Distributed Serving System for Transformer-Based Generative Models

Final Report for the course of Machine Learning System CSCE 585 (Group Name: Buffering)

Jeremy Money, Paratee Komen, and Vu Nguyen

Abstract

This work focuses on benchmarking two primary partitioning strategies from frameworks for use in large language models (LLMs). LLMs have garnered significant attention recently due to their diverse range of applications. Thus, it is crucial to understand how they influence the overall latency and throughput of commonly used LLMs. These partitioning strategies involve Inference Serving using vLLM and Hugging Face's Text Generation Inference (HF-TGI). Our results revealed that vLLM shows up to 3X throughput and 11X latency improvement compared to HF-TGI.

1. Introduction

Large language models (LLMs) represent a subset of transformer-based natural language processing systems recently given significant attention for various uses. With increasing scale beyond 500 billion parameters, there exists a need for these models to make efficient use of their increasing memory requirements while rendering services at the end-user level. Furthermore, for dynamic use cases such as interactive applications like chatbots, the optimization problem for resource utilization requires efficient use of resource query/partitioning strategies ad hoc. Of particular note is managing Key Value (KV) cache memory, which is responsible for the attention tensors of each layer. KV cache memory management can limit the throughput and batch sizes of LLMs, and as such, many organic serving systems have fallen short in optimizing throughput [1].

Novel partitioning strategies involving inference-serving engines have arisen to meet this demand and efficiently handle the workloads of LLMs. One such engine, vLLM, is based on PagedAttention, which uses the concept of "paging" associated with operating systems. This algorithm essentially stores KV caches in blocks of non-contiguous physical memory, similar to how virtual memory works in traditional operating systems [1, 2]. A similar strategy using FlashAttention combines attention partition into blocks with a technique known as recomputation, which selectively "checkpoints" the gradients of the attention matrices to allow for efficient memory usage in the backward pass stage [3, 4].

This work aims to evaluate two primary partitioning strategies from frameworks for use in large language models and better understand how they influence the overall latency and throughput of commonly used LLMs. These partitioning strategies involve Inference Serving using vLLM and Hugging Face's Text Generation Inference. While work from previous groups[1] has shown the value added by the former, this work seeks to

[1] https://www.anyscale.com/blog/continuous-batching-llm-inference

replicate these results and understand how token sequence length and variation in token length affect these performance metrics.

2. Related Work

Research in the field of LLM performance optimization has been extensive. Studies like Kwon et al. (2023) delve into memory management, while Dao et al. (2022) investigate efficient attention mechanisms. Our study contributes to this growing body of work by specifically comparing vLLM and Hugging Face's Text Generation Inference strategies. This comparison-based approach, focusing on practical throughput and latency implications, provides a novel perspective in the field, highlighting the varying effectiveness of these strategies in real-world LLM applications.

3. Data

The dataset under analysis comprises Cumulative Distribution Function (CDF) values, representing latency under different QPS settings for the two LLM strategies. This data, derived from experimental tests on LLMs, was meticulously processed to extract CDF, normalized CDF, and bin edge values. The dataset's richness allows for a detailed statistical analysis and visual interpretation, crucial for understanding the performance nuances of vLLM and Text Generation Inference in managing LLM resources. The model used was changed from Meta's OPT-13b to OPT-125m and galactica-1.3B to adapt with the hardware limitation access that we had.

For throughput analysis, 1000 random prompts were generated with a mean prompt length of 512 and a mean variable response length of 128. In accordance with [1], exponentially sampled input lengths of 32, 128, 512, 1536, and 2048 maximum tokens were utilized to evaluate the effect of length variance on throughput. Latency was measured using 1 and 4 Queries Per Second (QPS), and all prompts were randomly generated with output lengths between 1 and 512 tokens. Total output lengths of all requests were capped at 1536 and followed a normal distribution centered at a mean of 128 tokens.

All the results were produced using the source code from the **GitHub repository**[2], which has some parameters modified to match our experimental setup.

