TriForce: Lossless Acceleration of Long Sequence Generation with Hierarchical Speculative Decoding

Hanshi Sun[†], Zhuoming Chen[†], Xinyu Yang[†], Yuandong Tian[‡], and Beidi Chen[†‡]

[†]Carnegie Mellon University
[‡]Meta AI (FAIR)
`{hanshis, zhuominc,xinyuya2,beidic}@andrew.cmu.edu`, `yuandong@meta.com`

April 24, 2024

Abstract

With large language models (LLMs) widely deployed in long content generation recently, there has emerged an increasing demand for efficient long-sequence inference support. However, key-value (KV) cache, which is stored to avoid re-computation, has emerged as a critical bottleneck by growing linearly in size with the sequence length. Due to the auto-regressive nature of LLMs, the entire KV cache will be loaded for every generated token, resulting in low utilization of computational cores and high latency. While various compression methods for KV cache have been proposed to alleviate this issue, they suffer from degradation in generation quality. We introduce TriForce, a hierarchical speculative decoding system that is scalable to long sequence generation. This approach leverages the original model weights and dynamic sparse KV cache via retrieval as a draft model, which serves as an intermediate layer in the hierarchy and is further speculated by a smaller model to reduce its drafting latency. TriForce not only facilitates impressive speedups for Llama2-7B-128K, achieving up to 2.31$\times$ on an A100 GPU but also showcases scalability in handling even longer contexts. For the offloading setting on two RTX 4090 GPUs, TriForce achieves 0.108s/token—only half as slow as the auto-regressive baseline on an A100, which attains 7.78$\times$ on our optimized offloading system. Additionally, TriForce performs 4.86$\times$ than DeepSpeed-Zero-Inference [2] on a single RTX 4090 GPU. TriForce's robustness is highlighted by its consistently outstanding performance across various temperatures. The code is available at `https://github.com/Infini-AI-Lab/TriForce`.

1 Introduction

Large language models (LLMs) with long-context capability, such as GPT-4 [1], Gemini [42], and LWM [24] continue to emerge and gain proficient application in scenarios including chatbots, vision generation, and financial analysis [9, 35, 43, 54]. However, losslessly serving these LLMs efficiently is challenging. Because of the auto-regressive nature of LLMs, the entire key-value (KV) cache, which stores intermediate key-value states from previous contexts to avoid re-computation, together with model parameters will be loaded into GPU SRAM for every token generated, resulting in low utilization of computational cores. In addition to the large volume of model parameters, the memory footprint of KV cache, which grows linearly with sequence length [33], is emerging as a new bottleneck for long sequence generation.

Recent methodologies have proposed KV cache eviction strategies [13, 17, 28, 47, 53] to mitigate the substantial memory footprint of KV cache, which selectively discard KV pairs from the cache based on a designed eviction policy, allowing models to generate texts with a limited KV cache budget. However, considering that discarded KV pairs cannot be restored and the difficulty in precisely foreseeing which KV pairs will be crucial for future text generation, they struggle with potential information loss, including hallucination and contextual incoherency [49], particularly in long contexts. Such challenges prevent these approaches from boosting speed without sacrificing the performance of models, as illustrated in Figure 1.

Concurrently, speculative decoding, which leverages a lightweight draft model to sequentially predict the next few tokens and let the target model verify the predicted tokens in parallel, is introduced to accelerate LLM inference while provably precisely preserving model output [6, 22, 45]. Nonetheless, deploying it for

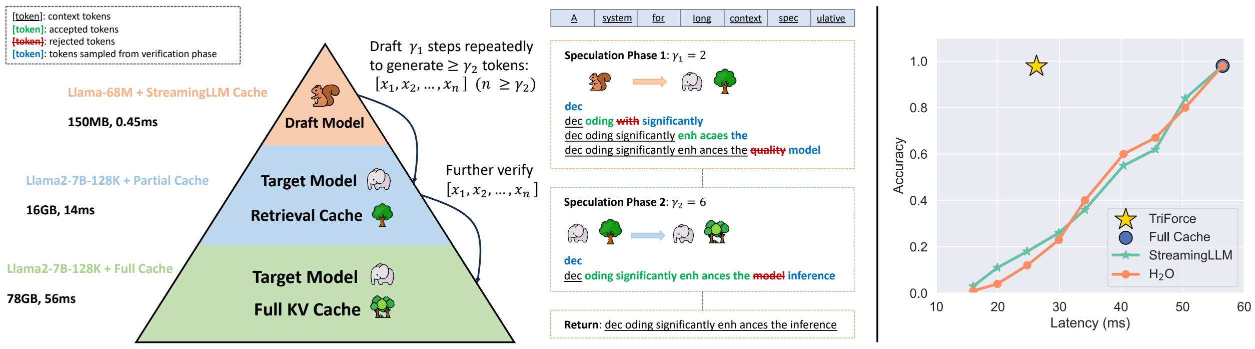

[token]: context tokens
[token]: accepted tokens
~~[token]~~: rejected tokens
[token]: tokens sampled from verification phase
Llama-68M + StreamingLLM Cache
150MB, 0.45ms
Llama2-7B-128K + Partial Cache
16GB, 14ms
Llama2-7B-128K + Full Cache
78GB, 56ms
Draft Model
Target Model
Retrieval Cache
Target Model
Full KV Cache
Draft γ_1 steps repeatedly to generate $\geq \gamma_2$ tokens: $[x_1, x_2, \dots, x_n]$ $(n \geq \gamma_2)$
Further verify $[x_1, x_2, \dots, x_n]$
A system for long context spec ulative
Speculation Phase 1: $\gamma_1 = 2$
dec
dec oding ~~with~~ significantly
dec oding significantly enh acaes the
dec oding significantly enh ances the ~~quality~~ model
Speculation Phase 2: $\gamma_2 = 6$
dec
dec oding significantly enh ances the ~~model~~ inference
Return: dec oding significantly enh ances the inference
Accuracy
Latency (ms)
TriForce
Full Cache
StreamingLLM
H_2O

