CRITIPREFILL: A SEGMENT-WISE CRITICALITY-BASED APPROACH FOR PREFILLING ACCELERATION IN LLMS

Junlin Lv, Yuan Feng, Xike Xie, Xin Jia, Qirong Peng, and Guiming Xie
AI Center, Guangdong OPPO Mobile Telecommunications Corp.,Ltd.
Guangdong, China
{junlin.lv, yuanfeng.yf}@outlook.com, xkxie@hotmail.com
{jiaxin, pengjirong, xieguiming}@oppo.com

ABSTRACT

Large language models have achieved notable success across various domains, yet efficient inference is still limited by the quadratic computation complexity of the attention mechanism. The inference consists of prefilling and decoding phases. Although several attempts have been made to accelerate decoding, the inefficiency of the prefilling phase, especially for long-context tasks, remains a challenge. In this paper, we observe a locality in query criticality during the prefilling phase of long-context processing: adjacent query tokens tend to focus on similar subsets of the past Key-Value (KV) cache. Based on this observation, we propose CritiPrefill, a criticality-based segment-wise prefilling method. This method partitions the input sequence's queries and KV cache into segments and blocks, utilizing a segment-wise algorithm to estimate the query criticality. By pruning non-critical computations between query segments and cache blocks in the self-attention mechanism, the prefilling process can be significantly accelerated. Extensive evaluations on multiple long-context datasets show up to 2.7x speedup on Llama3-8B and 3.0x speedup on Yi-9B for 128K context length on a single A100 GPU, with minimal quality degradation.

1 INTRODUCTION

Large Language Models (LLMs) have been widely applied across various fields, showcasing impressive capabilities in handling long-context tasks such as long text comprehension (Laban et al., 2023), multi-turn dialogue QA (Yang et al., 2018), in-context learning (Dong et al., 2022), and agent tasks(Wang et al., 2024). The rapid advancement of open-source LLMs has significantly reduced the training costs for downstream applications. However, a key challenge remains in the quadratic inference cost of the self-attention mechanism in Transformer layers while processing long sequences (Wan et al., 2024; Dong et al., 2023; Zhou et al., 2024).

Typically, LLMs are composed of multiple Transformer layers, each containing a self-attention layer that significantly contributes to the inference cost in long-sequence scenarios (Wang et al., 2020). Inference within each self-attention layer consists of two phases: the *prefilling phase* and the *decoding phase*. During the prefilling phase, LLMs calculate the Key-Value (KV) cache for all input tokens and predict the first output token, which takes the majority of the computation cost during inference (Qin et al., 2024a). In the subsequent decoding phase, the model generates tokens autoregressively by leveraging the most recent query token along with the entire KV cache from all previous steps. While this phase involves less computation, it is primarily constrained by the input/output (I/O) latency of the KV cache (Leviathan et al., 2023), resulting in a memory-bound bottleneck.

To address this challenge, plug-and-play (P&P) methods have been extensively explored for seamless integration into any LLM, to reduce inference costs without the need of substantial fine-tuning expenses (Dao et al., 2022; Kwon et al., 2023; Kachris, 2024). Among those works, recent advances in efficient decoding (Feng et al., 2024; Li et al., 2024; Zhang et al., 2024) have made remarkable

Figure 1: Acceleration of CritiPrefill

Figure 2: Framework of CritiPrefill

progress, especially by identifying key subsets of KV caches based on the last query token at each decoding step (Tang et al., 2024). By pruning non-critical KV caches, these methods minimize redundant computations, significantly reducing cache I/O latency and accelerating the decoding phase.

However, in typical long-sequence scenarios, such as long-context QA, the input sequence is often much longer than the output sequence, resulting in a significantly higher prefilling time than the total decoding time. As shown in Figure 1, when the sequence length reaches 128K, the prefilling time can account for over 95.6% of the total inference time, indicating that the main bottleneck in long-sequence tasks lies in the prefilling phase. However, accelerating the prefilling phase remains a significant challenge due to its highly parallel yet computationally intensive nature. Existing methods for prefilling acceleration typically require architectural changes (Shen et al., 2021; Qin et al., 2024b; Yang et al., 2023) or extensive fine-tuning (Beltagy et al., 2020; Zaheer et al., 2020; Tworkowski et al., 2024), which limiting their integration with pretrained LLMs (Rasley et al., 2020; Shoeybi et al., 2019). This raises a critical question: *Can the P&P strategy for critical KV cache identification, which has promise in the decoding phase, be extended to the prefilling phase?*

Nevertheless, it is non-trivial to identify critical KV cache. The token-wise criticality estimation (Tang et al., 2024), while effective in decoding, introduces significant computational overhead, making it impractical for the compute-intensive prefilling phase. However, we observe a key locality pattern: *although the critical subset of the KV cache varies with different query tokens, neighboring query tokens tend to share similar critical KV caches.* Based on this observation, we propose the *CritiPrefill*, which leverages segment-wise query criticality estimation for pruning attention computation, thus accelerating the prefilling phase.

