–16.

The memory layout amortizes updates and fetches Top-k



- Sink and recent Local tokens stay on GPU for dense attention.
- New tokens enter the GPU Update Buffer; a full buffer triggers encoding and offload of old Local tokens.
- GPU reranking returns Top-100 indices; UVA reads only those full KV pairs from the CPU Retrieval region.

Paper, Fig. 5, Sec. 4.2.1–4.2.3, PDF pp. 5–6. Update interval is task-dependent (Table 1).

Fewer transferred bytes are the main systems lever

Full attention	dense scores over $(\lvert S\rvert + n)$ keys, full dimension D
ParisKV	dense local $\lvert S\rvert$; compressed search over history n
	rerank $C = \lceil \beta n \rceil$; fetch only final $k \ll C$ KV

- Custom CUDA kernels: collision accumulation, integer bucket Top-k, fused rerank, UVA gather.
- Candidate work still grows with context length; compression and kernel fusion reduce the cost per key.

Paper, Sec. 4.3, PDF p. 6.

Evaluation spans long generation and long input

Workload	Metric	Models
GPQA-Diamond, MATH500	pass@1	Qwen3-4B, DS-R1-Llama-8B, Qwen3-8B
AIME25, up to 38.9K output	pass@8	same three model families
LongBench-V2	accuracy	same three model families
RULER 128K	accuracy	Llama3.1-8B

- ParisKV always retrieves Top-100; PQCache uses a 20% compression budget and MagicPIG a dynamic policy.
- Efficiency tests use Llama3.1-8B and Qwen3-8B, 64K–1024K contexts, with batch and single-request analyses.

Paper, Sec. 5, Table 1, PDF p. 6; Appendix D.1, PDF pp. 16–18. Baselines use recommended budgets.

On AIME25, ParisKV retains long-generation accuracy

AIME25 pass@8 (%)	Qwen3-4B	DS-R1-Llama-8B	Qwen3-8B
Full attention	86.67	50.00	83.33
PQCache	3.33	13.30	16.67
MagicPIG	6.67	13.30	10.00
ParisKV	**80.00**	**53.30**	**73.33**

- Gains over PQCache: +76.67, +40.00, +56.66 percentage points.
- ParisKV is below full attention on two Qwen AIME25 settings, above it on the DS setting.

Paper, Table 2, PDF p. 7. AIME25 reports pass@8; larger is better.

Seven of nine reasoning results meet or exceed dense attention

Qwen3-4B	Full	PQCache	ParisKV
GPQA-Diamond pass@1	64.14	38.38	**72.22**
MATH500 pass@1	88.60	58.80	**92.80**
AIME25 pass@8	**86.67**	3.33	80.00

- Across three models and three tasks, ParisKV meets or exceeds full attention in 7/9 cells.
- The task scores show generated-answer quality; they do not directly measure retrieval recall.

Paper, Table 2, Sec. 5.1, PDF p. 7. Scores are percentages; differences are percentage points.

LongBench-V2 gains vary by model

Overall accuracy (%)	Full	PQCache	MagicPIG	ParisKV
Qwen3-4B	**25.84**	17.91	16.70	24.60
Qwen3-8B	**33.59**	25.50	10.34	33.07
DS-R1-Llama-8B	13.12	19.90	13.92	**28.43**

- ParisKV exceeds PQCache and MagicPIG in all three model settings.
- It trails full attention by 1.24 and 0.52 points on the two Qwen models.
- The DS model's gain over full attention is observed, but its mechanism is not isolated by this table.

Paper, Table 3, PDF p. 8. LongBench-V2 with chain-of-thought; higher is better.

