PQ-HSA: REUSING PRODUCT-QUANTIZED SCORES FOR HYBRID SPARSE-APPROXIMATE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

At each decoding step a language model attends over the key–value (KV) cache of every earlier token, so at long context the attention call is bounded by memory bandwidth. Sparse attention reads only a subset of keys chosen by a cheap score estimate, and most methods give the unread tokens zero weight. The output then draws on only a small fraction of the KV cache, and accuracy drops at small budgets, most on tasks that aggregate information across the context. An inverted-file product-quantization (IVF-PQ) index over the cached keys computes an approximate score for every indexed token in order to rank them; after ranking, those scores approximate the attention logits of the tokens left out. PQ-HSA (hybrid sparse-approximate attention) attends the selected tokens with their original keys and values, and the unselected tokens, the background, enter the same softmax through those scores, summed per inverted list and multiplied by the list's mean value. At 128K and a 1–2% retrieval budget, PQ-HSA is more accurate than Quest and SnapKV on Llama-3.1-8B and Qwen3-30B-A3B and stays close to full attention in macro accuracy; with the same selector, the background term raises macro accuracy on the 8B model from 0.71 to 0.83. In the same 128K setting, inside vLLM on one NVIDIA H20, the decode attention call runs $1.6\times$ faster than the FlashAttention-3 kernel; the speedup grows with context length, and a cost model fitted on 8B to 30B models gives the context length at which it begins. A vLLM plugin runs PQ-HSA on two engine versions without changes to the engine source; code is available at https://anonymous.4open.science/r/pqhsa_release-71BF.

1 INTRODUCTION

A decoder-only language model generates one token at a time, and at each step the new query attends over the keys and values of every earlier token, stored in the key–value (KV) cache (Dao et al., 2022). The cache grows linearly with the context, so at long context the attention call is bounded by the memory bandwidth needed to read it. Yet the softmax of a single query is often concentrated on a small fraction of the tokens (Zhang et al., 2023; Tang et al., 2024). Sparse attention builds on this: it attends only to a subset of the cached tokens, chosen by an inexpensive estimate of their attention scores, and so reads a small part of the cache at each step.

Quest bounds the logits of each page of keys and keeps the highest-bound pages (Tang et al., 2024); SnapKV keeps the tokens that the end of the prompt attended to most during prefill (Li et al., 2024); PQCache, RetrievalAttention and ParisKV build an approximate nearest-neighbor (ANN) index over the keys and retrieve the top-k per query (Zhang et al., 2024a; Liu et al., 2024; Qi et al., 2026); MagicPIG samples tokens with locality-sensitive hashing (Chen et al., 2025). In the selection-based methods the softmax is normalized over the selected tokens only and the unselected tokens receive zero weight, which we refer to as truncation. When attention is spread over many tokens, as in tasks that aggregate evidence across the context, the omitted mass can carry part of the answer; among the methods above, MagicPIG accounts for it through the importance weights of its sample.

An IVF-PQ index (Jégou et al., 2011; Johnson et al., 2021) groups the keys into n_{list} clusters by k-means, the inverted lists (the inverted file, IVF), and compresses each key's offset from its cluster centroid into M short codes (product quantization, PQ); a query's inner product with any indexed key is then approximated by M table lookups.

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.

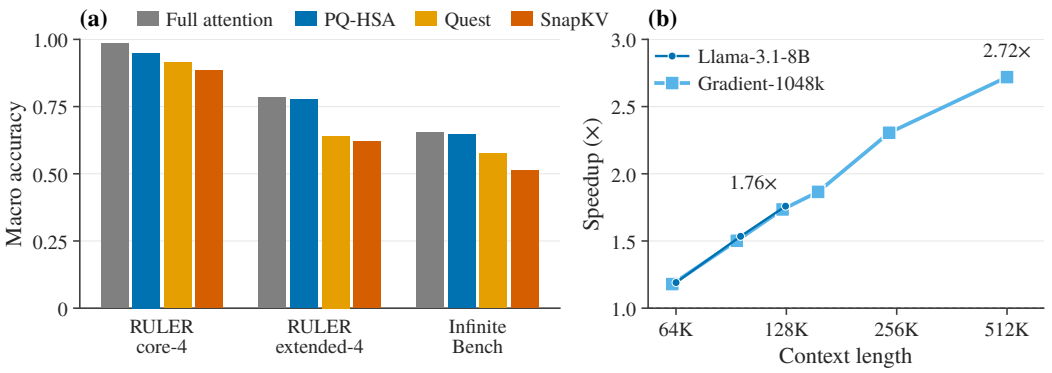


Figure 1: (a) Macro accuracy of Llama-3.1-8B-Instruct at a 1% retrieval budget on the RULER core-4 and extended-4 task groups (Section 4.1) and InfiniteBench at 128K. (b) Speedup of the decode attention call (summed over layers) over the FlashAttention-3 decode kernel (Shah et al., 2024) on one H20. Gradient-1048k, a Llama-3-8B checkpoint with a 1M-token window, has the architecture of Llama-3.1-8B (8 KV heads, $G = 4$) and extends the sweep beyond Llama-3.1's 128K window.

PQ-HSA (hybrid sparse-approximate attention) scans the codes of every indexed key at each decode step instead of probing a few lists, so one scan gives an approximate attention logit for every token, and PQ-HSA uses these logits twice. The top-scoring tokens are selected and attended with their original keys and values. Every other token, the background, keeps its approximate logit in the same softmax; these weights are summed per inverted list and multiplied by the list's mean value. The softmax denominator and the output thus account for the whole context, while only the selected KV rows are read in full (Figures 2 and 3). No second scoring pass is needed; the background adds one sum per list and a read of n_{list} value means (Section 3.2).

The construction is not specific to IVF-PQ: any index that scores every cached key cheaply yields both a selection and an estimate for the keys it leaves out. Because IVF-PQ is a classic vector-search index, the many optimizations developed for it can be brought into attention directly (Ge et al., 2014; Guo et al., 2020; Douze et al., 2026).

Three observations organize the evaluation. First, at the same budget PQ-HSA is more accurate than Quest, SnapKV and ANN retrieval baselines on Llama-3.1-8B and Qwen3-30B-A3B at 128K, and stays close to full attention (Section 4.2). Second, with the same selector and budget, the background term raises macro (mean per-task) accuracy by 0.12 on Llama-3.1-8B and also raises it on Qwen3-30B; the background needs both the PQ logits and the list value means (Section 4.3). Third, in the same 128K setting inside vLLM the attention call is 1.6× faster than the dense FlashAttention-3 kernel (Shah et al., 2024), rising to 2.7× at 512K on a 1M-window checkpoint. A cost model fitted on 8B to 30B models gives each model's break-even length from its number of KV heads and its group size under grouped-query attention (GQA), where a group of query heads shares one KV head (Ainslie et al., 2023) (Section 4.4).

PQ-HSA builds on published components: IVF-PQ search (Jégou et al., 2011; Johnson et al., 2021) with the key norm kept outside the quantizer (Dai et al., 2019), attention sinks (the first tokens) with a local window (Xiao et al., 2024b), ANN retrieval of KV rows at decode time (Zhang et al., 2024a; Liu et al., 2024; Qi et al., 2026), batched k-means (Yang et al., 2026a) and vLLM's paged KV storage in fixed-size blocks (Kwon et al., 2023). Our contributions are: (1) a construction of hybrid sparse–approximate attention in which the scores a vector-search index computes for ranking also serve as the logits of the unselected tokens; with IVF-PQ, they are aggregated per list with list value means under one normalizer (Section 3); (2) a controlled mechanism study that isolates the background term and the roles of its logits and list value means under the same selector and budget, on two model families (Section 4.3); (3) an empirical cost model for decode sparse attention across model sizes and GQA group sizes, with fitted break-even lengths (Section 4.4); (4) a vLLM plugin that leaves the engine source unchanged, reads the paged KV cache in place, and runs on vLLM 0.8.5 and 0.29 (Section 4.5); the operator, kernels and plugin are released as open source.