As outlined in Figure 2, the CritiPrefill partitions both query tokens and KV caches into fixed-size query segments and cache blocks, before the prefilling phase of each layer. A segment-wise algorithm then estimates the criticality of each cache block for each query segment. Additionally, inspired by inter-layer similarity in LLMs (Dai et al., 2019; Liu et al., 2023; Zhang et al., 2023), we propose a layer-fusion mechanism to further refine the final criticality score. By pruning non-critical computation between query segments and cache blocks in the self-attention mechanism, the CritiPrefill significantly accelerates the prefilling process. We evaluate CritiPrefill across various long-context QA tasks with different input lengths, showing substantial speedups with minimal accuracy loss. Additionally, tests on the "Needle-in-a-Haystack" task further confirm that CritiPrefill maintains information retrieval capacity while reducing prefilling time for long sequences, achieving up to a 2.7x speedup on Llama3-8B and 3.0x on Yi-9B using a single A100 GPU.

2 METHOD

2.1 QUERY CRITICALITY IN ATTENTION MECHANISM

Modern LLMs are generally constructed using multi-layer Transformer blocks, with the self-attention mechanism as the central operation. During the prefilling phase, the computation in each layer's self-attention is heavily dependent on the hidden states $X \in \mathbb{R}^{n \times d}$ of all input tokens from the previous layer, which are transformed into Query $Q \in \mathbb{R}^{n \times d}$, Key $K \in \mathbb{R}^{n \times d}$, and Value $V \in \mathbb{R}^{n \times d}$ matrices as follows:

$$Q = W_Q X; \ K = W_K X; \ V = W_V X \tag{1}$$

The output O is then computed using attention weights A: [1]

$$O = AV \text{ where } A = \text{softmax}(\frac{QK^T}{\sqrt{d}}) \tag{2}$$

The matrix multiplications in self-attention mechanisms result in a quadratic computational cost relative to sequence length n, which contributes to the significant computational load during the prefilling phase. In the decoding phase, LLMs use only the query of the last token $q = Q[-1]$ along with cached KV pairs to autoregressively generate the next token. Although the computational cost per decoding step is low, large sequence length n can lead to considerable KV cache I/O latency, creating a memory-bound bottleneck. Fortunately, due to the inherent sparsity of attention weights (Ge et al., 2023), only a small subset of critical KV cache significantly influences the output generation. Thus, excluding non-critical portions of the KV cache from self-attention computations does not degrade generation quality (Li et al., 2024). Furthermore, research on decoding speedup has shown that these critical KV cache subsets are query-dependent, a phenomenon referred to as query criticality (Tang et al., 2024). This means that each query token $q \in Q$ corresponds to a distinct critical subset of the KV cache. By identifying the relevant subset for the last query token in each decoding step, the decoding process can be significantly accelerated.

However, applying query criticality to the prefilling phase introduces unique challenges. First, the prefilling phase already incurs significant computational overhead, and an additional step to identify critical KV cache subsets would only increase this burden. Second, while token-wise query criticality reduces computation via sparsification, the resulting sparsity is unstructured, as different query tokens depend on varying critical KV cache subsets. This unstructured sparsity undermines the benefits of highly parallelized matrix multiplications in the prefilling phase, potentially decreasing the efficiency rather than achieving the intended acceleration.

2.2 Locality Pattern of Query Criticality

To leverage query criticality for computation pruning, we conduct a deliberated study focusing on the long-context question answering (QA) task. Our findings reveal a locality pattern in query criticality during the prefill process. Specifically, for a 4K-length QA context on the Llama3-8B model, we first identify the critical KV cache subset $\{\hat{K}_i, \hat{V}_i\}$ for each query $q_i \in Q$:

$$\langle \hat{K}_i, \hat{V}_i \rangle = \langle K[index_i], V[index_i] \rangle$$
$$\text{where } index_i = \text{argtop}_k(\text{softmax}(q_i K^T), k = 512) \tag{3}$$

Figure 3 visualizes the similarity of critical cache subsets for different queries q_i and q_j, using the metric $\frac{|\hat{K}_i \cap \hat{K}_j|}{k=512}$. The results show that nearby queries tend to share critical KV cache subsets, as evident by the higher similarity near the diagonal. As the distance between queries grows, this similarity declines. This criticality locality pattern suggests that adjacent queries show closer criticality, allowing for collective prediction of query segments. Thus, by leveraging the locality, it can significantly reduce the computational cost of criticality estimation, while exploiting the structured sparsity from segment-wise criticality.