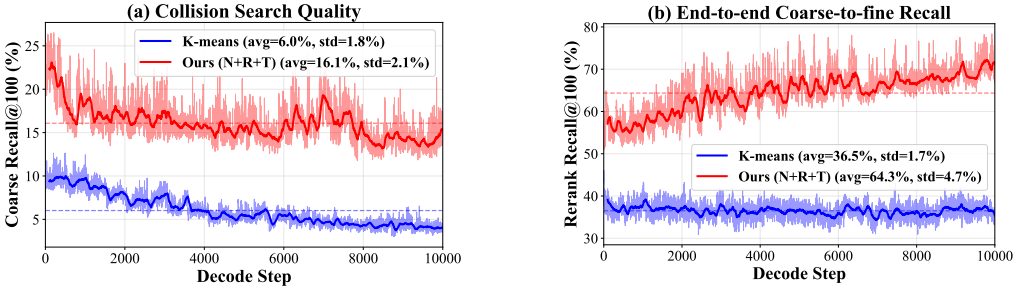
RULER shows a strong average with task-specific gaps

128K RULER, Llama3.1-8B	Avg. (%)	fwe (%)
Full attention	84.12	71.18
ShadowKV	80.29	61.11
Quest	79.34	62.85
ParisKV	**83.49**	56.25

- Best average among the listed sparse/retrieval systems; 0.63 points below full attention.
- The 14.93-point fwe gap shows that average accuracy does not imply uniform preservation.

Paper, Appendix Table 6, PDF p. 19. Higher accuracy is better.

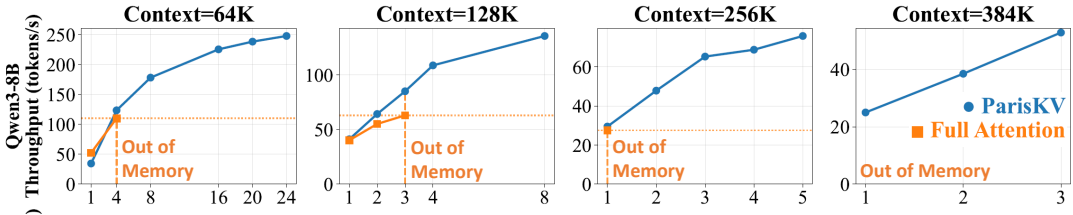
The routing design improves recall over long decoding



- Coarse Recall@100 average: K-means 6.0%, ParisKV 16.1%.
- Final two-stage Recall@100: 36.5% to 64.3% over 10K decode steps.

Paper, Sec. 5.3 and Fig. 10 panels (a) and (b), PDF pp. 7–8. The panels are cropped from the original figure.

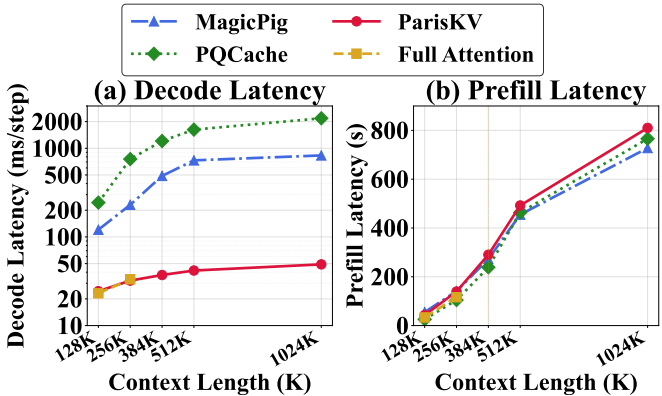
Batch throughput rises while dense attention runs out of memory



Horizontal axis: batch size; vertical axis: decode throughput (tokens/s).
Qwen3-8B peak throughput is 2.3×, 2.2×, and 2.8× dense attention at 64K, 128K, and 256K.

Paper, Fig. 7 top row, PDF p. 8; Appendix D.2, PDF p. 19. Peak values compare each method's runnable batch range.

At 1024K, on-demand KV access cuts decode latency



Llama3.1-8B, batch 1: ParisKV 49 ms/step; MagicPIG 830; PQCache 2179 (16.9× and 44.4× ratios).

Paper, Fig. 8, PDF p. 8; Sec. 5.2, PDF p. 7. Left axis is log-scale decode ms/step; right is prefill seconds.

One-time prefill cost and small-context latency matter

Context	Metric	Full attention	ParisKV
128K	Prefill / TTFT (s)	33.30	43.00
128K	Decode (ms/token)	23.13	24.42
256K	Prefill / TTFT (s)	116.20	139.50
256K	Decode (ms/token)	33.29	32.16

- These batch-1 measurements show the crossover depends on context and workload.
- Decode savings accumulate with output length; prefill cost is paid once per request.