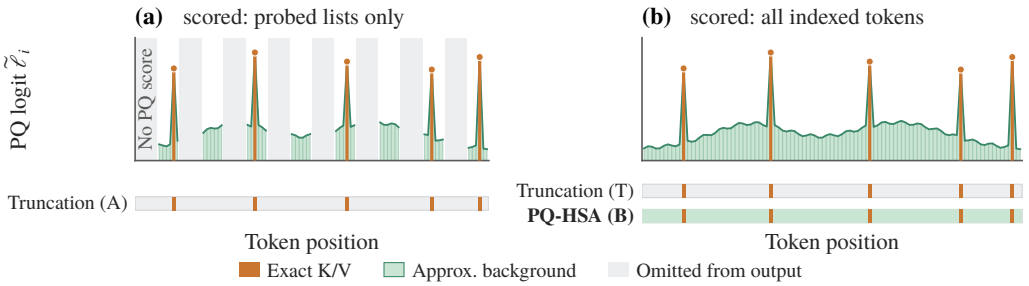


Figure 2: Token coverage (schematic). (a) Only the probed IVF lists are scored, as in standard ANN retrieval, and truncation (variant A) attends the selected tokens alone. (b) Every indexed token is scored; variant T attends only the top-k tokens, and PQ-HSA (variant B) adds the other scored tokens as an approximate background. The strips show contribution to the output, not exact KV reads.

2 Related Work

Decode-time methods that reduce the KV read can be grouped by what happens to the tokens they do not read.

Evicting. StreamingLLM keeps the first tokens, which act as attention sinks, and a recent window (Xiao et al., 2024b); H2O and TOVA drop tokens by accumulated or recent attention (Zhang et al., 2023; Oren et al., 2024); SnapKV selects once at the end of prefill from the attention of an observation window (Li et al., 2024). A dropped token is unavailable to later queries.

Selecting. Quest scores each page of keys by an upper bound on its logits and attends to the top pages (Tang et al., 2024); PQCache, RetrievalAttention and ParisKV index the keys with ANN structures and retrieve the top-k per query, ParisKV with a low-bit second-stage rerank (Zhang et al., 2024a; Liu et al., 2024; Qi et al., 2026). Loki, Double Sparsity and HashAttention rank tokens from low-rank keys, offline-calibrated channels and learned binary signatures (Singhania et al., 2024; Yang et al., 2024; Desai et al., 2025); Squeezed Attention and ClusterKV cluster the keys and keep whole clusters by centroid score (Hooper et al., 2025a; Liu et al., 2025); InfLLM looks up blocks of past tokens through representative keys (Xiao et al., 2024a); InfiniGen, ShadowKV and ClusterKV keep the KV cache, or its values, in host memory and fetch the selected rows (Lee et al., 2024; Sun et al., 2025; Liu et al., 2025). Speculative token selection runs a draft model at every step to nominate the tokens (Yu et al., 2026). Each of these normalizes the softmax over the selected tokens, so the unselected tokens receive zero weight.

Approximating the remainder. Splitting attention into an exact sparse part and an approximation of the rest goes back to Scatterbrain, which adds a low-rank estimate, and to clustered attention, which groups the queries; both approximate full-sequence attention rather than decoding over a KV cache (Chen et al., 2021; Vyas et al., 2020). At decode time, SparQ scores every token from a few query-selected key components and gives the mass outside its top-k to one global mean value (Ribar et al., 2024). RetroInfer and Multipole Attention cluster the keys, attend the highest-scoring clusters exactly and represent the other clusters by a centroid logit and an aggregated value, so both selection and approximation work at cluster granularity; RetroInfer also keeps the KV cache in host memory and drops the lowest-ranked clusters (Chen et al., 2026; Hooper et al., 2025b). MagicPIG samples tokens with locality-sensitive hashing and forms an importance-weighted estimate, computed on the CPU (Chen et al., 2025).

Quantizing. Product quantization encodes vectors as short codes in subspaces (Jégou et al., 2011; Johnson et al., 2021), norm-explicit quantization keeps the norm outside the quantizer (Dai et al., 2019), and Transformer-VQ, CommVQ and self-indexing caches store quantized keys (Lingle, 2024; Li et al., 2025; Yang et al., 2026b).

PQ-HSA is a selecting method whose unselected tokens keep the query-dependent PQ scores the selector already computed and contribute through their list's value mean, under the same normalizer

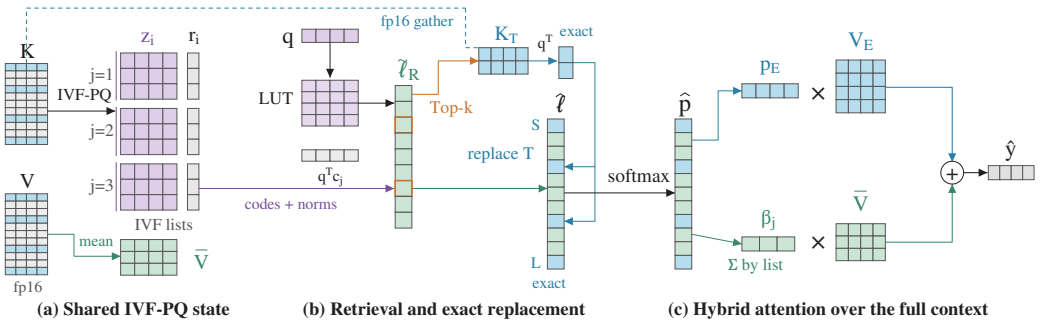


Figure 3: PQ-HSA shares one IVF-PQ score stream between retrieval and background attention. (a) Residual PQ codes, key norms and list value means are kept alongside the resident original KV cache. (b) The scan scores every indexed token in R from a lookup table (LUT) built from the query; the top-k tokens T are gathered and their logits replaced by exact ones. (c) The exact set $E = S \cup L \cup T$ and the background $B = R \setminus T$ share one softmax: selected weights multiply original values, and background weights are summed per list and multiply list means. Tensor sizes are schematic.

as the selected set. Unlike SparQ it uses one value mean per list rather than one global mean, and unlike the clustered methods it selects individual tokens from a scan of every key, with the KV cache kept on the GPU. It shares the IVF-PQ index with PQCache, which normalizes over the retrieved tokens alone; Appendix D compares these designs side by side.

3 The Proposed Product-Quantization based Hybrid Sparse-Approximate Attention (PQ-HSA)

For a decode query $q \in \mathbb{R}^d$ and cached keys and values $\{(k_i, v_i)\}_{i=1}^n$, dense attention computes $y = \sum_i e_i v_i / \sum_i e_i$ with $e_i = \exp(q^\top k_i / \sqrt{d})$. For any split of the positions into an exact set E and its complement B, the standard decomposition

$$y = \frac{N_E + N_B}{Z_E + Z_B}, \qquad N_X = \sum_{i \in X} e_i v_i, \quad Z_X = \sum_{i \in X} e_i \tag{1}$$

holds. A sparse method chooses E and decides what to do with N_B and Z_B: truncation sets both to zero; PQ-HSA estimates both from the approximate scores that chose E.

3.1 Direction–norm residual IVF-PQ over keys

The index is standard IVF-PQ (Jégou et al., 2011; Johnson et al., 2021) applied to key directions. Each key is written $k_i = r_i u_i$ with norm r_i and unit direction u_i. The directions are clustered into n_{list} inverted lists with centroids c_j, and each residual $u_i - c_{a_i}$ is product-quantized in M subspaces with 2^b codewords $p_{m,z}$ each, so that a key is stored as its list id a_i, its codes $z_{i1}, \ldots, z_{iM}$ and its norm. For a query $q = r_q u_q$ the index returns, for every indexed position,

$$\widetilde{\ell}_i(q) = \frac{r_q r_i}{\sqrt{d}} \left(u_q^\top c_{a_i} + \sum_{m=1}^{M} \left(u_q^{(m)}\right)^\top p_{m, z_{im}} \right), \tag{2}$$

computed as one table lookup per subspace once an $M \times 2^b$ lookup table (LUT) of the inner products of each subvector $u_q^{(m)}$ of u_q with each codeword has been formed. Keeping the norm outside the quantizer, as in norm-explicit quantization (Dai et al., 2019), puts $\widetilde{\ell}_i$ in the units of $q^\top k_i / \sqrt{d}$. The coarse term $u_q^\top c_{a_i}$ stays in the sum, so all members of a list share its offset. We use $n_{\text{list}} = 512$, $M = 8$ and $b = 4$, so the index holds four bytes of codes per token per KV head plus one norm and one list id. We scan every list at decode time, so every indexed token receives a score. The inverted lists thus serve both as the coarse quantizer of the residual codes and as the groups in which the background is aggregated. Probing a subset of lists, the usual ANN setting, appears only in comparison variants (Figure 2); even without a background, scanning every list selects more accurately (Section 4.3).