Figure 1: **Left**: TRIFORCE employs retrieval-based drafting and hierarchical speculation to effectively address the dual bottlenecks. It integrates two models and three caches, comprising a draft model, a target model, and a StreamingLLM cache for the draft model, alongside a retrieval cache and a full cache for the target model. The process initiates by repeatedly drafting for γ_1 steps, assisting the target model with retrieved partial KV cache in generating over γ_2 tokens, which will be further verified by the target model using full KV cache. **Right**: Evaluating the Llama2-7B-128K on a needle retrieval task indicates that KV cache eviction-based methods, such as StreamingLLM, require a trade-off between latency and accuracy. In contrast, our TRIFORCE successfully maintains low latency without sacrificing accuracy.

long sequence generation faces several challenges. First, training draft models to match the context length of target LLMs requires massive computation and it remains questionable whether these small models can achieve the same accuracy with a context length around 1M [4, 32, 48]. Second, we found that draft models with existing training-free methods (e.g., KV cache eviction strategies) can result in poor speculating performance. A continuously increasing divergence [22] is witnessed as the sequence length increases, as shown in Figure 2a.

In pursuit of lossless acceleration, we utilize the lossless feature of speculative decoding as the foundation of our system. An ideal speculative decoding algorithm should (i) be training-free, (ii) maintain a high acceptance rate with long contexts, and (iii) have low-cost drafting. However, two technical challenges need to be addressed to achieve the goal. First, it is not immediately apparent what we can use for low-latency drafting without training a smaller draft model to match the long context length. Second, the key factors for attaining a high acceptance rate with long contexts remain unclear.

Fortunately, based on our preliminary exploration, three key observations pave the way for designing an applicable system for serving LLMs with long contexts.

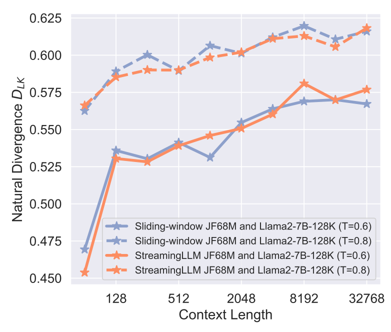
Hierarchical Speculation for Dual Memory Bottlenecks: As illustrated in Figures 2b and 2c, we recognize two memory bottlenecks: model weights and KV cache, and the latter gradually becomes the dominant bottleneck as context length increases. This inspires us to apply hierarchical speculation to tackle the two bottlenecks sequentially by different draft models.

Leveraging Attention Sparsity for Speculative Decoding: We identify considerable redundancy within KV cache, finding that a relatively small portion of it is sufficient to achieve a high acceptance rate by using partial KV cache as a draft cache for self-speculation.

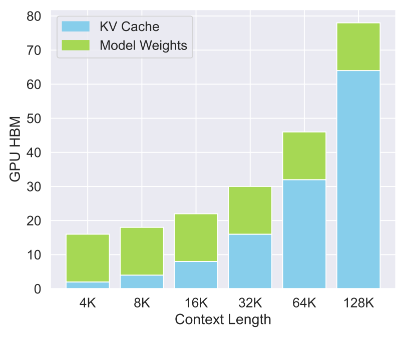
Exploiting Contextual Locality for Drafting Efficiency: We discover that the information from long context tokens needed by adjacent tokens tends to be similar. This observation suggests that a specific segment of the cache can be effectively reused across multiple decoding steps, amortizing the overhead of constructing draft cache and enhancing drafting efficiency.

Building on these insights, we introduce a hierarchical speculation approach. For a long-context target model (e.g., Llama2-7B-128K [32]), we leverage the original model weights but only with a small proportion (e.g., 3%) of KV cache as a draft to tackle the bottleneck of KV cache. Hierarchically, the draft model is further speculated by a lightweight model (e.g., Llama-68M) with StreamingLLM cache to address the bottleneck of model weights. We present TRIFORCE, depicted in Figure 1, a scalable and robust speculative decoding system that integrates retrieval-based drafting and hierarchical speculation, optimized for both on-chip and offloading scenarios. Specifically,

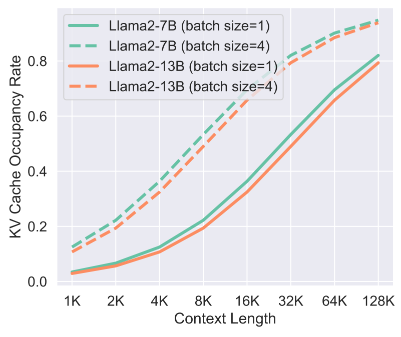
- In Section 4.1, by maintaining the full cache, we gain the flexibility to select KV pairs, allowing us to devise a superior KV cache selection method, termed retrieval-based drafting. This strategy retrieves required context information for future needs, which is characterized as lossless, particularly in comparison to eviction-based



(a)



(b)



(c)

Figure 2: (a) A continuously increasing natural divergence [22] between the draft model with StreamingLLM or Sliding-window with Re-computation and Llama2-7B-128K is witnessed as the sequence length increases, indicating a falling acceptance rate for speculative decoding with longer contexts. Additionally, temperature sensitivity signals a lack of robustness. (b) Compared with model weights, KV cache gradually becomes another bottleneck with long contexts. (c) KV cache occupies most of the memory as the context length increases.

methods such as StreamingLLM and H_2O. We further demonstrated its effectiveness and robustness on different datasets.

