FIER: Fine-Grained and Efficient KV Cache Retrieval for Long-context LLM Inference

Dongwei Wang[1], Zijie Liu[2], Song Wang[3], Yuxin Ren[1], Jianing Deng[4], Jingtong Hu[4], Tianlong Chen[2], Huanrui Yang[1†],
[1]The University of Arizona, [2]The University of North Carolina at Chapel Hill,
[3]The University of Virginia, [4]University of Pittsburgh
{dongweiw, yuxinr, huanruiyang}@arizona.edu
{jesseliu, tianlong}@cs.unc.edu, sw3wv@virginia.edu
{jthu, JID70}@pitt.edu

Abstract

The Key-Value (KV) cache reading latency increases significantly with context lengths, hindering the efficiency of long-context LLM inference. To address this, previous works propose retaining a small fraction of KV cache based on token importance. For example, KV eviction uses static heuristics to retain tokens, while KV retrieval dynamically selects query-relevant tokens for more adaptive cache management. However, we observe that important tokens are often sparsely distributed across the long context. This sparsity makes existing page-level KV retrieval inaccurate, as each page may include irrelevant tokens and miss critical ones. In this work, we propose Fier, a Fine-Grained and Efficient KV cache Retrieval method. Fier uses 1-bit quantized keys to estimate the importance of each token, resulting in efficient and precise retrieval. Experiments show that Fier matches full KV performance using only 11% of the cache budget across various long-context tasks, reducing decoding latency by 1.2× to 1.5×.

1 Introduction

KV caching is a memory-for-computation acceleration technique for LLM inference, enabling faster decoding by reusing intermediate hidden states (Waddington et al., 2013). However, during inference, each decoded token must attend to the full KV cache, causing cache reading latency to grow significantly with context length. For example, in LLaMA 7B (Touvron et al., 2023), a 32k-token KV cache requires 16GB of memory and over 11ms to read—accounting for more than half of the total inference time (Tang et al., 2024).

To address this issue, previous works have proposed to selectively retain only a subset of KV cache entries based on token importance. Among them, one line of work—known as KV eviction (Fig. 1(b))—focuses on retaining fixed-position tokens that typically receive higher attention weights, such as the initial tokens (due to the attention sink phenomenon) and the most recent tokens (Xiao et al., 2023; Zhang et al., 2023; Li et al., 2024; Liu et al., 2023). However, these approaches overlook the dynamic nature of KV criticality, i.e., tokens that are evicted earlier may become important in the future. Their inability to recall evicted tokens often results in degraded performance in multi-round QA or long conversation applications. Motivated by this, another line of work—KV retrieval (Fig. 1(c))—has been proposed to dynamically recall tokens that are most relevant to the current query during generation (Tang et al., 2024; Chen et al., 2024a).

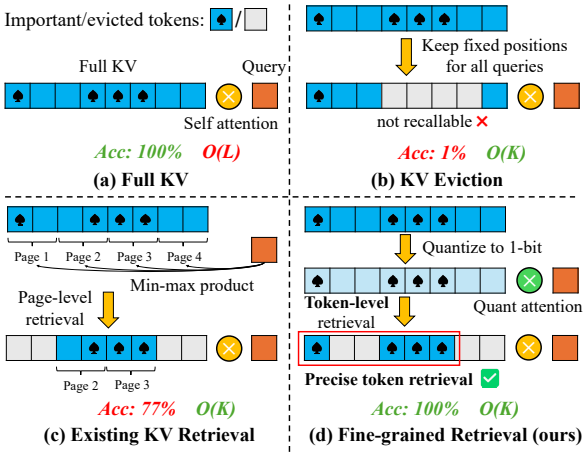

Figure 1: Comparison of KV eviction (b), KV retrieval (c) and Fier (d). While existing retrieval methods suffer from coarse granularity, Fier achieves higher accuracy through fine-grained *token-level* retrieval, and preserves selection efficiency by quantization.

Despite achieving better performance, KV retrieval requires frequent estimation of the token importance for every new query, introducing additional computational overhead. To mitigate this, existing methods perform page-level retrieval, where all tokens that belong to a certain page of the KV cache are retrieved (or not re-

†Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 9702–9713
November 4-9, 2025

trieved) simultaneously to avoid computing full attention scores, leading to a coarser granularity of retrieval.

However, we observe that in long-context tasks, information relevant to the current query may be scattered throughout the entire input, i.e., important tokens are often sparsely distributed across the KV cache (shown in Fig. 2). Consequently, page-level retrieval inevitably leads to imprecise selection: *retrieved pages may include irrelevant tokens, while evicted pages may exclude critical ones*, thereby affecting the model performance.

In this paper, we aim to address the retrieval inaccuracy while preserving selection efficiency. Specifically, we find that quantizing the key cache to as low as 1-bit has minimal impact on the accuracy of Top-k selection in token importance estimation (shown in Fig. 3). Despite quantization truncates large values, important tokens are still preserved in the Top-k after computing quantized attention. Based on this insight, we propose Fine-Grained and Efficient KV cache Retrieval (Fier), a 1-bit quantization-based KV retrieval method. As shown in Fig. 1(d), Fier enables more accurate recovery of important tokens and reduces the selection cost. This leads to improved model performance under the same cache budget.

We evaluate Fier across PG19 (Rae et al., 2019), LongBench (Bai et al., 2023), and the passkey retrieval benchmarks (Peng et al., 2023). The results demonstrate the effectiveness of Fier in both generative and retrieval-focused settings. Experiments show that Fier achieves performance comparable to using the full KV cache while requiring only 11% of the cache budget, and consistently outperforms existing KV eviction and retrieval baselines. Additionally, Fier achieves 1.2× to 1.5× decoding speedup across different context lengths on a single RTX 4090 GPU. In summary, we make the following contributions in this work:

- We observe the sparse distribution of important tokens within the KV cache in long-context scenarios, highlighting the necessity of fine-grained retrieval.
- We propose Fier, a KV retrieval method built on 1-bit key quantization, which enables efficient and accurate token-level retrieval.
- We conduct comprehensive evaluations of Fier across diverse long-context tasks and model architectures, demonstrating its superior performance and efficiency.

