A^{2}ATS: Retrieval-Based KV Cache Reduction via Windowed Rotary Position Embedding and Query-Aware Vector Quantization

Anonymous ACL submission

Abstract

Long context large language models (LLMs) pose significant challenges for efficient serving due to the large memory footprint and high access overhead of KV cache. Retrieval-based KV cache reduction methods can mitigate these challenges, typically by offloading the complete KV cache to CPU and retrieving necessary tokens on demand during inference. However, these methods still suffer from unsatisfactory accuracy degradation and extra retrieval overhead. To address these limitations, this paper proposes A^{2}ATS, a novel retrieval-based KV cache reduction method. A^{2}ATS aims to obtain an accurate approximation of attention scores by applying the vector quantization technique to key states, thereby enabling efficient and precise retrieval of the top-K tokens. First, we propose Windowed Rotary Position Embedding, which decouples the positional dependency from query and key states after position embedding. Then, we propose query-aware vector quantization that optimizes the objective of attention score approximation directly. Finally, we design the heterogeneous inference architecture for KV cache offloading, enabling long context serving with larger batch sizes. Experimental results demonstrate that A^{2}ATS can achieve a lower performance degradation with similar or lower overhead compared to existing methods, thereby increasing long context serving throughput by up to 2.7×.

1 Introduction

Large language models (LLMs) with long context windows (OpenAI, 2023; Reid et al., 2024; Dubey et al., 2024; Jiang et al., 2024; Yang et al., 2024a; DeepSeek-AI et al., 2024) are driving advancements in AI applications. However, these models pose significant challenges for efficient serving. Their Transformer-based (Vaswani et al., 2017) architecture generates and maintains a Key-Value (KV) cache during inference to store intermediate results and avoid re-computation. As the context length increases, the size of the KV cache grows proportionally, leading to severe overheads. First, the size of KV cache accessed when generating each token increases, resulting in a GPU memory bandwidth bottleneck. Moreover, the large KV cache size of each request limits the maximum feasible batch size, resulting in suboptimal GPU utilization.

Various methods were proposed to address these challenges from different perspectives. Quantization-based methods (Liu et al., 2024b; Hooper et al., 2024) compress KV cache by using lower bit-width representations for KV cache elements. Eviction-based methods (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024; Yang et al., 2024b) reduce the KV cache size by directly evicting unimportant tokens from memory. Retrieval-based methods (Tang et al., 2024; Singhania et al., 2024; Zhang et al., 2024; Liu et al., 2024a; Chen et al., 2024) offload the complete KV cache to CPU memory and retrieve necessary tokens on demand during inference. However, these methods still face challenges of limited compression ratio, unsatisfactory accuracy degradation, or extra retrieval overhead.

To address the above limitations, this paper proposes A^{2}ATS, a novel retrieval-based KV cache reduction method. A^{2}ATS aims to obtain an **A**ccurate **A**pproximation of **AT**tention **S**cores by applying vector quantization technique to key states, thereby enabling efficient and precise retrieval of the top-K tokens. In order to achieve this goal, we face two main challenges. First, the position-dependent nature of key states after applying position embedding hinders the direct application of shared codebooks across varying inputs. Second, directly utilizing the conventional vector quantization fails to guarantee an accurate approximation of attention scores. To overcome these challenges, the main contributions in this paper are as follows:

- We observe high inter-input similarities be-

tween codebooks of key states before position embedding, and the objective misalignment between vector quantization and attention score approximation by experimental and theoretical analysis;

- We propose Windowed Rotary Position Embedding to decouple the positional dependency from query and key states after position embedding,
- We propose and query-aware vector quantization that directly optimizes the objective of attention score approximation;
- We design the heterogeneous inference system for KV cache offloading, enabling long context serving with larger batch sizes.

Experimental results demonstrate that A^{2}ATS can achieve a low accuracy degradation of 2.2 on Llama-3.1-8B and 0.4 on Mistral-7B while accessing only 6% of the entire KV cache, thereby increasing long context serving throughput by up to 2.7×. Our source code is publicly available [1].

2 Preliminaries

2.1 Self-Attention Modules and Rotary Position Embedding

Self-attention modules (Vaswani et al., 2017) and Rotary Position Embedding (RoPE) (Su et al., 2021) have become the de facto standard components of state-of-the-art (SOTA) LLMs (Dubey et al., 2024; Yang et al., 2024a; Jiang et al., 2024; DeepSeek-AI et al., 2024).

In the self-attention module, during decoding phase, the inference process begins by linearly projecting the input states of the i-th token into query (q_i), key (k_i), and value (v_i) states, where $q_i, k_i, v_i \in \mathbb{R}^{1\times d}$, and d denotes the number of channels or hidden dimensions per head. To enable the model to effectively capture the positional relationships between tokens, position embeddings are then applied to the query and key states. These hidden states before and after this transformation are abbreviated as pre-PE and post-PE states, respectively.

