Training-Free Hashing-Based Attention via Binary Principal Components

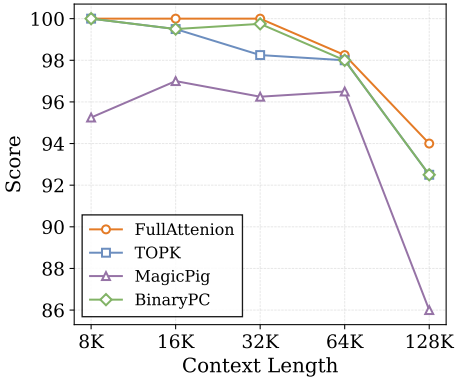
BinaryPC: data-aware retrieval with 64-bit key codes

Paper by Daohai Yu, Zhanpeng Zeng, Keyu Chen, Wenhao Li,
Zhifeng Shen, Luxi Lin, Ruizhi Qiao, Xing Sun, and Rongrong Ji

arXiv:2608.04405v1 | ICML 2026
Research group discussion | 16 October 2026

Reconstruct keys, scan compact codes, then attend to selected original KV.

The expensive step is repeatedly reading long KV histories



- Decode has few queries and many cached keys.
- Exact top-k still reads all keys to compute scores.
- A compact proxy can reduce the cost of finding useful tokens.

At 128K on NIAH multi-value:
TOPK / BinaryPC: 92.50;
MagicPIG: 86.00; full: 94.00.

Paper, Sec. 1, Fig. 1, PDF p. 1; exact subtask scores: Table 18, p. 21. TOPK budget: 2%.

Data-aware hashing need not require gradient training

Approach	Selection / encoding	Main trade-off
Static KV pruning	Prefill importance	Future evidence can be lost
Quest	Query-aware chunks	Coarse selection units
MagicPIG	Random LSH projections	Long codes / hash tables
Spotlight	Trained nonlinear hash	Model-specific training
BinaryPC	**Binary reconstruction**	**Greedy approximation**

Design question

Can key activations provide a useful retrieval representation through direct computation?

Paper, Secs. 2–3, PDF pp. 2–4; baseline configurations: Appendix A.1, p. 13.

A compact search stage selects original KV for attention

Prefill / preparation

1. Obtain projection **P** online or through offline calibration.
2. Encode keys into 64-bit codes.
3. Identify keys with large reconstruction errors.

Each decoding step

1. Project and quantize the query.
2. Score all key codes; take top-k.
3. Add protected keys, gather KV, and compute subset attention.

The full KV cache remains stored. The saving is in access and computation per step.

Paper, Sec. 3.2, Algorithms 1–3, PDF pp. 4–5; Appendix B.4, p. 17. Diagram is a teaching summary.

Reconstructing keys gives an asymmetric score approximation

$$\min_{\mathbf{H},\mathbf{P}} \|\mathbf{K} - \mathbf{H}\mathbf{P}\|_F, \qquad \mathbf{K} \in \mathbb{R}^{N\times D}, \quad \mathbf{H} \in \{-1,+1\}^{N\times H}, \quad \mathbf{P} \in \mathbb{R}^{H\times D}.$$

From key reconstruction to retrieval

$$\mathbf{k} \approx \mathbf{h}\mathbf{P} \quad \Longrightarrow \quad \mathbf{q}\mathbf{k}^\top \approx (\mathbf{q}\mathbf{P}^\top)\mathbf{h}^\top.$$

- N: cached tokens; D: key dimension; H: hash length.
- Keys use binary coefficients; query coefficients retain magnitude.
- The implementation uses $D = 128$ and $H = 64$.

Paper, Eqs. (2)–(3), PDF pp. 3–4; Appendix A.4, p. 14.