2 Related Work

Long-Context LLMs. Large Language Models (LLMs) have transformed the landscape of natural language processing, largely due to their strong ability to deal with long context. Their context length capacity has increased dramatically—from 4k to 128k (Grattafiori et al., 2024), 1M (Yang et al., 2025), and even 10M (Team et al., 2024) tokens. This expansion unlocks a range of advanced capabilities, including o1 long-range reasoning (Guo et al., 2025; OpenAI, 2024), in-context learning (Li et al., 2025), and multimodal intelligence (Weng et al., 2024). Fier aims to improve the inference efficiency of long-context LLMs by exploiting the sparsity of the KV cache.

KV Cache Eviction. Previous work identified the sparsity of attention matrices, showing that retaining only a small fraction of tokens is sufficient for the performance. For example, Xiao et al. (2023) propose to retain the first few tokens based on the "attention sink" phenomenon. H2O (Zhang et al., 2023) retains a limited set of KV cache by selecting tokens with the highest cumulative attention scores. SnapKV (Li et al., 2024) selects clustered historical tokens along with a localized window of recent tokens. However, these approaches ignore the fact that tokens evicted can become important in the future. Fier addresses this via query-specific KV retrieval, enabling dynamic reassessment of token importance at each decoding step.

KV Cache Retrieval. KV retrieval methods, including our proposed Fier, dynamically select tokens relevant to the current query. However, existing approaches like Quest (Tang et al., 2024), ArkVale (Chen et al., 2024a), PQCache (Zhang et al., 2025), SparQattn (Ribar et al., 2023) and MagicPIG (Chen et al., 2024b) retrieve at the page or cluster level or compute partial tokens' criticality for efficiency, overlooking the sparse distribution of important tokens. In this paper, we propose a fine-grained, token-level retrieval strategy that maintains efficiency while improving accuracy. This design better captures critical information missed by existing methods.

KV Cache Quantization. Another related line of work is KV quantization (Liu et al., 2024; Dong et al., 2024), which compresses the cache by performing self-attention in low-bit space. The objec-

tive of KV quantization is to minimize global reconstruction error. In contrast, Fier achieves cache compression by retaining only a subset of the full KV and adopts a relaxed quantization objective focused on preserving Top-ranked tokens, enabling the use of extremely low bit-widths.

3 Methodology

3.1 Preliminaries

In an autoregressive LLM, the inference process typically consists of two stages: the prefill stage and the decoding stage.

Prefill Stage. Given an input context $\mathbf{X} \in \mathbb{R}^{l_{prompt} \times d}$, the model computes the initial key and value representations, denoted as $\mathbf{K}_0 \in \mathbb{R}^{l_{prompt} \times d}$ and $\mathbf{V}_0 \in \mathbb{R}^{l_{prompt} \times d}$, which together form the initial KV cache.

Decoding Stage. At each step, a new query $\mathbf{q} \in \mathbb{R}^{1 \times d}$ is generated and the corresponding key $\mathbf{k}$ and value $\mathbf{v}$ are appended to the initial cache $(\mathbf{K}_0, \mathbf{V}_0)$ to form the current KV cache:

$$\mathbf{K} \leftarrow \text{Concat}(\mathbf{K}_0, \mathbf{k}), \mathbf{V} \leftarrow \text{Concat}(\mathbf{V}_0, \mathbf{v}).$$

The attention output is then computed as:

$$\mathbf{s} = \text{softmax}(\mathbf{q}\mathbf{K}^T), \quad \mathbf{o} = \mathbf{s}\mathbf{V}.$$

The major overhead of decoding comes from computation of attention score $\mathbf{s}$, which reflects the importance of KV token $\mathbf{k}_j$ to the query. At each step, the current query must attend to the entire KV cache. This cost becomes higher in long-context tasks. To address this, previous works have demonstrated attention sparsity, observing that initial tokens (attention sink) and recent tokens (locality) tend to receive higher attention scores. Based on this, they retain a fixed subset of tokens, denoted as $(\mathbf{K}', \mathbf{V}') \in \mathbb{R}^{n \times d}$ at these positions for all queries, where n is cache budget.

However, subsequent studies show that token criticality varies across different queries. As a result, query-specific token selection is necessary, where token importance needs to be recomputed for each query. To improve importance estimation efficiency, existing works tend to perform selection at a coarser, page-level granularity. For example, Quest (Tang et al., 2024) partitions the key cache $\mathbf{K} \in \mathbb{R}^{l \times d}$ into $\frac{l}{L}$ pages (L is typically 16 or 32). For each page $\mathbf{P}$, it extracts maximum and minimum vectors $\mathbf{k}_P^{\max}$ and $\mathbf{k}_P^{\min} \in \mathbb{R}^{1 \times d}$, performs point-wise multiplication with q,

$$\alpha^{\max} = \mathbf{q} \odot \mathbf{k}_P^{\max}, \tag{1}$$

$$\alpha^{\min} = \mathbf{q} \odot \mathbf{k}_P^{\min}, \tag{2}$$

and takes the maximum across both the hidden dimension and the two vectors to obtain the page importance score.

$$s_P = \max_{i=1,\ldots,d} \left(\max \left(\alpha_i^{\max}, \alpha_i^{\min} \right) \right). \tag{3}$$

Pages with the highest importance scores are then selected for self-attention computation.

During the selection, Quest loads only 2 $\mathbf{K}$ vectors per page. Assuming $\mathbf{K}$ is stored in `float16`, this results in a key cache load ratio of:

$$\frac{2 \times 16 \times l/L}{l \times 16} = \frac{2}{L}. \tag{4}$$

It is clear that larger page sizes reduce importance estimation costs, but lead to coarser granularity by grouping more tokens together. There exists a trade-off between efficiency and accuracy.

3.2 Fine-grained and Efficient KV Retrieval

To improve upon existing page-level KV retrieval methods, we make the following two observations on the token importance estimation process.