RoPE is a commonly used position embedding in SOTA LLMs. Specifically, for the i-th token, a position-dependent rotation matrix $R_i \in \mathbb{R}^{d\times d}$ is applied to the query q_i and key k_i, to obtain their post-PE counterparts, denoted by $\tilde{q}_i$ and $\tilde{k}_i$:

$$\tilde{q}_i = q_i R_i, \quad \tilde{k}_i = k_i R_i \tag{1}$$

Then the matrices of KV cache of the context can be denoted by $\tilde{K} = [\tilde{k}_1; \tilde{k}_2; \ldots; \tilde{k}_n] \in \mathbb{R}^{n\times d}$ and $V = [v_1; v_2; \ldots; v_n] \in \mathbb{R}^{n\times d}$ respectively, where n denotes the context length. Next, these post-PE states are used to compute the output state o_i as shown in formula (2):

$$o_i = \text{Softmax}\left(\frac{\tilde{q}_i \tilde{K}^\top}{\sqrt{d}}\right) V = \text{Softmax}\left(\frac{u_i}{\sqrt{d}}\right) V \tag{2}$$

where $u_i = \tilde{q}_i \tilde{K}^\top \in \mathbb{R}^{1\times n}$ denotes the attention scores before softmax.

Due to the inherent property of rotation matrices that $R_i R_j^\top = R_{i-j}$ (Su et al., 2021), the attention score $u_{i,j}$ between the i-th query and j-th key can be expressed as:

$$\begin{aligned} u_{i,j} = \tilde{q}_i \tilde{k}_j^\top = q_i R_i (k_j R_j)^\top &= q_i R_i R_j^\top k_j^\top \\ &= q_i R_{i-j} k_j^\top \end{aligned} \tag{3}$$

This equation illustrates how RoPE encodes the relative position $(i - j)$ directly into the attention scores, allowing the model to effectively capture the positional relationships between tokens.

2.2 Vector Quantization for Efficient Attention Score Approximation

Vector quantization (Buzo et al., 1980) is a data compression technique that maps input vectors to a finite set of codewords from a learned codebook.

Formally, given an input space $X \subseteq \mathbb{R}^{1\times d}$ with data distribution $\mathcal{D}$, vector quantization aims to construct a codebook $C = \{c_1, c_2, \ldots, c_L\} \subset \mathbb{R}^{1\times d}$ with a size of L codewords to minimize the following objective:

$$J(C) = \mathbb{E}_{x\sim\mathcal{D}}[\|x - \hat{x}\|^2] \tag{4}$$

where $x \in \mathbb{R}^{1\times d}$ denotes the input vector, $\hat{x} = c_{f(x;C)}$ denotes the quantized vector, and $f(x; C)$ denotes the quantization function that maps x to its nearest codeword:

$$f(x; C) = \underset{j}{\text{argmin}} \|x - c_j\|^2 \tag{5}$$

Finding the optimal codebook C is computationally expensive. Therefore, approximate algorithms such as LBG and k-means++ (Linde et al., 1980; Arthur and Vassilvitskii, 2007) are commonly used to find a suboptimal but effective codebook.

[1] https://anonymous.4open.science/r/4987d19d-f5cd-4f14-910d-c7141ce37f13/

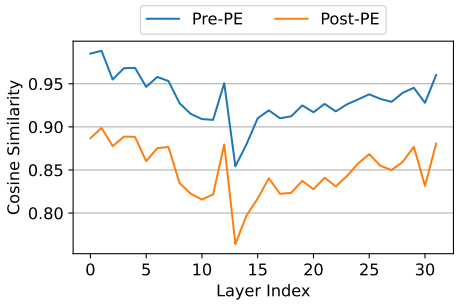
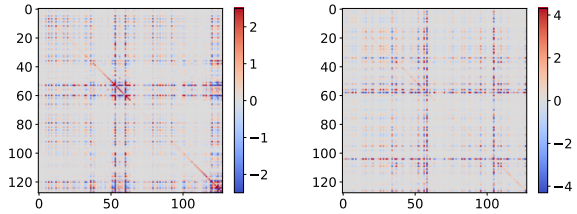


Figure 1: Inter-sample cosine similarities of pre-PE and post-PE codebooks.



(a) Layer 16, Head 8 (b) Layer 32, Head 8

Figure 2: Visualization of second-moment matrices H of post-PE query states. Each pixel represents an element in H. Warmer colors correspond to higher values, while cooler colors correspond to lower values.

After obtaining the codebook, vector quantization compresses an input x by replacing the original vector with its index $s = f(x; C)$. Since the storage requirement for the index is substantially lower than that of the original vector, vector quantization achieves significant data compression ratio.

Multiple studies (Lingle, 2024; Zhang et al., 2024; Liu et al., 2024a) have investigated applying vector quantization to post-PE key states of LLMs to efficiently approximate attention scores. Let $s \in \{1, 2, \ldots, L\}^{1\times n}$ denotes the codeword index vector of all post-PE key states, where the length of this vector is n, and each element $s_i \in \{1, 2, \ldots, L\}$ denotes the codeword index of the i-th key state. Then, the $\tilde{k}_i$ can be quantized as $\hat{k}_i = c_{s_i}$, and the attention score $u_{i,j}$ can be approximated as:

$$\hat{u}_{i,j} = \tilde{q}_i \hat{k}_j^\top = \tilde{q}_i c_{s_j}^\top \tag{6}$$

This equation illustrates the approximation of attention scores without the memory-intensive access to the $\tilde{k}_j$.

