Tactic: Thresholding Accumulated Weight Via Clustering and Curve Fitting for Sparse Attention in Long-Context Models

Anonymous Authors[1]

Abstract

Long-context models are increasingly important in daily applications, but their inference is hindered by inefficiencies caused by loading large KV Cache during decoding. Prior work has shown that attention is inherently sparse, with a small subset of tokens significantly influencing model output. To utilize sparsity, most existing methods rely on a predefined token budget for attention computation. However, our analysis reveals that attention sparsity is inherently context-dependent, varying significantly with the specific generation context and different types of tasks. This variability highlights the need for a dynamic token budget and exposes the limitations of current methods in adapting to changing sparsity patterns. To address these challenges, we propose Tactic, a context-adaptive sparse attention mechanism for efficient and accurate long-context model inference. The key insight behind Tactic is to focus on the accumulated attention score to dynamically adjust the token budget, rather than relying on a predefined fixed budget. Since computing attention weights for the entire sequence is computationally expensive, Tactic introduces a distribution fitting technique to efficiently estimate attention weights with minimal computation.

1. Introduction

Large language models (LLMs) have become an integral part of daily life, powering tools such as conversational assistants, document analysis systems, and search engines. The increasing demand for multi-turn conversations and large document processing has greatly expanded the context length for LLM, growing from thousands of tokens to as much as 1 million in recent models. (Liu et al., 2024b)

However, the inference procedure of long-context LLMs poses significant challenges, primarily due to the increasing size of the Key-Value (KV) cache (Zhao et al., 2024). The memory requirements of KV cache scale proportionally with the context length. For example, Llama3-8B model with an input sequence length of 128K requires up to 16GB of KV cache storage.[1] Furthermore, since the entire KV cache must be loaded for every generated token, this repeated loading becomes a major bottleneck, significantly increasing latency. In fact, loading the large KV cache can account for over 50% of the total latency during auto-regressive decoding, significantly impeding the efficiency of large-scale serving systems. (Tang et al., 2024)

Recent research reveals the inherent sparsity of attention mechanisms, where only a small subset of tokens significantly influences the model output (Zhang et al., 2023; Liu et al., 2023). Building on this insight, several methods have been proposed to reduce KV cache loading costs by selecting a fixed number of critical tokens for attention computation. (Liu et al., 2024a; Tang et al., 2024; Zhang et al., 2023; Xiao et al., 2023).

While these methods are simple and effective in some scenarios, they fail to account for the variability in attention sparsity, which is closely tied to the specific generation context and task type. Our observation reveals that attention sparsity adapts dynamically based on the context: for instance, decoding for language coherence often focuses on a small set of local tokens, while tasks involving reasoning with rich semantic content require attention to a broader set of tokens. This dynamic behavior becomes even more apparent in real-world inference tasks, where the diversity of task requirements, such as the significantly higher token demands of coding tasks compared to retrieval tasks, highlights the limitations of fixed token budgets. As a result, attention sparsity is inherently context-sensitive, rendering fixed-budget methods suboptimal in both accuracy and efficiency.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]KV cache size = 2(K and V) * Number of Layer * Sequence length * Number of Heads * Head Dimensions * FP16 = 2*32*128K*8*128*2 bytes = 16GB

To address the limitations of fixed token budget methods, we propose Tactic, a model-agnostic, training-free, and context-adaptive sparse attention mechanism to enhance the efficiency and accuracy of long-context LLM inference. The key insight behind Tactic is that a preset fixed token budget cannot generalize to the variability in context-adaptive sparsity. Instead, Tactic uses accumulative attention weight to dynamically determine the token budget on-the-fly. Specifically, Tactic selects the Top-K tokens that cumulatively reach a threshold T of accumulated attention weight, where K is derived from T.

However, directly calculating the attention weight requires loading the entire KV cache, which contradicts the goal of leveraging sparsity for efficiency. To tackle this challenge, Tactic operates in three stages: (1) Clustering: during the prefill phase, Tactic applies K-Means Clustering to group all keys based on their similarity. (2) Querying: In the decode phase, Tactic computes the dot product between the query and cluster centroids, generating a list of keys ranked by relevance. (3) Distribution Fitting: Using the key list, Tactic samples a small subset of keys to fit the distribution of attention weights. By loading only a small subset of keys, Tactic predicts the number of tokens needed to meet the desired threshold of accumulative attention weight at a very low cost. This approach identifies the minimal set of tokens needed to maintain model performance, significantly reducing KV cache access.

Our experimental results show that Tactic achieves superior and consistent accuracy compared to existing systems including StreamingLLM (Xiao et al., 2023) and Quest (Tang et al., 2024), offering a more effective solution for long-context LLM inference in accuracy-sensitive applications.

In summary, we contribute the following:

- A detailed analysis of the dynamic nature of attention sparsity across contexts.
- **Tactic**, a context-adaptive attention algorithm that uses clustering and distribution fitting to dynamically determine the token budget for high accuracy long context inference.
- A comprehensive evaluation of Tactic, demonstrating Tactic consistently achieves high accuracy within same token budgets.