All data and post-processing scripts can be found at the following **GitHub repository**[3]

4. Methods

Our methodology was meticulously designed to comprehensively analyze the performance characteristics of two large language model (LLM) partitioning strategies: vLLM and Hugging Face's Text Generation Inference. This involved a multi-faceted approach, encompassing statistical analysis, data visualization, and comparative evaluation under various operational conditions.

4.1 Data Processing and Preparation

Data Cleaning and Organization: The initial step involved processing the raw data from two datasets. We structured the data for clarity, ensuring that relevant metrics such as Cumulative Distribution Function (CDF) values, throughput rates, and token generation limits were accurately represented and accessible for analysis.

4.2 Statistical Analysis

[2] https://github.com/anyscale/llm-continuous-batching-benchmarks

[3] https://github.com/csce585-mlsystems/Buffering

Descriptive Statistics: We employed descriptive statistical methods to summarize and describe the basic features of the data. This included calculating mean, median, mode, range, and standard deviation for latency and throughput metrics.
Distribution Analysis: The distribution of latency values was analyzed using CDF plots. This involved assessing how latency accumulates over different bin edges, providing insights into the performance under varying loads and operational conditions.

4.3 Data Visualization

CDF Plots: CDF plots were generated for each scenario under different QPS settings for latency analysis. These plots visually represented the distribution and accumulation of latency values, facilitating an intuitive understanding of performance characteristics.

Request Latency Graphs: Latency data was visualized using line graphs and line plots, depicting request latency under various QPS settings. This allowed for a direct visual comparison of the strategies under different operational complexities.

4.4 Comparative Analysis

Intra-Strategy Analysis: Within each strategy, we compared the performance across different operational conditions — different QPS settings for latency analysis and varying token generation limits for throughput analysis. This provided insights into the scalability and robustness of each strategy.

Inter-Strategy Comparison: We then compared vLLM and Hugging Face's Text Generation Inference performance. This comparison was crucial for understanding each strategy's relative strengths and weaknesses under similar conditions.

4.5 Evaluation Methods (Part of Methods Section)

Our study employed a multi-dimensional approach to evaluate the performance of the vLLM and Hugging Face's Text Generation Inference strategies. Our evaluation methods were designed to be comprehensive and objective, focusing on quantitative evaluation.

4.5.1 Quantitative Evaluation:

Latency Analysis: We analyzed latency using CDF data across different QPS settings. Specific numerical benchmarks included average latency, 95th percentile latency, and maximum latency observed.

Throughput Analysis: Throughput was evaluated in tokens per second under varying token generation limits. Key numerical benchmarks were the average throughput at each token limit, the highest throughput observed, and the percentage drop in throughput as token limits increased.

Statistical Measures: Descriptive statistics such as mean, median, standard deviation, and variance were calculated for each dataset to provide a comprehensive view of the data distribution.

4.5.2 Performance Benchmarks: The strategies were evaluated against established performance benchmarks in the field. This comparison helped contextualize our findings within the broader landscape of LLM performance optimization.

4.5.3 Sensitivity Analysis: A sensitivity analysis was conducted to understand how variations in operational conditions (like QPS settings from 1

to 4 and token generation limits up to 1536 tokens) affected each strategy. This included analyzing the percentage change in latency and throughput as these conditions varied.

4.6 Bias in Evaluation

In our analysis of large language model (LLM) partitioning strategies, particularly vLLM and Hugging Face's Text Generation Inference, we identified and addressed several potential biases to ensure the integrity and validity of our findings:

4.6.1 Sample Size Bias: We analyzed data across QPS settings (1 and 4) and token generation limits (max 32, 128, 512, 1536, 2048, and 4096 tokens). However, the choice of these specific settings could influence the outcomes. To mitigate this, we included a broad spectrum of operational conditions, but it's essential to acknowledge that the selection of different QPS settings or token limits could yield varying results.