OB1: Important Token Sparsity Makes Page Retrieval Inaccurate. To understand the trade-off of page granularity, we visualize both the highest-attended tokens and those selected under page-level partitioning by mapping them back to the original context. As shown in Fig. 2, queries $Q1$ and $Q2$ attend to different regions of the context, which aligns with prior findings on the dynamic nature of token criticality. Moreover, tokens with high attention scores are sparsely distributed across the context, and we observe that pages 7, 16, and 54 each contain a mixture of both important and unimportant tokens. This overlap makes inaccurate retrieval, which pinpoints the necessity of fine-grained retrieval strategies that identify important information at the token level, rather than relying on coarse-grained page grouping.

Motivated by this, we aim to design a retrieval strategy that operates at fine-grained token level while incurring minimal additional computation overhead. Notably, quantization enables low-bit computation, achieving high efficiency while still allowing every token to participate in the criticality estimation. We begin by validating this insight through the following observation.

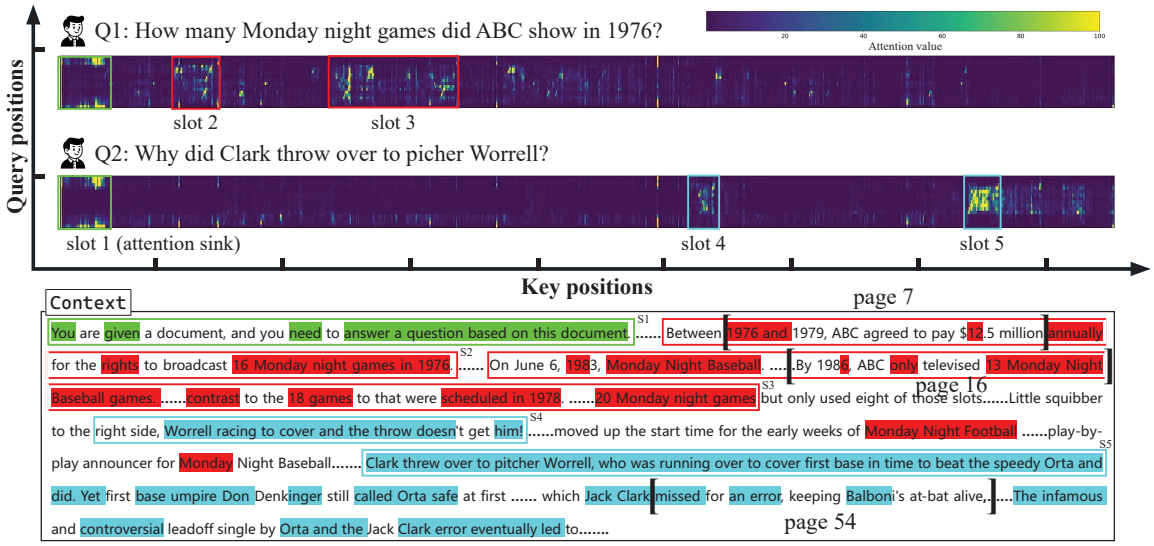

Figure 2: High-scoring tokens selected by two different queries in LLaMA (Grattafiori et al., 2024) are mapped to their corresponding text. Important tokens are query-dependent and sparsely distributed across the context, causing pages to contain a mix of important and unimportant tokens, which leads to inaccuracy in page-level retrieval.

OB2: Quantization has Minimal Impact on Top-k Selection, even at 1-bit. Quantizing KV cache values to a lower precision significantly reduces the cost of data movement and attention computation. Let $\mathbf{K} \in \mathbb{R}^{l \times d}$ denote the original key cache, where $\mathbf{k}_i \in \mathbb{R}^{1 \times d}$ is the key vector corresponding to the i-th token, quantization converts each $\mathbf{k}_i$ to its dequantized counterpart $\tilde{\mathbf{k}}_i$ as

$$\mathbf{k}_i^Q = \left\lfloor \frac{\mathbf{k}_i - \mathbf{z}_i^K}{\mathbf{s}_i^K} \right\rceil, \quad \tilde{\mathbf{k}}_i = \mathbf{k}_i^Q \odot \mathbf{s}_i^K + \mathbf{z}_i^K, \quad (5)$$

where $\mathbf{s}_i^K, \mathbf{z}_i^K \in \mathbb{R}^{1 \times d}$ are the per-channel calibrated scaling and bias vectors specific to the i-th key vector. Previous KV quantization methods (Zhang et al., 2024) aim to optimize these factors through the calibration process to minimize the impact of quantization on the computed attention score. This objective can be formulated as an ℓ_2 loss:

$$\min_{\mathbf{s}^K, \mathbf{z}^K} \sum_{i=1}^{l} \left(\mathbf{q}\mathbf{k}_i^\top - \mathbf{q}\tilde{\mathbf{k}}_i^\top \right)^2. \quad (6)$$

However, quantization introduces perturbations on the values, drawing computation results away from intended. The impact of quantization is more severe if outliers exist in the distribution. Previous methods like Kivi (Liu et al., 2024), equipped with advanced channel-wise grouping and rescaling scheme, cannot quantize the KV cache below 2 bit while retaining model performance.

Meanwhile, we observe that quantizing the key cache to low bits has significantly less impact on the token importance estimation than retaining the attention scores. Here we explore an extreme case by quantizing $\mathbf{K}$ to just 1-bit. Using the full-precision attention scores as ground truth, we evaluate whether Top-k token selection can still be accurately recovered under such an aggressive quantization setting. Specifically, we feed a long input context (14k tokens) into LlaMA during the prefill stage, and compute attention scores under both full-precision and 1-bit quantized $\mathbf{K}$ using the same query. As shown in Fig. 3, despite the fact that low-bit quantization truncates large values and distorts the overall distribution, the Top-k tokens remain largely unchanged. This suggests that token criticality is still well captured, even un-