Extract one binary rank-one component from each residual

Initialize $\mathbf{R} = \mathbf{K}$. For each of H bits:

$$\mathbf{v}^* \sim \mathcal{N}(0, \mathbf{I}_D), \qquad \mathbf{u} = \operatorname{sign}(\mathbf{R}\mathbf{v}^{*\top}), \qquad \mathbf{v} = \frac{\mathbf{u}^\top \mathbf{R}}{N}, \qquad \mathbf{R} \leftarrow \mathbf{R} - \mathbf{u}\mathbf{v}.$$

Store $\mathbf{u}$ as one column of $\mathbf{H}$, and $\mathbf{v}$ as one row of $\mathbf{P}$.

Why the residual decreases

For fixed $\mathbf{u}$, the update for $\mathbf{v}$ is least squares:

$$\|\mathbf{R} - \mathbf{u}\mathbf{v}\|_F^2 = \|\mathbf{R}\|_F^2 - \frac{\|\mathbf{u}^\top \mathbf{R}\|_2^2}{N}.$$

The binary partition uses a random direction. This is a greedy procedure; it does not compute exact PCA or a global binary optimum.

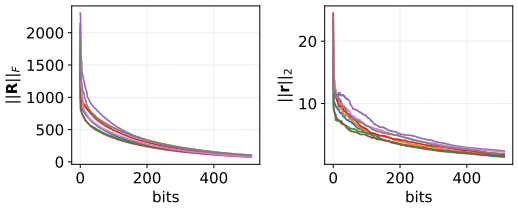
Paper, Algorithm 1, PDF p. 4. Column-vector convention for $\mathbf{u}$; residual identity is a discussion derivation.

New keys are encoded sequentially because the basis is not orthogonal

$$\mathbf{r}_0 = \mathbf{k},$$
$$h_i = \operatorname{sign}(\mathbf{r}_{i-1}\mathbf{P}_{i,:}^{\top}),$$
$$\mathbf{r}_i = \mathbf{r}_{i-1} - h_i\mathbf{P}_{i,:}.$$

Each bit describes what earlier bits have not reconstructed.

This differs from a single parallel operation $\operatorname{sign}(\mathbf{k}\mathbf{P}^{\top})$.



Left: constructing the basis.
Right: encoding with a fixed basis.
The curves show measured residuals.

Paper, Algorithm 2 and Fig. 3, PDF p. 4. Fixed-basis per-key descent is empirical, not unconditional.

Spend more bits on one query and fewer on every key

Key: 8 bytes

64 binary signs $\longrightarrow$ one 64-bit word per cached key.

Query projection: 64 bytes

64 signed quantized values $\longrightarrow$ one sign word and seven magnitude words.

Let $\mathbf{p} = \mathbf{q}\mathbf{P}^\top$. After quantization:

$$\hat{p}_i = s_i \sum_{j=0}^{6} 2^j M_{j,i}.$$

- XOR identifies sign agreement.
- AND and popcount collect contributions by bit plane.
- Weighted sums approximate inner products, then top-k selects candidates.

Paper, Algorithm 3, PDF p. 5; Appendix A.4, pp. 14–15. Zero-based bit indexing used here; details in backup.

Protect the keys whose compressed scores are least reliable

$$e_i = \|\mathbf{k}_i - \mathbf{h}_i\mathbf{P}\|_2, \qquad \mathcal{S}_{\text{attn}} = \mathcal{S}_{\text{hash}} \cup \mathcal{S}_{\text{err}}.$$

- Error-Aware Safeguard (EAS) retains large-error keys from prefill.
- Large error means a less reliable proxy, not necessarily large current attention.

Connection to score error

$$|\mathbf{q}\mathbf{k}^\top - (\mathbf{q}\mathbf{P}^\top)\mathbf{h}^\top| \le \|\mathbf{q}\|_2 \, \|\mathbf{k} - \mathbf{h}\mathbf{P}\|_2.$$

Default: reserve 10% of the attention budget for EAS.
Example: 100,000 input tokens, 2,000 total budget, about 200 EAS slots.

Paper, Eqs. (8)–(9), p. 5; budget setting, p. 6. Bound and numerical example are teaching derivations.

