HeteroCache: A Dynamic Retrieval Approach to Heterogeneous KV Cache Compression for Long-Context LLM Inference

Zhiyuan Shi[1], Qibo Qiu[2,3], Feng Xue[4], Zhonglin Jiang[4], Li Yu[2], Jian Jiang[2], Xiaofei He[3,5], Wenxiao Wang[1,†]
[1]School of Software Technology, Zhejiang University
[2]China Mobile (Zhejiang) Research & Innovation Institute
[3]State Key Lab of CAD&CG, Zhejiang University
[4]Geely Automobile Research Institute (Ningbo) Co., Ltd. [5]FABU Inc.
{zhiyuan0206, wenxiaowang}@zju.edu.cn

Abstract

The linear memory growth of the KV cache poses a significant bottleneck for LLM inference in long-context tasks. Existing static compression methods often fail to preserve globally important information. Although recent dynamic retrieval approaches attempt to address this issue, they typically suffer from coarse-grained caching strategies and incur high I/O overhead. To overcome these limitations, we propose **HeteroCache**, a training-free dynamic compression framework. Our method is built on two key insights: attention heads exhibit diverse temporal heterogeneity, and there is significant spatial redundancy among heads within the same layer. Guided by these insights, HeteroCache categorizes heads based on stability and similarity, applying a fine-grained weighting strategy that allocates larger cache budgets to heads with rapidly shifting attention to capture context changes. Furthermore, it features a hierarchical storage mechanism where representative heads monitor attention drift to trigger asynchronous, on-demand context retrieval, thereby hiding I/O latency. Experiments demonstrate that HeteroCache achieves state-of-the-art performance on long-context benchmarks and accelerates decoding by up to 3× compared to the original model with a 224K context. Our code is available at https://github.com/ponytaill/HeteroCache.

1 Introduction

LLMs have reshaped artificial intelligence, demonstrating powerful capabilities in conversational assistants and autonomous agents (Singh et al., 2025; Yang et al., 2025; Team et al., 2025; Wang et al., 2025). This success depends on the processing of long-form context, urging innovative architectures to extend current limits (DeepSeek-AI et al., 2025; Ye et al., 2025). However, Transformer-based models (Vaswani et al., 2017) rely on an optimization mechanism known as the KV cache during generation. Although this avoids redundant computation, its linear memory consumption makes the KV cache bandwidth a major bottleneck for efficient inference on resource-constrained hardware.

To address this problem, some work highlights the inherent sparsity of the attention mechanism in long contexts, such as MInference (Jiang et al., 2024), which shows that only 4K tokens of 128K can account for 96.4% of the total attention weight. Consequently, a major line of research has focused on static compression, which permanently evicts unimportant token entries from the cache. Methods like SnapKV and PyramidKV (Li et al., 2024; Cai et al., 2025) typically utilize heuristics based on historical attention scores to identify and retain only a critical subset of tokens. However, this irreversible discard policy poses a fundamental risk. Due to attention drift in complex reasoning tasks, information deemed unimportant early on may become pivotal later, leading to accuracy degradation. To mitigate this, recent work such as ShadowKV and OmniKV (Sun et al., 2025; Hao et al., 2025) has introduced dynamic compression, effectively improving performance by recalling the necessary tokens. Nevertheless, these approaches face two primary limitations: first, they often employ coarse-grained caching strategies that overlook the heterogeneity across layers or heads; second, their coarse-grained retrieval at every step incurs unnecessary I/O overhead and potential accuracy drops.

To overcome these limitations, we propose HeteroCache, a dynamic framework that avoids premature information loss. Our approach leverages two key observations: temporal heterogeneity, where some heads maintain long-term stability while others exhibit rapid shifts, and intralayer redundancy, where attention patterns of certain heads are highly similar within the same layer. Guided by these insights, HeteroCache optimizes the compression through a fine-grained, role-based strategy.

[†]Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 43172–43187

First, we employ a profiling strategy that categorizes heads into distinct functional roles. By explicitly modeling their similarity behavior, we cluster highly redundant heads to identify representative ones, subsequently dividing all heads into the full heads that retain complete context and the compressed heads designated for compression. For compressed heads, we implement a fine-grained stability-based strategy that allocates larger cache budgets to heads prone to rapid context shifts. This ensures the precise capture of dynamic information and maintains stable model performance across varying contexts. Second, we ensure efficient inference through a hierarchical storage mechanism with sparse monitoring. Rather than retrieving context at every step, we offload the majority of the KV cache to CPU memory, retaining full context on the GPU only for the full heads. These heads continuously monitor attention drift and trigger asynchronous on-demand retrieval only when a significant attention shift is detected, utilizing data fetched from the CPU to dynamically update the compressed heads. This approach maintains high fidelity with minimal overhead.

Our extensive experiments were conducted on mainstream LLMs, including the standard Llama and Qwen series, as well as the reasoning-specialized DeepSeek-R1-Distill-Llama-8B to assess performance on chain-of-thought tasks. We evaluated these models under various compression rates on multiple long-context benchmarks, such as LongBench (Bai et al., 2024), LongBench v2 (Bai et al., 2025), InfiniteBench (Zhang et al., 2024). The results demonstrate that, compared to other advanced compression algorithms, HeteroCache achieves state-of-the-art performance on a variety of tasks while realizing up to 3× inference acceleration compared to the baseline of the original model with a 224K context.

2 Related Work

KV Cache Eviction. Existing work has demonstrated the inherent sparsity of the attention mechanism in long contexts (Jiang et al., 2024). Therefore, many KV cache eviction works are studying how to effectively retain important tokens in the inference stage and evict unimportant ones. StreamingLLM (Xiao et al., 2023) relied on a simple positional heuristic to retain initial and final tokens. This simple eviction strategy leads to significant information loss. To address this, subsequent methods introduced more sophisticated indicators driven by attention to identify less important tokens. H2O (Zhang et al., 2023) uses cumulative attention to eliminate the last token. SnapKV (Li et al., 2024) uses window attention clustering in the prefill stage to obtain the set of important tokens. CAKE (Qin et al., 2025) introduces a cascading and adaptive strategy that evaluates layer-specific preferences to rationally distribute cache resources.

Nevertheless, the static methods above suffer from a critical limitation. Tokens that are regarded as irrelevant and evicted at a certain stage cannot be retrieved, which may become important in subsequent processes, resulting in significant information loss. Some works have recognized the importance of dynamic selection, such as Quest (Tang et al., 2024). However, it fails to reduce memory usage and suffers from accuracy degradation. ShadowKV (Sun et al., 2025) and OmniKV (Hao et al., 2025) attempt to handle infinite context by offloading KV pairs to CPU memory. Unfortunately, their coarse-grained retrieval strategies often overlook fine-grained dependencies and incur high I/O overhead.

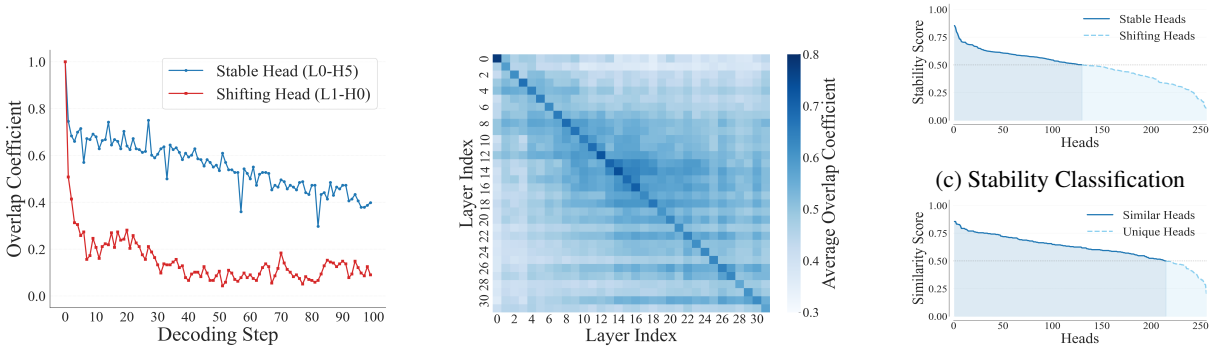
Some methods, such as KVQuant (Hooper et al., 2024) and KIVI (Liu et al., 2024), attempt to quantize the KV cache. These quantization methods are orthogonal to the token-level eviction strategy of the KV cache and can be combined to further substantially reduce GPU memory overhead.