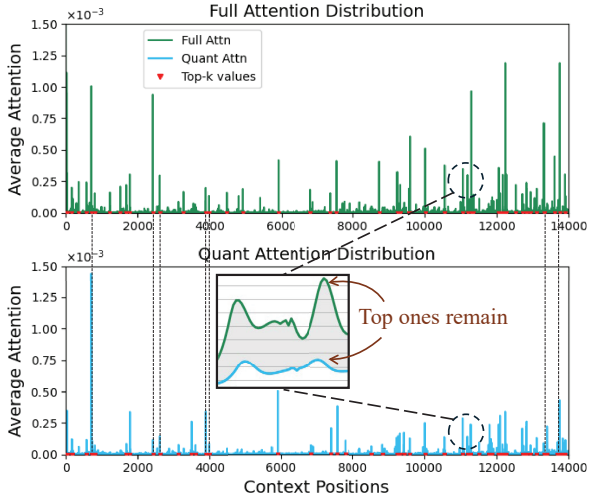

Figure 3: Averaged full/quantized attention scores along the sequence. Despite distribution distortion caused by low-bit quantization, the Top-k tokens are largely preserved, indicating that token criticality remains identifiable under extreme quantization.

der extreme quantization.

To understand the reason, we analyze the quantization objective implied by the goal of token importance estimation. For importance estimation, we aim to maintain the ranking of the Top-k tokens rather than preserving all attention scores precisely. Assume m is the minimum margin between the attention scores of Top-k and non-Top-k tokens in the full-precision setting. To preserve this ranking after quantization, it is sufficient to ensure that the attention score of each token deviates from its full-precision counterpart by at most $m/2$. This leads to the following hinge objective:

$$\min_{\mathbf{s}^K, \mathbf{z}^K} \sum_{i=1}^{l} \max\left(0,\ \frac{m}{2} - \left(\mathbf{q}\mathbf{k}_i^\top - \mathbf{q}\tilde{\mathbf{k}}_i^\top\right)\right). \quad (7)$$

Compared to the ℓ_2 loss, the hinge loss imposes a relaxed objective that prioritizes preserving the relative ranking of Top-k tokens (Fig. 4). More importantly, outlier tokens that lead to large attention scores enjoy larger margins under the hinge loss, making their quantization errors less impactful. This enables quantization with 1-bit and larger group sizes g while still maintaining Top-k accuracy.

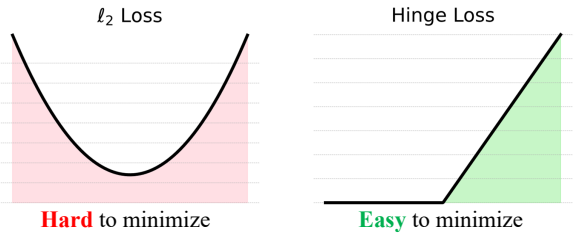

Figure 4: Intuition of Fier. Our relaxed quantization objective ignores errors smaller than $m/2$, allowing the use of extremely low bit-widths while preserving Top-k ranking accuracy.

3.3 Fier Workflow

Motivated by previous observations, we propose Fier, a token-level retrieval method based on 1-bit linear quantization. Fier compresses the key cache into 1-bit using a simple round-to-nearest (RTN) quantizer. Given a query $\mathbf{q}$, approximate attention scores are computed efficiently using quantized keys. Based on these scores, Fier selects the Top-k tokens and performs full-precision self-attention over the selected subset. We summarize the workflow of Fier in Algorithm 1.

Algorithm 1 Fier: Token-Level KV Retrieval via 1-Bit RTN Quantization

Input: Query $\mathbf{q}$, full-precision $(\mathbf{K}, \mathbf{V})$, group size g
Output: Attention output $\mathbf{o}$
// Step 1: Quantize $\mathbf{K}$ to 1-bit
Partition $\mathbf{K}$ into groups of size g along each channel
For each group, compute the scaling factors (s, z) and broadcast them to construct $\mathbf{s}^K, \mathbf{z}^K \in \mathbb{R}^{l\times d}$.
$\mathbf{K}_Q = \left\lfloor \frac{\mathbf{K} - \mathbf{z}^K}{\mathbf{s}^K} \right\rfloor, \mathbf{K}_Q \in \{-1, 1\}^{l\times d}$ # binary
$\tilde{\mathbf{K}} = \mathbf{K}_Q \odot \mathbf{s}^K + \mathbf{z}^K$
// Step 2: Compute Approximate Attention Scores
$\tilde{\mathbf{s}} = \mathbf{q} \cdot \tilde{\mathbf{K}}^\top$
// Step 3: Select Top-k Tokens
$\mathcal{S}_q = \text{Top-}k(\tilde{\mathbf{s}})$
// Step 4: Compute Real Attention on Selected Tokens
$\mathbf{K}' = \mathbf{K}[\mathcal{S}_q], \mathbf{V}' = \mathbf{V}[\mathcal{S}_q]$
Return: $\mathbf{o} = \text{softmax}(\mathbf{q}\mathbf{K}'^\top)\mathbf{V}'$

3.4 Theoretical Analysis of Efficiency

Beyond retrieval accuracy, the efficiency of the selection and decoding stage is also critical for practical deployment. We measure the efficiency using the key cache access ratio (CAR), defined as the fraction of the key cache accessed throughout the pipeline. Specifically, we compare both retrieval- and eviction-based methods under the same top-k setting. In the decoding phase, all methods (Fier, Quest, and H2O) access k tokens, so their CAR is identical. The key difference lies in the selection phase. For $\mathbf{K}$ stored in `float16`, we quantize it to 1-bit with group size g. Note that in addition to the 1-bit $\mathbf{K}_Q$, each group also needs to store a pair of (s, z) in `float16`. Therefore, the CAR of Fier will be calculated as:

$$\frac{l \times 1 + (l/g) \times 2 \times 16}{l \times 16} = \frac{1 + 32/g}{16}, \quad (8)$$

which decreases with a larger group size. Recall that Quest has a load ratio of $2/L$. For fairness, we set $g = 32$, which matches the load ratio of $1/8$ with page size $L = 16$ as implemented in the Quest baseline. In contrast, H2O maintains a dynamic pool of size k/l during the selection stage. Therefore, when $k = l/10$, the CAR of the three methods can be summarized in Tab. 1. While retrieval does introduce additional overhead during the selection compared to eviction, its goal is to improve performance with only a modest CAR increase.