3.2 ONE SCORE STREAM, TWO USES

Positions are partitioned into a sink prefix S, the first tokens, kept exact as in StreamingLLM (Xiao et al., 2024b); a local window L of the most recent, not-yet-indexed tokens; and the indexed retrieval region R. The retrieval fraction ρ is the share of R attended exactly; with $k = \lceil \rho |R| \rceil$,

$$T(q) = \mathrm{TopK}_{k,\, i \in R} \widetilde{\ell}_i(q), \qquad E = S \cup L \cup T(q), \qquad B = R \setminus T(q). \tag{3}$$

For $i \in E$ the original k_i, v_i are gathered and the exact logit $\ell_i = q^\top k_i / \sqrt{d}$ replaces the approximate one; no second selection is made. For $i \in B$ the approximate logit is kept. With a common shift m_0 (the maximum over exact and approximate logits), $e_i = \exp(\ell_i - m_0)$ on E and $\widetilde{e}_i = \exp(\widetilde{\ell}_i - m_0)$ on B. The background is aggregated per inverted list $R_j = \{i \in R : a_i = j\}$:

$$M_j = \sum_{i \in B \cap R_j} \widetilde{e}_i, \qquad \bar{v}_j = \frac{1}{|R_j|} \sum_{i \in R_j} v_i, \qquad \widehat{Z}_B = \sum_j M_j, \qquad \widehat{N}_B = \sum_j M_j \, \bar{v}_j, \tag{4}$$

where an empty list contributes zero. The output is

$$\widehat{y} = \frac{N_E + \widehat{N}_B}{Z_E + \widehat{Z}_B}. \tag{5}$$

At $\rho = 1$, B is empty and Equation 5 is dense attention. List means are maintained over all members of R_j at index updates, so the value side reads n_{list} list means and none of the $|B|$ unselected rows. Weighting each mean by its members' approximate mass would make it query-dependent and require reading every unselected value at each step. The list means also count the selected members, about a fraction ρ of each list; excluding them changes macro accuracy by $+0.001$ $[-0.010, +0.010]$ (Appendix B).

3.3 EXECUTION PATH

After a dense prefill, the index of each KV head is built from its prefilled keys by k-means, once per prompt. Training the k-means problems of all heads of a layer together (Yang et al., 2026a) builds the index of Llama-3.1-8B at 128K in 0.84 s on one H20, against 6.5 s when each head is built separately (Appendix C). New tokens stay exact in L and are encoded into the existing lists every 256 steps, without re-clustering; each update also refreshes the list means. Each decode step runs, per layer, the lookup-table construction, the code scan with a fused per-list reduction, a top-k over the score rows, a gather of the selected rows, and the combination of Equation 5 (Figure 3), replayed as one CUDA graph (NVIDIA Corporation, 2025). Under GQA every query head of a group shares the codes of its KV head and forms its own lookup table, so one pass over the codes yields one score row per query head. Inside vLLM (Kwon et al., 2023), whose PagedAttention stores the KV cache in fixed-size pages, the gather reads the engine's pages in place.

4 EXPERIMENTS

4.1 SETUP

Models and hardware. Llama-3.1-8B-Instruct is the primary model (Grattafiori et al., 2024). Qwen2.5-14B-1M and Qwen3-30B-A3B extend the speed study to larger models (Table 7), and Qwen3-30B-A3B also serves as the second model for quality (Table 1(c)). Gradient-1048k, a Llama-3-8B checkpoint extended to a 1M-token window, supplies the 256K–512K points beyond Llama-3.1's 128K window (Yang et al., 2025b;a; Gradient AI, 2024). Speed is measured on one NVIDIA H20 GPU (96 GB) inside the vLLM serving engine (Kwon et al., 2023), and quality in a HuggingFace implementation of the same operator (Appendix A).

Benchmarks. RULER (Hsieh et al., 2024) generates synthetic long-context tasks at a chosen length; at 128K we use four core tasks, core-4 (single- and multi-key needle retrieval, question answering, variable tracking), and four extended tasks, extended-4 or ext-4 (multi-query and multi-value needles, common- and frequent-word extraction). InfiniteBench (Zhang et al., 2024b) contributes six tasks at 128K, including question answering and multiple choice over full-length novels;

Table 1: Quality at 128K, higher is better. (a) Llama-3.1-8B-Instruct, macro accuracy at a 1% budget; bold marks the best budgeted method. (b, c) All methods on the same RULER prompts at a 2% budget, on Llama-3.1-8B (b) and Qwen3-30B-A3B (c); Δ = PQ-HSA $-$ method with its paired 95% interval, bold where the interval excludes zero. † full attention; reprod. our reimplementation.

(a) 1% budget Method	RULER core-4	RULER ext-4	InfiniteBench
full attention†	0.985	0.784	0.655
PQ-HSA	**0.950**	**0.777**	**0.646**
Quest	0.915	0.641	0.575
SnapKV	0.885	0.621	0.512
sink + window	0.050	0.207	0.105

(b) Llama-3.1-8B, 2% budget, RULER ext-4 Method	macro	Δ [95% CI]
PQ-HSA	0.795	—
full attention†	0.779	$+0.016$ [-0.011, $+0.044$]
ParisKV (reprod.)	0.738	$+\mathbf{0.056}$ [$+0.022$, $+0.092$]
SnapKV	0.646	$+\mathbf{0.148}$ [$+0.097$, $+0.200$]
(c) Qwen3-30B-A3B, 2% budget, ext-4 + single needle		
PQ-HSA	0.905	—
full attention†	0.914	-0.009 [-0.021, $+0.003$]
SnapKV	0.809	$+\mathbf{0.096}$ [$+0.071$, $+0.120$]
Quest	0.753	$+\mathbf{0.151}$ [$+0.117$, $+0.189$]

LongBench (Bai et al., 2024) contains sixteen English tasks on real documents, run at native lengths for the comparison with PQCache and MagicPIG. AIME-24 and MATH-500 test long generation on DeepSeek-R1-Distill-Llama-8B. Passkey retrieval, which asks for a random number hidden in the context (Kamradt, 2023), is the correctness check attached to speed measurements. Scores are macro accuracy, the mean of per-task accuracies (Appendix A).

Budgets and baselines. The retrieval fraction ρ is 1% unless stated; Table 1(b, c) uses 2%, and the mechanism study reports both. Every budgeted baseline attends exactly as many tokens as PQ-HSA (about 1.5K at 1% and 2.8K at 2% for a 128K input). Baselines are full attention (dense attention, timed with the FlashAttention-3 kernel), truncation (top-k from a subset of probed IVF lists, unselected tokens at zero weight), SnapKV, sink-and-window, Quest, RetrievalAttention, PQCache, MagicPIG, ParisKV and SparQ; ParisKV, RetrievalAttention and SparQ are our reimplementations of the published procedures in the same evaluation pipeline. Appendix A lists the baseline settings.

Statistics and timing. All methods are evaluated on the same prompts, and paired differences come with 95% bootstrap intervals from 10,000 resamples of the prompts within each task. Speed is reported in two ways: the attention segment, the GPU time of the decode attention call summed over layers, and the time per output token (TPOT) of the whole decode step. Table 5 gives the setting of every speed number.