3 Motivation

3.1 Inter-Input Similarity of Codebooks

PQCache (Zhang et al., 2024) and ClusterKV (Liu et al., 2024a) propose applying vector quantization to post-PE key states, with individual codebooks constructed for each input during the prefilling phase. However, constructing codebooks during inference requires iterative access to the entire KV cache, incurring high memory access overhead. To address this limitation, we investigate the feasibility of employing shared codebooks for all inputs based on the inter-input similarity of codebooks.

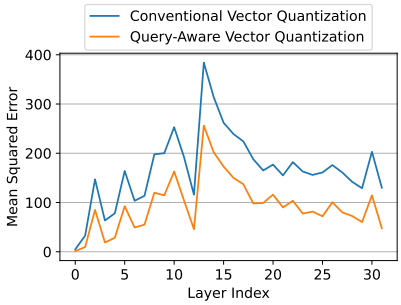


Figure 3: A comparison of MSE of attention score approximation between conventional vector quantization and query-aware vector quantization.

To quantify the similarity between codebooks C_1 and C_2, we define the cosine similarity between codebooks as metric, which can be formulated as:

$$\begin{aligned}\text{sim}(C_1, C_2) = &\frac{1}{2L}\sum_{i=1}^{L}\max_{j\in\{1,2,\ldots,L\}}\cos(c_{1i}, c_{2j}) + \\ &\frac{1}{2L}\sum_{i=1}^{L}\max_{j\in\{1,2,\ldots,L\}}\cos(c_{2i}, c_{1j})\end{aligned} \tag{7}$$

where $c_{1i}, c_{1j} \in C_1, c_{2i}, c_{2j} \in C_2$. This metric computes the average maximum cosine similarity from each codeword in one codebook to any codeword in the other codebook. A cosine similarity score closer to 1 indicates higher similarity between codebooks, while a score closer to 0 or negative values indicates lower similarity.

We utilize the Llama-3.1-8B-Instruct model (Dubey et al., 2024), and two random samples from the FineWeb dataset (Penedo et al., 2024) with a context length of approximately 32k tokens each for experiments. We collect pre-PE and post-PE (RoPE is used in here) key states from all attention heads on both samples, then employ k-means++ algorithm (Arthur and Vassilvitskii, 2007) to generate codebooks with a size of 4096 codewords for each set of key states, and calculate the inter-sample cosine similarities of pre-PE and post-PE codebooks using Equation (7). The similarity generated by pre-PE and

post-PE codebooks are shown in Figure 1. From the experimental results, we derive the following key observations:

Observation 1: High inter-input similarities of pre-PE codebooks suggest the potential for using shared codebooks to effectively approximate key states across various inputs. The cosine similarities for pre-PE codebooks remain remarkably high, exceeding 0.9 for the majority of layers, indicate a very strong similarity across codebooks of various inputs. This finding indicates that the semantic information in key states might be broadly similar.

Observation 2: The position-dependent nature of post-PE key states hinders the direct application of a shared codebook. The post-PE codebook similarities are consistently lower, fluctuating around 0.85, with some layers below 0.8, indicating weaker similarity across inputs. The reason is that RoPE causes semantically similar key states at different positions to have different representations. The position-dependent nature of post-PE key states makes it difficult to construct a single codebook that can effectively quantize representations with all semantic information at all possible positions. This representation divergence becomes a key challenge for directly applying a shared codebook on post-PE key states, necessitating the development of a novel vector-quantization-compatible position embedding method.

3.2 Objective Misalignment of Vector Quantization

The optimization objective of vector quantization is to minimize the mean squared error (MSE) of the approximate key states, while attention score approximation focuses on minimizing the MSE of attention scores. This discrepancy between their optimization objectives raises a fundamental question:

Does the optimal codebook for vector quantization necessarily yield the most accurate approximation of attention scores?

Formally, let C denote the codebook constructed by vector quantization, $J(C)$ denote the MSE of vector quantization with C, and $J'(C)$ denote the MSE of attention scores approximation with C. We need to investigate whether the following equation holds:

$$\operatorname*{argmin}_C J(C) \equiv \operatorname*{argmin}_C J'(C) \tag{8}$$

To address this question, we analyze the relationship between J and J'.

For vector quantization, the objective J can be reformulated as:

$$\begin{aligned} J(C) &= \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}}}[\|\tilde{k}-\hat{k}\|^2] \\ &= \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}}}[(\tilde{k}-\hat{k})(\tilde{k}-\hat{k})^\top] \end{aligned} \tag{9}$$

where $\mathcal{D}^{\text{key}}$ denotes the distribution of post-PE key states, and $\hat{k}$ denotes quantized key.