- In Section 4.2, we propose a hierarchical system to address the dual memory bottlenecks. Using a lightweight model paired with a StreamingLLM cache for initial speculations, we can reduce the drafting latency for the subsequent speculation stage, thereby accelerating end-to-end inference.

Empirically, in Section 5, we perform extensive experiments and ablation studies to demonstrate the effectiveness of TriForce. We show that TriForce achieves up to 2.31× speedup for Llama2-7B-128K on a single A100 GPU on top of Hugging Face [44] with CUDA graphs [30]. For the offloading setting, TriForce attains an impressive 7.78× on two RTX 4090 GPUs, reaching 0.108 s/token—only half as slow as the auto-regressive baseline on an A100. TriForce can efficiently serve a Llama2-13B with 128K contexts with 0.226s/token, which is 7.94× faster on our optimized offloading system. On a single RTX 4090 GPU, TriForce is 4.86× faster than DeepSpeed-Zero-Inference [2] [1]. Further, we show that: (i) TriForce has a theoretical 13.1× upper bound, demonstrating exceptional scalability when dealing with long contexts; (ii) TriForce is robust across various temperature settings, maintaining an acceptance rate above 0.9 even for a temperature of 1.0; and (iii) TriForce's ability to efficiently process large batches, consistently outperforming the small model with StreamingLLM cache across all configurations, achieving a 1.9× speedup for a batch size of six, where each sample consists of 19K contexts.

2 Background

2.1 Speculative Decoding

Speculative decoding [6, 15, 19, 22, 36, 40, 51] is featured by accelerating LLM decoding while precisely maintaining the model's output distribution. As the speed of the auto-regressive decoding process is mainly bound by the time for loading model weights and KV cache to GPU SRAM, speculative decoding leverages the observation that generating one token takes the same time as processing tens of tokens in parallel. Tree-based speculation methods are proposed to fully utilize the speculation budget [12, 23]. Instead of making one prediction for the next token, tree-based methods leverage multiple candidates to boost the acceptance rate so that more tokens can get accepted [7, 29, 41]. Staged speculation techniques [8, 39] have been suggested to further accelerate inference by using a cascade of draft models. Meanwhile, self-speculation approaches such as Medusa [3, 5], which are orthogonal to our method, require training efforts and can be integrated into our intermediate draft model.

[1]The official implementation of DeepSpeed-ZeRO-Inference with KV cache offloading currently only supports a single GPU, which computes attention on CPU. Our offloading system transfers KV cache from CPU to GPU, benefiting from Tensor Parallelism.

2.2 KV Cache Eviction Strategies

StreamingLLM [47] addresses the limitations of window attention and sliding window with re-computation by presenting a straightforward yet effective method that allows LLMs to handle infinitely long text sequences without fine-tuning. StreamingLLM stabilizes the performance by retaining critical attention sink tokens together with recent KV for attention computation. By prioritizing sink tokens, StreamingLLM ensures the attention score distribution remains stable, promoting consistent language modeling for long texts.

H_2O [53] introduces a greedy but low-cost approach to processing infinite-length input streams, inspired by a simplified version of the heavy-hitters (H_2) eviction policy. This method dynamically updates the KV cache based on the cumulative attention scores, systematically removing the least critical KV to maintain a fixed cache size. By leveraging a greedy algorithm based on local statistics, H_2O effectively selects which KV pairs to preserve in the cache, ensuring efficient inference without compromising quality.

However, it is important to recognize that these techniques do not increase the context window size [13, 17, 18, 52]. They focus on retaining only the most recent tokens along with either attention sinks or heavy-hitters, while discarding other tokens. These approaches limit the model to processing based on their designed eviction policies and recent tokens. Consequently, they might not be directly applicable to tasks that demand comprehensive, long-context understanding.

2.3 KV Cache Quantization

Several approaches to KV cache quantization have been introduced to enhance the efficiency of inference for long sequence generation, aiming to maintain generation quality while reducing the memory consumption [16, 26, 37, 46, 50, 55]. Quantization methods focus on compressing the bit width of KV cache activations, which is orthogonal to our approach.

3 Observation

Our design of TriForce is inspired by two critical empirical observations regarding LLMs when dealing with long contexts, detailed as follows.

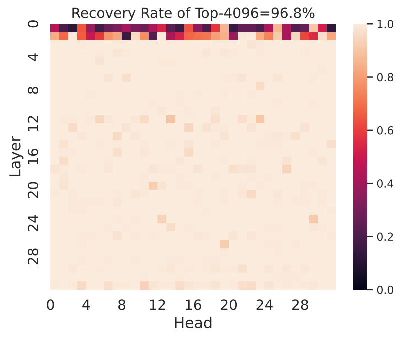
3.1 Leveraging Attention Sparsity for Speculative Decoding

Observation The phenomenon of attention sparsity in pre-trained LLMs has been discovered by numerous studies [27, 28, 47, 53]. In our study, we conduct zero-shot inference on the PG-19 test set [34] with Llama2-7B-128K model. By visualizing the sparsity across different attention heads, demonstrated in Figure 3a, we observe that with a context length of 120K, it is possible to recover over 96% of the attention score with merely 4K tokens across almost all layers.