2.3 Proposed Methods

In this section, we present CritiPrefill, a segment-wise criticality-based method for accelerating the prefilling process by leveraging the locality pattern in query criticality. This approach improves efficiency by estimating criticality at the segment level and utilizing structured sparsity to prune computations within the self-attention mechanism.

2.3.1 Estimating segment-wise query ciriticality

As described in Algorithm 1, CritiPrefill estimates segment-wise query criticality segment-wise, significantly outperforming token-wise approaches in terms of efficiency. First, the Query and KV

---

[1]For clarity, we illustrate only a single head in the multi-head attention mechanism, which can be easily extended to multiple heads in practice.

Figure 3: Locality Pattern

Cache are firstly divided into segments and blocks. Then, representative queries and keys are selected for each segment and block via element-wise max and min operations. Lines 9-13 calculate attention scores for these representatives, producing criticality scores between segments and blocks. Additionally, a layer-fusion mechanism is employed in line 14 to refine the estimation, drawing on insights from prior works on inter-layer similarity (Elhoushi et al., 2024).

Algorithm 1 Estimation of Query Criticality in One Layer

1: **Input:** $Q, K, V \in \mathbb{R}^{n \times d}$, query ciriticality score from last layer $S' \in \mathbb{R}^{n_1 \times n_2}$
2: **Output:** query ciriticality score in this layer $S \in \mathbb{R}^{n_1 \times n_2}$
3: segment number $n_1 = n/segment\ size$
4: block number $n_2 = n/block\ size$
5: $Q = Q.\text{reshape}(n_1, segment\ size, d)$
6: $K = K.\text{reshape}(n_2, block\ size, d)$
7: $Q_{max} = Q.\text{max(dim=1)}; Q_{min} = Q.\text{min(dim=1)}$
8: $K_{max} = K.\text{max(dim=1)}; K_{min} = K.\text{min(dim=1)}$
9: $S_1 = \text{softmax}(Q_{max}K_{max}^T); S_2 = \text{softmax}(Q_{max}K_{min}^T)$
10: $S_3 = \text{softmax}(Q_{min}K_{max}^T); S_4 = \text{softmax}(Q_{min}K_{min}^T)$
11: $S_{max} = (S_1 + S_3)/2; S_{min} = (S_2 + S_4)/2$
12: $S = \max(S_{max}, S_{min})$
13: apply causal mask to S
14: $S = \alpha S + (1-\alpha)S'$
15: **Return** S

2.3.2 PRUNING SELF-ATTENTION MECHNISUM

Based on the estimated segment-wise criticality scores, CritiPrefill applies structured pruning to the self-attention mechanism (Algorithm 2). Rather than standard dense attention, it eliminates non-critical KV Cache blocks for the computation of each query segment, thereby reducing redundant computation operations and speeding up the prefilling phase. Thus, the selective attention mechanism accelerates the overall process, while maintaining generation quality.

3 EXPERIMENT

In this section, we conduct experiments on datasets with various sequence lengths and scenarios to demonstrate the accuracy and acceleration effects of our approach.

3.1 SETTINGS

3.1.1 DATASET

We conduct thorough evaluations across four datasets in both Single-hop (Kočiský et al., 2018) and Multi-hop QA (Yang et al., 2018) scenarios. For Single-hop QA, where answers come from a single piece of evidence, we assess the Loogle Short-dependency QA (SD) (Dasigi et al., 2021) and

---

Algorithm 2 Pruned Attention

1: **Input:** $Q, K, V \in \mathbb{R}^{n \times d}$, Query Ciriticality Score $S \in \mathbb{R}^{n_1 \times n_2}$, Critical Budget B
2: **Output:** Attention Output O
3: segment number $n_1 = n/segment\ size$
4: block number $n_2 = n/block\ size$
5: $Q = Q.reshape(n_1, segment\ size, d)$
6: **for** $j = 1$ to n_1 **do**
7:     $Q_j = Q[j]$
8:     critical block index $\mathcal{I} = \mathrm{argtop}_k(S[j], B/block\ size)$
9:     extract critical KV cache $\hat{K}, \hat{V}$ in cache block $\mathcal{I}$
10:     $A_j = \mathrm{softmax}(\frac{Q_j \hat{K}^T}{\sqrt{d}})$
11:     $O_j = A_j \hat{V}$
12: **end for**
13: concate all O_j to O
14: **Return** O