For attention score approximation, the objective J' is formualted as:

$$\begin{aligned} J'(C) &= \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}},\ \tilde{q}\sim\mathcal{D}^{\text{query}}}[(\tilde{q}\tilde{k}^\top-\tilde{q}\hat{k}^\top)^2] \\ &= \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}},\ \tilde{q}\sim\mathcal{D}^{\text{query}}}[(\tilde{q}(\tilde{k}-\hat{k})^\top)^2] \\ &= \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}},\ \tilde{q}\sim\mathcal{D}^{\text{query}}}[(\tilde{k}-\hat{k})\tilde{q}^\top\tilde{q}(\tilde{k}-\hat{k})^\top] \\ &= \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}}}[(\tilde{k}-\hat{k})H(\tilde{k}-\hat{k})^\top] \end{aligned} \tag{10}$$

where $\mathcal{D}^{\text{query}}$ denotes the distribution of post-PE query states, $H = \mathbb{E}_{\tilde{q}\sim\mathcal{D}^{\text{query}}}[\tilde{q}^\top\tilde{q}] \in \mathbb{R}^{d\times d}$ denotes the second-moment matrix of query states.

The two objectives only align if the query-dependent H is proportional to the identity matrix. To examine this consistency, we visualize H on a set of input samples, as depicted in Figure 2. The visualization shows that H is not proportional to the identity matrix, displaying a non-uniform and non-diagonal structure. This result reveals a fundamental misalignment between the vector quantization objective J and the attention score approximation objective J', suggesting potential inaccuracy in attention score approximation.

To validate the impact of this objective mismatch, we compare conventional vector quantization that minimizes objective J, with its query-aware variant that directly minimizes objective J' (implementation details in Section 4.2) in Figure 3. From experimental results, the query-aware method consistently achieves lower squared error in attention score approximation. Thus, we derive the following observation:

Observservation 3: Optimizing vector quantization alone fails to guarantee accurate approrximation of attention scores due to objective misalignment, necessitating query-aware vector quantization for bridging this objective gap.

4 Method

In A^{2}ATS, there are two stages. During the offline pre-processing stage (1st stage), A^{2}ATS constructs

a shared codebook on a representative dataset for each attention head of each layer.

During the inference stage (2nd stage), A^{2}ATS applies quantization functions to key states to map them to the nearest codewords. At each autoregressive decoding step, A^{2}ATS first utilizes codebooks and codeword indices to approximate attention scores, then retrieves the top-K tokens with the highest attention scores for computation, thereby mitigating the memory overhead of accessing the entire KV cache.

4.1 Windowed Rotary Position Embedding

As discussed in Section 3.1, the position-dependent nature of post-PE key states hinders the direct application of a shared codebook in the vector quantization process. A seemingly straightforward solution would be to quantize pre-PE key states, and then incorporate RoPE when approximating attention scores. However, this approach is computationally expensive, as it necessitates calculating and applying the rotary matrices for each token at each inference.

To overcome this inefficiency, while eliminating the inherent position-dependent nature of post-PE key states, we propose Windowed Rotary Position Embedding (WRoPE). This approach builds on the findings by Su (2023b,a) that transformer-based models are nonsensitive to the positional information of non-local tokens. The core idea of WRoPE is to use standard RoPE for local tokens (i.e. those in the window) and use approximate positional information for non-local tokens (i.e. those not in the window). Specifically, WRoPE computes the attention scores as follows:

$$u_{i,j} = \begin{cases} q_i R_{i-j} k_j^\top, & i-j < w \\ q_i R_b k_j^\top, & i-j \geq w \end{cases} \tag{11}$$

where w is the window size, acting as a threshold for local vs. non-local tokens, and b is a constant value representing a fixed relative position approximation for non-local tokens.

For local tokens (i.e., $i-j < w$), WRoPE functions identically to standard RoPE, as defined in Equation 3. For non-local tokens outside the window (i.e., $i-j \geq w$), the position-dependent rotation matrix R_{i-j} is replaced by a fixed rotation matrix R_b, approximating the relative positional information $(i-j)$ with a constant offset b. Then, we can calculate the post-PE query and key states as:

$$\tilde{q}_i = q_i R_b, \quad \tilde{k}_i = k_i \tag{12}$$

Since post-PE key $\tilde{k}_i$ states are identical to their pre-PE counterparts k_i, WRoPE decouples the positional dependency from post-PE representations, therefore optimizes subsequent vector quantization.

4.2 Query-Aware Vector Quantization

As discussed in Section 3.2, conventional vector quantization fails to achieve accurate approximation of attention scores, due to the objective misalignment between vector quantization and attention score approximation.

To address this limitation, **we propose query-aware vector quantization, a custom vector quantization method that directly optimizes the objective of attention score approximation.** Specifically, we replace the squared Euclidean distance $\|\tilde{k}-\hat{k}\|^2$ of conventional vector quantization with a query-aware quadratic form $(\tilde{k}-\hat{k})H(\tilde{k}-\hat{k})^\top$ derived from formula (10), where H represents the second-moment matrix of query states.