3 Observation

To investigate the intrinsic attention behavior of the KV cache in long-context inference, we conducted a pilot study using a small calibration dataset. More details on the data collection process and visualization methodology are provided in the Appendix B. We focus on quantifying the behavior of attention heads across two dimensions: temporal heterogeneity and intralayer redundancy. Specifically, we employ the overlap coefficient (Szymkiewicz, 1934) to evaluate the alignment between the sets of the top-k important indices from any two sources, $\mathcal{K}(X)$ and $\mathcal{K}(Y)$, calculated as follows:

$$\mathrm{O}(\mathcal{K}(X), \mathcal{K}(Y)) = \frac{|\mathcal{K}(X) \cap \mathcal{K}(Y)|}{\min(|\mathcal{K}(X)|, |\mathcal{K}(Y)|)} \quad (1)$$

where X and Y represent attention scores (e.g., from different decoding steps or different attention heads) and $\mathcal{K}(\cdot)$ denotes the set of indices corresponding to the top-k values. This metric serves as a proxy for information, where a high overlap



(a) Temporal Heterogeneity (b) Intralayer Redundancy (c) Stability Classification (d) Similarity Classification

Figure 1: Analysis of attention heads heterogeneity and redundancy. (a) illustrates temporal heterogeneity: the blue line represents a stable head maintaining consistent focus, while the red line indicates a shifting head with rapid attention shift; (b) visualizes intralayer redundancy, where the darker diagonal blocks indicate that attention heads within the same layer share significantly higher similarity than those across different layers; (c) and (d) present the classification of heads based on their stability and similarity scores, respectively.

indicates that the head persistently attends to the same context.

Temporal Heterogeneity. We first examine how the focus of a single attention head evolves over time. By tracking the overlap coefficient between the top-k attended tokens in the decode stage and the prefill stage over a span of T steps (e.g., $T = 100$), we observe distinct behaviors. As shown in Figure 1 (a), certain heads exhibit slow decay, where the set of important tokens remains highly consistent with the prefill phase, indicating a persistent focus on specific contexts. Conversely, some heads show fast decay with a rapid decline in overlap, suggesting that their attention focus shifts dynamically as new tokens are generated. To quantify this property, we define the stability score $S^{(h)}_{\text{stable}}$ for a head h as the median of its overlap coefficients with the prefill stage across T decoding steps:

$$S^{(h)}_{\text{stable}} = \underset{t=1\ldots T}{\text{Median}} \left(\text{O}(\mathcal{K}^{(h)}_t, \mathcal{K}^{(h)}_{\text{prefill}}) \right) \tag{2}$$

Figure 1 (c) shows the stability scores of all heads ranked in descending order. Based on a predefined stability threshold τ_{stable}, we classify the heads into stable heads where $S_{\text{stable}} \geq \tau_{\text{stable}}$ and shifting heads where $S_{\text{stable}} < \tau_{\text{stable}}$.

Intralayer Redundancy. In addition to the temporal heterogeneity of attention, we also investigated spatial redundancy across attention heads. To quantify spatial relationships, we define layer-level similarity as the average of the overlap coefficients from each head in a source layer to a target layer. The result, shown in Figure 1 (b), reveals a strong intralayer similarity, which is significantly higher than the interlayer similarity. This high degree of intralayer redundancy suggests that not all heads contribute unique information; many are functionally similar. To measure this redundancy, we calculate the similarity score $S^{(h)}_{\text{sim}}$. For each step t, we first identify the maximum overlap between the head h and all other heads h' in the same layer l. The final score is the median of these maximums in the T steps:

$$S^{(h)}_{\text{sim}} = \underset{t=1\ldots T}{\text{Median}} \left(\max_{h' \in l} \text{O}(\mathcal{K}^{(h)}_t, \mathcal{K}^{(h')}_t) \right) \tag{3}$$

We plot the heads sorted by their similarity scores in Figure 1 (d). Specifically, based on similarity threshold τ_{sim}, we designate heads as similar heads where $S_{\text{sim}} \geq \tau_{\text{sim}}$, implying that their information can be approximated by others, and conversely as unique heads where $S_{\text{sim}} < \tau_{\text{sim}}$. Collectively, these two dimensions form the basis for our heterogeneous taxonomy strategy in HeteroCache.

4 Method

Building on the observations of temporal heterogeneity and intralayer redundancy presented in Section 3, we propose HeteroCache, a framework designed to compress the KV cache through heterogeneous head profiling and dynamic retrieval. The detailed pseudocode for our algorithm is provided in Appendix A. This section details the three phases of our approach: head profiling and taxonomy, stability-based budget allocation, and hierarchical storage with dynamic retrieval.

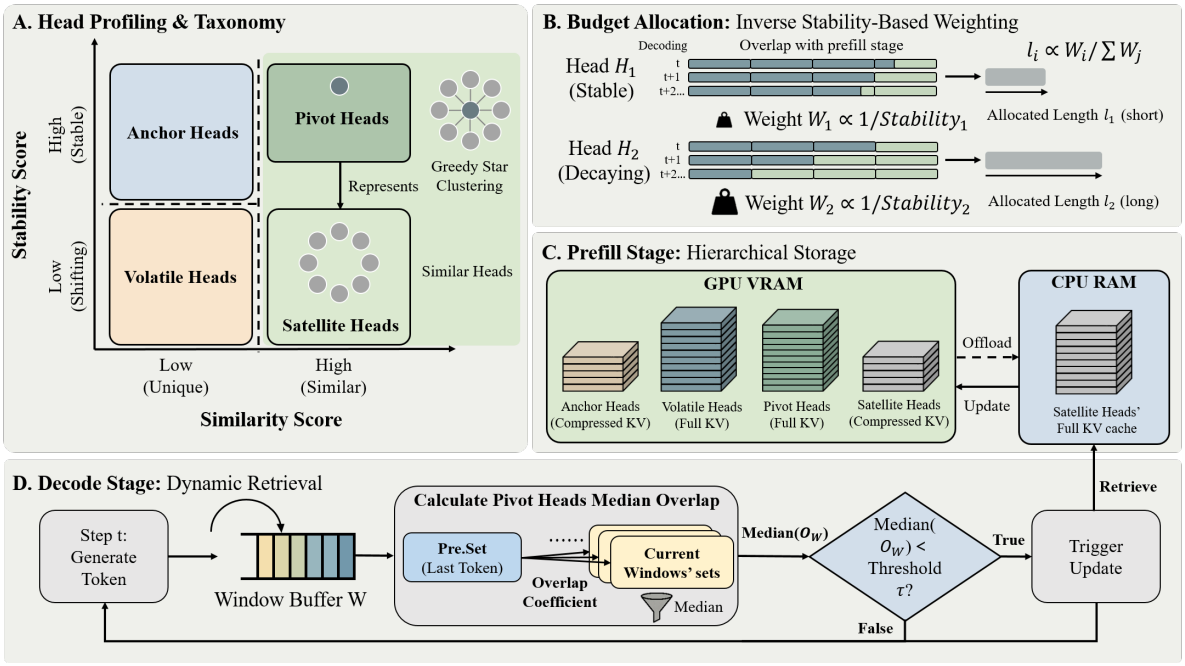


Figure 2: The workflow of HeteroCache. **(A, B) Offline Calibration:** Heads are categorized into functional roles (A) to determine stability-based budgets for compressed heads (B). **(C, D) Online Inference:** Guided by (B), (C) initializes hierarchical storage in prefill stage. In (D), Pivot heads monitor drift to trigger asynchronous CPU retrieval for updating satellite heads in decode stage.