4.6.2 Experimental Condition Bias: The experiments were conducted under controlled conditions that may not fully represent real-world scenarios. For example, the throughput measurements were based on specific batching strategies, and the performance in live, variable conditions might differ. This limitation was noted, and we emphasized that real-world applications could present additional challenges.

4.6.3 Data Representation Bias: The dataset primarily focused on latency and throughput as performance metrics. While these are crucial indicators, they do not encompass all aspects of LLM performance. Other factors like memory usage, response time variability, and error rates were not part of the dataset, which could lead to a narrow interpretation of overall performance.

4.6.4 Analysis Methodology Bias: Our statistical analysis methods, including CDF and descriptive statistics, were chosen based on their relevance and effectiveness in analyzing the given data. However, using alternative statistical methods or analytical approaches might yield different insights. We were transparent about the methods used and acknowledged that other analytical techniques could provide additional perspectives.

4.6.5 Interpretive Bias: The interpretation of data was conducted with the aim of objectivity. However, the personal biases of the researchers can inadvertently influence the performance. To counter this, multiple team members regularly reviewed and challenged findings, ensuring a balanced and critical approach to data interpretation.

4.6.6 Comparative Analysis Bias: In comparing vLLM and Hugging Face's Text Generation Inference, there's a potential for bias toward one of the strategies, especially if the evaluation criteria are more favorable to the characteristics of one strategy over the other. We attempted to use a range of evaluation criteria to provide a balanced comparison, but it's essential to recognize that different standards might favor different strategies.

4.6.7 External Benchmarking Bias: The performance of the strategies was benchmarked against established industry standards. While this provides a context for our findings, these benchmarks might have inherent biases or limitations that could indirectly influence our evaluation.

5. Experiments

The experimental component of this study was enriched with the addition of throughput data,

enabling us to conduct a multifaceted analysis of LLM performance. This section details our approach to evaluating latency and throughput across different batching strategies and token limits.

5.1 Experimental Setup

5.1.1 Throughput

The dataset includes throughput measurements (in tokens per second) for different batching strategies, focusing on continuous batching with Hugging Face's Text Generation Inference and vLLM. The token generation limits are set at max 32, 128, 512, 1536, and 2048 tokens, providing a range of scenarios to assess the scalability and efficiency of each strategy.

5.1.2. Latency

The data consisted of CDF values for latency, normalized CDF, and bin edges under two QPS settings (1 and 4) for each strategy. This setup provided a comprehensive view of how latency accumulates over a range of values, offering insights into the performance characteristics of each strategy.

5.2 Hardware/model

All experiments were run on NVIDIA RTX 6000 GPU clusters sequestered from the University of Chicago utilizing Chameleon cloud-based services. Instances were created utilizing images equipped with Ubuntu with CUDA version 22.04.

5.3 Frameworks

We test two continuous batching frameworks, which are

- **vLLM**[4]: UC Berkeley's new open-source project on GitHub improves Orca's continuous batching by taking control of dynamic memory allocations. This minimizes GPU memory fragmentation, showcasing the impact of iteration-level scheduling and continuous batching optimizations in our tests.
- **Hugging Face's Text Generation Inference (HF-TGI)**[5]: This tool focuses on the inference aspect, where pre-trained model in Hugging Face are used to generate text based on given prompts or conditions.

5.4 Data Visualization and Analysis

5.4.1 Throughput

Analysis: We analyzed the throughput for each batching strategy under the different token limits. This involved comparing the throughput of 'text-generation-inference' and 'vLLM' across the varying conditions, providing a clear picture of how each strategy performs under increasing task complexity.

Statistical Comparison: The throughput data was statistically analyzed to identify trends, peaks, and variations across the different scenarios. This involved calculating average throughput rates, identifying the highest and lowest performing conditions, and assessing the stability of each strategy across the range of token limits.