4.2 Quality against full attention and budgeted baselines

Table 1(a) reports macro accuracy at a fixed 1% budget, on RULER's synthetic tasks and on InfiniteBench's natural and synthetic ones. PQ-HSA is the most accurate budgeted method on all three Llama-3.1-8B benchmark groups and is close to full attention on RULER extended-4 and InfiniteBench. On paired extended-4 prompts it leads Quest by 0.136 [0.105, 0.165] and SnapKV by 0.156 [0.121, 0.190], and differs from full attention by -0.007 [-0.031, $+0.017$]. On RULER core-4 it scores 0.950 (0.970 at 2%), above our RetrievalAttention reimplementation (0.925; Appendix D) and IVF-probed truncation (0.645). In long chain-of-thought generation (AIME-24 and MATH-500 with DeepSeek-R1-Distill-Llama-8B), the prompt is short and the context is produced during decoding, so the index is extended many times per problem; there PQ-HSA scores 0.562 and full attention 0.554.

Table 2: Mechanism variants at 128K, in the notation of Section 3.2; $\bar{v}_{\text{global}}$ is the token-weighted global value mean, a logit of $-\infty$ drops the background (truncation), and — marks no background term. †128 probed lists (Appendix B).

Variant	Selector	Background logits	Background values
A	top-k, probed lists†	$-\infty$	—
T	top-k, all lists	$-\infty$	—
B (PQ-HSA)	top-k, all lists	$\widetilde{\ell}_i$	$\bar{v}_j$
C	top-k, all lists	$\widetilde{\ell}_i$	$\bar{v}_{\text{global}}$
C2	top-k, all lists	$\text{mean}_{i \in T}\, \widetilde{\ell}_i$	$\bar{v}_{\text{global}}$
C2b	top-k, all lists	$\min_{i \in T}\, \ell_i$	$\bar{v}_{\text{global}}$
Dense	all tokens	—	—

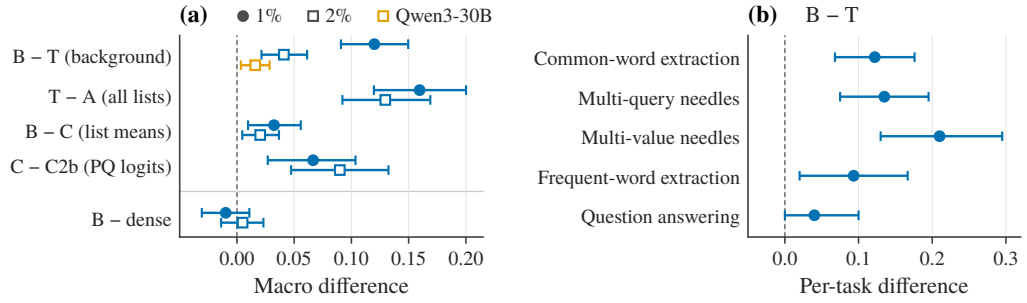


Figure 4: Paired differences between the variants of Table 2 at 128K, 95% bootstrap intervals, zero marked. (a) Macro differences for five contrasts on Llama-3.1-8B at 1% and 2% retrieval; the B minus T row adds Qwen3-30B at 2%. (b) Per-task B minus T on Llama-3.1-8B at 1%. Per-task scores are in Appendix B.

Table 1(b, c) evaluates every method on the same 128K prompts at a matched 2% budget. On both models PQ-HSA is more accurate than every budgeted baseline, with intervals that exclude zero, and on both the interval against full attention contains zero. PQ-HSA also leads our SparQ reimplementation (Ribar et al., 2024) by 0.049 [0.014, 0.086] at a 1% budget (Appendix D). On the sixteen LongBench tasks at their native lengths (0–100 scale), PQ-HSA scores 49.1, full attention 48.8, PQCache 47.6 and MagicPIG 48.6 (Appendix D).

4.3 Where the background term's accuracy comes from

To see which part of the operator carries the accuracy, we change one component at a time (Table 2), on five 128K RULER tasks (extended-4 and question answering). B is PQ-HSA. T keeps B's selector and budget and drops the background, so B versus T measures the background term itself. C replaces the list value means by the single global mean value that SparQ uses (Ribar et al., 2024), and C2b further gives every background token one shared logit, the smallest exact logit of the selected set, so C versus C2b measures what the PQ logits add. A is truncation with the usual IVF probing of a few lists, so T versus A measures the gain from scoring every token.

Removing the background under the same selector lowers accuracy on both models. On Llama-3.1-8B, B minus T is $+0.120$ [0.091, 0.150] at 1% and $+0.041$ [0.021, 0.061] at 2%, and at 1% B is ahead of T on every task (Figure 4(b)), including multi-value needles ($+0.210$). On Qwen3-30B at 2%, B minus T is $+0.016$ [0.003, 0.029]. Scoring every token also lifts truncation itself, by 0.160 [0.120, 0.200] (T minus A). On Llama-3.1-8B both components of the background term contribute, with intervals that exclude zero (Figure 4(a)): list value means add 0.02–0.03 in macro accuracy over a global mean (B minus C), and PQ logits add 0.067 at 1% and 0.090 at 2% over one shared logit (C minus C2b), up to 0.436 on common-word extraction at 1%. With both in place, B stays close to full attention: B minus full attention is -0.010 [-0.031, $+0.011$] at 1% and $+0.005$ [-0.014, $+0.023$] at 2%. Measured directly, the background halves the attention-output error of truncation: the median relative error to full attention is 0.063 for B against 0.124 for T, and list value means keep it below that of a global mean (0.070 for C; Appendix B).

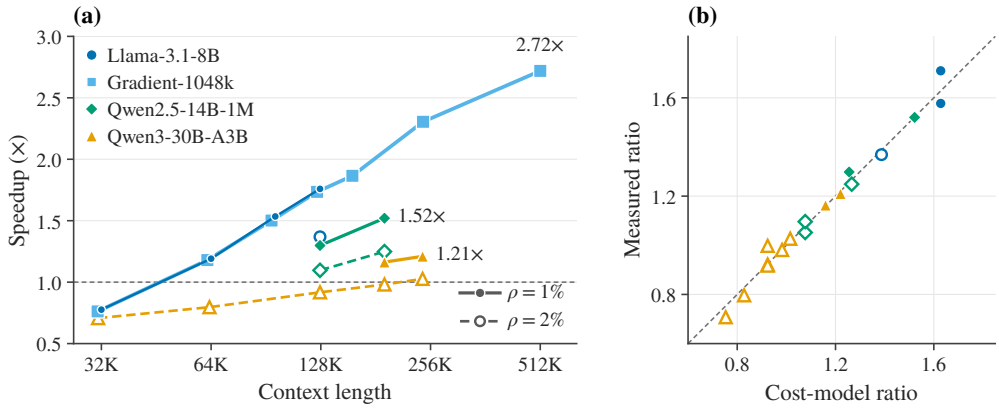


Figure 5: Speedup across model sizes and the empirical cost model. (a) Attention-segment speedup (dense/PQ-HSA) on one H20 against context length at budgets of 1% (solid, filled markers) and 2% (dashed, hollow markers); the Llama-3.1-8B and Gradient-1048k lines at 1% extend those of Figure 1(b) to 32K. (b) Measured ratio against the fitted cost model for the 17 measurements of Table 7; repeated measurements show the median.

4.4 Speed, context length, and the empirical cost model

Context length. Figure 1(b) reports the attention-segment speedup over the FlashAttention-3 decode kernel on one H20, and the speedup grows with context length. Near 128K, Llama-3.1-8B reaches $1.58\times$ when the plugin reads vLLM's KV pages in place (Section 4.5) and $1.76\times$ when PQ-HSA gathers from its own copy of the KV cache; Gradient-1048k, with the same architecture, reaches $2.72\times$ at 512K. At 512K the whole decode step is faster as well: PQ-HSA generates a token in 31.0 ms against 54.5 ms for dense attention (Table 5), as attention's share of a dense decode step on these 8B models rises from about half at 128K to about three quarters at 512K. On 512K multi-query needle retrieval, PQ-HSA gives the same score as full attention on every prompt (0.79; Appendix A). The advantage holds with concurrent requests and against dense attention over an FP8 KV cache ($1.37\times$ at 128K; Table 6).