4.1 Head Profiling and Taxonomy

To systematically exploit the heterogeneity of attention heads, we introduce an offline calibration phase to compute the stability score $S^{(h)}_{\text{stable}}$ and similarity score $S^{(h)}_{\text{sim}}$ for each head h. Based on these metrics, we categorize the set of all attention heads $\mathcal{H}$ into four functional roles, as shown in Figure 2 (A). We first differentiate heads according to the similarity threshold τ_{sim}, defining the sets of unique heads and similar heads as:

$$\begin{aligned} \mathcal{H}_{\text{unique}} &= \{h \in \mathcal{H} \mid S^{(h)}_{\text{sim}} < \tau_{\text{sim}}\} \\ \mathcal{H}_{\text{similar}} &= \{h \in \mathcal{H} \mid S^{(h)}_{\text{sim}} \geq \tau_{\text{sim}}\} \end{aligned} \tag{4}$$

The unique heads are characterized by their specialized functional roles, exhibiting attention patterns that diverge significantly from other heads. Given their distinct nature, we further classify them based on their stability scores. Using the stability threshold τ_{stable}, we delineate anchor heads and volatile heads as:

$$\begin{aligned} \mathcal{H}_{\text{anchor}} &= \{h \in \mathcal{H}_{\text{unique}} \mid S^{(h)}_{\text{stable}} \geq \tau_{\text{stable}}\} \\ \mathcal{H}_{\text{volatile}} &= \{h \in \mathcal{H}_{\text{unique}} \mid S^{(h)}_{\text{stable}} < \tau_{\text{stable}}\} \end{aligned} \tag{5}$$

Anchor heads maintain a consistent focus on specific contexts, making their history highly predictable and suitable for compression. In contrast, volatile heads exhibit rapid attention shift; therefore, we retain their full KV cache on the GPU to prevent information loss. Empirically, volatile heads are sparse and their proportion is tunable through τ_{stable}, ensuring that they do not exhaust the memory budget.

For similar heads $\mathcal{H}_{\text{similar}}$, characterized by high redundancy, we employ the greedy star clustering algorithm (Aslam et al., 2004) to divide them into clusters $\mathcal{C} = \{C_1, \ldots, C_m\}$, ensuring that every head in a cluster is similar to its central head. Within each cluster C_j, the head with the highest centrality is designated as the pivot head $h^{(j)}_{\text{pivot}}$. Since the attention distribution of the pivot head effectively approximates that of the other members in the cluster, we retain its full KV cache to serve as a representative monitor for the attention shift. The remaining heads are classified as satellite heads:

$$\mathcal{H}_{\text{satellite}} = \bigcup_j (C_j \setminus \{h^{(j)}_{\text{pivot}}\}) \tag{6}$$

where $\setminus$ denotes the set difference operation. Consequently, satellite heads undergo compression to significantly minimize GPU memory usage, yet remain subject to dynamic updates guided by the pivot heads during inference. For a comprehensive description of the implementation pipeline, in

conjunction with evaluations of the robustness, efficiency, and transferability of our offline calibration, see the Appendix C.

4.2 Stability-Based Budget Allocation

Although numerous KV cache compression algorithms (Cai et al., 2025; Qin et al., 2025) have demonstrated varying degrees of redundancy across model layers, few approaches utilize fine-grained compression at the individual attention head level. Using the functional head classification of the previous section, we achieve this granularity by incorporating both the anchor and satellite heads into the compressed head set defined as $\mathcal{H}_{\text{comp}} = \mathcal{H}_{\text{anchor}} \cup \mathcal{H}_{\text{satellite}}$. Our intuition is that heads with higher stability scores exhibit concentrated attention regions, which permit a reduced budget, whereas heads with lower stability display shifting attention patterns that necessitate a larger allocation. To address this, we propose a fine-grained inverse stability-based weighting strategy, as illustrated in Figure 2 (B), where we assign a weight $w_i = 1/S^{(i)}_{\text{stable}}$ to each compressed head $h_i \in \mathcal{H}_{\text{comp}}$ inversely proportional to its stability score. To determine the allocated KV cache length l_i for each compressed head, we first calculate a compressed base length L_{base}. Let ρ be the KV cache budget ratio, N be the total number of heads, and L be the sequence length. The base length L_{base} is calculated as:

$$L_{\text{base}} = \frac{(\rho N - N_{\text{full}})L}{N_{\text{comp}}} \tag{7}$$

Given the total budget $N_{\text{comp}} \cdot L_{\text{base}}$, l_i is:

$$l_i = (N_{\text{comp}} \cdot L_{\text{base}}) \cdot \frac{w_i}{\sum_{h_j \in \mathcal{H}_{\text{comp}}} w_j} \tag{8}$$

This inverse weighting allows HeteroCache to adaptively safeguard context-sensitive heads against premature eviction.

4.3 Hierarchical Storage & Retrieval

To efficiently manage memory consumption, HeteroCache implements a hierarchical storage mechanism bridging GPU VRAM and CPU RAM, as illustrated in Figure 2 (C).

During the prefill stage, we maintain the full KV caches for the $\mathcal{H}_{\text{full}}$ volatile and pivot heads on the GPU. For compressed heads $\mathcal{H}_{\text{comp}}$, which comprise both anchor heads and satellite heads, we adopt a split storage strategy. Based on Section 4.2, we assign a specific budget l_i to each compressed head. In the GPU VRAM, we only retain the KV cache corresponding to the top-l_i attention scores for these heads. Simultaneously, the full KV contexts of the satellite heads are offloaded to the CPU RAM to serve as a retrieval reservoir.

We formally describe the dynamic retrieval process for satellite heads as follows, also shown in Figure 2 (D). Let h_p be a pivot head. We initialize a baseline set $\mathcal{K}_{\text{base}}$ consisting of the top-L_{base} attended indices of the prefill stage. During the decoding step t, the pivot head monitors the shift of the attention distribution by calculating the overlap coefficient o_t utilizing the base length L_{base} as normalization factor:

$$o_t = \text{O}(\mathcal{K}_t, \mathcal{K}_{\text{base}}) = \frac{|\mathcal{K}_t \cap \mathcal{K}_{\text{base}}|}{L_{\text{base}}} \tag{9}$$

To differentiate meaningful attention shift from transient noise, we employ a sliding window of size W. A retrieval signal $r_t \in \{0, 1\}$ is triggered when the median overlap within the window falls below a predefined drift threshold τ_{drift}:

$$r_t = \mathbb{I}\left(\text{Median}\left(\{o_i\}_{i=t-W+1}^{t}\right) < \tau_{\text{drift}}\right) \tag{10}$$

Upon detecting a drift where $r_t = 1$, the system triggers an update guided by the pivot head.

Specifically, we utilize the indices of the top-l_i tokens from the pivot head's current attention distribution to retrieve the corresponding KV tensors from the CPU, updating the satellite heads on the GPU. Subsequently, we update the historical baseline state to prepare for re-initiating the next dynamic retrieval. Since this process is executed asynchronously, I/O latency is effectively hidden within the attention computation overhead, ensuring efficient inference without stalling. Additionally, anchor heads maintain their statically compressed state without compromising performance, benefiting from their high temporal stability.

5 Experiments

5.1 Experiment Settings

Backbone LLMs and Baselines. We selected multiple open-source LLMs for our experiments to verify the performance across different paradigms. These include standard non-reasoning models, specifically Llama-3.1-8B-Instruct and Qwen2.5-14B-Instruct, as well as the reasoning model DeepSeek-R1-Distill-Llama-8B.