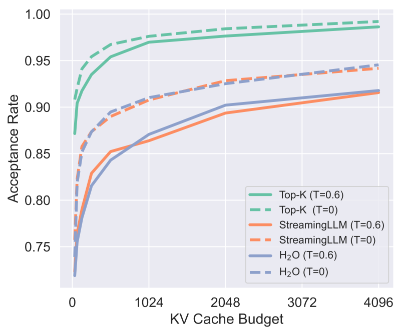
Analysis The presence of sparsity within the attention blocks suggests that a fraction of KV cache could serve as a draft cache to attain a high acceptance rate during self-speculative decoding. Since KV cache is the bottleneck under this setting, we can load whole model weights with partial KV cache as a draft model. Figure 3b demonstrates that utilizing only 1K tokens could theoretically achieve a 97.6% acceptance rate with Top-K selection method. While this scenario represents an optimal theoretical upper bound, practical implementations like H_2O and StreamingLLM exhibit promising results, achieving over 90.5% acceptance rates with 1K KV cache budget. It should be noted that we maintain a full cache for the initial two layers for illustration purposes, while no layers are skipped in our practical system implementation for efficiency.

3.2 Exploiting Contextual Locality for Drafting Efficiency

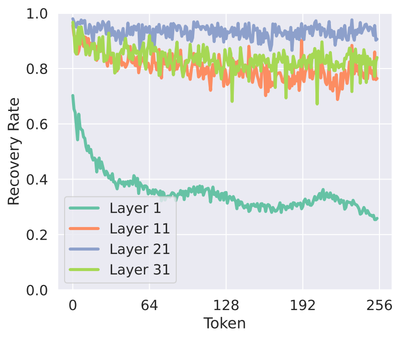
Observation Our exploration reveals that the information from long context tokens needed by adjacent tokens tends to be similar. In our experiments, with the context length established at 120K, we instruct the model to generate 256 tokens. By choosing the top-4K indices according to the attention score of the last prefilled token, we use these indices to gather the attention scores for the subsequently generated tokens and assess the score's recovery rate for the initially prefilled 120K tokens. As shown in Figure 3c, it leads to high recovery across almost all layers and a slowly decreasing trend as the number of tokens increases.



(a)



(b)



(c)

Figure 3: (a) The Llama2-7B-128K model demonstrates significant attention sparsity with a 120K context. Apart from the first two layers, the rest exhibit significant sparsity. (b) We can utilize partial KV cache and whole model weights to perform self-speculation. High acceptance rates are attainable using existing methods with a limited budget. (c) A notable degree of locality is observed in most layers, which gradually diminishes as context evolves.

Insights This observation allows for a single construction of the cache to suffice for multiple decoding steps, thereby amortizing the latency of constructing draft cache and boosting efficiency. As new KV cache are introduced, guided by the understanding that recent words are more strongly correlated with the tokens currently being decoded, these entries will replace the less significant ones. Cache re-building operations can be scheduled at regular intervals or adaptively in response to a drop in the acceptance rate, which ensures that the cache remains dynamically aligned with the evolving context. Notably, both StreamingLLM and H_2O incorporate this principle implicitly. H_2O consistently retains tokens with high scores, and StreamingLLM reuses extensive local information and sink tokens, which both reduce the necessity for complete cache reconstruction.

4 TriForce

This section aims to introduce the TriForce, which leverages a retrieval-based KV cache selection policy and a hierarchical speculation system. We first argue that our retrieval-based drafting approach is intuitive and lossless compared to existing strategies such as StreamingLLM and H_2O. Subsequently, we introduce the hierarchical system designed to effectively address the dual bottlenecks in speculative decoding, facilitating a substantial improvement in overall speed-up. Finally, TriForce is elaborated in Section 4.3.

4.1 Retrieval-based Drafting

In scenarios requiring long-term contextual dependencies, methods like StreamingLLM and H_2O underperform due to their cache updating strategies, which are ineffective at accurately retrieving detailed contextual information because they inevitably and irrecoverably discard KV pairs. In our experiment, we challenge StreamingLLM and H_2O with a needle retrieval task [24, 25, 32]. As detailed in Table 1, there is a notable drop in their acceptance rates compared to their performance on the PG-19 dataset, highlighting their limitations. Essentially, StreamingLLM and H_2O operate on a lossy principle, as evicted tokens are permanently discarded, making them a poor fit for settings requiring the preservation of full KV cache for the target model.

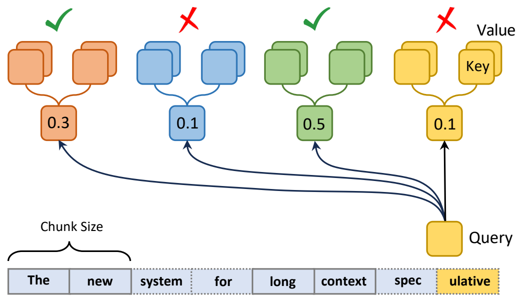


Figure 4: Retrieval-based drafting

The necessity of keeping the entire KV cache in our settings allows us to select KV cache more freely [38]. This insight leads us to develop a more effective selection policy for lossless approximations. In our approach, demonstrated in Figure 4, KV cache is segmented into small chunks. During the retrieval phase, we calculate the attention between a given query and the average key cache within each chunk. This method effectively highlights the most relevant chunks, enabling us to gather KV cache with a fixed budget based on the scores. As illustrated in Table 1, retrieval-based method excels by actively identifying the most crucial information for the task rather than relying on passive and time-based cache management methods. By focusing on relevance over recency, retrieval-based policy demonstrates its potential to handle contextually dense datasets.