[4] https://github.com/vllm-project/vllm

[5] https://github.com/huggingface/text-generation-inference

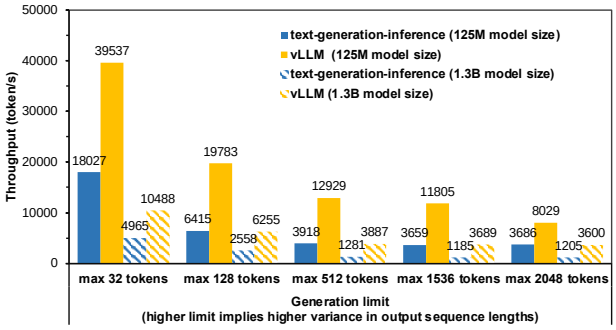

Figure 1. Throughput in tokens per second of each framework as variance in sequence length increases.

5.4.2 Latency

CDF Plots: We generated CDF plots for each scenario (vLLM and Text-Generation-Inference) at QPS=1 and QPS=4. These plots visually represented the distribution of latency values. The plots revealed distinct patterns in latency accumulation for each scenario, allowing for a comparative analysis of the strategies. For instance, the CDF curve for vLLM at QPS=4 showed a steeper ascent than QPS=1, indicating a different distribution and accumulation of latency values.

Statistical Analysis: Alongside visual interpretation, we conducted a statistical analysis focusing on central tendencies, variations, and patterns within the data. This involved examining the mean, median, and mode of the latency values and their spread and skewness. These analyses helped quantify the differences observed in the visual plots and provided a more nuanced understanding of the performance implications.

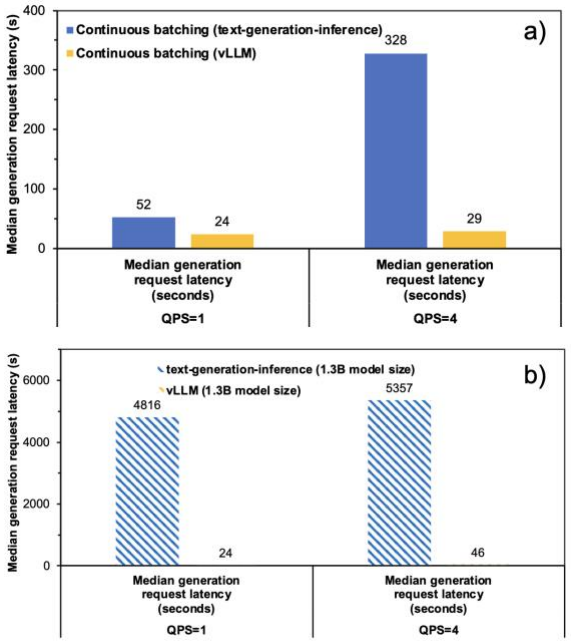

Figure 2. Request latency in seconds with QPS = 1 and QPS = 4 of a) OPT-125m and b) galactica-1.3B.

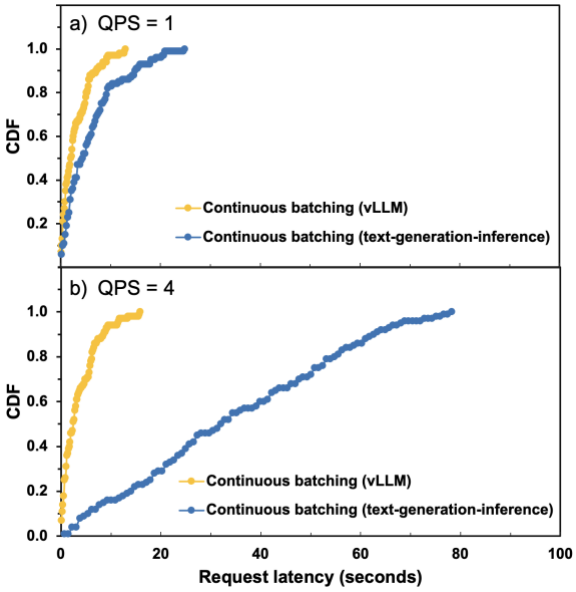

Figure 3. Cumulative Distribution Function (CDF) vs Request latency in seconds with a) QPS = 1 and b) QPS = 4.