MultiFieldQA (MF) (Bai et al., 2024) datasets. For Multi-hop QA, which requires integrating information from multiple sources, we assess the Multiple Information Retrieval (MIR) (Yuan et al., 2024) and Comprehension and Reasoning (CR) tasks in Loogle datasets. Following the methodology of (Yuan et al., 2024), we categorize each dataset into two groups based on context length: 64k and 128k for evaluation under different context lengths. Additionally, the widely-used "Needle-in-a-Haystack" (Liu et al., 2024) test, a synthetic QA task designed to assess long-context retrieval capabilities, is employed to facilitate a detailed evaluation across various context lengths.

3.1.2 BASELINE

We compare the CritiPrefill method with vanilla LLMs to demonstrate that our approach significantly accelerates prefilling with minimal quality loss. Additionally, a CritiPreill method without the layer-fusion mechanism is also included as an ablation study to illustrate that this mechanism effectively enhances the accuracy of segment-wise criticality estimation. All methods are based on two open-source LLMs, Llama3-8B-1M (Touvron et al., 2023), Yi-9B-200K (AI et al., 2024), which are widely used for long context evaluation due to their moderate parameter size and remarkable capability of long context processing(Xiong et al., 2023).

3.1.3 PARAMETER SETTINGS

In all experiments, the query segment size is set to 512, the key-value block size to 32, and the budget size to 1024. The layer-fusion hyperparameter, α, is fixed at 0.25. All experiments employ Flash Attention (Dao, 2023) and FP16 for efficient computation on Nvidia A100 GPUs. For more details, please refer to our code at https://github.com/66RING/CritiPrefill.

3.2 EVALUATIONS CROSS MULTIPLE SCENARIOS

The test results for each model across all datasets are summarized in Table 1. Overall, our CritiPrefill method achieves an average speedup of over 3x across all datasets, with minimal degradation in quality. Specifically, on the Llama3-8B model, CritiPrefill achieves a 3.0x speedup with only a

Table 1: Quality Scores on Long-Context QA Datasets

	Single-Hop. QA				Multi-Hop. QA				
	SD 64k	SD 128k	MF 64k	MF 128k	CR 64k	CR 128k	MIR 64k	MIR 128k	Ave.
Llama3-8B	41.77(1.00x)	40.9(1.00x)	26.24(1.00x)	24.08(1.00x)	17.79(1.00x)	18.39(1.00x)	18.01(1.00x)	18.35(1.00x)	25.69(1.00x)
Llama3-8B(ours w/o fusion)	**39.72(2.22x)**	**39.41(3.38x)**	25.35(2.49x)	24.05(3.94x)	17.67(2.43x)	15.87(3.66x)	**17.31(2.44x)**	13.94(3.68x)	24.16(3.03x)
Llama3-8B(ours)	39.55(2.22x)	38.62(3.38x)	**27.09(2.48x)**	**28.56(3.94x)**	**19.35(2.42x)**	16.72(3.67x)	16.96(2.44x)	**15.07(3.68x)**	25.24(3.03x)
Yi-9B	31.03(1.00x)	16.04(1.00x)	14.23(1.00x)	12.23(1.00x)	10.6(1.00x)	6.77(1.00x)	8.99(1.00x)	5.19(1.00x)	13.13(1.00x)
Yi-9B(ours w/o fusion)	26.37(2.51x)	18.34(4.01x)	9.02(2.86x)	7.51(4.77x)	**12.32(2.78x)**	12.03(4.39x)	8.49(2.80x)	**6.96(4.42x)**	12.63(3.56x)
Yi-9B(ours)	**27.89(2.52x)**	**22.19(3.98x)**	**9.59(2.86x)**	**8.32(4.71x)**	11.16(2.79x)	**12.34(4.35x)**	**9.11(2.80x)**	4.56(4.37x)	**13.14(3.55x)**

Figure 4: CritiPrefill on Needle-in-a-Haystack Test. This test involves inserting an answer within a large context and evaluating the retrieval ability in response to the corresponding question. The Average Score is derived from averaging scores across various context lengths (x-axis) and insertion depths (y-axis), with a maximum score of 10.