2. Background & Motivations

2.1. Large Language Models

The core component of Large Language Models is the transformer blocks. Each transformer block consists of an attention layer and a feed-forward layer. In each attention layer, input embeddings are first projected into Query (Q), Key (K), and Value (V) vectors. The model then takes the dot product of the Q vector with the K vectors to measure their similarity, followed by a softmax function to generate *attention weights*. For a given Q vector, the attention weight a_i for the i-th token can be formulated as $softmax(\frac{QK^\top}{\sqrt{d}})_i$. These weights are then multiplied by the V vectors to produce the final output. This result is passed to the feed-forward layer, which, together with residual connections and layer normalization, refines the final token representation.

On the request level, LLM inference consists of two phases: the prefill phase and the decode phase. In the prefill phase, all input tokens are processed simultaneously to generate Q, K, and V vectors, with the K and V vectors stored as KV cache to avoid recomputation. In the decode phase, only the last generated token is processed, with its Q, K, and V vectors computed. The current Q vector interacts with the cached K and V vectors to generate the output.

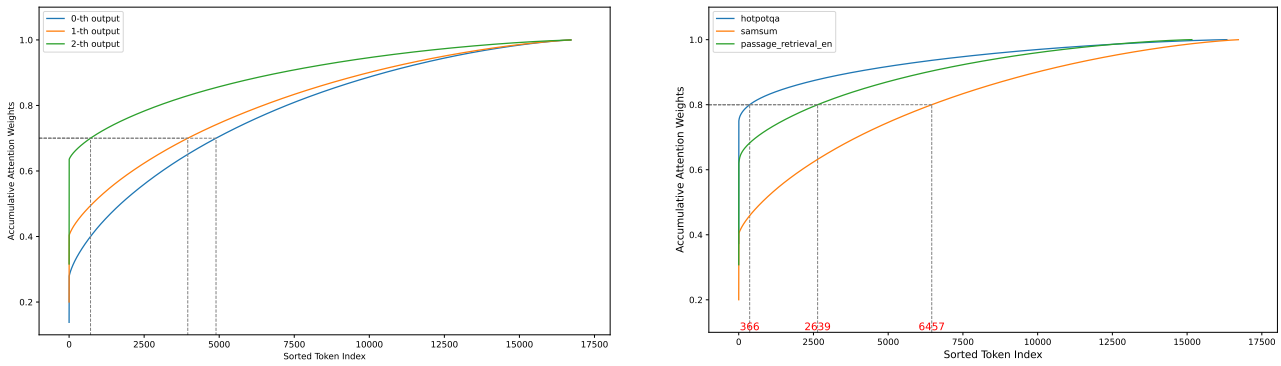
Unlike the prefill phase, the decode phase is repeated for every generated token, making it the dominant factor during inference. In long-context scenarios, this issue is exacerbated as all cached K and V vectors must be loaded at each decoding step to compute self-attention.

2.2. Long-Context Models

The demand for long-context models has driven advancements in extending the context window of large language models. Techniques like Rotary Position Embeddings (Su et al., 2023) have enabled a significant expansion in context length, such as increasing LLaMA-2's window to 32K in LongChat (Li et al., 2023) and 128K in Yarn-Llama-2 (Peng et al., 2023), and even 1 million recently (Liu et al., 2024b). However, as context length grows dramatically, managing the KV cache becomes an increasingly significant bottleneck, with loading the KV cache contributing substantially to latency and sometimes accounting for nearly half of the total decode time (Tang et al., 2024).

2.3. Attention Approximation

Prior work has shown that only a small subset of tokens have a significant impact on the model output (Zhang et al., 2023; Xiao et al., 2023), indicating that the attention operation in LLMs is inherently sparse. In order to leverage the sparsity and reduce the significant time and memory burden imposed by the KV cache, many methods have been proposed to approximate attention by compressing the size of this cache. The most common strategy is to use a fixed token budget to retain those tokens identified as most relevant based on their attention weights. StreamingLLM (Xiao et al., 2023) retains only a fixed number of initial and the most recent tokens for attention calculations. SparQ (Ribar



(a) (b)

Figure 1. Attention Sparsity Variation. These two figures show the cumulative density function (CDF) of sorted attention weights of a specific layer and head. Input data is from LongBench (Bai et al., 2024). (a) During inference: The CDF demonstrates significant variation during inference process. (b) Different Contexts: The CDF in different contexts demonstrates significant variation.

et al., 2023) approximates attention weights by performing channel pruning. Similarly, Quest (Tang et al., 2024) evaluates the importance of fixed chunks of tokens at decode time, selecting a predefined number of chunks based on their relevance. These methods use the same token budget for all attention heads and layers.

Some recent work has highlighted the limitations of fixed token budget methods and proposed adaptive token selection strategies to better capture the variability of sparsity. PyramidKV (Cai et al., 2024) and SqueezeAttention (Hooper et al., 2024) focus on dynamically allocating KV cache budgets across layers, optimizing memory usage at the layer level. In contrast, Ada-KV (Feng et al., 2024) targets variability across attention heads by using an adaptive budget allocation algorithm that aligns memory usage with the characteristics of individual heads.