Table 1: Acceptance rates are shown across various tasks, utilizing a 120K context and a 4K budget, while bypassing the initial two layers. There is a notable drop in StreamingLLM and H_2O's acceptance rate on the needle retrieval task. For reference, Top-K is the theoretical upper bound.

Method	Top-K (Ref.)	StreamingLLM	H_2O	Retrieval
PG-19	0.9921	0.9156	0.9179	**0.9649**
Needle Retrieval	0.9989	0.0519	0.0739	**0.9878**

4.2 Hierarchical Speculation

While addressing the KV cache bottleneck enhances efficiency, the requirement to load whole model weights for drafting reintroduces latency, shifting the bottleneck to model weights again. To tackle this challenge, we implement a hierarchical system, as illustrated in Figure 1. This system employs a secondary, lightweight model with StreamingLLM cache to perform initial speculations for the target model with retrieval-based draft cache (which serves as a draft model for the target model with full KV cache). By establishing this sequential speculation hierarchy, we effectively reduce the drafting latency, thereby accelerating the overall inference.

Correctness: The original output distribution is preserved during the final speculation phase, which is identical to the standard speculative decoding algorithm [6, 22], and the proof is trivial.

4.3 Algorithm

TriForce is devised to exploit the bottlenecks associated with both model weights and KV cache to enhance the inference speed of LLMs for long sequence generation. We present the pseudocode for the TriForce in Algorithm 1. It starts by prefilling the target model M_p with full cache C_p and draft model M_q with StreamingLLM cache C_q using a given input prefix, and then constructs the retrieval cache C_r. The initialization and update mechanism for the retrieval cache C_r is guided by the insights of contextual locality discussed in Section 3.2. We first construct C_r using the last token of the prefix, arranging tokens by their descending order of importance. In subsequent inferences, we overwrite the tokens with the least importance, maintaining the relevance and utility of the cache. A reconstruction of C_r is triggered either when the rolling average acceptance rate drops below a threshold or at a designed stride.

The inference progresses iteratively until it reaches the target sequence length T. After each iteration, cache C_r and C_q are updated to prepare for the subsequent speculation phase. Each iteration encompasses two speculations: initially, M_q utilizes C_q to predict M_p with C_r for γ_1 steps until $n \geq \gamma_2$. Subsequently, these n tokens are self-verified [51] by M_p with C_p. This process constructs a hierarchy: the first layer of hierarchy employs a smaller, faster model M_q with local context C_q to speculate the large model M_p with partial but high-quality global context C_r, addressing the model weights bottleneck. The second layer utilizes model M_p with retrieval cache for self-speculation, overcoming the bottleneck caused by KV cache. This hierarchical speculation algorithm boosts efficiency by effectively addressing both bottlenecks.

System Implementation. We implement the draft and target models using Transformers [44]. Leveraging a predetermined cache budget enables the effective use of PyTorch CUDA graphs [30, 31], significantly minimizing the kernel launching overhead during speculative decoding. FlashAttention is used to accelerate attention operations [10, 11, 14, 21]. Notably, we maintain full layer sparsity without omitting the initial two layers for system efficiency, ensuring our approach stays within the fixed KV cache budget. Although this strategy might lead to a lower acceptance rate, the benefit of maintaining a constant drafting cost makes it a worthwhile compromise. Additionally, to facilitate a faster speculation phase, we also implement two extra speculation cache structures, including our retrieval cache and StreamingLLM cache.

Algorithm 1 TriForce

```
Input: Prefix $[x_1,\cdots,x_t]$, target model $M_p$ with full cache $C_p$, draft model $M_q$ with StreamingLLM cache $C_q$, target sequence length $T$, speculation length $\gamma_1,\gamma_2$, auto-regressive drafting phase Draft, verification phase Verify, and correction phase Correct;
Initialize: Prefill $M_p$, $M_q$ with Prefix, construct retrieval cache $C_r$ using $x_t$, $N \leftarrow t$
while $N < T$ do
    $n \leftarrow 0$
    while $n < \gamma_2$ do
        Set $q_1,\cdots,q_{\gamma_1} \leftarrow$ Draft$(M_q, C_q, x_{\leq N})$    ▷ Run $M_q$ with eviction cache $C_q$
        Sample $\tilde{x}_i \sim q_i, i = 1,\cdots,\gamma_1$
        Set $\hat{p}_1,\cdots,\hat{p}_{\gamma_1+1} \leftarrow M_p(C_r, x_{\leq N}, \tilde{x}_{\leq \gamma_1})$    ▷ Run $M_p$ with retrieval cache $C_r$
        for $i = 1$ to $\gamma_1$ do
            if Verify$(\tilde{x}_i, q_i, \hat{p}_i)$ then
                $\hat{x}_{n+i} \leftarrow \tilde{x}_i$ and $n \leftarrow n+1$
            else
                $\hat{x}_{n+i} \leftarrow$ Correct$(q_i, \hat{p}_i)$ and $n \leftarrow n+1$
                Break
            end if
        end for
        If all drafted tokens are accepted, sample next token $\hat{x}_{n+1} \sim \hat{p}_{\gamma_1+1}$ and $n \leftarrow n+1$
    end while
    Collect $\hat{p}_1,\cdots,\hat{p}_n$ for $\hat{x}_1,\cdots,\hat{x}_n$
    Set $p_1,\cdots,p_{n+1} \leftarrow M_p(C_p, x_{\leq N}, \hat{x}_{\leq n})$    ▷ Run $M_p$ with full cache $C_p$
    for $i = 1$ to $n$ do
        if Verify$(\hat{x}_i, \hat{p}_i, p_i)$ then
            $x_{N+i} \leftarrow \hat{x}_i$ and $N \leftarrow N+1$
        else
            $x_{N+i} \leftarrow$ Correct$(\hat{p}_i, p_i)$ and $N \leftarrow N+1$
            Break
        end if
    end for
    If all drafted tokens are accepted, sample next token $x_{N+1} \sim p_{n+1}$ and $N \leftarrow N+1$
    Update $C_r$, $C_q$ based on the accepted tokens    ▷ Update KV cache for the next iteration
end while
```

