Evaluating Speculative Decoding and PagedAttention on Consumer GPUs [E]

Vimalnath Achuthan, Gopinath Balaji, Yuhao Ge, Gisaldjo Purbollari
{va33,gbalaji4,yuhaoge2,gp27}@illinois.edu
University of Illinois Urbana-Champaign, Urbana, Illinois, USA

1 Introduction

Large language models (LLMs) have become essential for tasks such as text generation, summarization, and conversational agents, and they have revolutionized natural language processing. However, their widespread adoption is hindered by inference latency and computational demands, especially in resource-constrained environments. Although recent advances like vLLM[1] have dramatically improved memory efficiency through techniques like *PagedAttention*, [1], the fundamental limitations of sequential autoregressive decoding persist, particularly on consumer-grade hardware where computational resources are restricted.

Speculative decoding [2] has emerged as a promising approach that allows a smaller draft model to propose multiple tokens in parallel, which are subsequently verified or corrected by a larger target model. This technique reduces latency by minimizing the dependency on sequential token generation. Although speculative decoding has shown impressive performance improvements, such as up to 2.8x speedups[2] in certain configurations, its performance varies significantly depending on the system settings, including the choice of the draft model, batch size, and hardware resources.

vLLM has recently integrated speculative decoding support[3], however, its performance characteristics are not fully understood, particularly in resource-constrained environments. The current implementation faces optimization challenges, with inconsistent latency reductions in different workloads and configurations, as documented by the owners of vLLM[4] and reported by their users[5]. This motivates our systematic evaluation of speculative decoding in vLLM on consumer-grade GPUs, examining its behavior across various parameters including speculative token length, batch sizes, draft model sizes, etc. Our investigation aims to provide practical insights for deploying these optimizations in environments where high-end computational resources are not available.

2 Related Work

2.1 vLLM: An Efficient Inference Framework for Large Language Models

vLLM is an efficient system for LLM inference that introduces novel techniques such as *PagedAttention* [1] and continuous batching to optimize memory management and maximize hardware utilization. *PagedAttention* improves key-value (KV) cache management by enabling on-demand memory allocation and sharing, which significantly reduces the memory overhead of LLM inference. The continuous batching feature processes multiple requests concurrently, resulting in higher throughput and lower latency.

Despite these advancements, the sequential nature of autoregressive decoding remains a bottleneck in vLLM, particularly for latency-sensitive applications. To address this, the vLLM team recently integrated speculative decoding into its framework, a feature designed to mitigate the limitations of sequential token generation.

2.2 Speculative Decoding: Accelerating Token Generation

Speculative decoding [2] aims to reduce inference latency by parallelizing token generation. In this approach, a smaller draft model proposes multiple tokens in a single forward pass, and a larger target model verifies or corrects these tokens. This reduces the dependency on sequential token generation, enabling faster inference without compromising accuracy. The process involves two key components:

- **Draft Model:** A lightweight model predicts a sequence of candidate tokens.
- **Target Model:** A larger, more accurate model verifies the candidate tokens and corrects any mismatches.

2.3 Limitations

Speculative decoding represents a significant advance in LLM inference optimization, with recent studies demonstrating promising speedups in datacenter environments[6] [7]. However, existing research primarily focuses on high-performance systems like NVIDIA A100 GPUs, creating a significant gap in understanding performance characteristics in more common deployment scenarios.

Our research aims to analyze how speculative decoding adapts to consumer GPU constraints and identify configurations that optimize the tradeoff between latency and throughput. We focus on hardware configurations constrained by low memory bandwidth, fewer computing cores, and lower power efficiency - conditions faced by many independent developers, academic researchers, and small businesses without access to enterprise-grade infrastructure.

Current studies are insufficient for understanding the behavior of speculative decoding in consumer-grade settings for several key reasons:

Hardware Bias. Most benchmarks neglect the unique constraints of consumer-grade GPUs, such as limited memory bandwidth and susceptibility to thermal throttling, which can significantly affect performance.

Oversimplified Configurations. Existing studies often assess performance under ideal conditions with large batch sizes and optimal

[1]https://github.com/vllm-project/vllm
[2]https://blog.vllm.ai/2024/10/17/spec-decode.html
[3]https://blog.vllm.ai/2024/10/17/spec-decode.html
[4]https://docs.vllm.ai/en/latest/models/spec_decode.html
[5]https://github.com/vllm-project/vllm/issues/10318

[6]https://huggingface.co/blog/dynamic_speculation_lookahead
[7]https://pytorch.org/blog/hitchhikers-guide-speculative-decoding/

memory availability. These configurations are impractical for consumer GPUs with constrained VRAM, where smaller batch sizes and frequent memory swaps are necessary.