Formally, the query-aware vector quantization minimizes the following objective:

$$J'(C) = \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}}}\left[(\tilde{k}-\hat{k})H(\tilde{k}-\hat{k})^\top\right] \tag{13}$$

where $\hat{k} = c_{f'(\tilde{k};C)}$ denotes the quantized $\tilde{k}$, and the corresponding query-aware quantization vector quantization is formulated as:

$$f'(\tilde{k};C) = \underset{j}{\text{argmin}}(\tilde{k}-c_j)H(\tilde{k}-c_j)^\top \tag{14}$$

For the codebook construction process, we reformulate the objective function to utilize conventional efficient vector quantization algorithms like k-means++ (Arthur and Vassilvitskii, 2007). Specifically, we apply Cholesky decomposition to the positive definite matrix $H = LL^T$, where $L \in \mathbb{R}^{d\times d}$ denotes the Cholesky factor. Let

$$z = \tilde{k}L, \quad C^z = CL, \quad \hat{z} = \hat{k}L \tag{15}$$

where $z \in \mathbb{R}^{1\times d}$ denotes the transformed key state. Then, we can re-derive the objective of attention score approximation J' as:

$$\begin{aligned} J'(C) &= \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}}}\left[(\tilde{k}-\hat{k})H(\tilde{k}-\hat{k})^\top\right] \\ &= \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}}}\left[(\tilde{k}-\hat{k})LL^\top(\tilde{k}-\hat{k})^\top\right] \\ &= \mathbb{E}_{\tilde{k}\sim\mathcal{D}^{\text{key}}}\left[(\tilde{k}L-\hat{k}L)(\tilde{k}L-\hat{k}L)^\top\right] \\ &= \mathbb{E}_{z\sim D^z}\left[(z-\hat{z})(z-\hat{z})^\top\right] \end{aligned} \tag{16}$$

And the quantization function f' can be re-derived as:

$$\begin{aligned} f'(\tilde{k}; C) &= \underset{j}{\operatorname{argmin}}(\tilde{k} - c_j)H(\tilde{k} - c_j)^\top \\ &= \underset{j}{\operatorname{argmin}}(\tilde{k}L - c_j L)(\tilde{k}L - c_j L)^\top \\ &= \underset{j}{\operatorname{argmin}}(z - c_j^z)(z - c_j^z)^\top \\ &= f(z; C^z) \end{aligned} \tag{17}$$

where $f(z; C^z)$ denotes the quantization function of conventional vector quantization. Then, we can derive that

$$\hat{z} = \hat{k}L = c_{f'(\tilde{k};C)}L = c^z_{f(z;C^z)} \tag{18}$$

Equations 16, 18 reveal that **the objective of attention score approximation is equivilant to that of conventional vector quantization on transformed** z. This alignment enables the application of conventional efficient vector quantization algorithms in codebook construction process.

During the offline pre-processing stage, we collect z on a representative dataset and construct its codebook C^z using k-means++ (Arthur and Vassilvitskii, 2007). Then, The original shared codebook C for $\tilde{k}$ is calculated as:

$$C = C^z L^{-1} \tag{19}$$

During inference, the codeword index of $\tilde{k}$ is computed through query-aware quantization function:

$$f'(\tilde{k}; C) = \underset{j}{\operatorname{argmin}}(\tilde{k}L - c_j L)(\tilde{k}L - c_j L)^\top \tag{20}$$

Let $s \in \{1, 2, \ldots, L\}^{1\times n}$ denotes the codeword index vector of all key states after applying query-aware vector quantization, where $s_j = f'(\tilde{k}_j; C)$. Then, the attention score is approximated as:

$$\hat{u}_{i,j} = \tilde{q}_i \hat{k}_j = \tilde{q}_i c_{s_j} \tag{21}$$

4.3 Heterogeneous Inference Design

Although approximating attention scores via vector quantization and then selectively retrieving top-K tokens for computation reduces memory access overhead, the issue of KV Cache occupying substantial GPU memory remains unresolved. To reduce the memory footprint of KV Cache and enable larger batch sizes for improved GPU utilization, we design a heterogeneous inference system.

We partition the decoding process of our proposed A^{2}ATS into three components:

(1) **GPU-based model execution**: All model weights reside on the GPU memory. Computations involving model weights are executed on the GPU during inference.

(2) **GPU-based approximation of attention scores**: The codebook is stored on the GPU. During inference, the GPU first executes the quantization function to assign codewords to key states, then computes attention weight approximations using the codebooks and indices, and lastly gathers the indices of top-K tokens.

(3) **CPU-based selective attention**: The full KV Cache is maintained on the CPU memory. During decoding, the top-K token indices and the current query state are transferred to the CPU, where selective attention computation is performed to derive the attention output. This output is then transferred back to the GPU for subsequent computations.

This design aims to minimize data transfer between CPU and GPU, thereby reducing latency. Furthermore, to fully leverage the CPU's thread-level parallelism and SIMD capabilities, we implement a custom selective attention kernel optimized for CPU execution.

5 Experiments

5.1 Experimental Setup

Tasks. We utilize RULER (Hsieh et al., 2024) as our benchmark for downstream tasks evaluation. This synthetic benchmark contains thirteen subtasks organized into four categories: information retrieval, multi-hop tracing, information aggregation, and question answering. It evaluates long-context comprehension and reasoning capabilities of LLMs, while effectively revealing the accuracy drop caused by KV cache reduction methods.

Models. We conduct our main experiments on Llama-3.1-8B-Instruct (Dubey et al., 2024) and MegaBeam-Mistral-7B-512k (Wu et al., 2024). These models feature long-context processing capabilities with context windows of up to 128K and 512K tokens, respectively. As for the ablation study and end-to-end throughput evaluation, we apply the Llama-3.1-8B-Instruct model.