2.4. Motivations

2.4.1. Variation of Attention Sparsity

While previous works have demonstrated the existence of attention sparsity in LLMs, most methods still assign the same token budget within the same head or layer and rely on a fixed overall token budget. Our observations reveal that sparsity varies significantly with the inference process and input context, as illustrated in Figure 1. These findings indicate that focusing solely on sparsity variation at the head or layer level is insufficient, underscoring the need for a more flexible, context-adaptive approach to token selection that dynamically adjusts to varying sparsity patterns during inference.

2.4.2. Token budget is context-dependent

Due to varying sparsity in contexts, fixed token budget strategies, although simple and intuitive, are suboptimal in the trade-off between efficiency and accuracy. These methods lack a clear rationale for determining the appropriate token budget, often excluding critical tokens or retaining irrelevant ones, ultimately leading to excessive memory movement and inconsistent output qualities. As shown in (b) of Figure 1, if we set the token budget to be 366, then it is insufficient for task *samsum* and *passage_retreival_en*. However, setting the token budget to 6457 results in minimal improvement on task *hotpotqa* and *passage_retreival_en*, while significantly increasing costs.

2.4.3. Accumulative attention weights is context-adaptive

To achieve consistently good performance over different contexts without including irrelevant tokens, we need to select tokens based on some sort of performance requirement. The output of self-attention is of form $\sum_i a_i \cdot V_i$, where a_i is the attention weight of i-th token. Therefore,accumulative attention weights is directly related to attention output. By setting a threshold of accumulative attention weights, no matter how the distribution of attention weights is, we can select adequate amount of tokens without being wasteful.

3. Methodology

3.1. Weight Percentage Is The Goal

To be context-adaptive, Tactic targets *weight percentage* in token selection, which can be formulated as:

$$P_w(q, K_s, K) = \frac{\sum_{k \in K_s} \exp(k^\top q)}{\sum_{k \in K} \exp(k^\top q)}. \quad (1)$$

K is the set of all previous key vectors, K_s is the subset of keys selected to perform self-attention. *weight percentage* is directly linked to the self-attention output . The higher the value is, the more accurate the self-attention is. Improv-

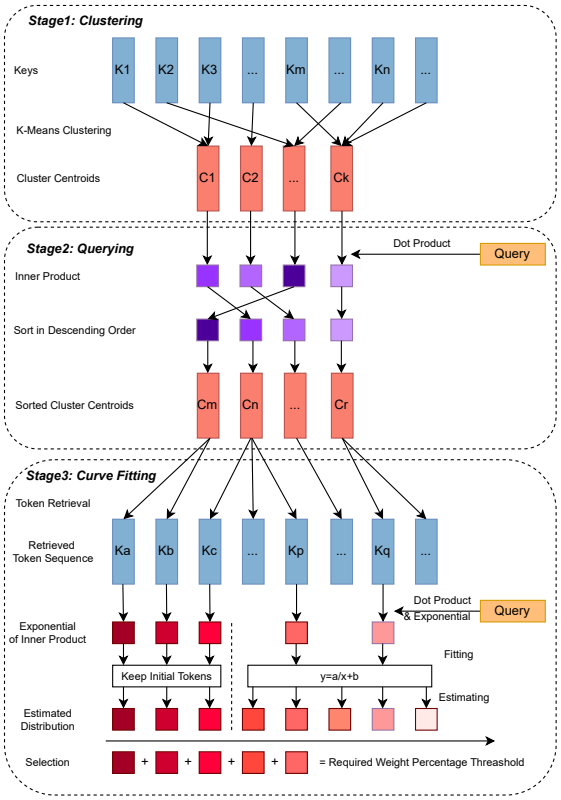


Figure 2. Tactic operates in three sequential stages. (1) In the Clustering stage, key vectors are grouped into clusters using K-means clustering; (2) In the Querying stage, cluster importance is computed as the dot product between the cluster centroid and the query, then clusters are sorted by criticality; (3) In the Curve Fitting stage, tokens from the sorted clusters are processed. The initial tokens are retained directly, while others are modeled using an estimated attention weight distribution $y = \frac{a}{x} + b$.

ing *weight percentage* intrinsically requires dynamic token selection for different input contexts.

Given a threshold of *weight percentage*, the Optimal Selection Strategy to reach the threshold is to sort all the tokens based on their attention weights in descending order, and then select tokens sequentially until the desired *weight percentage* is reached.