Model size. Figure 5(a) adds Qwen2.5-14B-1M, with 8 KV heads and $G = 5$, and Qwen3-30B-A3B, with 4 KV heads and $G = 8$. On every model the ratio grows with context, and each model is faster than dense attention above its break-even length L^* (Equation 6). At 1%, Qwen2.5-14B reaches $1.52\times$ at 192K, and Qwen3-30B reaches $1.16\times$ at 192K and $1.21\times$ at 250K. Fewer KV heads and larger groups move L^* to longer contexts.

Cost model. For a model with H_{kv} KV heads, GQA group size G and context length L, we fit the per-layer times $t_{\text{dense}} = d_0 + aH_{\text{kv}}L$ and $t_{\text{pq}} = F + cH_{\text{kv}}GL$. The dense kernel streams H_{kv} heads of L tokens once, so its time grows with $H_{\text{kv}}L$ above a fixed launch cost d_0. PQ-HSA forms one lookup table and one score row per query head, $H_{\text{kv}}G$ rows per layer, so its time grows with $H_{\text{kv}}GL$ above a fixed per-layer overhead F, and c is the per-row, per-token cost of the scan and of attending the selected rows. We fit F and d_0 per model and c per budget. Fitted to the 17 measurements of Table 7, the model stays within 7.4% of every measured ratio (Figure 5(b); constants in Appendix C). Setting the two times equal gives

$$L^* = \frac{F - d_0}{H_{\text{kv}}(a - cG)}, \tag{6}$$

with a crossover when $F > d_0$ and $a > cG$. Above L^* the ratio rises toward $a/(cG)$, so the group size bounds the attainable speedup and the number of KV heads sets how early the crossover comes. At 1%, the fitted break-even lengths are 57K for Llama-3.1-8B, 85K for Qwen2.5-14B and 100K for Qwen3-30B (both budgets in Appendix C).

4.5 Serving-stack integration

The plugin replaces vLLM's single-token decode attention call without changing the engine source. Prefill and multi-token steps keep the engine's own attention, so PQ-HSA acts on the memory-bound

decode phase and runs alongside concurrent requests and prefix caching, which reuses the KV cache of a shared prompt prefix; the gather reads the engine's paged KV cache in place. The same package runs on vLLM 0.8.5 and 0.29, which differ in their attention-backend interface and KV layout, and gathers bit-identical rows on both (Appendix E). The plugin speeds up the attention call by $1.58\times$ on Llama-3.1-8B at 128K on vLLM 0.8.5; on vLLM 0.29 the speedup again grows with context length, to $1.89\times$ at 256K and $2.22\times$ at 512K on Gradient-1048k (Table 5).

5 Limitations

Our quality evaluation centers on two GQA models with a 128K window, and a broader study over longer native contexts and other attention designs, such as multi-head latent attention, would show how the background estimate transfers across architectures. Our speed evaluation targets single-GPU deployment, where the advantage lies in the memory-bound regime of long contexts; with short contexts, few KV heads or large GQA groups, a fixed per-layer overhead and the per-query-row scan dominate, and the fitted cost model of Section 4.4 places this boundary for the measured models. Reducing that overhead would widen the regime where PQ-HSA is faster.

6 Conclusion

PQ-HSA reuses the IVF-PQ scores that rank the cached keys as logits for the unselected tokens, aggregated per list with list value means under one softmax with the exact selection. At 128K it is more accurate than Quest and SnapKV at the same budget on Llama-3.1-8B and Qwen3-30B-A3B and approaches full attention on both, and under the same selector the background term adds 0.12 macro accuracy at 1% retrieval. In the same setting inside vLLM the decode attention call runs $1.6\times$ faster than the FlashAttention-3 kernel, and its advantage grows with context length; a fitted cost model summarizes when the speedup appears across 8B, 14B and 30B models, and a vLLM plugin runs the operator on two engine versions without engine changes. Because the operator asks the index only for a cheap score of every key, other quantizers and indexes from vector search can take the place of IVF-PQ.

AI use statement

We used generative AI tools to help organize and monitor experiments, write analysis and checking code, draft and revise the manuscript, and prepare figures and tables. They also produced internal reviews of drafts, which are separate from conference peer review. All reported measurements and statistical summaries are backed by recorded execution outputs and analysis code that the authors checked. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

Reproducibility statement

The anonymized code at https://anonymous.4open.science/r/pqhsa_release-71BF, also included in the supplementary material, contains the operator and its CUDA kernels, the vLLM plugin for both engine versions, the quality and speed evaluation scripts with the configuration behind every speed number, the paired-bootstrap script and CPU unit tests; it will be released publicly. Public model checkpoints and prompt generators are identified in Appendix A, which also lists budgets, index settings, clocks and sample sizes; variant definitions and per-task results are in Appendix B, fitted constants in Appendix C and serving-stack settings in Appendix E. Run scripts record seeds, engine settings, precision and timing boundaries.

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 (EMNLP)*, 2023. arXiv:2305.13245.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://aclanthology.org/2024.acl-long.172/. arXiv:2308.14508.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2110.15343.

Yaoqi Chen, Jinkai Zhang, Baotong Lu, Qianxi Zhang, Chengruidong Zhang, Jing Liu, Jingjia Luo, Di Liu, Huiqiang Jiang, Qi Chen, Bailu Ding, Xiao Yan, Jiawei Jiang, Chen Chen, Mingxing Zhang, Cheng Li, Yuqing Yang, Fan Yang, and Mao Yang. RetroInfer: A vector storage engine for scalable long-context LLM inference. *Proceedings of the VLDB Endowment (PVLDB)*, 19(5): 1016–1031, 2026. arXiv:2505.02922.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Léon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=ALzTQUgW8a. arXiv:2410.16179.

Xinyan Dai, Xiao Yan, Kelvin Kai Wing Ng, Jie Liu, and James Cheng. Norm-explicit quantization: Improving vector quantization for maximum inner product search. *arXiv preprint arXiv:1911.04654*, 2019.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2205.14135.

Aditya Desai, Shuo Yang, Alejandro Cuadron, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. HashAttention: Semantic sparsity for faster inference. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. arXiv:2412.14468.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library. *IEEE Transactions on Big Data*, 12(2):346–361, 2026. arXiv:2401.08281.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):744–755, 2014.

Gradient AI. Llama-3-8B-Instruct-Gradient-1048k. Hugging Face model card, https://huggingface.co/gradientai/Llama-3-8B-Instruct-Gradient-1048k, 2024. URL https://huggingface.co/gradientai/Llama-3-8B-Instruct-Gradient-1048k. Accessed 2026-09-20.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning (ICML)*, pp. 3887–3896, 2020. arXiv:1908.10396.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Monishwaran Maheswaran, Sebastian Zhao, June Paik, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Squeezed attention: Accelerating long context length LLM inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025a. arXiv:2411.09688.

Coleman Hooper, Sebastian Zhao, Luca Manolache, Sehoon Kim, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Multipole attention for efficient long context reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b. arXiv:2506.13059.

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? In *Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy. arXiv:2404.06654.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572. arXiv:1702.08734.

Gregory Kamradt. Needle in a haystack: Pressure testing LLMs. https://github.com/gkamradt/needle-in-a-haystack, 2023. URL https://github.com/gkamradt/needle-in-a-haystack.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. arXiv:2309.06180.

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)*, 2024. arXiv:2406.19707.

Junyan Li, Yang Zhang, Muhammad Yusuf Hassan, Talha Chafekar, Tianle Cai, Zhile Ren, Pengsheng Guo, Foroozan Karimzadeh, Colorado J. Reed, Chong Wang, and Chuang Gan. CommVQ: Commutative vector quantization for KV cache compression. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://proceedings.mlr.press/v267/li25du.html. arXiv:2506.18879.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2404.14469.