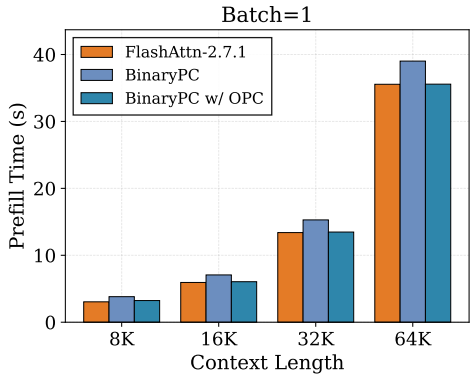
Offline projection calibration shifts work out of prefill

Online: construct **P** from the current input keys.

OPC: collect keys from 180 calibration samples:

- 60 PG19: literary text;
- 60 ProofPile: formal reasoning;
- 60 CodeParrot: code.

Apply Algorithm 1 without backpropagation. Inference still encodes incoming keys.



OPC prefill latency is close to full attention in this measurement.

Paper, Sec. 3.2, p. 4; Fig. 7 right, p. 14; Appendix B.2, p. 16. Consumer-grade GPU setup.

The evaluation changes budgets across context regimes

Evaluation	Main quantity	BinaryPC budget
Short tasks	GSM8K / MMLU accuracy; CoQA F1	5%
LongBench	Mean of 16 task scores	2% and 1K
InfiniteBench	Mean of seven task scores	2% and 2K
LongBench v2	Multiple-choice accuracy	2% and 2K
RULER, 8K–128K	13-task score at each length	2% and 2K
NIAH	Retrieval by depth and length	2%

- Fixed counts and percentages are different: at 8K, 2K is about 25%.
- MagicPIG's effective budget is data-dependent: about 5–7% on short tasks, 2–3% elsewhere.

Paper, Sec. 4.1, pp. 5–7; Appendix A.1–A.2, pp. 13–14. All task scores shown later are higher-is-better.

OPC substantially improves short-context fidelity on Llama-3

Method	Llama-3-8B	Llama-3.1-8B
Full attention	64.92	72.74
MagicPIG (default)	56.46	64.23
Spotlight (5%)	58.97	—
BinaryPC (5%)	59.31	70.81
BinaryPC + OPC (5%)	63.82	71.16

- Llama-3 online average drops by 5.61 points from full attention.
- Its GSM8K score is 54.50 full, 42.60 online, and 52.00 with OPC.

OPC matters for this setting; "training-free" does not imply no quality loss.

Paper, Table 2, PDF p. 6. Average of GSM8K, CoQA, and MMLU; mixed accuracy / F1 scores.

LongBench averages remain close to full attention at 2%

Model	Full	BinaryPC	BinaryPC + OPC
Llama-3-8B	30.63	30.18	30.35
Llama-3.1-8B-Instruct	49.64	49.50	49.50
Mistral-7B-Instruct-v0.3	47.31	47.07	47.24
Qwen2.5-7B-Instruct-1M	47.03	46.48	46.30

- Llama-3.1: a 0.14-point gap at 2%; 49.66 / 49.67 with a 1K budget.
- Task-specific gaps remain: Qwen Lcc is 46.19 full vs. 42.01 online and 39.94 with OPC.

Paper, Table 3, p. 6; Table 9, p. 15. Mean over 16 datasets, not an equal mean of six category averages.

Long-context quality depends on the model and benchmark

Model	Method	InfiniteBench	LongBench v2
Llama-3.1-8B	Full	43.30	29.4
	BinaryPC, 2%	43.21	29.8
	OPC, 2%	43.09	29.4
Qwen2.5-7B	Full	31.15	30.4
	BinaryPC, 2%	29.83	31.2
	OPC, 2%	29.31	30.4

Llama tracks the full baseline closely. Qwen loses 1.32 / 1.84 points on InfiniteBench, while maintaining LongBench v2 accuracy.