5 Empirical Evaluation

In this section, our goal is to showcase the capabilities of TriForce, a scalable and robust speculative decoding algorithm designed to expedite the inference of LLMs for long sequence generation, which significantly reduces the wall-clock time. We first present our end-to-end system, highlighting the overall speedup achieved, including both on-chip and offloading settings, followed by detailed ablation experiments.

- In Section 5.1, we illustrate TriForce's remarkable end-to-end performance, achieving up to 2.31× speedup for Llama2-7B-128K on an A100 and 7.78× on two RTX 4090 GPUs with offloading (achieving the latency as low as 0.108 s/token). We also examine the scalability and robustness of TriForce. Moreover, our results show that TriForce enhances large batch inference efficiency by 1.9×.

- In Section 5.2, we present detailed studies on the KV cache budget and chunk size, which provide valuable insights into selecting the appropriate hyperparameters to maximize efficiency. Additionally, we discuss the compatibility of TriForce with tree-based speculative decoding.

5.1 End-to-end Results

We now demonstrate that TriForce accelerates long sequence generation, up to 2.31× on an A100 GPU in the on-chip setting and 7.78× on two RTX 4090 GPUs with offloading for Llama2-7B-128K.

Table 2: **On-chip results (A100)**: We indicate the average acceptance rate in parentheses alongside the speedup factor. T means sampling temperature. In the A100 on-chip experiments, with a prompt length of 122K, and a generation length of 256, we evaluate TriForce against the JF68M model with StreamingLLM cache (Naive Policy). The results clearly demonstrate that TriForce significantly surpasses its performance.

Method	**T**	**Speedup**	**Naive Policy**
TriForce	0.0	**2.31**× (0.9234)	1.56× (0.4649)
TriForce	0.2	**2.25**× (0.9203)	1.54× (0.4452)
TriForce	0.4	**2.20**× (0.9142)	1.47× (0.4256)
TriForce	0.6	**2.19**× (0.9137)	1.42× (0.4036)
TriForce	0.8	**2.08**× (0.8986)	1.34× (0.3131)
TriForce	1.0	**2.08**× (0.9004)	1.29× (0.2872)
TriForce	1.2	**2.02**× (0.8902)	1.27× (0.2664)
Retrieval w/o Hierarchy	0.6	1.80× (0.9126)	-
StreamingLLM w/ Hierarchy	0.6	1.90× (0.8745)	-

Table 3: **Offloading results (RTX 4090 GPUs)**: We present latency comparison between TriForce and Auto-regressive (AR) baseline for various models on different GPU setups. The sampling temperature is set to 0.6. The results indicate that TriForce achieves significant speedups across a range of models and hardware configurations. The entries marked with an asterisk represent the baseline using DeepSpeed-ZeRO-Inference [2].

GPUs	**Target Model**	**TriForce (ms)**	**AR (ms)**	**Speedup**
2× RTX 4090s	Llama2-7B-128K	108	840	7.78×
2× RTX 4090s	LWM-Text-Chat-128K	114	840	7.37×
2× RTX 4090s	Llama2-13B-128K	226	1794	7.94×
1× RTX 4090	Llama2-7B-128K	312	1516*	4.86×
1× RTX 4090	LWM-Text-Chat-128K	314	1516*	4.83×

Setup. Our experiments are based on Llama2 and LWM models with 128K context window size [24, 32, 43], which serve as our target models. In this setup, we utilize a 4K retrieval cache as an intermediate draft cache in our hierarchical system, while leveraging the JackFram/Llama68M (JF68M) [29] model as the initial draft model. For experiments involving offloading, we aim to maximize memory utilization by filling it up as much as possible and offloading the remaining KV cache to the CPU (AMD EPYC 9754 @ 2.25 GHz), while keeping the model weights on the GPU. Our evaluation is carried out on the PG-19 [34] and NarrativeQA [20] dataset, each testing on 100 examples, configured to a prompt length of 122K for on-chip settings and 127K for offloading settings, and aiming for a generation of 256 tokens. The performance of TriForce is analyzed across various hardware configurations, including on-chip experiments on an A100, and offloading experiments on RTX 4090 GPUs.