Lucas D. Lingle. Transformer-VQ: Linear-time transformers via vector quantization. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.16354.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. RetrievalAttention: Accelerating long-context LLM inference via vector retrieval. *arXiv preprint arXiv:2409.10516*, 2024.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. ClusterKV: Manipulating LLM KV cache in semantic space for recallable compression. In *62nd ACM/IEEE Design Automation Conference (DAC)*, 2025. arXiv:2412.03213.

NVIDIA Corporation. CUDA programming guide: CUDA graphs. https://docs.nvidia.com/cuda/cuda-programming-guide/04-special-topics/cuda-graphs.html, 2025. URL https://docs.nvidia.com/cuda/cuda-programming-guide/04-special-topics/cuda-graphs.html. Accessed 2026-09-25.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. *arXiv preprint arXiv:2401.06104*, 2024.

Yanlin Qi, Xinhang Chen, Huiqiang Jiang, Qitong Wang, Botao Peng, and Themis Palpanas. ParisKV: Fast and drift-robust KV-cache retrieval for long-context LLMs. *arXiv preprint arXiv:2602.07721*, 2026. URL https://arxiv.org/abs/2602.07721v3. Version 3; accepted to ICML 2026.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. SparQ attention: Bandwidth-efficient LLM inference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. URL https://proceedings.mlr.press/v235/ribar24a.html. arXiv:2312.04985.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2407.08608.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.02542.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. arXiv:2410.21465.

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)*, 2024. URL https://proceedings.mlr.press/v235/tang24l.html. arXiv:2406.10774.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:2007.04825.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. *arXiv preprint arXiv:2402.04617*, 2024a.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024b. arXiv:2309.17453.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, et al. Qwen2.5-1M technical report. *arXiv preprint arXiv:2501.15383*, 2025b. URL https://arxiv.org/abs/2501.15383.

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.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Xiaoze Fan, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Kurt Keutzer, Song Han, Chenfeng Xu, and Ion Stoica. Flash-KMeans: Fast and memory-efficient exact K-Means. *arXiv preprint arXiv:2603.09229*, 2026a. URL https://arxiv.org/abs/2603.09229.

Xu Yang, Jiapeng Zhang, Dongyang Zhao, Guo Chen, and Zhuo Tang. Self-indexing KVCache: Predicting sparse attention from compressed keys. *arXiv preprint arXiv:2603.14224*, 2026b. URL https://arxiv.org/abs/2603.14224.

Jiangnan Yu, Ceyu Xu, Yongji Wu, and Yuan Xie. STS: Efficient sparse attention with speculative token sparsity. *arXiv preprint arXiv:2605.15508*, 2026. URL https://arxiv.org/abs/2605.15508.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. PQCache: Product quantization-based KVCache for long context LLM inference. *arXiv preprint arXiv:2407.12820*, 2024a.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024b. URL https://aclanthology.org/2024.acl-long.814/. arXiv:2402.13718.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2306.14048.

A EVALUATION PROTOCOL

Tasks. RULER tasks follow the released generators at 131,072 tokens with a noise haystack (Hsieh et al., 2024): single needle (`niah_s`), multi-key needle (`niah_mk`, four keys), multi-query needle (`niah_multiquery`, four queries), multi-value needle (`niah_multivalue`, four values), common-word extraction (`cwe`: ten common words at frequency 30 against three uncommon words at frequency 3, one few-shot example), frequent-word extraction (`fwe`, $\alpha = 2$, three answers), variable tracking (`vt`) and a single-fact question-answering task (`qa`, a city–year fact). Output caps depend on the study: the multi-query and multi-value needle runs of the mechanism study use 64 tokens; the simpler needle and question-answering settings use 16, frequent-word extraction 64 and common-word extraction 128. InfiniteBench (Zhang et al., 2024b) contributes key–value retrieval, long-book multiple choice, long-book question answering, math find, number string and passkey at 128K. LongBench uses the sixteen English tasks with official metrics and output caps (Bai et al., 2024). Long chain-of-thought generation uses AIME-24 and MATH-500 with greedy decoding and a 16K generation cap on DeepSeek-R1-Distill-Llama-8B; the paired unit is the problem. Generations average 6.8K tokens, and the index is extended about 26 times per problem. The 512K check on Gradient-1048k runs the RULER generators at 524,288 tokens in the vLLM setting of the 512K speed measurement, with six prompts each of multi-query needle retrieval and common-word extraction. On multi-query needles, PQ-HSA and dense attention score 0.79 with the same score on every prompt; common-word extraction is near zero for dense attention itself at this length (0.02, PQ-HSA 0.03).

Budgets. The exact set contains four sink tokens, a 128-token local window and a retrieved portion controlled by ρ. Final budgets are rounded to whole pages of 16 tokens: 1,456 tokens at 1% and 2,752 at 2% for the 128K setting. Budgets are token counts in the exact set. Because ρ is a fraction of the indexed region, the exact set grows with the context, to about 5.4K tokens at 512K and 1%. SnapKV receives the same number of tokens (observation window 32, pooling kernel 7, max pooling); Quest receives the same number of tokens in pages of 16; sink-and-window keeps the four sink tokens and the most recent tokens up to the budget. In the mechanism study (Appendix B) the truncation variant A selects from IVF-probed candidates (128 lists), whereas B, C, C2 and C2b rank every indexed token by its PQ score. SparQ follows its Algorithm 1 (Ribar et al., 2024): it scores tokens from the $r = 32$ largest query components (summed over the GQA group), attends the same number of tokens as PQ-HSA exactly, and gives the remaining mass to the running mean of all values; the variant without this mean, which the SparQ authors use for GQA models, scores lower on the same prompts (0.751 against 0.777). ParisKV uses 16 subspaces, collision fraction 0.30 and $\beta = 0.08$ for its first stage and reranks to the same final token budget; the tokens it selects need not coincide with PQ-HSA's. This reimplements the published ranking procedure. The LongBench comparison with PQCache and MagicPIG (Table 9) uses a separate BF16 pipeline at native task lengths with per-prompt truncation caps; its 800 prompts reach about 65.4K tokens. There PQCache runs at compression ratio 0.20 and MagicPIG with $K = 11$, $L = 300$.

Implementation and serving stack. Measurements use CUDA 12.4. The quality runs use an FP16 HuggingFace implementation in which PQ-HSA replaces the decode attention of every layer after a dense prefill; prefill is chunked at 8,192 tokens. The index uses 512 lists, eight subspaces of four bits, residual encoding, one coarse and two PQ training iterations (four and eight on Qwen3-30B-A3B), direction normalization and a 1,024-token block size for top-k; new tokens are encoded every 256 steps. Speed runs use vLLM 0.8.5.post1 with the FlashAttention-3 backend, FP16 weights and KV, and the operator installed through the plugin of Section 4.5; the vLLM 0.29.0 runs use the same plugin with FP16 KV and the engine's default FlashAttention backend. The attention-segment and decode wall-clock runs use one GPU.

Statistics. Every comparison between two methods is computed on the same prompts. For a paired difference we resample prompts within each task with replacement 10,000 times (seed 20260828), recompute the macro over tasks, and report the 2.5th and 97.5th percentiles. Sample sizes are 50 prompts per task for the primary Llama-3.1-8B RULER studies; the mechanism study lists its per-variant counts in Appendix B. Other studies use 25 prompts per task for the comparisons of Table 1(b, c), 200 prompts for the RULER-core comparisons at 1%, and 130 problems for the chain-of-thought study.