Underexplored Parameter Space. Critical parameters such as the draft model size and speculative token length are not thoroughly examined in resource-constrained environments. The relationship between these parameters and performance could vary significantly across different model configurations.

While vLLM provides techniques like continuous batching and memory sharing, their interaction with speculative decoding remains unstudied in consumer hardware contexts. Understanding these relationships is crucial for optimizing performance. Our research addresses several key questions:

- *Resource Constraints:* How do memory limitations and computational constraints affect speculative decoding performance on different consumer hardware?
- *Performance Analysis:* How do different configurations perform on consumer GPUs?
- *Future Directions:* What insights from our evaluation can inform the design of self-adapting speculative decoding implementations?

By shedding light on these gaps, our study aims to drive innovation in memory-efficient and latency-optimized inference techniques specifically targeted at consumer hardware constraints, contributing to the democratization of advanced language model inference.

3 Our Approach and Experiment Settings

To assess the performance of speculative decoding in vLLM[8], we conducted extensive experiments. Our implementation leveraged the most recent vLLM package, which supports speculative decoding, as detailed in their blog post[9].

Our experiments were executed on two different GPUs to evaluate performance across varying hardware capabilities:

- **NVIDIA A40 GPU**: Equipped with 48 GB of memory, suitable for large-scale inference tasks.
- **NVIDIA GeForce RTX 3080 GPU**: Offers 10 GB of memory, representing a consumer-grade hardware scenario.

We systematically swept over several critical parameters to understand their impact on the performance of speculative decoding:

- **Number of Requests**: We tested with 32, 128, and 512 concurrent requests to simulate different server load conditions.
- **Batch Sizes**: Batch sizes of 1, 4, 8, 16, and 256 were used to analyze the effect of batching on throughput and latency.
- **Speculative Models**: We evaluated performance without speculative decoding (`null`), and with pre-trained models including `facebook/opt-125m` and `facebook/opt-350m`[10].
- **Number of Speculative Tokens**: Values of 1, 3, 5, and 10 were used to assess how the number of speculative tokens influences performance and resource utilization.

[8]https://github.com/vllm-project/vllm
[9]Accelerating Language Model Decoding via Speculative Sampling, vLLM Blog, https://blog.vllm.ai/2023/10/17/spec-decode.html
[10]Available at https://huggingface.co/facebook

The target language model for all was `facebook/opt-1.3b`[11], a 1.3-billion-parameter model known for its balance between performance and computational requirements. The base prompt for text generation was set to "The future of AI is," allowing for consistent evaluation across different configurations. We adjusted the `gpu_memory_utilization` parameter to 0.9, which permits vLLM to utilize up to 90% of the available GPU memory for caching and internal buffers, optimizing resource usage without overcommitting memory.

We aimed to measure key performance metrics under varying conditions:

- **Latency**: The average time taken to generate responses for a batch of requests.
- **Output Token Throughput**: The rate at which output tokens are generated per second.

By systematically varying one parameter at a time while holding others constant, we isolated the effects of each factor on the performance of speculative decoding. This approach allowed us to identify optimal configurations and understand the trade-offs involved in different settings. All experiments were reproducible using the codebase we developed[12], and the configurations are provided in the `config.yaml` file included in our repository.

4 Evaluations

4.1 Impact of Number of Requests on Throughput and Latency

We evaluated the effect of varying the number of concurrent requests on the performance of speculative decoding in vLLM. With the number of speculative tokens fixed at 3, batch size at 4, and using the NVIDIA A40 GPU, we compared two configurations: without speculative decoding (`None`) and using `facebook/opt-125m` as the speculative model.

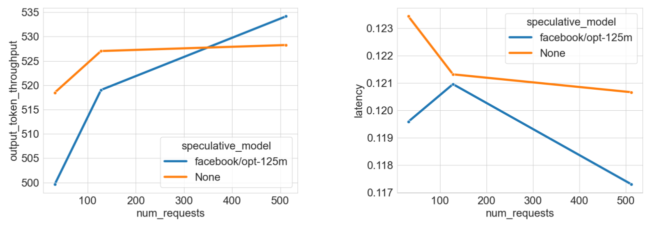

Figure 1: Performance metrics vs. number of requests. Left: Throughput; Right: Latency.

As shown in Figure 1 (left), increasing the number of requests leads to higher throughput for both configurations, due to better utilization of system resources and parallelism in vLLM. The speculative decoding configuration with `facebook/opt-125m` exhibits a steeper increase in throughput compared to the baseline, indicating more effective scaling. Figure 1 (right) shows that speculative decoding achieves lower latency than the baseline no matter what is the number of requests.