Table 1: CAR comparison of different methods (Assuming $k = l/10$).

Method	Sel. (%)	Dec. (%)	Total CAR (%)
H2O	10.0	10.0	20.0
Quest	12.5	10.0	22.5
Fier	12.5	10.0	22.5

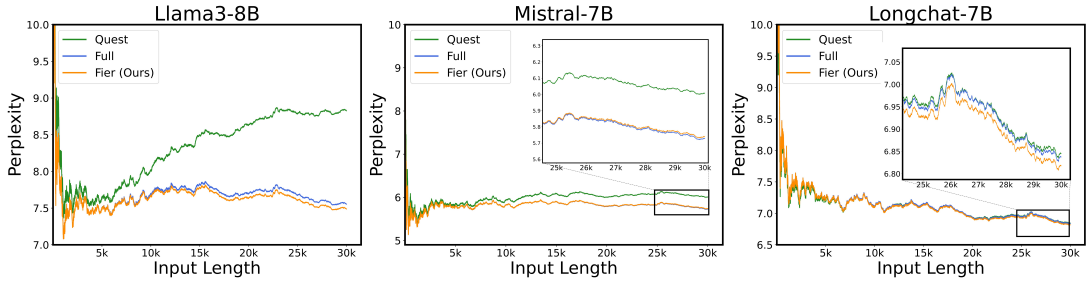

Figure 5: Language modeling evaluation. We measure output perplexity by prompting the model with input lengths ranging from 0 to 32k tokens. Fier achieves performance comparable to full KV and significantly surpasses Quest.

4 Experiments

4.1 Setting

Datasets. We evaluate Fier on the language modeling task PG19 (Rae et al., 2019). To assess its performance on long-context QA, we further conduct experiments on LongBench (Bai et al., 2023) using six representative datasets: NarrativeQA, HotpotQA, Qasper, TriviaQA, GovReport, and MultifieldQA. The detailed information about the six datasets is in Appendix A. We evaluate Fier on the passkey retrieval task (Peng et al., 2023) to assess its ability to model long-range dependencies. We also compare responses from Fier and Quest enabled chatbots in Appendix E.

Models. We apply our method to three open-sourced models: LLaMA-3-8B-Instruct (Grattafiori et al., 2024), LongChat-v1.5-7B-32k (Li et al., 2023), and Mistral-7B-Instruct (Jiang et al., 2023). Following the same setup as in Quest, neither Fier nor the baseline methods are applied to the first two layers of the model. We evaluate the performance of each method under varying KV cache budgets.

Baselines. We thoroughly compare the performance of Fier and Quest (Tang et al., 2024) across various benchmarks. Note that in all performance evaluation, we set the page size of Quest to 16 and the grouping size of Fier to 32 for a fair comparison. We also compare Fier with four KV eviction baselines: H2O (Zhang et al., 2023), StreamingLLM (Xiao et al., 2023), SnapKV (Li et al., 2024) and TOVA (Oren et al., 2024). The results in the experiments are either taken from the original paper or obtained by running open-source code. More implementation details can be found in Appendix B.

4.2 Insight Verification

More Accurate Retrieval of Important Tokens. In Fig. 6, we visualize the positions of Top-64 tokens selected by Quest with different page sizes and by Fier-1bit-g32, all mapped back onto the full KV cache. We then compute the recall rate, defined as the overlap between the retrieved tokens and those selected using the full attention score. The experiment is conducted on LLaMA.

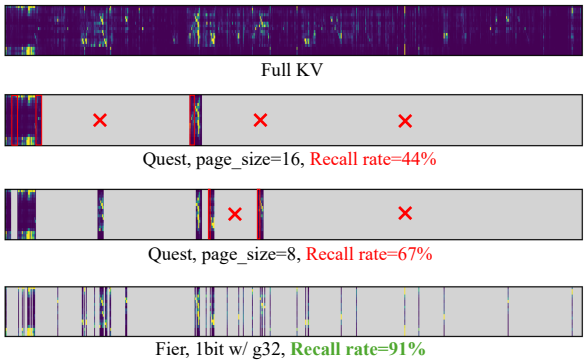

Figure 6: Fier's token-level retrieval preserves more Top-k tokens compared to Quest's page-level approach, resulting in higher recall and better alignment with full attention.

We observe that Quest with either small or large page sizes tends to retain unimportant tokens and evict important ones, due to its coarse-grained page-level retrieval. In contrast, Fier performs token-level retrieval through low-bit quantized attention computation, resulting in a significantly higher recall rate and better alignment with the full attention distribution.

4.3 Performance Evaluation

4.3.1 PG19 Results

We begin by evaluating language modeling perplexity on PG19, a benchmark consisting of 100 books with an average length of 70k tokens. We evaluate three different models by feeding them texts of varying lengths and compare the results against both Full KV and Quest (Fig. 5). Note that both Fier and Quest are evaluated under the same KV cache budget of 4096 tokens. Fier achieves performance close to that of Full KV and significantly outperforms Quest on both the LLaMA and Mistral models.