Methods. For main experiments, we compare the proposed A^{2}ATS with the following four KV cache reduction methods, along with the full attention baseline: H2O (Zhang et al., 2023), SnapKV (Li et al., 2024), Quest (Tang et al., 2024), MagicPIG (Chen et al., 2024). For a fair comparison, the sparsity ratios of all KV cache reduction meth-

ods are controlled around 0.06. Detailed discussions are presented in Appendix A.

For ablation studies, we evaluate the following configurations on Llama-3.1-8B-Instruct:

- **Baseline**: It utilizes standard RoPE and conventional vector quantization.
- **WRoPE**: It utilizes WRoPE and conventional vector quantization.
- **QAVQ**: It utilizes standard RoPE and query-aware vector quantization.
- **A^{2}ATS**: The proposed method with WRoPE and query-aware vector quantization.

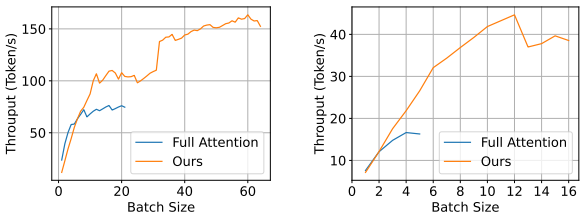
Implementation Details. The hyper-parameters w and b of WRoPE are set to 64 and 2048, respectively. The codebooks of query-aware vector quantization, with a size of 4096 each, are constructed from a set of sample inputs consisting of approximately 64K tokens of text randomly sampled from FineWeb (Penedo et al., 2024) and 16K tokens of randomly generated uuid strings. The end-to-end speedup experiments are conducted on a server equipped with an NVIDIA H800 GPU with 80GB memory, and an Intel Xeon Platinum 8469C CPU.

5.2 Main Results on Downstream Tasks

Table 1 compares the accuracies of different methods on RULER, along with attention sparsity ratios and auxiliary memory overhead. Experimental results draw the following conclusions:

A^{2}ATS minimizes accuracy degradation under comparable sparsity ratios. For Llama models, A^{2}ATS achieves an average accuracy of 86.6, outperforming H2O (27.0), SnapKV (72.7), Quest (80.7), and MagicPIG (85.7). For Mistral models, A^{2}ATS achieves an average accuracy of 86.3, surpassing H2O (22.3), SnapKV (67.6), Quest (78.4), and MagicPIG (84.6). Notably, A^{2}ATS achieves these accuracies with comparable sparsity ratios (0.060 for Llama, 0.062 for Mistral) to H2O, SnapKV and Quest, lower than those of MagicPIG (0.068 for Llama, 0.064 for Mistral). These results demonstrate the superior accuracy preservation of A^{2}ATS.

A^{2}ATS causes comparable or lower auxiliary memory overhead compared to existing methods. A^{2}ATS causes auxiliary memory usage (0.008) identical to eviction-based methods, i.e., H2O and SnapKV, while being significantly more memory efficient than retrieval-based approaches, i.e., Quest (0.031) and MagicPIG (2.344). This efficiency comes from the inherent nature of vector quantization, requiring only one codeword index for each token in each attention head, eliminating the need for storing either per-page metadata (Quest) or large LSH tables (MagicPIG).



(a) Inference throughput with a context length of 16K.

(b) Inference throughput with a context length of 64K.

Figure 4: End-to-end inference throughput of Llama-3.1-8B-Instruct across varying context lengths and batch sizes.

5.3 Ablation Study

Table 2 validates the effectiveness of WRoPE and query-aware vector quantization on improving model accuracy. Experimental results draw the following conclusions: (1) WRoPE is fundamental to attention score approximation using shared codebooks. (2) Query-aware vector quantization provides a further improvement in model accuracy by aligning the objectives of vector quantization and attention score approximation. Detailed discussions are presented in Appendix B.

5.4 End-to-End Inference Speedup

Figure 4 compares the inference throughput of the proposed A^{2}ATS against full attention on Llama-3.1-8B-Instruct. At a context length of 16K, A^{2}ATS initially exhibits marginally lower throughput than full attention for small batch sizes (≤ 5), primarily due to CPU-GPU data transfer overhead. As batch sizes increase, the throughput of A^{2}ATS grows linearly to exceed 100 tokens/s, while full attention plateaus below 80 tokens/s due to memory bandwidth bottleneck and ends up out of memory at a batch size of 22. A^{2}ATS achieves a peak throughput of over 160 tokens/s, a 2.1× speedup over full attention, with a maximum batch size of over 64. This trend becomes more pronounced at 64K context lengths, where full attention struggles with batches over 5, while A^{2}ATS serving a batch size of up to 16 with a throughput of up to 45 tokens/s, delivering a 2.7× performance advantage. These results highlight A^{2}ATS's potential in mitigating the memory bottleneck in long context LLM serving.