[11]OPT Models by Facebook AI Research, https://huggingface.co/facebook/opt-1.3b
[12]https://github.com/Geyuhao/cs511_final

In summary, increasing the number of concurrent requests improves throughput and reduces latency, with speculative decoding providing significant advantages over the baseline. This demonstrates the effectiveness of speculative decoding in high-load scenarios.

4.2 Impact of Batch Size on Throughput and Latency

We investigated the effect of varying batch sizes on the performance of speculative decoding in vLLM. In this experiment, we fixed the number of speculative tokens at 3, the number of requests at 128, and used the NVIDIA A40 GPU. We compared two configurations: without speculative decoding (`None`) and using `facebook/opt-125m` as the speculative model, considering batch sizes up to 32.

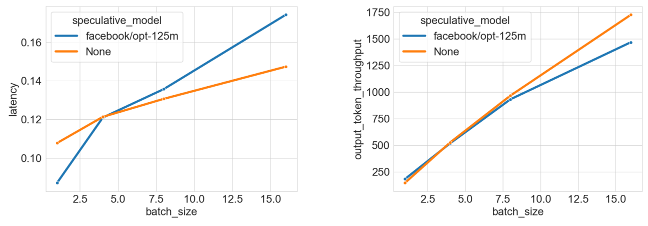

Figure 2: Performance metrics vs. batch size. Left: Latency; Right: Throughput.

As shown in Figure 2 (left), for smaller batch sizes (up to 8), speculative decoding with `facebook/opt-125m` outperforms the baseline in terms of output token throughput. However, when the batch size is greater than or equal to 8, the throughput of the speculative decoding configuration becomes worse than the baseline. Similarly, Figure 2 (right) indicates that latency increases more rapidly for speculative decoding as batch size increases.

This counterintuitive result can be attributed to the overhead introduced by the speculative model during larger batch processing. Speculative decoding involves running both the target model and the speculative model. While the speculative model is smaller and faster, coordinating between the two models introduces additional computational overhead, especially noticeable at larger batch sizes. As the batch size increases, the synchronization and communication overhead between the models becomes more significant, diminishing the benefits of speculative decoding.

In contrast, the baseline without speculative decoding relies solely on the target model. At larger batch sizes, the target model can better utilize GPU resources due to increased parallelism within a single model, leading to improved throughput and relatively lower latency compared to the speculative decoding configuration.

In summary, while speculative decoding offers performance advantages at smaller batch sizes by reducing latency and increasing throughput, it may not scale as effectively at larger batch sizes due to the overheads that outweigh its benefits. This suggests that speculative decoding is more beneficial in scenarios with smaller batch sizes or when low latency is crucial.

4.3 Optimal Number of Speculative Tokens for Different Batch Sizes

We examined the effect of varying the number of speculative tokens on latency when using different speculative models. The experiments were conducted on the NVIDIA A40 GPU with the number of requests fixed at 32. We compared three configurations: without speculative decoding (`None`), using `facebook/opt-125m`, and using `facebook/opt-350m` as the speculative models.

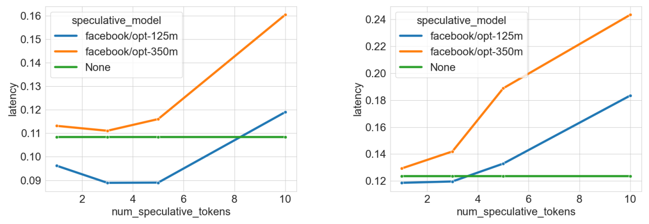

Figure 3: Latency vs. Number of Speculative Tokens. Left: Batch Size 1; Right: Batch Size 4.

As shown in Figure 3, when the batch size is 1 (left), the `facebook/opt-125m` speculative model generally achieves better latency compared to the baseline without speculative decoding. The optimal number of speculative tokens for `facebook/opt-125m` is at 3 or 5, where the latency is minimized. However, when the number of speculative tokens increases to 10, the latency worsens and becomes even higher than the baseline. This suggests that beyond a certain point, the overhead due to the need of managing more speculative tokens and a lower acceptance rate outweighs the benefits, leading to increased latency.

For the `facebook/opt-350m` speculative model, the latency improvements are less pronounced, and the optimal number of speculative tokens is lower. This is likely due to the larger size of the speculative model, which introduces more computational overhead, diminishing the benefits of speculative decoding at higher token counts.

When the batch size is increased to 4 (Figure 3, right), the optimal number of speculative tokens for `facebook/opt-125m` shifts to 1 or 3. The latency improvements over the baseline are less significant compared to the batch size of 1. This could be because, at higher batch sizes, system resources are already more effectively utilized, and the additional overhead of speculative decoding provides diminishing returns, especially with larger speculative models like `facebook/opt-350m`.