To demonstrate the effectiveness of HeteroCache, we compare it with several compression

Methods	Mem%	Avg.	Single-Doc QA	Multi-Doc QA	Summarize	Few-Shot	Synthetic	Code
Llama-3.1-8B-Instruct	100%	49.77	43.53	40.82	29.04	69.48	53.75	62.01
Quest	100%	48.60	42.79	40.12	**29.11**	67.50	52.52	59.55
ShadowKV	50%	48.03	42.10	40.42	23.34	**69.48**	**53.75**	59.08
OmniKV	50%	49.22	43.28	40.33	29.09	68.94	53.50	60.20
CAKE	50%	49.16	43.37	40.48	27.95	68.97	**53.75**	60.45
HeteroCache	50%	**49.42**	**43.38**	**40.54**	28.59	**69.48**	**53.75**	**60.77**
Qwen2.5-14B-Instruct	100%	50.55	43.01	52.54	24.99	71.68	54.35	56.70
ShadowKV	30%	46.93	35.77	50.80	17.18	70.65	54.00	53.20
OmniKV	30%	46.48	37.90	50.21	23.10	68.44	50.25	48.95
CAKE	30%	46.92	40.74	42.95	23.69	66.77	54.25	53.09
HeteroCache	30%	**49.97**	**42.46**	**52.12**	**24.12**	**71.18**	**54.63**	**53.33**

Table 1: Inference performance on LongBench (Bai et al., 2024). *Italics* indicate that the model uses FullAttention baseline. **Bold** indicates the best performance among compression methods.

			Difficulty		Length		
Methods	Mem%	Overall	Easy	Hard	Short	Medium	Long
Llama-3.1-8B-Instruct	100%	31.2	32.8	30.2	36.1	27.9	29.6
Quest	100%	28.0	29.7	27.0	33.3	**27.4**	20.4
ShadowKV	50%	30.0	30.2	**29.9**	**36.1**	26.5	26.9
OmniKV	50%	30.4	31.8	29.6	**36.1**	26.0	29.6
CAKE	50%	29.4	31.8	28.0	33.9	25.6	29.6
HeteroCache	50%	**30.6**	**32.3**	29.6	35.6	26.0	**31.5**
Qwen2.5-14B-Instruct	100%	33.2	35.4	31.8	42.8	29.3	25.0
ShadowKV	30%	33.0	**36.5**	30.9	42.8	29.3	24.1
OmniKV	30%	33.3	36.2	31.4	43.6	28.5	26.0
CAKE	30%	31.6	33.3	30.5	37.2	**30.2**	25.0
HeteroCache	30%	**33.6**	35.9	**32.2**	**44.4**	27.4	**27.8**
DeepSeek-R1-Distill-Llama-8B	100%	29.2	32.8	27.0	32.8	25.1	31.5
Quest	100%	24.1	30.2	20.3	23.3	25.1	23.1
ShadowKV	50%	25.0	28.4	21.6	23.7	26.1	25.2
OmniKV	50%	26.6	26.6	26.7	28.9	**27.4**	21.3
CAKE	50%	27.2	28.6	26.4	**33.9**	26.5	17.6
HeteroCache	50%	**28.9**	**32.5**	**26.8**	32.5	24.9	**30.8**

Table 2: Inference performance on LongBench v2 (Bai et al., 2025). *Italics* indicate that the model uses FullAttention baseline. **Bold** indicates the best performance among compression methods.

methods: (1) FullAttention. The original model without KV cache compression. (2) Quest (Tang et al., 2024). An algorithm that estimates the criticality of KV cache pages using the statistics of key values, loading only critical pages for attention calculation; (3) ShadowKV (Sun et al., 2025). A dynamic compression method that retains low-rank key projections on the GPU while offloading the value cache to the CPU, dynamically reconstructing minimal sparse KV pairs on the fly; (4) OmniKV (Hao et al., 2025). A dynamic compression method by offloading KV pairs to CPU memory and retrieving relevant tokens based on layer attention patterns; (5) CAKE (Qin et al., 2025). A cascading static compression framework that allocates adaptive cache budgets between layers based on their spatiotemporal attention preferences.

Implementation. Detailed system configurations are provided in Appendix D.1. To demonstrate the orthogonality of HeteroCache with quantization, we applied 4-bit weight quantization to the DeepSeek-R1-Distill-Llama-8B model using bitsandbytes (Dettmers et al., 2021). To ensure a fair comparison, we standardized the evaluation criteria by requiring all algorithms to retain a prefetched KV cache equivalent to a memory budget ratio ρ, rather than a fixed token budget, and we used the Qwen2.5 series to ensure a fair comparison, as implementations of certain baselines do not yet support the latest architecture Qwen3. See Ap-

Methods	Mem%	Avg.	En.Sum	En.QA	En.MC	En.Dia	Zh.QA	Code.Debug	Math.Find	Retr.PassKey	Retr.Number
Llama-3.1-8B-Instruct	100%	47.56	32.93	27.41	65.94	19.50	34.60	25.38	24.00	99.66	98.64
Quest	100%	46.68	28.86	26.55	64.00	18.50	31.78	24.65	22.86	99.15	98.01
ShadowKV	50%	44.26	23.59	18.62	64.19	15.50	30.55	**25.38**	19.71	98.31	98.14
OmniKV	50%	46.92	31.46	**27.03**	**65.94**	19.00	33.98	23.10	**24.00**	**99.66**	98.14
CAKE	50%	46.91	31.51	26.15	**65.94**	19.00	33.22	25.13	**24.00**	**99.66**	97.46
HeteroCache	50%	**47.34**	**32.05**	26.94	**65.94**	**19.50**	**34.14**	**25.38**	**24.00**	**99.66**	**98.47**

Table 3: Inference performance on InfiniteBench (Zhang et al., 2024). *Italics* indicate that the model uses FullAttention baseline. **Bold** indicates the best performance among compression methods.

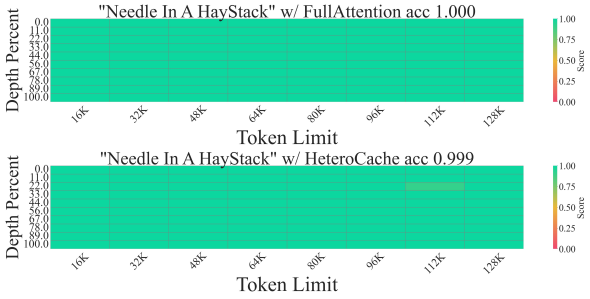


Figure 3: Results for Llama-3.1-8B-Instruct on NIAH, evaluated on context lengths from 16K to 128K tokens.

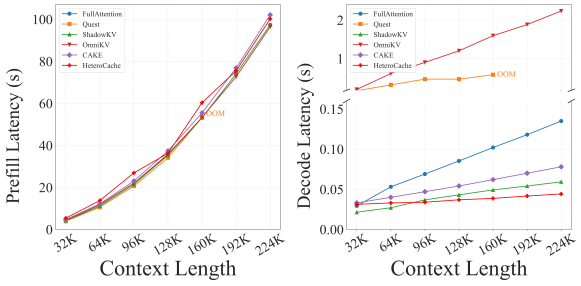


Figure 4: End-to-end latency results for HeteroCache and the baselines. The left and right plots show the latency for the prefill and decode stage, respectively.

pendix D.2 for detailed system footprint and bandwidth scaling.

Hyperparameter Configuration. For the hyperparameter configuration, we standardized both the stability threshold τ_{stable} and the similarity threshold τ_{sim} at 0.5 for Llama and the stability threshold 0.4 for Qwen due to the lower memory budget. Consistent with this setting, the drift threshold τ_{drift} was set equal to τ_{stable}. Regarding the KV cache memory budget ρ, we selected 0.3 or 0.5, tailored to the specific characteristics of different datasets and models. Ablation studies on parameters are presented in the following sections.

5.2 Performance on Various Tasks

Standard Benchmarks and Retrieval. To comprehensively validate the effectiveness of HeteroCache in long-context inference tasks, we conducted extensive experiments on LongBench and LongBench v2 (Bai et al., 2024, 2025). As summarized in Table 1 and Table 2, the results demonstrate that even under significantly compressed KV cache budgets, HeteroCache exhibits superior performance that is nearly indistinguishable from the FullAttention mechanism and significantly outperforms existing state-of-the-art baselines. This advantage is further substantiated by extending it to the more challenging InfiniteBench (Zhang et al., 2024). As shown in Table 3, HeteroCache maintains exceptional inference accuracy even with low memory budgets, achieving leading or highly competitive results across various subtasks, which fully proves its robustness in handling ultra-long context tasks.