Models	Sparsity↓	Aux Mem↓	Accuracy↑				
			16K	32K	64K	96K	Average
Llama-3.1-8B-Instruct	1.000	0.000	94.4	91.9	85.9	83.1	88.8
H2O	0.060	0.008	27.6	30.6	24.9	25.0	27.0
SnapKV	0.060	0.008	72.7	75.1	72.2	70.7	72.7
Quest	0.060	0.031	84.3	84.0	80.0	74.4	80.7
MagicPIG	0.068	2.344	**92.3**	87.6	83.9	79.1	85.7
A^2ATS	0.060	0.008	92.2	**90.4**	**84.3**	**79.6**	**86.6**
MegaBeam-Mistral-7B-512K	1.000	0.000	91.8	88.2	83.3	83.4	86.7
H2O	0.060	0.008	22.5	23.4	20.7	22.6	22.3
SnapKV	0.060	0.008	69.3	68.5	69.5	65.2	67.6
Quest	0.060	0.031	81.5	80.8	76.7	74.4	78.4
MagicPIG	0.064	2.344	88.7	85.2	82.6	81.8	84.6
A^2ATS	0.062	0.008	**91.6**	**88.1**	**83.4**	**82.2**	**86.3**

Table 1: Comparison of sparsity ratio, auxiliary memory usage and accuracy on RULER benchmark. 'Aux Mem' refers to 'Auxialiary Memory Usage', which denotes the extra memory usage caused by KV cache reduction methods compared to the original key cache. '16K', '32K', '64K' and '96K' denote the input context length.

Config	16K↑	32K↑	64K↑	96K↑	Average↑
Baseline	86.4	86.3	81.5	71.3	81.4
WRoPE	**92.3**	90.0	82.8	78.4	85.9
QAVQ	91.7	86.9	76.3	69.4	81.1
A^2ATS	92.2	**90.4**	**84.3**	**79.6**	**86.6**

Table 2: Ablation study on the importance of WRoPE and query-aware vector quantization for accuracy.

6 Related Work

Quantization-based KV cache reduction. This method aims to compress KV cache by using lower bit-width representations for KV cache elements (Liu et al., 2024b; Hooper et al., 2024). However, it typically faces challenges of limited compression ratio and extra computational overhead caused by the dequantization process.

Eviction-based KV cache reduction. This method aims to reduce the KV cache size by directly evicting unimportant tokens from memory (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024; Yang et al., 2024b). These methods typically record statistics of attention weights of each token. When the KV cache reaches its capacity limit, they utilize heuristic rules and historical statistics to predict which tokens are more likely to get high attention weights in future decoding, then retain these tokens while evicting the rest. Although these methods generally have low additional overhead, they often lead to noticeable performance degradation.

Retrieval-based KV cache reduction. This method keeps the entire KV cache in memory while selectively retrieving tokens crucial for the current inference. Quest (Tang et al., 2024) chunks the continuous KV cache into pages and pre-calculates necessary metadata for each page during prefilling. For decoding, it selects the top-K critical cache pages to participate in selective attention computation. PQCache (Zhang et al., 2024) and ClusterKV (Liu et al., 2024a) perform vector quantization on the key states with individual codebooks constructed for each input during prefilling. For decoding, the system uses codewords of the codebooks to approximate attention scores, then retrieves the top-K tokens for computation. MagicPIG (Chen et al., 2024) employs Locality-Sensitive Hashing on query and key states for token retrieval. Although these methods generally achieve lower performance degradation compared to eviction-based methods, they still suffer from unsatisfactory performance and lead to extra overhead.

7 Conclusion

In this paper, we propose A^2ATS, a novel retrieval-based KV cache reduction method. First, we propose Windowed Rotary Position Embedding to decouple the positional dependency from query and key states after position embedding. Then, we propose query-aware vector quantization to achieve an accurate attention score approximation. Next, we introduce the heterogeneous inference design for KV cache offloading which increases available batch size. Experimental results demonstrate that A^2ATS achieves lower performance degradation with comparable or lower overhead compared to existing methods, thereby boosting long context serving throughput by up to $2.7\times$.

8 Limitations

The limitations of this work can be summarized in two main aspects.

First, while A^{2}ATS demonstrates lower accuracy degradation compared to existing methods while accessing a comparable proportion of KV cache, it still exhibits non-negligible performance degradation. This suggests opportunities for future work to investigate adaptive attention sparsity allocation strategies that dynamically optimize the sparsity ratios across layers and attention heads, based on their contextual importance.

Second, while A^{2}ATS increases long context serving throughput by up to 2.7×, our current implementation is limited to single-GPU deployment. Future research could further explore (1) distributed multi-GPU system designs for scaled deployment, (2) integration with disaggregated LLM serving architectures like MoonCake (Qin et al., 2024).

References

David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035. SIAM.

Andres Buzo, Augustine H. Gray Jr., Robert M. Gray, and John D. Markel. 1980. Speech coding based upon vector quantization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '80, Denver, Colorado, USA, April 9-11, 1980*, pages 15–18. IEEE.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Léon Bottou, Zhihao Jia, and Beidi Chen. 2024. Magicpig: LSH sampling for efficient LLM generation. *CoRR*, abs/2410.16179.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, and Wangding Zeng. 2024. Deepseek-v3 technical report. *CoRR*, abs/2412.19437.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. RULER: what's the real context size of your long-context language models? *CoRR*, abs/2404.06654.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bam-

ford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *CoRR*, abs/2401.04088.