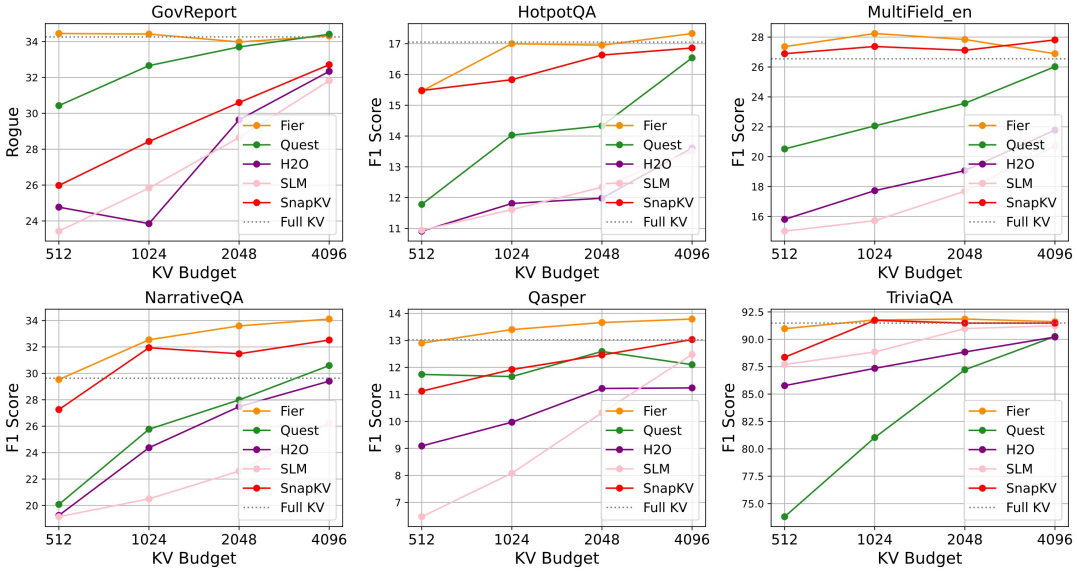

Figure 7: LongBench evaluation on LaMA-3-8B-Instruct. Fier outperforms all baselines across six long-context datasets and matches full KV performance with just 1k cache budget.

Table 2: LongBench evaluation on LongChat and Mistral. Consistent performance gains of Fier on two models.

LLMs	Method	Multifield_en				NarrativeQA				GovReport				Avg.
		512	1024	2048	4096	512	1024	2048	4096	512	1024	2048	4096	
LongChat-7B	Full KV	43.2				20.88				30.89				31.66
	SLM	21.17	21.29	26.55	34.82	10.69	12.46	17.55	18.94	16.85	21.88	22.77	26.96	20.99
	H2O	21.15	25.07	30.28	37.75	10.67	12.96	14.75	19.31	19.73	22.69	26.15	27.55	22.34
	SnapKV	36.74	37.93	40.26	42.21	**19.21**	**19.32**	19.28	**20.68**	22.57	23.45	26.3	28.55	28.04
	Quest	38.05	**41.95**	**44.03**	42.41	16.51	18.76	19.37	20.12	27.54	30.12	**31.27**	31.21	30.11
	Fier	**39.05**	39.85	42.4	**42.54**	17.23	17.83	**19.96**	19.51	**30.18**	**30.89**	30.85	**31.67**	**30.16**
Mistral-7B	Full KV	52.92				28.49				34.81				38.74
	SLM	29.91	31.16	35.75	44.12	24.21	24.79	25.91	28.9	22.09	24.6	27.57	31.19	29.18
	H2O	47.39	48.43	49.03	49.95	23.04	27.79	28.6	30.2	24.24	26.15	27.19	30.04	34.34
	SnapKV	53.05	52.64	52.92	**53.44**	25.57	28.09	**30.27**	29.76	25.83	28.28	30.91	32.74	36.96
	Quest	48.07	50.67	53.7	51.76	20.25	25.71	28.31	27.28	31.42	32.57	33.07	33.52	36.36
	Fier	**53.97**	**54.67**	**54.37**	53.32	**26.75**	**28.75**	29.11	**31.11**	**34.47**	**34.53**	**34.65**	**34.9**	**39.22**

4.3.2 Longbench Results

We evaluate on the LongBench benchmark using LLaMA-3-8B-Instruct across a diverse set of long-context tasks, including single-document QA (NarrativeQA, Qasper, MultiFieldQA), multi-document QA (HotpotQA), summarization (GovReport), and few-shot learning (TriviaQA). We also compare Fier with H2O, StreamingLLM, SnapKV, and Quest under varying KV cache budget settings. In addition, we perform evaluations using the Mistral-7B and LongChat-7B models to verify the generality of our method across different model architectures.

As shown in Fig. 7, Fier consistently achieves superior performance compared to all baselines across six long-context datasets under various KV cache budgets. Overall, Fier surpasses all baselines at cache budgets of 512, 1024, and 2048 tokens, and achieves comparable performance to full KV using only 1k tokens, which is only 11% of the full cache. This suggests that Fier benefits from accurately retrieving critical tokens, enabling efficient use of limited cache resources without compromising model quality. Similar results are observed on the other two models. As shown in Tab. 2, Fier shows consistent improvements on both single-document and multi-document tasks.

4.3.3 Passkey Retrieval

We further evaluate Fier's ability to handle long-distance dependencies. Specifically, we employ the passkey retrieval task (Peng et al., 2023) as our benchmark. This task assesses whether the model can retrieve a simple passkey from a large amount of irrelevant content. Following the setup in Tang et al. (2024), we use a context length of 10k tokens for evaluation with both LongChat-7B and a smaller model, LLaMA3-1B. KV eviction methods perform poorly due to their inability to recall discarded tokens, while Quest provides noticeable improvements over them. Fier, however, achieves even higher retrieval accuracy, performing well under extremely low budgets of just

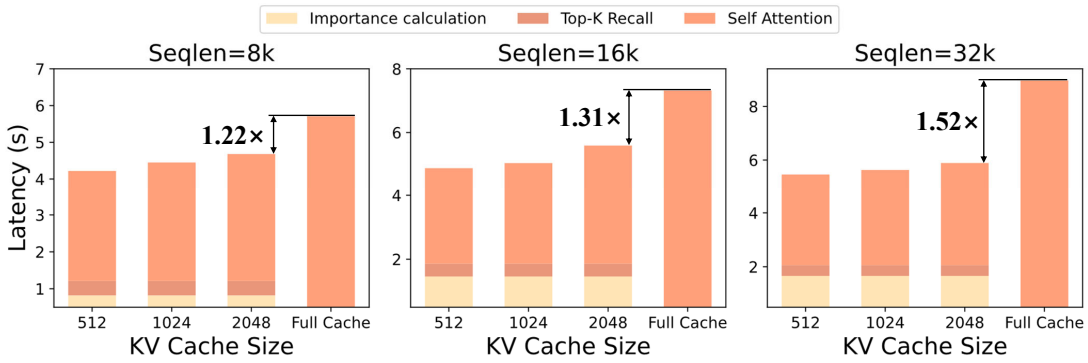

Figure 8: Decoding latency of 256 tokens on LLaMA-2-7B (Touvron et al., 2023) under varying prefill context lengths. Fier achieves increasing speedup over full KV by restricting attention to a small subset of the cache. At 32k context length, it delivers over 1.5× acceleration.