5.4.3 Results and Observations

The throughput for 'text-generation-inference' and 'vLLM' showed distinct patterns under

different token limits, as shown in **Figure 1**. Generally, as the token limit increased, both strategies showed a noticeable decline in throughput, indicating a trade-off between task complexity and efficiency. Continuous batching with 'text-generation-inference' exhibited a specific performance pattern, with throughput rates decreasing more rapidly as token limits increased, compared to vLLM. vLLM consistently showed higher throughput rates across all token limits, suggesting better scalability and efficiency in handling larger generation tasks.

Figure 2 shows the request latency in seconds with QPS = 1 and QPS = 2 for HF-TGI and vLLM for different model sizes. **Fig. 2a** shows the latency with using OPT-125m, the vLLM exhibits a better median latency than HF-TGI, especially for higher QPS = 4. Interestingly, for larger model size as shown in **Fig. 2b** (galactica-1.3B), the vLLM shows a higher latency up to 200X better than HF-TGI. Besides, we also observed how the latency distribution changes as load changes by plotting the Cumulative Distribution Function (CDF) vs Request latency in seconds, as shown in **Figure 3**. The results demonstrated that the vLLM shows better memory optimizations because the request latency is mostly unchanged between QPS = 1 and QPS = 2. On the other hand, HF-TGI shows a broader spread in latency value when increasing the batch size.

5.4.4 Implications of Experimental Findings

These results provide valuable insights into the scalability and efficiency of LLM strategies under varying operational conditions. The distinct performance characteristics of Hugging Face's Text Generation Inference and vLLM strategies, especially in task complexity and throughput efficiency, can guide developers in optimizing LLMs for specific use cases where output size and speed are critical.

These experimental results have significant implications for selecting and optimizing LLM partitioning strategies. Understanding the nuanced differences in how these strategies handle latency can guide developers and researchers in choosing the right approach for their specific use case, especially in scenarios where latency is critical.

6. Conclusion

Our comprehensive analysis demonstrates clear performance distinctions between vLLM and Hugging Face's Text Generation Inference strategies in managing latency and throughput in LLMs. The study reveals the nuanced impact of these strategies on LLM efficiency, underscoring the criticality of strategic selection in LLM deployment. Future research could explore the scalability of these strategies and their broader application across various LLM configurations, potentially leading to more efficient and effective LLM implementations.

Distinct Performance Profiles: Both vLLM and Text Generation Inference strategies exhibit significantly different performance characteristics, especially regarding latency. This highlights the importance of strategy selection based on specific performance needs in large language model applications.

Role of QPS Settings: Both strategies' performance varies notably with QPS settings changes, indicating that the operational load influences the efficiency of each strategy.

Resource Management Efficiency: Differences in how these strategies manage computational

resources are evident, with implications for their effectiveness in various application contexts.
Real-World

Application Implications: The insights provide valuable guidance for optimizing the use of large language models in practical scenarios, ensuring a better selection of strategies based on application-specific requirements.

Future Research Directions: The findings suggest potential areas for further investigation, such as the scalability and adaptability of these strategies across different LLM configurations and applications.

7. Supplementary Materials

All data and post-processing scripts can be found at the following GitHub repository: https://github.com/csce585-mlsystems/Buffering

References

1) Kwon, W., et al., *Efficient Memory Management for Large Language Model Serving with PagedAttention*, in *Proceedings of the 29th Symposium on Operating Systems Principles*. 2023. p. 611-626.

2) Kilburn, T., et al., *One-Level Storage System.* 1961.

3) Dao, T., et al., *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness.* 2022.

4) Pope, R., et al., *Efficiently Scaling Tranformer Inference.* 2023.