This remarkable performance superiority stems from effectively overcoming the limitations of existing methods. Specifically, although ShadowKV and OmniKV also employ dynamic compression and offloading strategies, they primarily rely on coarse-grained compression for retrieval, which fails to capture fine-grained semantic dependencies and results in wasted cache space. In contrast, HeteroCache achieves more efficient information retention with a minimal memory footprint through fine-grained compression and retrieval at the head level. Regarding CAKE, although it rationally distributes cache resources based on layer importance, its lack of dynamic compression and recall mechanisms makes it unable to address attention drift during long-text generation. Furthermore, while Quest achieves certain acceleration effects through sparsity, results indicate that it fails to effectively reduce memory usage and instead leads to a noticeable drop in accuracy.

Our performance on the Needle In A Haystack (NIAH) (Fu et al., 2024) further corroborates this analysis. As illustrated in Figure 3, HeteroCache achieves perfect retrieval accuracy across all con-

Methods	Mem%	Avg.	Prefill (s)	Decode (s)
HeteroCache	50%	46.00	2.30	0.032
w/o Allocation	50%	44.97	2.30	0.035
w/o Retrieval	50%	45.32	2.30	0.027

Table 4: Ablation study of the core components of HeteroCache on Qasper with Llama-3.1-8B-Instruct.

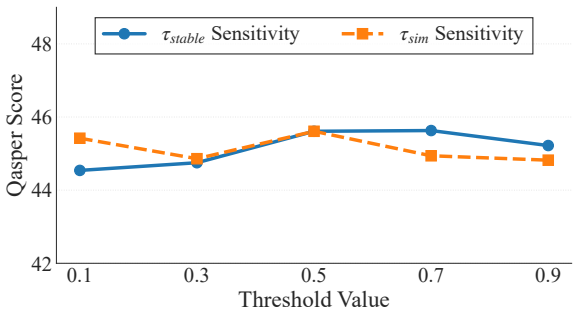


Figure 5: Ablation study of the threshold with Llama-3.1-8B-Instruct.

text length settings from 16K to 128K, exhibiting robust information retention capabilities comparable to the FullAttention mechanism. Such consistent performance strongly demonstrates that our approach, through its fine-grained dynamic retrieval mechanism, can high-fidelity restore critical historical dependencies while significantly reducing GPU memory overhead, thereby providing reliable support for long-document question answering and complex reasoning tasks. Beyond standard single-needle retrieval, we evaluate the robustness of our method on U-NIAH (Gao et al., 2026), detailed in the Appendix D.3.

Reasoning Capabilities on DeepSeek-R1-Distill. Emerging reasoning models (Guo et al., 2025) have improved their ability to solve complex problems through the chain of thought mechanism. However, the extremely long intermediate processes they generate pose a severe challenge to the efficiency of KV caching, making them an ideal scenario for testing dynamic retrieval capabilities. In the generation of long sequences, the loss of any key intermediate step may lead to the breaking of reasoning logic. As shown in Table 2, our test results using the DeepSeek-R1-Distill-Llama-8B model on the LongBench v2 benchmark indicate that HeteroCache demonstrates significantly better performance than baselines under limited memory and is closest to the FullAttention mechanism. This confirms that its fine-grained dynamic strategy can precisely retain key thinking paths in long-distance reasoning, effectively ensuring the coherence of complex reasoning tasks under the condition of limited memory.

5.3 Latency

We evaluated the end-to-end latency of HeteroCache against the baselines on the Llama-3.1-8B-Instruct model. For the prefill stage, we measured the Time-To-First-Token (TTFT), while for the decode stage, we calculated the average latency per token over 50 generated tokens.

The results presented in Figure 4 illustrate the efficiency of the proposed method. In the prefill stage, although compression algorithms typically necessitate additional computations for metric statistics and cache selection, experimental results indicate that these overheads are minimal. Consequently, the HeteroCache TTFT curve overlaps with that of the FullAttention baseline, indicating that no significant latency bottleneck is introduced. In the decode stage, HeteroCache shows exceptional performance. As the context length increases, it maintains consistently low inference latency, achieving approximately 3× acceleration compared to the FullAttention mechanism in the 224K context. More notably, in stark contrast to OmniKV which incurs substantial latency overhead due to frequent I/O operations, HeteroCache achieves a speedup of over 40× compared to OmniKV. Furthermore, starting from a context length of 128K, HeteroCache consistently outperforms all other compression algorithms, achieving the lowest per-token generation latency and verifying its superior efficiency and scalability in ultra-long context scenarios.

Furthermore, to comprehensively evaluate the robustness of our asynchronous retrieval mechanism against potential I/O bottlenecks, we provide detailed analyses of the retrieval trigger rate, tail latency, and performance under constrained interconnects in the Appendix D.4.

5.4 Ablation Studies

To systematically validate the effectiveness of the core components within HeteroCache, we designed two key ablation variants and presented the results in Table 4. We first examined the impact of the budget allocation strategy. The performance decline in the w/o Allocation variant, where a uniform cache budget is assigned, confirms that ignoring stability differences prevents compressed heads from retaining sufficient context. Subsequently, we evaluated

the dynamic retrieval mechanism. The degradation in the w/o Retrieval setting verifies the existence of attention drift and highlights the necessity of pivot-guided updates for maintaining long-range reasoning. Furthermore, the integration of these components imposes a negligible latency overhead. Regarding the hyperparameters as illustrated in Figure 5, our algorithm demonstrates robustness. Consequently, we selected the optimal value of 0.5 as the experimental threshold to balance the compression rate with the retention of information.

6 Conclusion

We propose HeteroCache, a training-free dynamic compression framework. By combining stability-based budget allocation with a head-guided asynchronous retrieval mechanism, HeteroCache effectively addresses the limitations of existing dynamic compression algorithms. It significantly outperforms prior methods in long-context tasks while drastically reducing the memory footprint. Furthermore, it achieves a $3\times$ decoding acceleration compared to the original model, offering an efficient and robust solution for resource-constrained inference.

Limitations

Although HeteroCache demonstrates state-of-the-art performance in efficient long-context inference, we acknowledge a few limitations in the current implementation. First, our prototype is primarily based on high-level PyTorch operations to validate the algorithmic effectiveness. We have not yet implemented custom CUDA kernels for the sparse attention and dynamic retrieval modules. Consequently, the current speedup does not fully reflect the theoretical upper bound of our method, and further engineering optimizations could yield significant latency reductions. Second, as a heterogeneous storage framework, the efficiency of our asynchronous retrieval mechanism inherently relies on the interconnect bandwidth (e.g., PCIe) between the CPU and the GPU. While our supplementary experiments in Appendix D.4 demonstrate that HeteroCache maintains highly robust performance even on consumer-grade hardware with restricted bandwidth, extreme low-bandwidth environments might still constrain the ability to hide I/O latency completely. Future work will focus on developing optimized kernels and exploring hardware-aware prefetching strategies to fully mitigate these potential bottlenecks.

Ethical Considerations

Our work introduces HeteroCache, a framework for compressing the KV cache in LLMs to reduce its memory overhead and accelerate the decoding process. The research is purely algorithmic in nature, with a primary focus on memory optimization.

This study did not involve human subjects and all experiments were conducted on publicly available open-source models and standard academic benchmarks. No personal or private information was used. We do not foresee any direct ethical concerns arising from this work. We declare no conflict of interest.

Acknowledgments

This work was supported in part by The National Nature Science Foundation of China (Grant Nos.: 62303406, 62432014), in part by Ningbo Key R&D Program (No.: 2025Z055).

References

Javed Aslam, Ekaterina Pelekhov, and Daniela Rus. 2004. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applications*, 8(1):95–129.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, Vienna, Austria. Association for Computational Linguistics.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2025. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *Preprint*, arXiv:2406.02069.