Table 3: Passkey retrieval accuracy under 10k context length. KV eviction methods struggle to recall discarded tokens, while Quest improves retrieval performance. Fier achieves the highest accuracy, even with extremely low budgets (32 and 64), effectively enhancing smaller models.

Longchat-7B	**Cache Budget**				
Method	32	64	128	256	512
H20	0%	1%	1%	1%	3%
StreamingLLM	1%	1%	1%	3%	5%
TOVA	0%	1%	1%	3%	8%
Quest	65%	99%	99%	99%	100%
Fier (ours)	**87%**	**99%**	**99%**	**100%**	**100%**
LlaMA3-1B	**Cache Budget**				
Method	32	64	128	256	512
H20	0%	1%	0%	1%	2%
StreamingLLM	0%	0%	1%	2%	4%
TOVA	0%	1%	1%	3%	6%
Quest	36%	60%	92%	99%	99%
Fier (ours)	**63%**	**87%**	**97%**	**100%**	**100%**

32 and 64 tokens, especially improving the long-context processing capability of smaller models (shown in Tab. 3).

4.4 Ablation Study

Token Granularity vs. Quantized Attention. To understand whether Fier's performance gain primarily stems from its fine-grained token-level selection (as opposed to page-level) or from the use of quantized attention as an importance metric, we conduct an ablation study on LLaMA-3-8B-Instruct, isolating these two factors. Specifically, we reduce the page size in Quest to approximate finer granularity and compare performance. In addition, we apply the averaged quantized attention score as the page-level importance metric under the same page size, and evaluate its effect on Quest. As shown in Tab. 4, using smaller page sizes in Quest leads to improved performance. However, it also increases the cache load ratio. Additionally, incorporating quantized attention for scoring further enhances its effectiveness. Notably, Fier can be viewed as quantized attention with a page size of 1, achieving the best overall results. These results suggest that Fier benefits from both the token-level granularity and the use of quantized attention as a lightweight yet effective importance estimator.

Table 4: Ablation study. Fier benefits from both token granularity and quantized attention. Larger group sizes yield better efficiency but may reduce accuracy.

Method	**Load R.**	**HotpotQA**		
		512	1024	2048
Quest-p32	0.063	7.16	8.98	11.39
Quest-p16	0.125	11.78	14.03	14.33
Quest-p16-w/quant	0.125	13.54	14.77	15.26
Quest-p8	0.25	15.16	16.66	**17.3**
Fier-g256	0.07	12.55	14.51	16.73
Fier-g128	0.08	13.96	15.53	17.04
Fier-g32	0.125	**15.46**	**17.0**	16.95

Fier w/ Different Group Sizes. We also investigate how the group size used during key quantization affects Fier's performance. We find that as the group size increases, the cache load ratio decreases, but this comes at the cost of reduced performance. Nevertheless, Fier consistently outperforms Quest under the same cache load ratio.

4.5 Efficiency Profiling

Inference Efficiency. In Fig. 8, we present the decoding latency of 256 tokens on LLaMA-2-7B under different prefill context lengths. To ensure a fair comparison with Full KV, we include both the time spent on computing quantized attention and the time required to recall the selected Top-k tokens. We implement the group-wise quantization kernel using Triton, and employ the Top-k

CUDA operator to efficiently perform Top-k token recall. Fier's efficiency gain is mainly attributed to the speedup in the self-attention computation, as it restricts attention to only a small subset of the KV cache. This acceleration becomes more pronounced as the context length increases; for instance, at a context length of 32k tokens, Fier achieves over 1.5× decoding speedup. We also provide a detailed latency breakdown comparison of Fier and Quest in Appendix D.

5 Conclusion

We present Fier, a fine-grained and efficient KV cache retrieval algorithm that selects important tokens using 1-bit quantization. By involving all tokens in the computation, Fier enables token-level criticality estimation, leading to improved recall rate and model performance. Extensive experiments across various tasks and model architectures show that Fier consistently outperforms existing methods. Notably, Fier matches full cache performance using only 11% of the KV budget and achieves a 1.2×–1.5× decoding speedup.

Limitations

Model Scale. Due to limited computational resources, our experiments are restricted to models up to 8B parameters. Evaluating Fier on larger models (e.g., 13B, 70B) may reveal further insights into its scalability and effectiveness.

System Optimization. Our current implementation uses Triton to develop low-bit operators for quantized attention. While Triton offers flexibility and ease of development, it does not match the low-level optimization potential of custom CUDA kernels, potentially limiting the achievable inference speedup.

Compatibility with GQA. Fier is not yet integrated with grouped-query attention (GQA) (Ainslie et al., 2023). This is because token pruning and grouped-query attention are orthogonal in principle: GQA reduces the number of KV heads, while token pruning reduces the number of tokens. Exploring their compatibility remains an important direction for future work.

Acknowledgments

This work was based upon High Performance Computing (HPC) resources supported by the University of Arizona TRIF, UITS, and Research, Innovation, and Impact (RII) and maintained by the UArizona Research Technologies department and the computational resource supported by TetraMem Inc. Partial support for this work was provided by NSF grants CNS-2122320, CNS-2133267, and CNS-2328972. Additional support was received through the Amazon Research Award, Cisco Faculty Award, UNC Accelerating AI Awards, NAIRR Pilot Award, OpenAI Researcher Access Award, and the Gemma Academic Program GCP Credit Award.

References

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Xiao Wen. 2024. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.