To further demonstrate that targeting at high *weight percentage* can achieve consistently good performance across different tasks, we evaluated Llama-3.1-8B-Instruct with different values of *weight percentage* on KL-Divergence and RULER (Hsieh et al., 2024), using the optimal selection strategy. KL-Divergence is calculated as the cross-entropy between the probability distribution of the model output under the selected tokens and the distribution under full

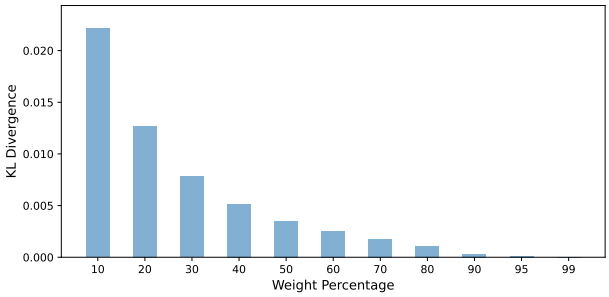


Figure 3. The relationship between KL Divergence and *weight percentage*, evaluated on the optimal selection strategy. The figure shows that KL Divergence decreases as *weight percentage* increases, highlighting its direct link to output quality.

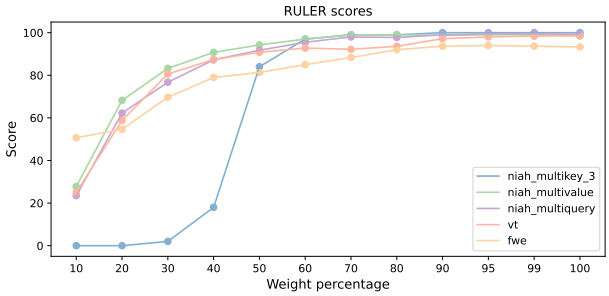


Figure 4. The relationship between 5 RULER task scores and *weight percentage*, evaluated on the optimal selection strategy.

attention, minus the entropy of the full attention distribution. A lower KL-Divergence indicates that the model's output closely matches that of full attention. As shown in Figure 3, higher *weight percentage* yields closer outputs to full attention. Figure 4 shows a similar trend, where the scores of RULER tasks, such as NIAH (Needle-In-A-Haystack), VT (Variable Tracing) and FWE (Frequent Word Extraction), improves as *weight percentage* increases.

However, performing the optimal selection strategy requires the knowledge of the attention weights in advance, which is impossible before the attention computation. Additionally, calculating the exact value of *weight percentage* necessitates the sum of the exponential weights of all tokens, which is also infeasible.

3.2. Algorithm Overview

To address these challenges, we propose Tactic, an algorithm that efficiently and accurately selects a minimal set of tokens in the KV cache for the self-attention given a *weight percentage* requirement. Figure 2 provides an overview of the workflow of Tactic. Tactic operates in 3 stages: Clustering, Querying and Curve Fitting. In the Clustering stage, Tactic performs K-Means Clustering algorithm on key vec-

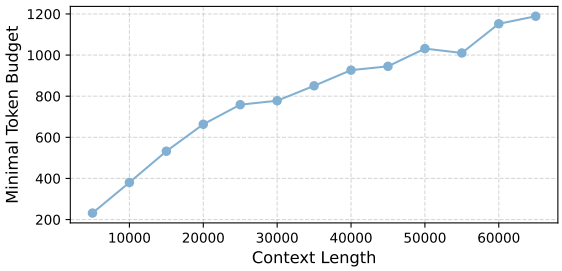


Figure 5. The relationship between Context Length and Minimal Token Budget. Minimal Token Budget is obtained by fully sorting the attention weights of each token at the first decoding step, sequentially selecting tokens until the cumulative attention weight reaches 90%, and then averaging the results across heads and layers. The figure demonstrates that the Minimal Token Budget scales linearly with Context Length.

tors produced from the prefill phase to group similar tokens. In the Querying stage, Tactic ranks clusters of the key vectors based on the dot product between cluster centroids and the given query vector in the decode phase. In the Curve Fitting Stage, we model the distribution of exponential weights with a fitted curve, which allows us to approximate the sum of exponential weights at a low cost and determine the optimal number of tokens to meet the desired *weight percentage* threshold.

3.3. Clustering for Token Organization

To organize tokens for efficient downstream querying, we perform K-means Clustering on the key vectors for each head in every layer after the prefill phase. The dot product serves as the similarity metric, with larger values indicating shorter distances. Clustering begins by randomly sampling N data points as the initial cluster centroids. In each iteration, the dot product between each data point and all centroids is computed, and each data point is assigned to the centroid with the highest dot product. After the assignment step, the centroids are updated as the mean of the key vectors assigned to each cluster. This process repeats until convergence or until a maximum of 10 iterations is reached. Unlike the common practice of running multiple clustering processes and selecting the best result, we perform the clustering only once to minimize computational overhead.[2]

3.4. Querying for Critical Clusters

Once the key vectors are organized into clusters, the querying stage identifies critical clusters for a given query vector Q in the decode phase. The criticality of each cluster is determined by the dot product between Q and each cluster centroid. This process produces a sequence of clusters sorted by the criticality. By extracting tokens from these clusters, we obtain a sequence of tokens which approximates the optimal sorted token sequence.

3.5. Fitting Attention weight distribution for Token Budget Optimization

Algorithm 1 Estimating Token Budget via Curve Fitting

Input: Token sequence unpacking from sorted clusters $\{x_1, x_2, \ldots, x_n\}$, query Q, weight percentage threshold T, initial token count N
Output: Token budget k

Compute μ_1 and μ_2 as the means of $\exp(x_i \cdot Q)$ within fixed windows around p_1 and p_2. Solve for parameters a and b in $y = a/x + b$ the two data points.