Ziyang Chen, Xing Wu, Junlong Jia, Chaochen Gao, Qi Fu, Debing Zhang, and Songlin Hu. 2026. Long-

bench pro: A more realistic and comprehensive bilingual long-context evaluation benchmark. *Preprint*, arXiv:2601.02872.

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, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024. Data engineering for scaling language models to 128k context. *Preprint*, arXiv:2402.10171.

Yunfan Gao, Yun Xiong, Wenlong Wu, Bohan Li, Yijie Zhong, and Haofen Wang. 2026. U-niah: Unified rag and llm evaluation for long context needle-in-a-haystack. *ACM Trans. Inf. Syst.*, 44(3).

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Jitai Hao, Yuke Zhu, Tian Wang, Jun Yu, Xin Xin, Bo Zheng, Zhaochun Ren, and Sheng Guo. 2025. Omnikv: Dynamic context selection for efficient long-context llms. In *The Thirteenth International Conference on Learning Representations*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Minference 1.0: accelerating pre-filling for long-context llms via dynamic sparse attention. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

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.

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*.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. 2025. Cake: Cascading and adaptive kv cache eviction with layer preferences. *arXiv preprint arXiv:2503.12491*.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, and 465 others. 2025. Openai gpt-5 system card. *Preprint*, arXiv:2601.03267.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. 2025. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *Preprint*, arXiv:2410.21465.

Dezydery Szymkiewicz. 1934. Une conlribution statistique à la géographie floristique. *Acta Societatis Botanicorum Poloniae*, 11(3):249–265.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1332 others. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, and 5 others. 2025. Openhands: An open platform for ai software developers as generalist agents. *Preprint*, arXiv:2407.16741.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Streamingllm: Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao,

Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. 2025. Flashinfer: Efficient and customizable attention engine for llm inference serving. *Preprint*, arXiv:2501.01005.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand. Association for Computational Linguistics.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2o: heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Appendix

A Pseudocode

To provide a clear perspective on our proposed method, we present the detailed HeteroCache workflow in the form of pseudocode. The entire step-by-step procedure is systematically illustrated in Algorithm 1.

B Details of Observation

In this section, we provide a comprehensive description of the observation. This includes the data acquisition process and the methodology used to generate the motivational observations presented in Figure 1.

B.1 Data Collection and Preprocessing

To construct a representative calibration dataset that ensures the generalization of our attention head taxonomy, we performed extensive web crawling from Wikipedia. We curated a diverse set of 50 samples that cover a wide range of domains, including science, entertainment, literature, and technology, to prevent topic-specific biases in attention patterns. Each collected sample was preprocessed and truncated to a standardized length of 10K tokens. This sequence length was chosen to be sufficiently long to trigger potential attention drift phenomena while remaining computationally manageable for the profiling phase. All subsequent metric calculations and profilings are based on the statistical average across this dataset to ensure robustness.

B.2 Analysis of Temporal Heterogeneity

The temporal heterogeneity observed in Figure 1 (a) was generated using a specific experimental setup derived from our calibration dataset. We used the Llama-3.1-8B-Instruct model and selected a sample context of 10K tokens as input. The model then generated 100 subsequent tokens in a token-by-token decoding manner. To visualize the attention stability, we tracked two distinct heads: Layer 1 Head 5 as the representative stable head and Layer 2 Head 1 as the representative shifting head. For the calculation of the representative attention set, we implemented a rigorous extraction pipeline in each decoding step. Specifically, we first obtained the raw attention distribution of the last query token attending to all historical keys. To ensure the robustness of the selected indices against transient local noise and to capture the broader attention peaks, we applied average pooling operation motivated by the findings of SnapKV (Li et al., 2024). Following this smoothing process, we identified the indices corresponding to the top-1K attention scores to form the active token set $\mathcal{T}_k$. The overlap coefficient was then calculated at each step by measuring the intersection size between the $\mathcal{T}_k$ of the current decoding step and the $\mathcal{T}_k$ obtained during the prefill stage. A high overlap indicates that the head maintains its focus on the same historical context, whereas a declining overlap signifies attention drift.

B.3 Analysis of Intralayer Redundancy

To quantify the spatial redundancy visualized in the heatmap of Figure 1 (b), we extended the overlap analysis from individual heads to layer-level interactions. Utilizing the same smoothed Top-k extraction methodology described previously, we define the layer-level similarity as the average of the overlap coefficients from each head in a source layer to a target layer. This metric effectively captures the degree to which information in one layer is statistically correlated with that of another. The resulting visualization reveals that the diagonal blocks ex-

Algorithm 1 HeteroCache Inference Process

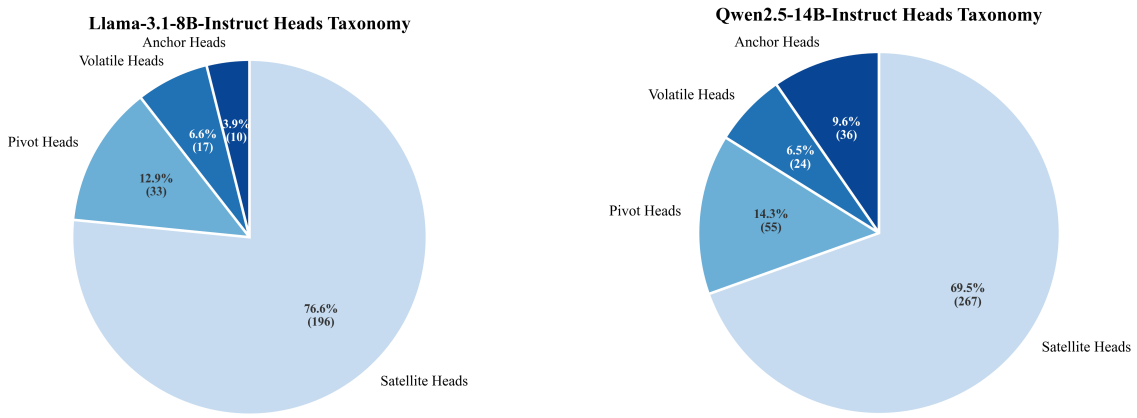
Require: Prompt P, Model $\mathcal{M}$, Thresholds $\tau_{\text{stable}}, \tau_{\text{sim}}, \tau_{\text{drift}}$, Budget Ratio ρ, Window W
Ensure: Generated sequence Y

