Revisiting Service Level Objectives and System Level Metrics in Large Language Model Serving

Anonymous ACL submission

Abstract

Large language models (LLMs) have achieved remarkable performance and are widely deployed in various applications, while the serving of LLM inference has raised concerns about maintaining high user experience and achieving sufficient throughput. Balancing these factors is crucial for reducing operational costs while ensuring optimal performance. Accordingly, service level objectives (SLOs) and system level metrics have been introduced as key performance measures for LLM serving. However, current metrics fall short in accurately capturing user experience. We find two notable issues: 1) manually delaying the delivery of some tokens can improve metrics of requests, and 2) actively abandoning requests that do not meet SLOs can improve system level metrics.

In this paper, we revisit SLOs and system level metrics in LLM serving and propose a comprehensive metric framework called *smooth goodput*, which integrates SLOs and system level metrics to reflect the nature of user experience in LLM serving. It is designed to be adaptable, with parameters that can be tailored to the specific objectives of various tasks. Through this unified framework, we reassess the performance of different LLM serving systems under multiple workloads. We aspire for this framework to establish a standardized method for evaluating LLM serving, thereby encouraging cohesive advancements in future research.

1 Introduction

Large language models (LLMs) have achieved remarkable performance in many tasks and are widely deployed in various applications, such as chatbots (OpenAI, 2024; Zheng et al., 2024; Montagna et al., 2023) and virtual assistants (Vu et al., 2024; Dong et al., 2023). With the increasing demand for LLM services, researchers have proposed various optimization strategies for LLM serving systems. Initially, the LLM serving systems are designed to maximize the throughput(Yu et al., 2022; Kwon et al., 2023). A straightforward approach is to increase the batch size of the requests to improve the resource utilization, thereby increasing the throughput. However, large batch sizes may lead to high latency, which may degrade the user experience. We notice in the real-world LLM serving applications, the user need to interact with the system, such as (OpenAI, 2024; DeepSeek-AI et al., 2025; OpenAI et al., 2024; Vu et al., 2024; Dong et al., 2023), which requires a real-time response.

Specially, to evaluate user experience in LLM serving systems, many metrics of single request that measures the token delivery time of a request have been used to in previous work (Patel et al., 2023; Agrawal et al., 2024b; Cheng et al., 2024; Patke et al., 2024), such as time-to-first-token (TTFT), time-between-tokens (TBT), and time-per-output-token (TPOT). For the first token generation, it is costly to process the prefill stage (Vaswani et al., 2023; Zhong et al., 2024), thereby introducing the TTFT for the first token generation, which may significantly larger than the TBT/TPOT. TPOT measures the average time between tokens in a request, while it is too loose to reflect the user experience, as a long stall in the middle of the request can be averaged out by short intervals between other tokens, which actually degrades the user experience. Therefore, (Agrawal et al., 2024b) introduces the TBT metric to constrain the time interval between two consecutive tokens. To further evaluate the performance of LLM serving systems ensuring the SLOs, system level metrics that measure the performance of each request of the system such as SLO attainment and goodput are proposed (Zhong et al., 2024; Agrawal et al., 2024b). The SLO attainment measures the proportion of requests that meet the SLOs, which can be viewed as the constraint of the serving system, while the goodput measures the number of completed requests that meet the SLOs per second, which can be viewed as the performance of the serving system. We also notice that various systems and optimization strategies have been proposed to improve the system level metrics under the SLOs (Patel et al., 2023; Agrawal et al., 2024b; Zhong et al., 2024; Cheng et al., 2024; Patke et al., 2024).

However, we observe that *these metrics fail to capture the nature of user experience*. Real-time LLM service is a rapidly interactive activity, just like web browsing (Weinreich et al., 2008; Skadberg and Kimmel, 2004). Users do not perceive them as a sequence of single isolated token generation events, but as a continuous stream of information. The evaluation bias caused by ignoring the inherent nature of user experience in streaming LLM serving can even lead optimization efforts based on these metrics to develop in a suboptimal direction. We identify several limitations in the existing metrics as follows:

TBT is too tight for overall user experience while TPOT and E2E latency are too loose. TBT measures the time interval between each token within a request, while TPOT reflects the average interval. As indicated in (Egger et al., 2012), user experience in streaming services is influenced by waiting times without information to process. If users have enough information to process, occasional stalls (i.e., high TBT) may not degrade the experience. For example, if a system delivers 10 tokens in the first second, then stalls for 1 second, users reading at 4 tokens per second will still have a good experience, although the TBT is up to 1 second. Conversely, if only 2 tokens are delivered before a 1-second stall, users will suffer from the waiting time, although the TPOT is only 0.1s. In other words, the cost of high latency iterations is shared with *previous* iterations.

Goodput and SLO attainment are not able to reflect the benefits of requests that exceed the SLO. Goodput is a system level metric that can reflect the number of completed request that meet the SLOs per second, while SLO attainment reflects the proportion of requests that meet the SLOs. However, existing metrics definitions ignore the contribution of requests that are missed. Therefore, the optimal strategy seems to be to give up or reject the requests that have already missed the SLOs, which is not a good choice for users obviously. We argue that the requests that missed the SLO requirements are still valuable, and the benefits of all the requests should be carefully considered.