These findings highlight the importance of tuning the number of speculative tokens based on the batch size and the size of the speculative model to achieve optimal performance.

4.4 Impact of Number of Speculative Tokens on Latency Across GPUs

We analyzed how the number of speculative tokens affects latency on different GPUs when using speculative decoding in vLLM. The experiments were conducted with fixed parameters: batch size of 4, number of requests set to 128, and using either `facebook/opt-125m` or `facebook/opt-350m` as the speculative model. We considered two GPUs: the NVIDIA A40 and the RTX 3080.

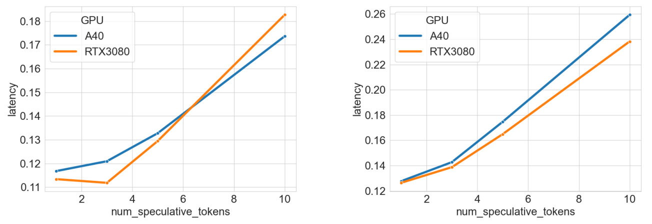

Figure 4: Latency vs. Number of Speculative Tokens. Left: `facebook/opt-125m`; Right: `facebook/opt-350m`.

As shown in Figure 4, for the smaller speculative model (`facebook/opt-125m`), the RTX 3080 GPU achieves optimal latency at 3 speculative tokens, while the A40 GPU reaches optimal latency at 1 speculative token. When the number of speculative tokens exceeds these optimal points, latency increases for both GPUs. Notably, when the number of speculative tokens exceeds 6, the A40 GPU demonstrates better performance than the RTX 3080.

In contrast, for the larger speculative model (`facebook/opt-350m`), both GPUs achieve optimal latency at 1 speculative token, and the RTX 3080 consistently exhibits better latency than the A40 GPU across all numbers of speculative tokens. Additionally, when using the larger speculative model, the A40 GPU's latency remains higher than that of the RTX 3080 regardless of the number of speculative tokens.

These results suggest that the optimal number of speculative tokens depends on both the size of the speculative model and the GPU used. For smaller speculative models, the RTX 3080 benefits from a moderate number of speculative tokens, while the A40 GPU performs best with fewer speculative tokens. With larger speculative models, both GPUs perform best with the minimal number of speculative tokens, and the RTX 3080 outperforms the A40 GPU in terms of latency.This behavior may be due to differences in GPU architecture, memory capacity, and computational capabilities.

In summary, selecting the optimal number of speculative tokens requires considering both the speculative model size and the GPU characteristics. Balancing these factors can lead to improved latency and overall performance in speculative decoding.

5 Conclusion

This study provides a comprehensive evaluation of speculative decoding in vLLMs, focusing on its performance across varying configurations and hardware settings, including consumer-grade GPUs such as the NVIDIA RTX 3080 and A40. Our experiments demonstrate that speculative decoding offers significant improvements in throughput and latency under specific configurations, though its benefits are highly sensitive to batch size, speculative model size, and the number of speculative tokens.

Our key findings include the following:

- **Batch Size Dependency:** Speculative decoding achieves its greatest benefits with smaller batch sizes, where latency and throughput improve significantly. However, its advantages diminish at larger batch sizes due to the synchronization and computational overhead introduced by the speculative model.
- **Speculative Model Selection:** Smaller speculative models, such as `facebook/opt-125m`, are better suited for consumer GPUs, offering a favorable trade-off between computational overhead and performance improvements. Larger models often negate the benefits due to increased resource demands.
- **Optimal Speculative Token Count:** The number of speculative tokens significantly affects latency and throughput, with an optimal range dependent on batch size, GPU architecture, and speculative model size. Beyond the optimal range, performance deteriorates due to lower token acceptance rates and synchronization overhead.
- **Hardware-Specific Performance:** Consumer-grade GPUs like the RTX 3080 exhibit distinct performance characteristics compared to datacenter-grade GPUs like the NVIDIA A40. Resource constraints such as limited VRAM and lower computational throughput necessitate careful tuning of speculative decoding configurations for optimal performance.

These results emphasize the importance of tailoring speculative decoding settings to the specific workload and hardware environment. Developers and researchers utilizing resource-constrained GPUs can leverage these findings to maximize performance by employing smaller speculative models, limiting batch sizes, and optimizing the number of speculative tokens.

Future work will explore dynamic speculative decoding techniques that adapt configurations in real time based on workload characteristics and system load. These strategies have the potential to further enhance the efficiency of speculative decoding in vLLMs, bridging the gap between enterprise-level and consumer-grade hardware for advanced language model inference.

References

[1] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management For Large Language Model Serving With PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 611–626.

[2] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast Inference From Transformers Via Speculative Decoding. In *International Conference on Machine Learning*. PMLR, 19274–19286.