Naive Policy. Since it is hard to train a draft model with long contexts, we consider JF68M with StreamingLLM cache as a naive policy approach, and its budget is set to 1K. Additionally, we experiment with various temperatures to test its robustness.

Main Results. We evaluate TriForce using different temperatures, as depicted in Table 2. We observe that TriForce reaches up to 2.31× speedup for the on-chip setting with a minimal 4K KV cache budget for Llama2-7B-128K. For offloading settings, we provide end-to-end results on consumer GPUs for more models, including Llama2-7B-128K, Llama2-13B-128K, and LWM-Text-Chat-128K. Remarkably, in Table 3 we demonstrate that TriForce can efficiently serve a Llama2-13B with 128K contexts on two RTX 4090 GPUs, reaching an average time between tokens as low as 0.226 seconds, which is

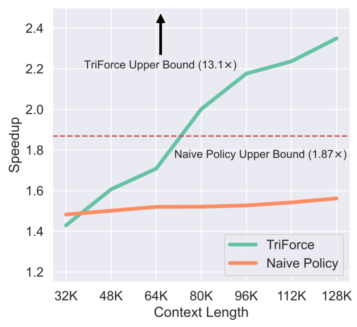


Figure 5: TriForce's excellent scalability with longer contexts

7.94× faster than a highly optimized offloading system. Furthermore, with TriForce, Llama2-7B-128K can be served with 0.108s/token—only half as slow as the auto-regressive baseline on an A100. We also illustrate how TriForce boosts the efficiency of batched inference, a more frequently employed setting in real-world model serving. TriForce achieves 1.9× for a batch size of six, with each sample in the batch having 19K contexts, which is demonstrated in Table 4.

Table 4: **Batching results (A100)**: TriForce showcases its exceptional capability in efficiently handling large batch sizes, consistently exceeding the performance of the JF68M model with StreamingLLM cache across all configurations for Llama2-7B-128K.

Batch	**Budget**	**T**	**Speedup**	**Naive Policy**
(2,56K)	(2,1024)	0.0	**1.89**×	1.46×
(2,56K)	(2,1024)	0.6	**1.75**×	1.35×
(6,19K)	(6,768)	0.0	**1.90**×	1.39×
(6,19K)	(6,768)	0.6	**1.76**×	1.28×
(10,12K)	(10,768)	0.0	**1.72**×	1.34×
(10,12K)	(10,768)	0.6	**1.61**×	1.21×

Analysis. (1) Effectiveness: TriForce's integration of the hierarchical system significantly enhances speedup, with TriForce showing marked improvements over both the StreamingLLM method with hierarchical speculation and retrieval method without the hierarchical system. (2) Scalability: As depicted in Figure 5, TriForce demonstrates excellent scalability with longer context lengths. This scalability is attributed to its high acceptance rate and the growing gap between the draft and the target model's latencies. Theoretically, TriForce could achieve a speedup of up to 13.1×, 7 times higher than the naive policy, underscoring its significant scaling potential. (3) Robustness: Unlike vanilla speculative decoding methods, TriForce maintains relatively consistent performance across various temperature settings. It exhibits less temperature sensitivity, maintaining an acceptance rate above 0.9 even when the temperature is set to 1.0, highlighting its stability and reliability.

5.2 Ablation Results

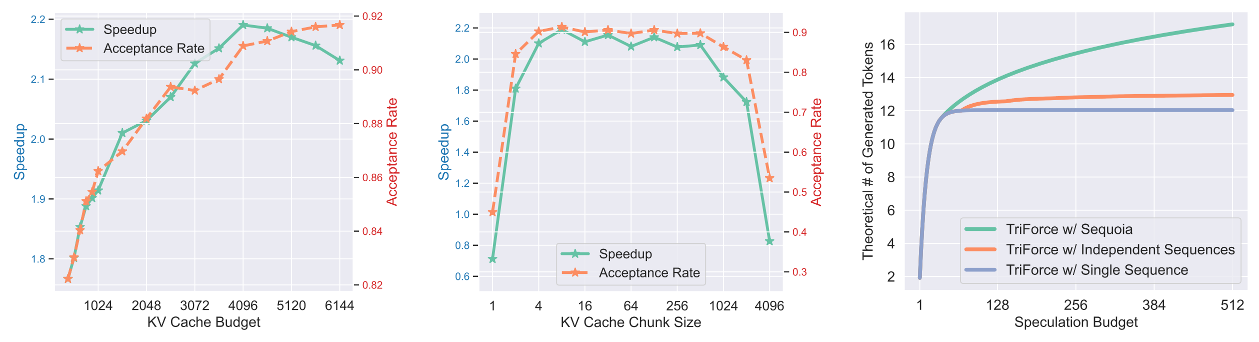
We present extensive ablation studies of TriForce, focusing on three key points: (1) the influence of different KV cache budgets, (2) the impact of chunk size selection, and (3) TriForce's compatibility with tree-based speculative decoding.

5.2.1 KV Cache Budget

As illustrated in Figure 6a, for Llama2-7B-128K, there is a notable rise in the acceptance rate as the cache budget is increased to 4K, beyond which the rate starts to plateau, moving towards 1.0 gradually. This pattern indicates that while increasing the cache size up to a certain point can improve the acceptance rate, any further expansion beyond 4K may lead to diminishing benefits due to the increased latency of drafting. Consequently, setting the KV cache budget at 4K emerges as the most effective strategy for TriForce, ensuring a balance between achieving a high acceptance rate and minimizing additional drafting overhead.