Initialize array w_i to store the simulated attention scores for all tokens
for $i = 1$ to n **do**
    If $i \leq N$, $w_i = \exp(\text{dot product of } x_i)$
    Else, $w_i = a/i + b$
end for
Compute the minimal k such that the cumulative sum $\sum_1^k w_i \geq T \cdot \Sigma_1^n w_i$.

return k

With the sequence of tokens obtained from the relevant clusters, the next step is to determine the token budget required to meet a specific attention weight threshold. To achieve this, we model the distribution of attention weights across the sorted tokens with a fitted curve, enabling us to dynamically adjust the token budget based on the desired attention *weight percentage* threshold.

Directly fitting the attention weight distribution is challenging, as the calculation depends on the sum of the exponential values of the dot products between the keys and the query, which cannot be determined in advance. An additional challenge is the need for efficient curve fitting, as this must be done for each query vector. We address these challenges by modeling the distribution of the exponential values of the dot products for each token using a lightweight function $y = \frac{a}{x} + b$, where a and b are parameters to be determined. To estimate these parameters, we select two locations based on their relative positions within the sequence, and calculate the average of several tokens within a small context window around the two locations. This approach provides sufficient data to solve for a and b, effectively capturing the overall trend of the distribution.

[2]We evaluated the cluster quality (Average radius of all clusters) across different random initializations and number of iterations. Results from multiple initializations and additional iterations showed no improvement.

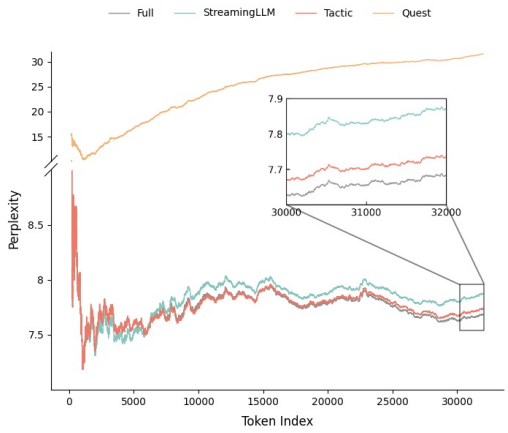


Figure 6. Perplexity comparison of Full Attention, Quest, StreamingLLM and Tactic methods on the PG19 dataset. Tactic significantly outperforms all of the other methods, maintaining perplexity close to Full Attention throughout the token range.

nd Tactic methods on the PG19 dataset. Tactic significantly outperforms all of the other ghout the token range

In the initial 128 tokens, there may be a significant concentration of critical tokens whose attention weights disproportionately contribute to the sum, a phenomenon known as the attention sink. (Xiao et al., 2023) These tokens act as outliers, and their values cannot be effectively predicted using the general distribution. To address this, we directly calculate the exponential values of the dot products for these tokens, rather than relying on the fitted curve. This ensures that the attention scores for the most critical tokens are accurately captured, while the remaining tokens are efficiently modeled through the fitted curve. A detailed description of the Curve Fitting stage is provided in Algorithm 1.

4. Experiments

4.1. Setting

We evaluate Tactic on the PG19 language modeling dataset (Rae et al., 2019), six tasks from the LongBench dataset(Bai et al., 2024), including HotpotQA(Yang et al., 2018), NarrativeQA(Kočiský et al., 2018), Qasper(Dasigi et al., 2021), TriviaQA(Joshi et al., 2017), MultifieldQA(Bai et al., 2024), and Passage Retrieval(Bai et al., 2024). Additionally, we conduct experiments on the Ruler benchmark(Hsieh et al., 2024), using 100 examples for each dataset.

We use the Llama-3.1-8b-Instruct model (Grattafiori et al., 2024) for our evaluation and compare our method with the most popular topK KV cache eviction algorithms, StreamingLLM (Xiao et al., 2023) and Quest (Tang et al., 2024). Note that we do not skip the first two layers, applying the algorithms to all layers of the model. To ensure consistency, we set the page size in Quest and the cluster size in our method to 32. For the clustering process, we limit the maximum number of iterations to 10 and perform a single clustering step, without repeating or selecting the best result.

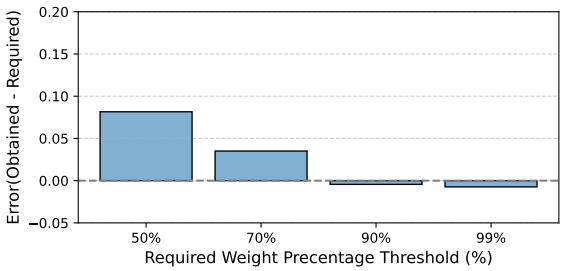


Figure 7. We compared the absolute error, defined as the difference between the obtained weight percentage of Tactic and the targeted weight percentage threshold. The relative error remains well below 0.01 for thresholds of 90% and 99%.