Paper, Table 1, p. 5; Table 8, p. 14. Qwen is the Instruct-1M model; InfiniteBench reports a seven-task mean.

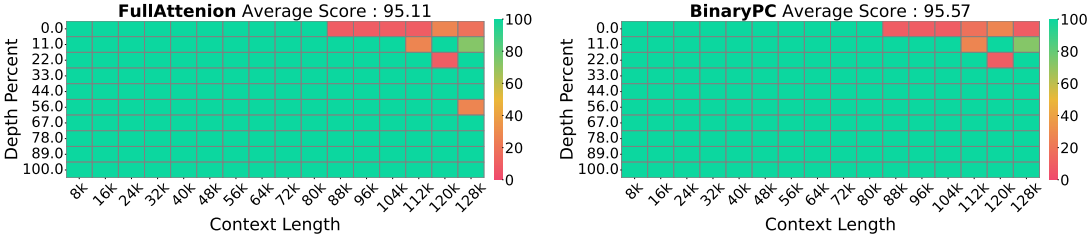
RULER separates hash approximation from sparse-budget loss

Context	Full	Exact TOPK 2%	BinaryPC 2%	BinaryPC 2K
8K	94.32	91.53	91.67	94.47
16K	93.96	91.37	90.86	93.98
32K	87.10	88.80	87.73	88.86
64K	85.19	83.39	84.36	85.09
128K	76.49	73.62	73.23	72.89
Average	87.41	85.74	85.57	87.06

At 128K, BinaryPC is 0.39 points below exact sparse TOPK and 3.26 below full attention. The 2K setting uses much more budget at short lengths.

Paper, Table 4, PDF p. 7. Llama-3.1-8B-Instruct. TOPK is an exact-score reference, not a task-score upper bound.

NIAH retrieval patterns closely follow full attention



- Horizontal axis: context length; vertical axis: needle depth; greener is better.
- Llama-3.1 average: 95.11 full, 95.57 online, 95.85 with OPC.
- Low-scoring cells remain near the beginning of longer contexts.

Paper, Fig. 4, p. 7; Fig. 10, p. 20. BinaryPC budget 2%; Qwen results in Fig. 9 reach 100.00.

EAS restores passkey retrieval for online BinaryPC

EAS share	Online R.PK	Online avg.	OPC R.PK	OPC avg.
0%	64.00	41.30	99.00	46.93
2%	99.00	47.42	99.00	47.34
5%	99.00	47.63	99.00	47.57
10% (default)	99.00	47.41	99.00	47.51
15%	99.00	47.51	99.00	47.41

- Just 2% of the attention budget recovers 35 percentage points on R.PK.
- Authors attribute failures to passkeys being hard-to-encode outliers.
- This uses 100 samples per subtask; full attention averages 47.92.

Paper, Sec. 4.3 and Table 5, PDF pp. 8–9. InfiniteBench, Llama-3.1-8B-Instruct.

64 bits and simple calibration are effective in the tested settings

Hash length: NIAH

Bits	104K	Average
32	71.82	86.37
64	92.73	90.18
128	90.91	90.00

64 bits fit in one machine word. Doubling the code gives no mean gain here.

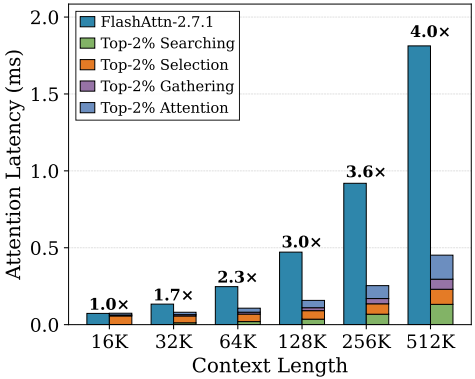
Calibration: LongBench

Setting	Average
Full attention	30.57
OPC, mixed data	30.39
OPC, PG19 only	30.43

Literary calibration also works on this mix of QA, code, and synthetic tasks.