5.2.2 KV Cache Chunk Size

Since we utilized contextual locality to reuse the constructed retrieval cache, we need to examine how the choice of KV cache chunk size affects performance. Figure 6b shows that smaller chunk sizes may become too specialized, overfitting on a single token at the expense of generalization for future tokens, especially given the high attention sparsity in our methodology. Meanwhile, selecting chunk sizes that are too large can lead to the risk of high-score tokens being neutralized by surrounding low-score tokens, potentially resulting in a lack of differentiation among chunks. Such a strategy also limits our selection flexibility, constraining the diversity achievable within a fixed KV cache budget. For example, we are only allowed to select two chunks when the chunk size is set to 2K with 4K budget.



(a) (b) (c)

Figure 6: (a) Analyzing speedup and acceptance rates across varying KV cache budgets reveals that a 4K budget is the optimal selection, balancing between acceptance rates and the overhead of drafting. (b) For 4K KV cache budget, it indicates that excessively small chunk sizes might overfit individual tokens, whereas overly large chunk sizes could constrain our diversity in selection. (c) TriForce is compatible with tree-based speculations such as Sequoia and Independent Sequences, enhancing the theoretical average number of tokens generated per decoding step of the target model by employing larger speculation budgets.

5.2.3 Compatibility with Tree-based Speculative Decoding

Tree-based speculative decoding methods [5, 7, 29, 41] leverage multiple candidates to enhance the acceptance rate so that more tokens can be accepted. We explore the possibility of integrating TriForce with tree-based speculative decoding. Specifically, for Llama2-7B-128K on an A100, we estimate the theoretical number of generated tokens when TriForce is combined with tree structures, including Sequoia and Independent Sequences [7]. As depicted in Figure 6c, combining TriForce with tree-based speculation can potentially improve the overall end-to-end speedup by utilizing additional speculation budgets.

6 Conclusion

In this work, we introduced TriForce, a hierarchical speculative decoding system aimed at significantly enhancing the efficiency of serving LLMs with long contexts. Leveraging insights from attention sparsity and contextual locality, TriForce mitigates the dual bottlenecks associated with KV cache and model weights. Our empirical experiments demonstrate TriForce's remarkable performance, including a notable speedup of up to 2.31× on A100 GPUs and an extraordinary 7.78× in offloading scenarios on two RTX 4090 GPUs, achieving 0.108s/token—only half as slow as the auto-regressive baseline on an A100. On a single RTX 4090 GPU, TriForce is 4.86× faster than DeepSpeed-Zero-Inference [2]. Additionally, it attains a 1.9× speedup with large batches. These achievements illustrate TriForce's potential to revolutionize the serving of long-context models for long sequence generation.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15. IEEE, 2022.

[3] Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

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

[5] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

[6] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[7] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024.

[8] Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Jie Huang, and Kevin Chen-Chuan Chang. Cascade speculative drafting for even faster llm inference. *arXiv preprint arXiv:2312.11462*, 2023.

[9] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

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

[11] 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.

[12] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

[13] 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.

[14] Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Hanyu Dong, and Yu Wang. Flashdecoding++: Faster large language model inference on gpus. *arXiv preprint arXiv:2311.01282*, 2023.

[15] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Hasan Genc, Kurt Keutzer, Amir Gholami, and Sophia Shao. Speed: Speculative pipelined execution for efficient decoding. *arXiv preprint arXiv:2310.12072*, 2023.

[16] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

[17] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[18] Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*, 2024.

[19] Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. Speculative decoding with big little decoder. *Advances in Neural Information Processing Systems*, 36, 2024.

[20] 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, 2018. doi: 10.1162/tacl_a_00023. URL `https://aclanthology.org/Q18-1023`.

[21] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

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

[23] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

[24] Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with ringattention. *arXiv preprint arXiv:2402.08268*, 2024.

[25] 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, 2024.

[26] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

[27] 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*, pages 22137–22176. PMLR, 2023.

[28] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 2023.

[30] Péter Vingelmann NVIDIA and Frank HP Fitzek. Cuda, release: 10.2. 89. *URL https://developer. nvidia. com/cuda-toolkit. Cited*, page 148, 2020.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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

[33] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

[34] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL `https://arxiv.org/abs/1911.05507`.

[35] Varshini Reddy, Rik Koncel-Kedziorski, Viet Dac Lai, and Chris Tanner. Docfinqa: A long-context financial reasoning dataset. *arXiv preprint arXiv:2401.06915*, 2024.

[36] Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. *arXiv preprint arXiv:2305.10427*, 2023.

[37] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.

[38] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[39] Benjamin Spector and Chris Re. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

[40] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

[41] Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[43] 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.

[44] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[45] Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*, 2024.

[46] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

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

[48] Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. *arXiv preprint arXiv:2402.01528*, 2024.

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

[50] Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. Wkvquant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*, 2024.

[51] Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*, 2023.

[52] Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Soaring from 4k to 400k: Extending llm's context with activation beacon. *arXiv preprint arXiv:2401.03462*, 2024.

[53] 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.

[54] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[55] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi : Plug-and-play 2bit kv cache quantization with streaming asymmetric quantization. 2023. doi: 10.13140/RG.2.2.28167.37282. URL `https://rgdoi.net/10.13140/RG.2.2.28167.37282`.

Acknowledgments

We thank Yang Zhou, Ranajoy Sadhukhan, Harry Dong, and Jian Chen for their helpful discussions and feedback on early drafts of the paper.