Sehoon Kim, Coleman Hooper, Thanakul Wattanawong, Minwoo Kang, Ruohan Yan, Hasan Genc, Grace Dinh, Qijing Huang, Kurt Keutzer, Michael W. Mahoney, Yakun Sophia Shao, and Amir Gholami. 2023. Full stack optimization of transformer inference: a survey. *CoRR*, abs/2302.14017.

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. *CoRR*, abs/2404.14469.

Yoseph Linde, Andres Buzo, and Robert M. Gray. 1980. An algorithm for vector quantizer design. *IEEE Trans. Commun.*, 28(1):84–95.

Lucas D. Lingle. 2024. Transformer-vq: Linear-time transformers via vector quantization. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. 2024a. Clusterkv: Manipulating LLM KV cache in semantic space for recallable compression. *CoRR*, abs/2412.03213.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. 2024. Mooncake: A kvcache-centric disaggregated architecture for LLM serving. *CoRR*, abs/2407.00079.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, James Molloy, Jilin Chen, Michael Isard, Paul Barham, Tom Hennigan, Ross McIlroy, Melvin Johnson, Johan Schalkwyk, Eli Collins, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Clemens Meyer, Gregory Thornton, Zhen Yang, Henryk Michalewski, Zaheer Abbas, Nathan Schucher, Ankesh Anand, Richard Ives, James Keeling, Karel Lenc, Salem Haykal, Siamak Shakeri, Pranav Shyam, Aakanksha Chowdhery, Roman Ring, Stephen Spencer, Eren Sezener, and et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, abs/2403.05530.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. 2024. Loki: Low-rank keys for efficient sparse attention. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Jianlin Su. 2023a. Expand the context length with rope, part 3 – unlocking the unlimited extrapolation potential with rerope. `https://normxu.github.io/Rethinking-Rotary-Position-Embedding-3/`.

Jianlin Su. 2023b. Rectified rotary position embeddings. `https://github.com/bojone/rerope`.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. QUEST: query-aware sparsity for efficient long-context LLM inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Chen Wu, Yin Song, and Eden Duthie. 2024. aws-prototyping/MegaBeam-Mistral-7B-512k.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei

Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024a. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024b. Pyramidinfer: Pyramid KV cache compression for high-throughput LLM inference. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3258–3270. Association for Computational Linguistics.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. 2024. Pqcache: Product quantization-based kvcache for long context LLM inference. *CoRR*, abs/2407.12820.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Method Configurations

For the main experiments, we compare the following five KV cache reduction methods, along with the full attention baseline:

- **H2O** (Zhang et al., 2023): An eviction-based method that preserves heavy hitter tokens and recent tokens;
- **SnapKV** (Li et al., 2024): An eviction-based method that preserves important tokens based on statistics within an observation window;
- **Quest** (Tang et al., 2024): A retrieval-based method that selects tokens based on metadata of KV cache pages;
- **MagicPIG** (Chen et al., 2024): A retrieval-based method that utilizes LSH for token sampling;
- **A^{2}ATS**: The proposed method.

For a fair comparison, the sparsity ratios of all KV cache reduction methods are controlled around 0.06, which means approximately 6% of KV cache is accessed at each inference. It is important to note that the sparsity ratio discussed in this paper differs in definition from the cost_2 in MagicPIG (Chen et al., 2024). Specifically, cost_2 measures the ratio of computation overhead (FLOPs) compared to full attention, whereas our sparsity ratio measures the ratio of memory access overhead (MOPs) relative to full attention. Since attention modules are typically considered memory-bound (Kim et al., 2023), we argue that the latter metric provides more meaningful insights on potential overhead reduction. Additionally, the initial 4 tokens and the most recent 64 tokens are statically preserved to align with MagicPIG (Xiao et al., 2024). The detailed configurations for each methods are shown in Table 3.

Method	Configurations
H2O	`hh_size` = 0.06 × `input_length`
SnapKV	`prompt_capacity` = 0.06 × `input_length`
Quest	`page_size` = 32, `ratio` = 0.06
MagicPIG	K = 10, L = 150
A^{2}ATS	`topk` = 0.03

Table 3: Configurations of KV cache reduction methods.

B Detailed Discussions of Ablation Study

Table 2 validates the effectiveness of WRoPE and query-aware vector quantization on improving model accuracy. Experimental results draw the following conclusions:

WRoPE is fundamental to attention score approximation using shared codebooks. WRoPE achieves an average improvement of +4.5 over the baseline, with consistent gains across all context lengths. This result confirms WRoPE's critical role in preventing representation divergence of key states caused by positional embedding.

Query-aware vector quantization provides a further improvement in model accuracy by aligning the objectives of vector quantization and attention score approximation. Our full method, incorporating query-aware vector quantization, demonstrates further improvements, particularly at longer context lengths (+1.5 at 64K, +1.2 at 96K, respectively). However, query-aware vector quantization alone underperforms the baseline, exhibiting a more significant drop at longer context lengths. This performance degradation suggests that representation divergence is not solely limited to key states but also affects query states, hindering the effectiveness of query-aware vector quantization. With WRoPE mitigating positional dependencies, query-aware vector quantization further optimizes the attention score approximation, achieving state-of-the-art performance.