Paper, Tables 6–7, p. 9. NIAH mean: 88K–120K, Llama-3.1-8B. Table 7 is a separate reported setting.

A fourfold layer speedup includes retrieval overhead



At 512K context:

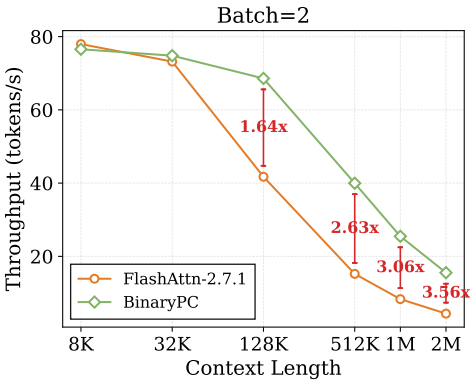
- Full attention: 1.814 ms.
- Subset attention: 0.163 ms.
- Search + top-k + gather: 0.295 ms.

$$\frac{1.814}{0.163 + 0.295} \approx 4.0 \times .$$

The complete layer cost matters, including scanning all key codes.

Paper, Sec. 4.2 and Fig. 6, PDF p. 8. Llama-3.1-8B-Instruct; 2% attention budget.

The reported 3.56-fold decoding gain occurs at 2M tokens

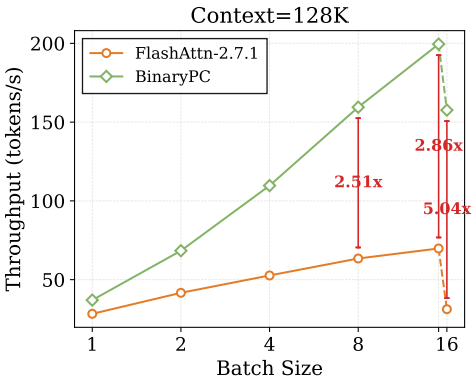


- Llama-3.1-8B-Instruct + OPC.
- Eight data-center GPUs, pipeline parallelism, static KV cache.
- Batch size 2, 2% budget.
- Warm up 32 steps; generate 64 tokens; average three runs.

Position encoding is extended and output quality is not evaluated in this throughput experiment.

Paper, Sec. 4.2, pp. 7–8; Fig. 5, second panel. Decode throughput excludes prefill and offline calibration.

The largest batch-size gain includes a kernel fallback effect



- At batch sizes above 15, the full-attention implementation switches from Split-K to a standard kernel.
- BinaryPC calls attention on a much smaller selected subset.
- At 128K and batch 16, the measured ratio reaches $5.04\times$.

This ratio combines sparse access with the implementation's kernel-selection behavior.

Paper, Sec. 4.2 and Fig. 5 rightmost, PDF p. 8. Eight data-center GPUs; FlashAttention 2.7.1.

The useful contribution is the combination of three decisions

1. **Reconstruct before retrieving.** Data-aware binary coefficients yield a useful score proxy without gradients.
2. **Allocate precision asymmetrically.** One 8-byte code per key; a more precise query is reused across the scan.
3. **Protect hard-to-encode evidence.** EAS repairs a substantial online passkey failure.

Practical scope

Keep the full KV cache, adding about 1.56% for key codes at $D = 128$ BF16. Long contexts offer the strongest speed benefit; task quality and short-context latency still require attention.

Paper, Secs. 3–4; memory calculation: Appendix B.4, p. 17.

Reference and discussion

Daohai Yu et al.
Training-Free Hashing-Based Attention via Binary Principal Components.
ICML 2026; arXiv:2608.04405v1, submitted 5 August 2026.

arxiv.org/abs/2608.04405v1
Author code: github.com/yudaohai666/BPC

Questions suggested by the method

Can gradient-free construction target ranking error more directly?
How should a static basis and protected set evolve during long generation?

All reported results in this presentation come from the version above. Additional derivations are labeled.