// Stage 1: Head Profiling & Taxonomy
for each head $h \in \mathcal{H}$ **do**
  Compute $S^{(h)}_{\text{stable}}$ and $S^{(h)}_{\text{sim}}$ via calibration
end for
$\mathcal{H}_{\text{unique}} \leftarrow \{h \in \mathcal{H} \mid S^{(h)}_{\text{sim}} < \tau_{\text{sim}}\}$; $\mathcal{H}_{\text{similar}} \leftarrow \{h \in \mathcal{H} \mid S^{(h)}_{\text{sim}} \geq \tau_{\text{sim}}\}$
$\mathcal{H}_{\text{anchor}} \leftarrow \{h \in \mathcal{H}_{\text{unique}} \mid S^{(h)}_{\text{stable}} \geq \tau_{\text{stable}}\}$; $\mathcal{H}_{\text{volatile}} \leftarrow \{h \in \mathcal{H}_{\text{unique}} \mid S^{(h)}_{\text{stable}} < \tau_{\text{stable}}\}$
$\mathcal{C}_{\text{clusters}} = \{C_1, ..., C_m\} \leftarrow \text{GreedyStarClustering}(\mathcal{H}_{\text{similar}})$
$\mathcal{H}_{\text{pivot}} \leftarrow \{h^{(j)}_{\text{pivot}} \mid \forall C_j \in \mathcal{C}_{\text{clusters}}\}$; $\mathcal{H}_{\text{satellite}} \leftarrow \bigcup_j (C_j \setminus \{h^{(j)}_{\text{pivot}}\})$
// Stage 2: Stability-Based Budget Allocation
$\mathcal{H}_{\text{comp}} \leftarrow \mathcal{H}_{\text{anchor}} \cup \mathcal{H}_{\text{satellite}}$
Calculate $L_{\text{base}} \leftarrow \frac{(\rho N - N_{\text{full}})L}{N_{\text{comp}}}$
for each head $h_i \in \mathcal{H}_{\text{comp}}$ **do**
  $w_i \leftarrow 1/S^{(i)}_{\text{stable}}$
  $l_i \leftarrow L_{\text{base}} \cdot \frac{w_i}{\sum_{h_j \in \mathcal{H}_{\text{comp}}} w_j}$
end for
// Stage 3: Hierarchical Storage & Dynamic Inference
$\mathcal{C}_{\text{GPU}} \leftarrow \{\text{FullKV}^{(h)} \mid h \in \mathcal{H}_{\text{volatile}} \cup \mathcal{H}_{\text{pivot}}\} \cup \{\text{Top}_{l_i}(\text{KV}^{(h)}) \mid h \in \mathcal{H}_{\text{comp}}\}$
$\mathcal{C}_{\text{CPU}} \leftarrow \{\text{FullKV}^{(h)} \mid h \in \mathcal{H}_{\text{satellite}}\}$
$\mathcal{K}^{(p)}_{\text{base}} \leftarrow \text{TopIndices}_{L_{\text{base}}}(\text{Attn}^{(p)}_{\text{prefill}}) \quad \forall p \in \mathcal{H}_{\text{pivot}}$
$\mathcal{Q}_{\text{buffer}} \leftarrow \emptyset$; $x_1 \leftarrow \text{LastToken}(P)$
for $t = 1$ **to** T **do**
  $y_t, q_t, \mathcal{C}_{\text{GPU}} \leftarrow \text{ModelForward}(\mathcal{M}, x_t, \mathcal{C}_{\text{GPU}})$
  $Y \leftarrow Y \cup \{y_t\}$
  $\mathcal{Q}_{\text{buffer}} \leftarrow \mathcal{Q}_{\text{buffer}} \cup \{q^{(p)}_t \mid p \in \mathcal{H}_{\text{pivot}}\}$
  if $t \pmod W = 0$ **then**
    for each pivot head $h_p \in \mathcal{H}_{\text{pivot}}$ **do**
      $\mathcal{W}_p \leftarrow \emptyset$
      for $k = t - W + 1$ **to** t **do**
        $q^{(p)}_k \leftarrow \mathcal{Q}_{\text{buffer}}[k]$
        $\mathcal{K}^{(p)}_k \leftarrow \text{TopIndices}_{L_{\text{base}}}(\text{Attention}(q^{(p)}_k, \mathcal{C}_{\text{GPU}}))$
        $o_k \leftarrow |\mathcal{K}^{(p)}_k \cap \mathcal{K}^{(p)}_{\text{base}}| / L_{\text{base}}$
        $\mathcal{W}_p \leftarrow \mathcal{W}_p \cup \{o_k\}$
      end for
      $r_t \leftarrow \mathbb{I}(\text{Median}(\mathcal{W}_p) < \tau_{\text{drift}})$
      if $r_t = 1$ **then**
        for each $s \in \text{Cluster}(h_p) \cap \mathcal{H}_{\text{satellite}}$ **do**
          $\mathcal{I}_s \leftarrow \text{TopIndices}_{l_s}(\text{Attention}(q^{(p)}_t, \mathcal{C}_{\text{GPU}}))$
          AsyncUpdate: $\mathcal{C}^{(s)}_{\text{GPU}} \leftarrow \mathcal{C}^{(s)}_{\text{CPU}}[\mathcal{I}_s]$
        end for
        $\mathcal{K}^{(p)}_{\text{base}} \leftarrow \mathcal{K}^{(p)}_t$
      end if
    end for
    $\mathcal{Q}_{\text{buffer}} \leftarrow \emptyset$
  end if
  $x_{t+1} \leftarrow y_t$
end for
return Y

hibit significantly higher values, confirming that heads within the same layer share a high degree of functional redundancy compared to cross-layer relationships.

B.4 Head Classification and Thresholding

The taxonomy of attention heads presented in Figure 1 (c) and Figure 1 (d) was derived by statistical analysis of the distribution of stability and similarity scores. For the stability classification shown in Figure 1 (c), we calculated the stability score S_{stable} for each head by averaging its overlap coefficients throughout the recorded decoding trajectory. We then ranked all heads in descending order based on these scores. Applying a predefined stability threshold τ_{stable} of 0.5, we divide the distribution curve into two distinct regions. Heads with scores above this threshold were rendered in solid dark blue to designate stable heads, while those falling below were depicted with a dashed light blue line



(a) Llama-3.1-8B-Instruct

(b) Qwen2.5-14B-Instruct

Figure 6: The distribution of functional roles derived from our profiling algorithm. The dominance of anchor and satellite heads across both models demonstrates the generalizability of the HeteroCache taxonomy.

to represent shifting heads. Parallel to this, Figure 1 (d) illustrates the spatial similarity distribution. We computed the similarity score S_{sim} for each head by measuring the maximum overlap it shares with any other head within the same layer at a representative decoding step. Similarly to the stability analysis, we ranked the heads according to their S_{sim} values and applied a similarity threshold τ_{sim} of 0.5. The curve clearly differentiates between high similarity heads, which are redundant and suitable for clustering, and unique heads, which possess distinct attention patterns.

C Algorithmic Implementation Details

In this section, we provide supplementary algorithmic details for HeteroCache. We divide the process into preprocessing and metric computation, taxonomy and budget allocation, robustness analysis, and calibration transferability to illustrate the specific implementation logic.

C.1 Preprocessing and Metric Computation

The pipeline begins with the extraction and preprocessing of raw attention weights. For models employing grouped query attention, such as Llama-3.1 and Qwen2.5, we first aggregate the query attention scores by averaging the weights within the same KV group to align with the actual KV cache structure. To mitigate the impact of transient noise on the raw attention distribution, we apply a 1D average pooling operation to the attention scores of the last token before identifying significant entries. Based on these smoothed scores, we extract the top-k indices to form the active token sets, which serve as the basis for calculating the stability and similarity metrics. Specifically, the stability score is computed as the median of the historical overlap coefficients between the decoding steps and the prefill stage to ensure robustness against outliers. Simultaneously, we construct a layer-wise adjacency graph by establishing edges between heads whose pairwise overlap ratio exceeds the predefined similarity threshold.

C.2 Taxonomy and Budget Allocation

Building on the constructed adjacency graph and stability metrics, we execute the taxonomy using a greedy star clustering algorithm. The process iterates through all attention heads to identify their functional roles. We first calculate the effective degree for each unassigned head, which represents the count of its neighbors that are also currently unassigned. In each iteration, the head with the highest effective degree is selected as a pivot head, and all its unassigned neighbors are immediately grouped in its cluster as satellite heads. This greedy strategy effectively maximizes the coverage of redundant structures. Upon completion of clustering, any remaining heads that were not assigned to a cluster are classified based on their temporal behavior. Those with stability scores exceeding the threshold are labeled as anchor heads, while the few remaining unstable heads are designated as volatile heads. Following this classification, we determine the cache budget for the compressible subset which includes both anchor and satellite heads. We assign a weight to each head that is inversely proportional to its stability score. These weights are then nor-

malized to derive the final cache allocation ratio to ensure that heads prone to context shifts receive a larger share of the budget.

C.3 Robustness Analysis

To empirically verify the robustness and generalizability of our proposed taxonomy, we applied this profiling pipeline to two distinct mainstream model architectures, including Llama-3.1-8B-Instruct and Qwen2.5-14B-Instruct. Regarding the hyperparameter configuration, we standardized both the stability and similarity thresholds at 0.5 for the Llama-3.1 model. For Qwen2.5, we utilized a stability threshold of 0.4 and a similarity threshold of 0.5 to accommodate its specific attention dynamics. The resulting distributions of the four functional roles are visualized in Figure 6. Consistent with our preliminary observations, the classification results for both model families reveal a striking commonality where the vast majority of attention heads are categorized as satellite heads. These heads exhibit high redundancy and are suitable for aggressive compression. Crucially, the proportion of volatile heads remains minimal across both architectures. This pervasive pattern confirms that the sparsity of attention and head redundancy are intrinsic characteristics shared by most mainstream LLMs, rather than being model-specific quirks. Consequently, our approach demonstrates strong robustness by guaranteeing that retaining the full KV cache for these highly dynamic heads does not occupy a substantial portion of the memory budget while maintaining high model performance.