Renze Chen, Zhuofeng Wang, Beiquan Cao, Tong Wu, Size Zheng, Xiuhong Li, Xuechao Wei, Shengen Yan, Meng Li, and Yun Liang. 2024a. Arkvale: Efficient generative llm inference with recallable key-value eviction. *Advances in Neural Information Processing Systems*, 37:113134–113155.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and 1 others. 2024b. Magicpig: Lsh sampling for efficient llm generation. *arXiv preprint arXiv:2410.16179*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. 2024. Qaq: Quality adaptive quantization for llm kv cache. *arXiv preprint arXiv:2403.04643*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong

Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, and 1 others. 2025. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023. How long can context length of open-source llms truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

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

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*.

OpenAI. 2024. Learning to reason with large language models. https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2025-05-10.

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

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. 2023. Sparq attention: Bandwidth-efficient llm inference. *arXiv preprint arXiv:2312.04985*.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context llm inference, 2024. *URL https://arxiv.org/abs/2406.10774*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Daniel Waddington, Juan Colmenares, Jilong Kuang, and Fengguang Song. 2013. Kv-cache: A scalable high-performance web-object cache for manycore. In *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 123–130. IEEE.

Yuetian Weng, Mingfei Han, Haoyu He, Xiaojun Chang, and Bohan Zhuang. 2024. Longvlm: Efficient long video understanding via large language models. In *European Conference on Computer Vision*, pages 453–470. Springer.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, and 1 others. 2025. Qwen2. 5-1m technical report. *arXiv preprint arXiv:2501.15383*.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. 2024. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. 2025. Pqcache: Product quantization-based kvcache for long context llm inference. *Proceedings of the ACM on Management of Data*, 3(3):1–30.

Tianyi Zhang, Jonah Yi, Zhaozhuo Xu, and Anshumali Shrivastava. 2024. Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization. *Advances in Neural Information Processing Systems*, 37:3304–3331.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

A Dataset Details

We use a subset of LongBench (Bai et al., 2023) for long-context QA evaluation. Tab. 5 shows the statistics and evaluation metrics used in the experiments.

Table 5: Dataset Statistics and Evaluation Metrics

Dataset	Avg len	Metric	#data
NarrativeQA	18,409	F1	200
Qasper	3,619	F1	200
MultiFieldQA-en	4,559	F1	150
HotpotQA	9,151	F1	200
GovReport	8,734	Rouge-L	200
TriviaQA	8,209	F1	200

B Implementation Details

For experiments on LongBench (Bai et al., 2023) and PG19 (Rae et al., 2019), we use three models: LLaMA-3-8B-Instruct (Grattafiori et al., 2024), LongChat-v1.5-7B-32k (Li et al., 2023), and Mistral-7B-Instruct (Jiang et al., 2023), with the maximum input length uniformly set to 32k tokens to ensure a fair comparison. All inference runs are conducted on NVIDIA A6000 GPUs. During the self-attention phase, we utilize FlashAttention (Dao, 2023) for acceleration, except for H2O (Zhang et al., 2023), which requires computation of historical attention scores and therefore cannot benefit from FlashAttention.

Except for H2O (Zhang et al., 2023) and Quest (Tang et al., 2024), which are run using their publicly released implementations, all other baselines are run using the unified KV Cache-Factory framework (Cai et al., 2024). Experiments on observations and decoding latency are conducted separately on NVIDIA RTX 4090 GPUs.

C Comparison of Fier and MiKV

Different from the methods discussed in the main text, MiKV (Yang et al., 2024) adopts a hybrid strategy: during decoding, it stores top tokens in high precision while retaining the remaining tokens in a low-bit format. The main distinction between Fier and MiKV is that MiKV maintains the entire KV cache in mixed precision, whereas Fier only preserves top tokens during decoding. To ensure fairness, we evaluated Fier on the MMLU dataset (Hendrycks et al., 2020) using the same cache size as reported for MiKV. Note that the cache size of Fier accounts for the CAR in the selection phase, which is not explicitly stated in the MiKV. As shown in Tab. 6, Fier demonstrates better performance compared to MiKV under the same cache budget.

Table 6: Comparison of Fier and MiKV on MMLU.

Cache Size (%)	Method	Acc. (%)
25	MiKV	43.9
25	Fier	**45.12**
20	MiKV	42.7
20	Fier	**43.75**

D Latency breakdown comparison with Quest

We provide a detailed latency breakdown between the pipelines of Fier and Quest in Tab. 7. The primary difference lies in the importance estimation stage: Fier uses an extreme 1-bit quantized attention, while Quest adopts page-level attention. In the top-k recall stage: Quest retrieves pages, while Fier directly retrieves top-k tokens. The final top-k self-attention is identical for both methods. In Tab. 7, we compare the average per-layer latency for each token, as well as the end-to-end generation latency. With the same k, Fier has higher latency than Quest due to its fine-grained top-k sorting in the recall stage. However, Fier achieves better performance. By reducing k , Fier can achieve lower latency than Quest while still maintaining better performance. Overall, Fier offers a better trade-off between latency and performance compared to Quest.

E Comparison of Chatbot Responses

To better illustrate the practical differences between the two retrieval methods, we deploy a LLaMA-3-8B-Instruct chatbot using Fier and Quest, respectively. Given the same long context and user question, we present the corresponding responses from each chatbot for a qualitative comparison (Fig. 9). In the first example, which asks about the orders in which Mufti-e-Azam-e-Hind received Khilafat, the Fier-enabled chatbot

Table 7: Time breakdown of Fier and Quest with different Top-k.

Stage	Fier ($k = 2048$)	Fier ($k = 512$)	Quest ($k = 2048$)
Importance Estimation	16μs	11μs	14μs
Top-k Recall	41μs	25μs	23μs
Top-k Self-Attn	68μs	45μs	68μs
End-to-End Latency	20.5ms	**16.7ms**	18.3ms
GovQA Perf.	**34.42**	33.98	33.7

Figure 9: Chatbot responses from Fier and Quest. Fier provides more complete and accurate answers in both examples.

correctly identifies all five orders, while the Quest-enabled chatbot only retrieves a single name, missing key information. A consistent trend is observed in the second example, which involves a scholarly article. When asked about the generative model adopted in the paper, the Fier-based chatbot accurately identifies the overall framework, whereas the Quest-based chatbot focuses narrowly on a sub-module mentioned in a later section.