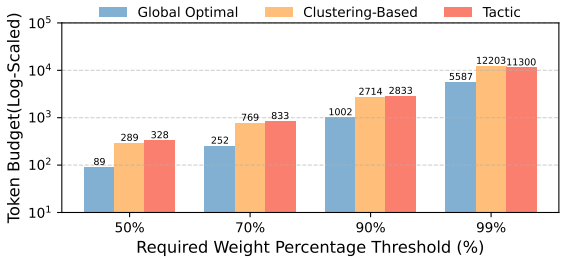


Figure 8. We compared the number of token budgets under three settings: global optimal, clustering-based optimal, and Tactic estimation. Global Optimal is obtained using Optimal Selection Strategy. Clustering-based optimal is obtained by selecting sequentially from the token list generated by the Querying stage until the desired threshold is reached. The Tactic estimation is the output token budget provided by the Curve Fitting stage. The results show that the estimated token budget provided by Tactic closely matches the Clustering-base optimal, demonstrating the effectiveness of our curve fitting technique.

4.2. Accuracy Evaluation

4.2.1. Effectiveness of Token Budget Estimation

We evaluate the effectiveness of our token budget estimation method on the PG19 dataset. The primary objective is to assess how closely our method aligns with the target cumulative attention weight percentage and how accurately it estimates the corresponding token budget.

To test the alignment between the target and obtained weight percentages, we set specific target weight percentage thresholds and measure the actual cumulative attention weight achieved. Figure 7 shows the relative error between the target and obtained weight percentages. Our results demonstrate that the estimation method consistently achieves low

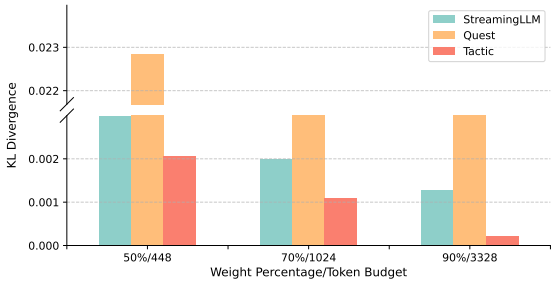


Figure 9. KL-Divergence with full attention evaluation of Quest, StreamingLLM and Tactic methods on the PG19 dataset. Tactic maintains the most accurate output in three configurations.

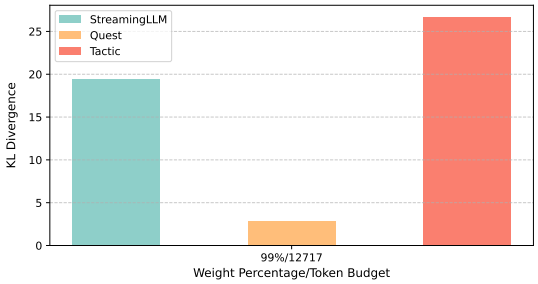


Figure 10. Token match length evaluation of Quest, StreamingLLM and Tactic methods on the PG19 dataset. Tactic achieves the highest match length against baselines.

relative errors across a wide range of thresholds, confirming the reliability of our approach in capturing the intended attention distribution.

To analyze the token budget estimation process, we set specific target weight percentages and compare the estimates obtained from three methods, the global optimal, clustering optimal and ours. The first method calculates the global optimal token budget, which is the theoretically minimal budget required to reach the target cumulative attention weight. This is achieved by directly summing tokens in descending order of their attention weights. The second method derives the token budget based on the clustering approach, summing the real attention weights of token sequence unpacked from the sorted clusters. Figure 8 illustrates the comparison among these three methods, highlighting how closely the clustering and estimation approaches approximate the global optimal while maintaining computational efficiency.

4.2.2. Language Modeling on PG19

To evaluate the effectiveness of our clustering and querying mechanisms, we first assess language modeling perplexity, KL-Divergence relative to the full attention, and token match under maximum sampling, using the PG19 test set(Rae et al., 2019).

For the perplexity evaluation, we select sequences of 32k tokens from PG19. These tokens are fed to the model sequentially, one at a time, and perplexity is computed at each position. As our primary focus in this evaluation is the clustering and querying components, we use the top-K version of Tactic without simulating the number of tokens. The KV cache budget is uniformly set to 1024 tokens for Tactic, Quest, and StreamingLLM. As shown in Figure 6, Tactic achieves perplexity nearly indistinguishable from full attention, significantly outperforming both StreamingLLM and Quest.

For the KL-Divergence evaluation, we include all texts in PG19 with number of tokens larger than 32k. In the prefill stage, we truncate the input to 32k tokens and feed it into the model. In the decode stage, we feed the following tokens one by one and collect the output logits of each decode step. We collect 64 decode steps in total. As shown in Figure 9, Tactic achieves the most accurate output compared to StreamingLLM and Quest.

For the token match evaluation, we include all texts in PG19 with number of tokens larger than 32k. Prefill stage is the same as in KL-Divergence evaluation. In decode stage, we choose the next token using maximum probability sampling and compare the token with that of full attention. If they are the same, then we continue to the next decode step. Otherwise we terminate the generation process and store the length of output tokens. We set the maximal output length to be 128. We only evaluate on the configuration where *weight percentage* equals to 99%, because token match requires high accuracy. As shown in Figure 10, Tactic achieves the highest match length against baselines.