C.4 Calibration and Transferability

Calibration Cost. The offline calibration process is highly efficient. It requires only a single forward pass along with minimal statistical computation. Empirically, the entire clustering process can be completed in a few minutes, introducing negligible overhead prior to deployment. Since the profiling pipeline only needs to extract and aggregate attention weights rather than storing massive intermediate hidden states, the entire process easily fits within a single consumer-grade GPU.

Sensitivity and Transferability. To evaluate sensitivity to calibration data and transferability to longer contexts, we conducted additional experiments using the LongBench Pro (Chen et al., 2026), which consists of real natural long documents. We selected twenty samples each from three distinct tasks: Version and Code Diff Analysis, Structured and Numeric Reasoning, and Dialogue Memory and Long Horizon Tracking.

To prove that offline clustering is robust to extended input lengths, we evaluated all these samples using a 128K context length. We used the Llama-3.1-8B-Instruct model and set the top-k threshold to one-fifth of the length, keeping all other configurations identical to those in the paper. The clustering results of these three different tasks show that the newly identified satellite heads overlap with our original clustering results by more than 80%. This demonstrates that our head taxonomy is highly robust, and the stability and redundancy patterns hold consistently at the 100K scale and beyond.

Furthermore, to ensure that this structural consistency does not significantly affect downstream accuracy, we tested the model's performance on the Qasper dataset from LongBench using these new task-specific clustering configurations. As shown in Table 5, these consistent results confirm that our calibration is effectively transferred to extremely long contexts and diverse domains without performance degradation.

Task (Calibration Source)	Qasper Score
Version & Code Diff Analysis	45.72
Structured & Numeric Reasoning	45.61
Dialogue Memory & Long-Horizon Tracking	45.68

Table 5: Performance on the Qasper dataset using clustering configurations derived from different 128K LongBench Pro tasks.

D Additional Experiments and Analyses

D.1 Detailed System Specifications

To ensure reproducibility and facilitate fair comparison across different hardware environments, we provide the complete system configuration used for our experimental evaluations in Table 6. Our analysis emphasizes the importance of the memory hierarchy and interconnect bandwidth, as these factors significantly influence the efficiency of the asynchronous KV cache retrieval mechanism.

D.2 Memory and Bandwidth Profiling

To provide a holistic view of the system's resource consumption, we measured the peak CPU memory usage and the corresponding PCIe bandwidth utilization during inference.

Component	Specification
CPU	Intel® Xeon® Gold 6338 @ 2.00 GHz
CPU RAM	256 GiB DDR4
GPU	NVIDIA A100 PCIe 80GB
GPU Memory	80 GB HBM2e
GPU Memory Bandwidth	1,935 GB/s
PCIe Interface	PCIe Gen4 x16
Theoretical PCIe Bandwidth	≈ 31.5 GB/s (per direction)
Interconnect	PCIe 4.0

Table 6: Detailed hardware and system configuration for experimental evaluation.

As summarized in Table 7, the peak CPU RAM usage scales linearly with the context length. For a 128K context, the CPU memory footprint reaches 17.95 GB, which remains well within the capacity of standard modern servers. Furthermore, peak bandwidth utilization reaches 21.07 GB/s, which is safely below the theoretical PCIe 4.0 limit. These results demonstrate that HeteroCache is highly scalable and does not introduce severe system-level memory or I/O bottlenecks when extending to long contexts.

Context Length	CPU Memory Peak	Bandwidth Utilization
32K	5.85 GB	16.91 GB/s
64K	9.88 GB	19.64 GB/s
128K	17.95 GB	21.07 GB/s

Table 7: Scaling of peak CPU memory footprint and PCIe 4.0 bandwidth utilization across different context lengths.

D.3 Multi-Needle Retrieval Evaluation

While the standard NIAH evaluation demonstrates HeteroCache's fundamental capability to retain critical information, it primarily assesses single-needle retrieval and may not fully reflect the complexities of multi-hop reasoning or the presence of adversarial distractors. To address this and avoid overgeneralizing single-needle performance, we conducted additional experiments using the more challenging U-NIAH (Gao et al., 2026) for multi-needle evaluation.

Following the exact same configuration for the Llama-3.1-8B-Instruct model used in our main experiments, we measured the performance using the ROUGE-1 metric across various context lengths. As shown in Table 8, HeteroCache maintains a highly competitive performance compared to the FullAttention baseline. These results confirm that our dynamic retrieval mechanism successfully preserves complex, reasoning-centric dependencies in long-context environments.

Method	16K	32K	64K	128K
FullAttention	7.379	3.338	1.306	2.195
HeteroCache	7.312	3.327	1.213	2.195

Table 8: ROUGE-1 scores on the multi-needle U-NIAH benchmark across different context lengths.

D.4 Detailed Latency Analysis

Retrieval Frequency and Tail Latency. A potential concern with dynamic retrieval mechanisms is the risk of excessive CPU-GPU communication causing severe latency spikes. However, our fine-grained tracking effectively minimizes unnecessary data fetching. To comprehensively evaluate performance during long context reasoning, we extended the testing window to a continuous generation of 1K tokens. In this configuration, the retrieval-trigger rate is only 0.823%.

To further verify the stability of the generation, we report the latency of the tail (P95 and P99) across various context lengths in Table 9. The results indicate that the worst-case decoding steps remain highly efficient. These results thoroughly demonstrate that the system maintains an extremely low stall rate and stable low latency even during prolonged generation periods, ensuring that the asynchronous retrieval mechanism does not introduce severe stalling.

Context Length	P95 Latency (s)	P99 Latency (s)
32K	0.068	0.074
64K	0.092	0.108
96K	0.095	0.114
128K	0.116	0.123
160K	0.118	0.126
192K	0.090	0.127
224K	0.096	0.132

Table 9: Tail decoding latency (P95 and P99) across various context lengths.

Performance under Constrained Interconnects. Whether the CPU↔GPU fetch overhead can be fully hidden depends heavily on the interconnect bandwidth. To validate the practicality of HeteroCache under hardware-constrained scenarios, we evaluated its performance on a consumer-grade setup featuring an NVIDIA GeForce RTX 4090 GPU and an Intel Xeon Gold 5218R processor. This setup utilizes a PCIe Gen 3 x16 interface, which restricts the theoretical bidirectional bandwidth to approximately 15.75 GB/s.

As shown in Table 10, despite the restricted in-

terconnect bandwidth, HeteroCache maintains prefill and decode times that are nearly identical to the FullAttention baseline across various sequence lengths. This demonstrates that its asynchronous, infrequent retrieval strategy effectively mitigates I/O contention, allowing it to accelerate inference even on standard consumer-grade hardware.

Method	4K (Pre/Dec)	8K (Pre/Dec)	16K (Pre/Dec)	32K (Pre/Dec)
FullAttention	0.497s / 0.038s	1.169s / 0.041s	2.496s / 0.039s	4.943s / 0.042s
CAKE	0.624s / 0.041s	1.332s / 0.038s	2.757s / 0.037s	5.008s / 0.039s
HeteroCache	0.502s / 0.039s	1.185s / 0.040s	2.508s / 0.038s	4.951s / 0.039s

Table 10: Prefill and Decode latency evaluated on NVIDIA GeForce RTX 4090.

E Statement on the Use of LLMs

We report on the use of LLMs in the preparation of this paper. The use of LLMs was strictly limited to the role of a general-purpose writing assistant. Specifically, we used these tools to proofread, correct grammatical errors, and rephrase sentences to improve clarity and readability. The LLMs did not contribute to the core scientific aspects of this work, such as research ideation, experimental design, or the generation of substantive content.