Paper, Appendix Table 7, PDF p. 20. Separate efficiency setting from the batch throughput figure.

Interpret the gains under their exact conditions

- **Model and task:** two Qwen LongBench-V2 settings and individual RULER tasks remain below full attention.
- **Budget:** ParisKV uses fixed Top-100; baselines follow different recommended policies, including PQCache at 20%.
- **Theory:** Beta priors are exact for Haar rotation; practical SRHT uses them approximately.
- **System:** CPU offload needs host memory and adds prefill work; the paper reports a 167 GB free-memory limit in one scaling setup.

Paper, Tables 2–3, 6–7; Appendix B.1 and D.2, PDF pp. 14, 19–20.

Take-home messages

1. Fixed, data-independent routing keeps a usable index as decode keys accumulate.
2. Collision pruning plus calibrated 4-bit reranking selects KV without CPU-side search.
3. GPU-resident summaries and on-demand UVA fetch make long-context, large-batch decode practical in the reported tests.

Paper: Qi et al., *ParisKV: Fast and Drift-Robust KV-Cache Retrieval for Long-Context LLMs*, ICML 2026, arXiv:2602.07721v3.
Code: github.com/amy-77/ParisKV.

Backup: why the rotation prior is Beta

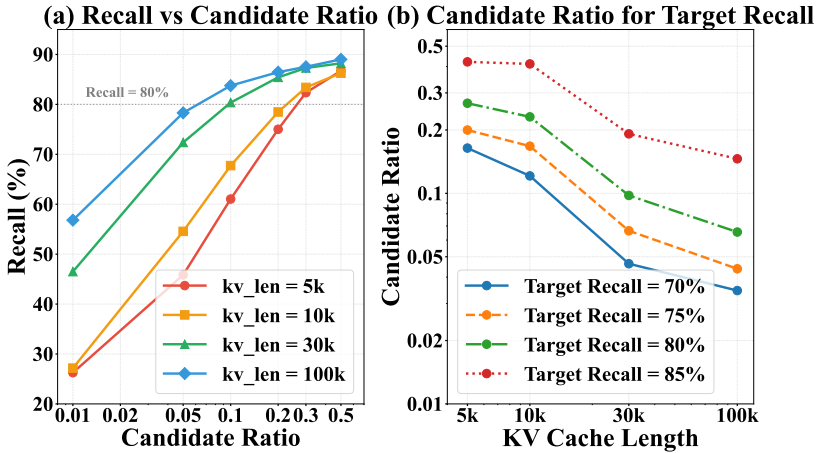
- If R is Haar-random, $R\hat{\mathbf{k}}$ is uniform on the unit sphere and has the same distribution as $\mathbf{g}/\|\mathbf{g}\|_2$ for $\mathbf{g} \sim N(0, I_D)$.
- A block energy is a ratio of independent chi-squared sums:

$$r_b^2 = \frac{\|\mathbf{g}_b\|_2^2}{\|\mathbf{g}\|_2^2} \sim \text{Beta}\left(\frac{m}{2}, \frac{D-m}{2}\right).$$

- Within a block, the squared coordinate of $\mathbf{u}_b = \mathbf{g}_b/\|\mathbf{g}_b\|_2$ is $\text{Beta}(1/2, (m-1)/2)$.
- The magnitude prior sets offline 3-bit Lloyd–Max thresholds.

Paper, Appendix B.1.1–B.1.2, PDF p. 14.

Backup: candidate ratio must track KV length



Paper, Fig. 9, PDF p. 8. Left: recall rises with candidate ratio; right: ratio needed for a given target recall falls as KV length grows.

Backup: source notation needs care during implementation

- Appendix Table 4 lists $D = 256$, $B = 16$, and $m = D/B = 8$ simultaneously; those numbers are arithmetically inconsistent.
- Appendix B's setup calls a center "learned on prefill", whereas Sec. 4.1.2 defines a fixed analytic codebook.
- Sec. 4.2.1 reuses m for Update Buffer length, while Table 1 sets Update to 256 or 512 tokens by task.

Paper, PDF pp. 5–6, 14, 17. Equations should use $D = Bm$; window sizes come from Table 1.