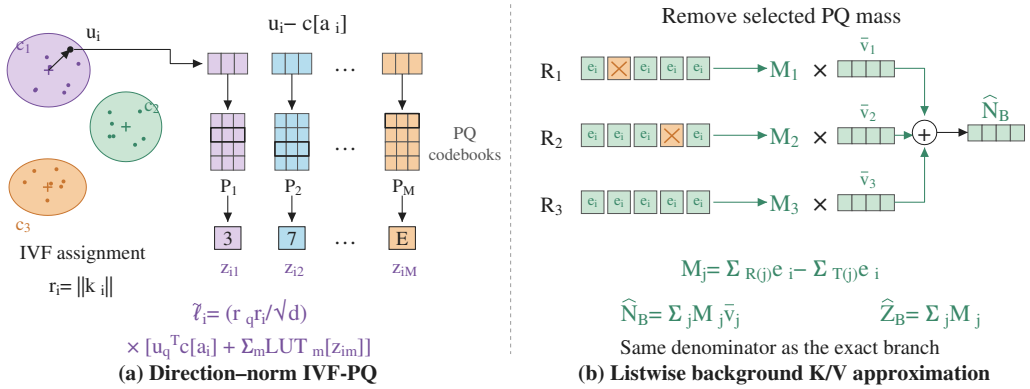


Figure 6: (a) Direction–norm residual IVF-PQ: a key direction is assigned to an IVF centroid and its residual is encoded in M PQ subspaces; multiplying by the key norm restores logit units. (b) Listwise background aggregation: each list contributes the approximate mass of its unselected tokens, weighted by the list's value mean. Here $e_i = \exp(\widetilde{\ell}_i - m_0)$, and $R(j)$ and $T(j)$ denote the indexed and selected tokens of list j. Clusters and tensor sizes are schematic.

Clocks. The attention segment is measured with CUDA events around the decode attention call on the compute stream and summed over layers; for PQ-HSA it covers all of its decode kernels: the lookup-table construction, scan, top-k, gather and combination. Time per output token (TPOT) is the host wall time per generated token (Table 5).

B Mechanism study: variants and per-task results

Figure 6 details the index and the listwise aggregation of the background term.

Variant definitions and execution checks. Table 2 defines the variants: A (truncation), B (PQ-HSA), C (global value mean) and the shared-logit variants C2 and C2b. Bx keeps B's selector, background logits and normalizer and replaces each list mean by the mean of the list's unselected members, $\bar{v}_j^B = (|R_j|\,\bar{v}_j - \sum_{i \in T \cap R_j} v_i)/|B \cap R_j|$, formed per query head from the rows already gathered for the exact set; on Llama-3.1-8B at 1%, Bx minus B is $+0.001$ $[-0.010, +0.010]$ over 250 paired prompts. C uses a token-count-weighted global value mean accumulated in FP32; C2 and C2b keep this mean and give all background tokens one shared logit, set after selection as in Table 2; A uses the IVF-probed selector with zero background. T, B, Bx, C, C2 and C2b use the same selector and budget and differ only in how the background term is formed. We verified that each variant's code path is executed.

Attention-output error. On two 128K prompts (common-word extraction and multi-query needles) we recompute, for every layer, query head and the first eight decode steps of Llama-3.1-8B at 1%, the relative error $\|\widehat{y} - y\|/\|y\|$ of each variant's attention output against full attention, 16,384 rows in total. The median (90th percentile) is 0.124 (0.476) for T, 0.070 (0.281) for C and 0.063 (0.228) for B. B has a lower error than T in 99.1% of the rows and than C in 84.0%, in both halves of the network.

Per-task results. Tables 3 and 4 report every variant on the same prompts, 25 per task, so rows can be compared directly; the paired contrasts in the main text use all the prompts each pair shares.

C Cost model, fitted constants and index build

Measurement protocol. Table 5 lists the run behind each speedup in the paper.

Table 3: Per-variant quality on Llama-3.1-8B at 128K (higher is better). A selects from IVF-probed candidates; T, B, Bx, C, C2 and C2b rank every indexed token by its PQ score, and T drops the background. MQ/MV: multi-query/multi-value needles; CWE/FWE: common-/frequent-word extraction; QA: question answering. All rows use the same 25 prompts per task; bold marks PQ-HSA (B).

ρ	Variant	n/task	MQ	MV	CWE	FWE	QA	Macro
1%	Dense	25	0.970	0.960	0.332	0.853	0.960	0.815
1%	B	25	0.930	0.920	0.492	0.827	0.960	**0.826**
1%	Bx	25	0.930	0.920	0.500	0.827	0.960	0.827
1%	T	25	0.780	0.720	0.360	0.760	0.920	0.708
1%	C	25	0.880	0.880	0.472	0.720	0.920	0.774
1%	C2b	25	0.910	0.860	0.036	0.733	1.000	0.708
1%	A	25	0.550	0.390	0.332	0.747	0.680	0.540
1%	C2	25	0.680	0.650	0.004	0.000	0.960	0.459
2%	Dense	25	0.970	0.960	0.332	0.853	0.960	0.815
2%	B	25	0.970	0.940	0.468	0.800	0.960	**0.828**
2%	T	25	0.910	0.920	0.364	0.813	0.920	0.785
2%	C	25	0.950	0.930	0.420	0.787	0.920	0.801
2%	C2b	25	0.950	0.930	0.024	0.693	0.960	0.711
2%	A	25	0.700	0.590	0.376	0.773	0.840	0.656
2%	C2	25	0.840	0.760	0.260	0.053	0.920	0.567

Table 4: Per-variant quality on Qwen3-30B-A3B at 128K and 2% retrieval. All rows use the same 25 prompts per task. Single: single needle; other columns and bold as in Table 3.

Variant	n/task	Single	MQ	MV	CWE	FWE	Macro
Dense	25	1.000	1.000	1.000	0.648	0.920	0.914
B	25	1.000	1.000	1.000	0.616	0.907	**0.905**
T	25	1.000	1.000	1.000	0.536	0.907	0.889

Table 6 extends the single-request protocol to concurrent requests and to an FP8 KV cache. With an FP8 KV cache (e4m3, bf16 weights, FlashAttention-3), dense attention reads half the bytes; PQ-HSA keeps fp16 keys and values and remains faster, by $1.37\times$ at 128K.

Derivation. Per layer, the dense decode kernel streams the H_{kv} KV heads of L tokens once, so its time is $t_{\mathrm{dense}} = d_0 + aH_{\mathrm{kv}}L$ with a launch and reduction overhead d_0. PQ-HSA reads the packed codes of the H_{kv} heads once but forms one lookup table and one score row per query head, and reduces, ranks and gathers per row, so its time is $t_{\mathrm{pq}} = F + cH_{\mathrm{kv}}GL$. The fixed per-layer overhead F collects the lookup-table construction, the top-k launch, the per-list reduction, the graph replay overhead and the part of the per-row work that does not grow with L; c is the per-row per-token cost of the scan and of attending the ρL selected rows, and depends on ρ. The ratio $R = t_{\mathrm{dense}}/t_{\mathrm{pq}}$ equals one at L^* of Equation 6 and tends to $a/(cG)$ as $L \to \infty$.

Fit. The fit uses the 17 single-device measurements of Table 7, two that gather from a separate KV copy and fifteen that read the engine's pages in place, with repeated configurations entered as separate points; a point measured in three processes enters with the median of each per-layer time. Each model has one value of $H_{\mathrm{kv}}G$ on one device, so a length-independent per-row term is absorbed into F. Least squares uses a shared a, per-model d_0 and F, and per-budget c, on the per-layer times; the dense fit has $R^2 = 0.998$ and the PQ-HSA fit $R^2 = 0.985$. The fitted ratios lie within 7.4% of the measured ones on every point, with a median deviation of 1.4%, and fall on the same side of one as the measurements.

Break-even lengths. At $\rho = 1\%$ and 2%: Llama-3.1-8B, $L^* = 57$K and 67K; Qwen2.5-14B, 85K and 109K; Qwen3-30B-A3B, 100K and 217K. The asymptotes $a/(cG)$ are 3.3, 2.7 and 1.7 at $\rho = 1\%$ for $G = 4$, 5 and 8, and 2.5, 2.0 and 1.2 at $\rho = 2\%$. Every measured point above its model's L^* in Table 7 and on the Llama-3.1-8B line of Figure 1(b) is faster than dense attention.

Table 5: Protocol of every speedup quoted in the paper. All points use one H20, a single request, a 1% retrieval budget, fp16 weights and KV, and vLLM's FlashAttention-3 kernel for dense attention. CUDA events time the attention call summed over layers, including all of PQ-HSA's decode kernels. The last row reports time per output token for the whole decode step (median process). Rows with 3×64 steps report the median of three processes. "Copy" gathers from a separate copy of the KV cache held by PQ-HSA; "paged" reads the engine's KV pages in place.