Figure 5: Prefilling Time on Need-in-a-Haystack Test

slight drop in quality score from 25.69 to 25.24. The Yi-9B model attains a 3.4x speedup, while the quality score remains nearly unchanged, shifting from 13.13 to 13.14. This demonstrates that CritiPrefill's segment-wise estimation algorithm not only effectively approximates query critical-ity but also successfully implements structured attention pruning, resulting in tangible acceleration gains. Furthermore, when examining datasets with varying sequence lengths, CritiPrefill's accel-eration becomes more pronounced with longer sequences. For instance, on the SD dataset using Llama3-8B, CritiPrefill achieves speedups of 2.2x and 3.3x at the 64K and 128K sequence lengths, respectively. This is primarily because, with longer sequences, CritiPrefill can prune more irrelevant operations during AttentionPrefill, leading to greater performance improvements.

Furthermore, comparing CritiPrefill with CritiPrefill without layer-fusion shows that while there is almost no difference in speed, our method without layer-fusion shows a significantly lower gener-ation quality. This indicates that the layer-fusion mechanism effectively enhances the accuracy of criticality estimation without introducing excessive computational overhead.

3.3 EVALUATIONS ON NEEDLE-IN-A-HAYSTACK TEST

The Needle-in-a-Haystack test, as shown in Figure 4, is conducted to thoroughly evaluate the impact of the information retrieval capabilities across varying lengths. Both the Llama-3 and Yi models exhibit nearly lossless performance in the Needle-in-a-Haystack test, ranging from 0 to 128K context lengths when utilizing CritiPrefill methods. As illustrated in Figure 5, the acceleration provided by

CritiPrefill becomes increasingly significant as sequence lengths grow. It effectively reduces the quadratic prefilling time to linear without compromising performance at lower sequence lengths like 8K-16K.

4 CONCLUSION

In this work, we propose CritiPrefill, a plug-and-play method to accelerate the prefilling phase in LLMs. Our key insight is identifying a locality pattern where neighboring query tokens rely on similar critical KV cache. This enables efficient segment-wise criticality estimation and structured pruning of self-attention computation, significantly reducing the computational load during the prefilling phase. By introducing a layer-fusion mechanism to refine criticality across layers, CritiPrefill enhances generation quality without compromising efficiency. Experiments on long-context QA tasks demonstrate up to 3.0x speedup with minimal accuracy loss on models such as Llama3-8B and Yi-9B. CritiPrefill provides an effective solution to the long-sequence inference bottleneck, requiring no architectural changes or fine-tuning, making it highly adaptable for practical applications.

REFERENCES

01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. Yi: Open foundation models by 01.ai, 2024. URL https://arxiv.org/abs/2403.04652.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding, 2024. URL https://arxiv.org/abs/2308.14508.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*, 2021.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Zican Dong, Tianyi Tang, Lunyi Li, and Wayne Xin Zhao. A survey on long text modeling with transformers. *arXiv preprint arXiv:2302.14502*, 2023.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference, 2024. URL https://arxiv.org/abs/2407.11550.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

Christoforos Kachris. A survey on hardware accelerators for large language models, 2024. URL https://arxiv.org/abs/2401.09890.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA Reading Comprehension Challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 05 2018. ISSN 2307-387X. doi: 10.1162/tacl_a_00023. URL https://doi.org/10.1162/tacl_a_00023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Philippe Laban, Wojciech Kryściński, Divyansh Agarwal, Alexander Richard Fabbri, Caiming Xiong, Shafiq Joty, and Chien-Sheng Wu. Summedits: measuring llm ability at factual reasoning through the lens of summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9662–9676, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 02 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00638. URL https://doi.org/10.1162/tacl_a_00638.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: A kvcache-centric disaggregated architecture for llm serving, 2024a. URL https://arxiv.org/abs/2407.00079.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. *arXiv preprint arXiv:2401.04658*, 2024b.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3505–3506, 2020.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3531–3539, 2021.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. Efficient large language models: A survey, 2024. URL `https://arxiv.org/abs/2312.03863`.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024. ISSN 2095-2236. doi: 10.1007/s11704-024-40231-1. URL `http://dx.doi.org/10.1007/s11704-024-40231-1`.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. URL `https://arxiv.org/abs/2006.04768`.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. Lv-eval: A balanced long-context benchmark with 5 length levels up to 256k, 2024. URL `https://arxiv.org/abs/2402.05136`.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Qingru Zhang, Dhananjay Ram, Cole Hawkins, Sheng Zha, and Tuo Zhao. Efficient long-range transformers: You need to attend more, but not necessarily at every layer. *arXiv preprint arXiv:2310.12442*, 2023.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024.