To validate that Tactic can outperform the baselines on long-context tasks, we test our method alongside other baselines on 6 tasks from the LongBench (Bai et al., 2024) and 10 tasks from the RULER(Hsieh et al., 2024).

4.2.3. Results on LongBench

For the LongBench benchmark, we include tasks NarrativeQA, HotpotQA, Qasper, TriviaQA, MultiFieldQA, and PassageRetrieval, spanning a wide range of scenarios such as single-document QA, multi-document QA, few-shot learning and synthesis tasks. For each dataset, we first evaluate Tactic using weight percent thresholds of 50%, 70%, and 90%. The average number of tokens selected at each threshold serves as the token budget for evaluating Quest and StreamingLLM.

The results, shown in the Figure 11, indicate that Tactic consistently outperforms all other baselines across all attention *weight percentage* thresholds. At the 90% threshold, Tactic achieves near-optimal performance comparable to

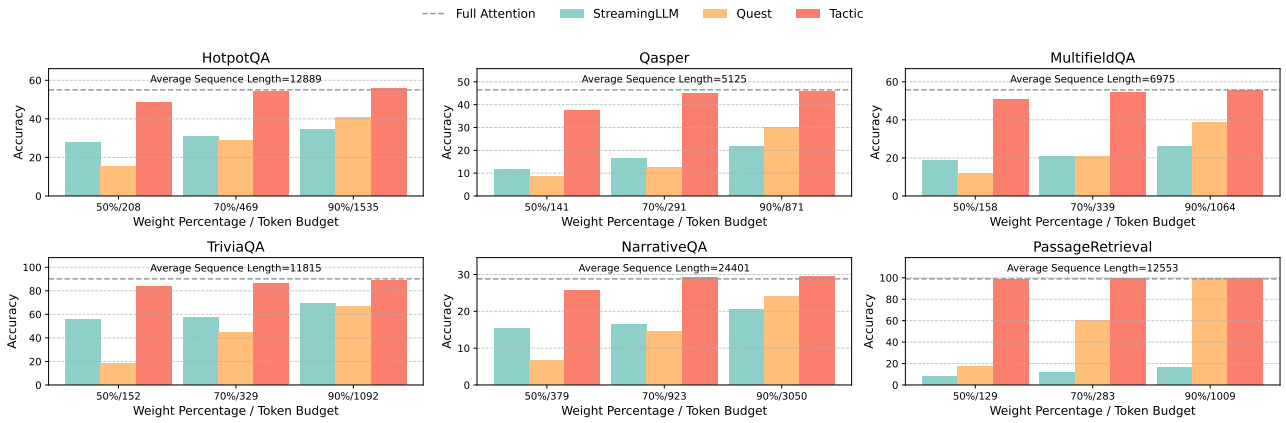


Figure 11. Accuracy comparison across LongBench tasks. The results are evaluated for different weight percentages (50%, 70%, 90%) with corresponding token budgets annotated.

full attention, while operating with different token budgets tailored to each dataset.

4.2.4. Results on RULER

Methods	Config	16K	32K	64K	Avg.
Llama-3.1-8B-Instruct	Full	91.6	87.4	84.7	87.9
Tactic	50%	80.0	76.1	68.9	75.0
Tactic	70%	87.3	80.0	75.8	81.0
Tactic	90%	87.7	81.9	79.0	82.8
Quest	448	54.1	40.0	34.1	42.7
Quest	1024	64.2	51.8	46.0	54.0
Quest	3328	74.5	64.6	60.6	66.6
StreamingLLM	(4,444)	18.8	2.9	18.1	13.3
StreamingLLM	(4,1020)	20.5	3.0	18.9	14.1
StreamingLLM	(4,3324)	29.6	3.5	21.0	18.0

Table 1. Performance comparison on RULER. Each score is computed by averaging accuracy of 10 tasks in RULER. More details on scores are in AppendixA

For the RULER benchmark, we include NIAH(w/o single-NIAH tasks), QA, VT and Aggregation (FWE/CWE) tasks. As shown in Table 1, Tactic consistently outperforms Quest and StreamingLLM in each configuration in terms of average accuracy. Tactic achieves similar performance as full attention with 90% *weight percentage*.

5. Conclusion

References

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. LongBench: A bilingual, multitask benchmark for long context understanding. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

Cai, Z., Zhang, Y., Gao, B., Liu, Y., Liu, T., Lu, K., Xiong, W., Dong, Y., Chang, B., Hu, J., and Xiao, W. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling, 2024. URL https://arxiv.org/abs/2406.02069.

Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., and Gardner, M. A dataset of information-seeking questions and answers anchored in research papers. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365.