Backup: construction and fixed-basis encoding have different guarantees

Construction, fixed binary u:

$$\mathbf{v}_{\mathrm{LS}} = \frac{\mathbf{u}^\top \mathbf{R}}{N}, \qquad \|\mathbf{R} - \mathbf{u}\mathbf{v}_{\mathrm{LS}}\|_F^2 = \|\mathbf{R}\|_F^2 - \frac{\|\mathbf{u}^\top \mathbf{R}\|_2^2}{N}.$$

The aggregate reconstruction residual cannot increase in this update.

New key, fixed component v:

$$\|\mathbf{r} - \mathrm{sign}(\mathbf{r}\mathbf{v}^\top)\mathbf{v}\|_2^2 - \|\mathbf{r}\|_2^2 = \|\mathbf{v}\|_2^2 - 2|\mathbf{r}\mathbf{v}^\top|.$$

This quantity can be positive. Residual curves for fixed-basis encoding are empirical evidence on the tested data.

Discussion derivations from Algorithms 1–2, PDF p. 4; empirical curves: Fig. 3.

Backup: make the sign convention and bit weights explicit

Encode $+1$ as 0 and -1 as 1 on both sides. Let W be the XOR of the sign words; $W = 1$ means disagreement.

$$\hat{p}_i = s_i \sum_{j=0}^{6} 2^j M_{j,i}.$$

$$\sum_i h_i \hat{p}_i = \sum_{j=0}^{6} 2^j \left[\text{popcount}(M_j \wedge \neg W) - \text{popcount}(M_j \wedge W)\right].$$

- Paper Eqs. (10)–(12) use indices $1, \ldots, 7$ with weights 2^j: one shift from standard seven-bit magnitudes.
- A common positive scale preserves top-k; reversing the sign does not. Eq. (13) requires a clear bit convention.

Explicit implementation convention derived for discussion. Paper, Algorithm 3, p. 5; Eqs. (10)–(13), p. 15.

Backup: retrieval success and aggregation success differ

Context	RULER task	Full	TOPK 2%	BinaryPC 2%
8K	Common-word extraction	96.30	75.40	75.30
128K	Multi-key 3	70.00	59.00	45.00
128K	Multi-value	94.00	92.50	92.50

First-decode-step output cosine similarity to full attention:

Method	8K	16K	32K	64K
BinaryPC (2.0%)	.9835	.9811	.9870	.9895
MagicPIG (2.2%)	.9388	.9651	.9595	.9725

Paper, Tables 14 and 18, pp. 19 and 21; Table 10, p. 15. Quest 128K differs between Tables 4 and 18: 65.39 vs. 63.59.

Backup: BinaryPC is competitive, but not always the fastest sparse method

Single data-center GPU, 64K, batch 4:

Method	FlashAttention	MagicPIG	BinaryPC
Throughput (tokens/s)	87.62	42.05	148.46

Eight consumer-grade GPUs, 128K, sparse budget 2,048:

Method	Full	PyramidKV	CompressKV	BinaryPC
Throughput (tokens/s)	21.31	28.14	29.01	27.66

The second comparison follows official baseline implementations; FlashAttention uses StaticCache. BinaryPC retains the full KV cache.

Paper, Tables 12–13 and Appendix B.5–B.6, PDF p. 18. Throughput comparisons are within each hardware setup.

Backup: 70B results support scale transfer with a length-dependent crossover

Llama-3-70B-Instruct, LongBench category scores:

Method	S-Doc	M-Doc	Sum.	F-shot	Syn.	Code
Full	42.18	47.51	27.31	70.83	38.50	54.24
BinaryPC	42.06	47.78	26.98	70.96	38.25	55.22

Single attention layer, 70B (ms; lower is better):

Method	32K	64K	128K	256K
Full	.181	.343	.654	1.289
BinaryPC, 2,048 tokens	.243	.243	.243	.245

The fixed-budget sparse layer is slower at 32K and faster at longer lengths.

Paper, Table 11 and Appendix B.4, PDF p. 17. The accuracy block does not specify the same 2,048-token budget.