Quoted in	Model, input tokens	vLLM	KV	Steps	Ratio
Tab. 7	Llama-3.1-8B, 130,400	0.8.5	copy	3×64	1.71×
Fig. 1(b)	Llama-3.1-8B, 130,400	0.8.5	copy	32	1.76×
Figs. 1(b), 5(a)	Llama / Gradient, 32K–250K	0.8.5	copy	32–64	0.76–2.31×
§1, §4.5, Tab. 7	Llama-3.1-8B, 130,400	0.8.5 plugin	paged	3×64	**1.58**×
Fig. 1(b)	Gradient-1048k, 524,288	0.8.5	paged	32	**2.72**×
Fig. 5(a)	Qwen2.5-14B, 195,936	0.8.5	paged	3×64	1.52×
Fig. 5(a)	Qwen3-30B, 195,936	0.8.5	paged	3×64	1.16×
Fig. 5(a)	Qwen3-30B, 249,328	0.8.5	paged	3×64	1.21×
§4.5	Llama-3.1-8B, 32K	0.29 plugin	paged	256	1.07×
§4.5	Gradient-1048k, 256K	0.29 plugin	paged	256	1.89×
§4.5	Gradient-1048k, 524,288	0.29 plugin	paged	256	2.22×
§4.4	Gradient-1048k, 524,288	0.8.5	paged	256/512	**1.76**×

Table 6: Attention-segment speedup (dense/PQ-HSA) per decode step on the concurrent-request path (Llama-3.1-8B, 1% budget, vLLM 0.8.5 with paged KV; median of three processes, lower median of four for 64K with 4 requests), and against dense attention over an FP8 KV cache at 128K (median of three processes per arm).

Context	Setting	Speedup
64K	1 request	1.03×
64K	2 requests	**1.71**×
64K	4 requests	1.39×
128K	1 request	1.53×
128K	2 requests	**2.60**×
128K	FP8 KV for dense, 1 request	1.37×

Index build. The per-head build of Llama-3.1-8B solves 2,304 small k-means problems, one coarse and eight subspace problems for each of its 256 heads, and its time is dominated by kernel launches and host synchronization. The batched build solves one coarse problem per layer over its eight heads and one PQ problem over its 64 head–subspace pairs, computes key norms in FP32 and re-seeds empty clusters with the rule of the per-head build. On a 128K RULER prompt it builds all 32 layers in 0.84 s against 6.53 s per head on one H20, and assigns 99.999% of the keys to the same list and 99.98% to the same PQ codes as the per-head build.

D Position in the retrieval design space

Table 8 places the operator among the retrieval-family methods by the unit of selection, how the exact rows are chosen, what the unselected tokens receive, and the index state. PQ-HSA attends the retrieved rows plus its exact sink and local regions; the background costs one reduction over the score rows the selector already produced and n_{list} value means. Page-level selection fetches whole pages around each hit; two-stage rerankers fetch a candidate set several times larger than the final k at reduced precision before the exact pass; LSH sampling fetches a sample whose size is set by the estimator's variance; speculative selection runs a second model.

Request times of the released PQCache and MagicPIG stacks are compared in Appendix F.

Table 9 collects the quality comparisons with PQCache and MagicPIG on LongBench and with our RetrievalAttention reimplementation on RULER. Settings of the LongBench methods are listed in Appendix A.

Table 7: Attention-segment speedup R (dense/PQ-HSA) on one H20 across model sizes, context lengths L and budgets ρ. H_{kv} is the number of KV heads and G the GQA group size. KV: "copy" gathers from a separate copy of the KV cache and "paged" reads the engine's pages in place (Table 5). R_{model} is the cost model of Section 4.4 fitted to all rows; bold marks each model's largest measured ratio. Rows at 192K and 250K and both Llama-3.1-8B rows at 1% report the median of three processes.

Model	L	ρ	H_{kv}	G	KV	R / R_{model}
Llama-3.1-8B	128K	1%	8	4	copy	**1.71** / 1.63
	128K	1%	8	4	paged	1.58 / 1.63
	128K	2%	8	4	paged	1.37 / 1.39
Qwen2.5-14B-1M	128K	1%	8	5	paged	1.30 / 1.26
	128K	2%	8	5	paged	1.05 / 1.08
	128K	2%	8	5	paged	1.10 / 1.08
	192K	1%	8	5	paged	**1.52** / 1.52
	192K	2%	8	5	paged	1.25 / 1.27
Qwen3-30B-A3B	32K	2%	4	8	paged	0.71 / 0.75
	64K	2%	4	8	paged	0.80 / 0.83
	128K	2%	4	8	copy	1.00 / 0.92
	128K	2%	4	8	paged	0.92 / 0.92
	128K	2%	4	8	paged	0.92 / 0.92
	192K	1%	4	8	paged	1.16 / 1.16
	192K	2%	4	8	paged	0.98 / 0.98
	250K	1%	4	8	paged	**1.21** / 1.22
	250K	2%	4	8	paged	1.03 / 1.02

Table 8: Retrieval-family methods compared by design. Index is the state kept besides the KV cache; "host" marks state kept in host memory in the released stacks. [a]In SparQ's fastest kernel.

Method	Unit	Selection	Unselected tokens	Index
Quest	page	bound-based top-k	dropped	page bounds
ParisKV	token	low-bit pass, rerank	dropped	ANN index
PQCache	token	PQ top-k	dropped	PQ codes (host)
SparQ	token	r of d key components	one global mean value	second key copy[a]
RetroInfer	cluster	centroid top clusters	centroid logit, value sum; rest dropped	clusters (host KV)
Multipole	cluster	centroid top clusters	centroid logit, mean value	key clusters
MagicPIG	token	LSH sampling	sampling estimate	LSH tables (host)
SnapKV	token	prefill eviction	evicted	none
Spec. selection	token	1B draft model	dropped	none
PQ-HSA	token	IVF-PQ top-k	per-token PQ logit, list mean value	IVF-PQ codes

E SERVING-STACK DETAILS

Plugin. The package registers itself through vLLM's plugin entry point and modifies no engine source. When enabled it wraps the decode attention implementation of the FlashAttention backend, keeps a per-layer adapter that owns the index, and routes single-token decode steps through Equation 5; prefill chunks and multi-token steps call the original implementation. The adapter gathers selected rows directly from the engine's paged KV cache, so no second copy of keys or values is kept. Between vLLM 0.8.5.post1 and 0.29.0 the attention backend's interface and the paged KV layout changed (keys and values are now interleaved in a non-contiguous $[\text{blocks}, \text{heads}, \text{block}, 2d]$ view); the plugin handles each difference behind a version check, reads the new layout in place through its strides, and a 48-case test confirms that the gather is bit-identical to one from the earlier layout and from a flat buffer.

Table 9: Quality comparisons with retrieval-family baselines: the sixteen LongBench tasks at native lengths (up to 65.4K tokens; 0–100 scale, 50 prompts per task) in the BF16 pipeline, and RULER core-4 at 128K and a 1% budget in the primary FP16 pipeline, where RetrievalAttention and SparQ (at 1%) are our reimplementations under the protocol of Appendix A.

Method	benchmark	n	method	PQ-HSA
full attention	LongBench-16	16×50	48.8	**49.1**
PQCache	LongBench-16	16×50	47.6	**49.1**
MagicPIG	LongBench-16	16×50	48.6	**49.1**
RetrievalAttention	RULER core-4	4×50	0.925	**0.950**
SparQ	RULER, 5 tasks	5×25	0.777	**0.826**

F ADDITIONAL MEASUREMENTS

Offload-based baselines. End-to-end request time was also compared with the released PQCache and MagicPIG stacks, which keep part of their state on the host, in BF16 at 126,720 input and 4,096 output tokens. The paired geometric-mean ratios PQ-HSA/PQCache and PQ-HSA/MagicPIG are 0.0264 and 0.2448 (about $38\times$ and $4.1\times$ less time). This comparison uses end-to-end request time in BF16, a different clock from the attention-segment ratios elsewhere in the paper.