Feng, Y., Lv, J., Cao, Y., Xie, X., and Zhou, S. K. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference, 2024. URL https://arxiv.org/abs/2407.11550.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Hooper, C., Kim, S., Mohammadzadeh, H., Maheswaran, M., Paik, J., Mahoney, M. W., Keutzer, K., and Gholami, A. Squeezed attention: Accelerating long context length llm inference, 2024. URL https://arxiv.org/abs/2411.09688.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. Ruler: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Barzilay, R. and Kan, M.-Y. (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023.

Li, D., Shao, R., Xie, A., Sheng, Y., Zheng, L., Gonzalez, J. E., Stoica, I., Ma, X., and Zhang, H. How long can open-source llms truly promise on context length?, June 2023. URL https://lmsys.org/blog/2023-06-29-longchat.

Liu, G., Li, C., Zhao, J., Zhang, C., and Guo, M. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression, 2024a. URL https://arxiv.org/abs/2412.03213.

Liu, X., Yan, H., Zhang, S., An, C., Qiu, X., and Lin, D. Scaling laws of rope-based extrapolation, 2024b. URL https://arxiv.org/abs/2310.05209.

Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time, 2023. URL https://arxiv.org/abs/2305.17118.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models, 2023.

Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling, 2019. URL https://arxiv.org/abs/1911.05507.

Ribar, L., Chelombiev, I., Hudlass-Galley, L., Blake, C., Luschi, C., and Orr, D. Sparq attention: Bandwidth-efficient llm inference, 2023.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Tang, J., Zhao, Y., Zhu, K., Xiao, G., Kasikci, B., and Han, S. Quest: Query-aware sparsity for efficient long-context llm inference, 2024. URL https://arxiv.org/abs/2406.10774.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv*, 2023.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z., and Chen, B. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023.

Zhao, Y., Lin, C.-Y., Zhu, K., Ye, Z., Chen, L., Zheng, S., Ceze, L., Krishnamurthy, A., Chen, T., and Kasikci, B. Atom: Low-bit quantization for efficient and accurate llm serving, 2024. URL https://arxiv.org/abs/2310.19102.

A. Ruler Scores Details

We present the details of Ruler evaluation here.

Methods	Config	16K	32K	64K	Avg.
Llama-3.1-8B-Instruct	Full	99.7	99.4	97.9	99.0
Tactic	50%	88.6	89.9	80.8	86.4
Tactic	70%	96.3	94.8	90.5	93.9
Tactic	90%	97.9	98.0	96.4	97.4
Quest	448	56.7	36.6	29.3	40.9
Quest	1024	66.1	53.1	41.4	53.5
Quest	3328	77.0	67.7	61.3	68.7
StreamingLLM	(4,444)	3.2	0.50	2.0	1.9
StreamingLLM	(4,1020)	7.0	0.50	2.0	3.2
StreamingLLM	(4,3324)	18.8	2.1	5.3	8.7

Table 2. Performance comparison of NIAH tasks on RULER.

Methods	Config	16K	32K	64K	Avg.
Llama-3.1-8B-Instruct	Full	67.3	65.8	63.7	65.6
Tactic	50%	70.0	67.5	62.5	66.7
Tactic	70%	71.0	67.5	67.5	68.7
Tactic	90%	70.5	68.5	66.0	68.3
Quest	448	56.0	44.5	38.0	46.2
Quest	1024	66.5	57.0	53.0	58.8
Quest	3328	68.5	65.5	64.0	66.0
StreamingLLM	(4,444)	45.0	13.0	46.5	34.8
StreamingLLM	(4,1020)	44.0	13.5	46.5	34.7
StreamingLLM	(4,3324)	48.0	11.5	44.0	34.5

Table 3. Performance comparison of QA tasks on RULER.

Methods	Config	16K	32K	64K	Avg.
Llama-3.1-8B-Instruct	Full	99.7	98.8	97.6	98.7
Tactic	50%	70.4	72.4	75.4	72.7
Tactic	70%	87.6	87.6	84.6	86.6
Tactic	90%	96.4	93.4	90.0	93.3
Quest	448	76.2	71.6	64.4	70.7
Quest	1024	81.6	74.0	79.2	78.3
Quest	3328	97.6	92.4	88.0	92.7
StreamingLLM	(4,444)	3.6	0.0	0.8	1.5
StreamingLLM	(4,1020)	7.8	0.0	2.6	3.5
StreamingLLM	(4,3324)	23.2	0.0	4.6	9.3

Table 4. Performance comparison of VT task on RULER.

Methods	Config	16K	32K	64K	Avg.
Llama-3.1-8B-Instruct	Full	79.2	54.1	43.5	58.9
Tactic	50%	73.2	51.9	42.2	55.8
Tactic	70%	80.7	51.8	42.7	58.4
Tactic	90%	74.8	49.2	43.2	55.7
Quest	448	34.7	28.1	26.9	29.9
Quest	1024	48.4	32.4	33.9	38.2
Quest	3328	62.6	42.2	42.0	49.0
StreamingLLM	(4,444)	39.4	0.0	38.8	26.1
StreamingLLM	(4,1020)	37.0	0.3	41.5	26.3
StreamingLLM	(4,3324)	41.4	0.8	45.3	29.2

Table 5. Performance comparison of Aggregation tasks on RULER.