Figure 1 illustrates how the streaming output

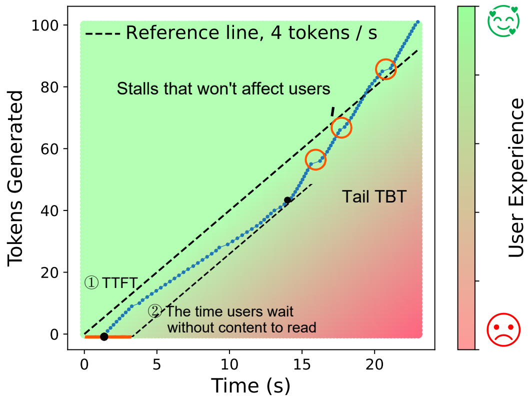


Figure 1: Token generation timeline in LLM serving systems and its impact on user experience. The red area indicates affected user experience. The overall experience is determined by the total waiting time (red line). Some longer TBTs do not degrade perceived experience due to user processing characteristics.

affects user experience. The horizontal red line marked on the time-axis indicates two types of waiting times: 1) the time to receive the first token (TTFT) and 2) the time for subsequent tokens that are generated slower than the user's reading speed (indicated by the reference line). At the beginning, the user experience is poor when the user has to wait for the first token. Subsequently, when users finish reading all tokens delivered, they still suffer from waiting. On the other hand, occasional stalls in the middle of the line will not affect the user experience as long as the user has enough tokens to read. Specifically, the user may not even notice the stalls in the red circles (the user is reading the delivered information) although the TBT is large.

In this paper, we revisit SLOs and system level metrics in LLM serving systems and identify the limitations of existing metrics. To better model the user experience in LLM serving systems, we redesigned the SLO to define reasonable deadlines for each token relative to the commitment of a request, rather than relative to the previous token. Upon the new SLO metric, we introduce the *smooth goodput* to evaluate the performance of the service. The smooth goodput considers the benefits of token generation as well as the punishment of user waiting time without tokens to read.

Based on this unified framework, we re-evaluate the performance of different LLM serving systems under multiple workloads, aiming to help unify the development direction of research on LLM serving focused on user experience optimization.

2 Background and Related Works

In this section, we revisit the background of LLM serving, including the autoregressive inference, mainstream LLM serving systems, and metrics to evaluate their serving quality. Based on these metrics, many scheduling strategies have been proposed.

2.1 Streaming LLM Serving

LLMs process autoregressive inference to generate output tokens based on input prompts. Specifically, a prompt of length k can be represented as a token sequence $(t_1, t_2, ..., t_k)$. The output generated by the LLM is also a token sequence of length n, denoted as $(t_{k+1}, t_{k+2}, ..., t_{k+n})$. The entire process consists of n iterations, where each iteration generates a token. In the current iteration, the prompt and the tokens generated in previous iterations are concatenated as the input. Based on the characteristics of computation and memory access, these iterations can be divided into two phases: *prefill* and *decode*. As shown in Fig. 2, in the prefill phase, the LLM processes the entire prompt within a single iteration and generates the first token A_0. The following decode phases generate the subsequent tokens $(A_1, A_2, ..., A_n)$ one by one, ending with the generation of the EOS token A_E.

2.2 User Experience in LLM Serving

Online LLM serving systems are often designed to provide real-time services to users, which is a rapidly interactive activity like web browsing (Weinreich et al., 2008; Skadberg and Kimmel, 2004). When interacting with LLMs, users expect the system to respond quickly and provide instant feedback. During this continuous stream of information, always lefting enough information to process makes users feel comfortable (Egger et al., 2012).

Exsiting works (Brysbaert, 2019) has studied the speed of reading and processing text. The average reading speed of an adult is about 3-4 words per second. Based on the granularity of tokenization in different languages, we can roughly estimate the token generation speed target.

For offline LLM serving, user experience is not as stringent as in online scenarios. Users generally focus on the end-to-end metrics of batched offline tasks, and typically do not have specific requirements for streaming-specific metrics like TBT and TPOT.

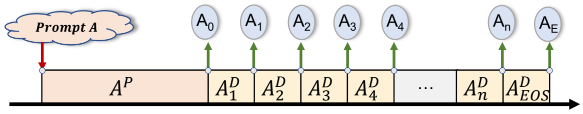


Figure 2: LLM Autoregressive Inference.

2.3 Metrics of LLM Serving

The metrics used to evaluate the performance of LLM serving can be divided into two main groups: SLOs that represent user experience and system level metrics that assess performance under SLO constraints.

As the protocol between the service provider and the user, SLOs have been widely used in LLM serving systems to support better user experience (Patel et al., 2023; Agrawal et al., 2024b; Zhong et al., 2024; Stojkovic et al., 2024; Cheng et al., 2024). As shown in Figure 3, the mainstream SLOs in LLM serving systems are discussed as follows:

- **TPOT (Time-per-Output-Token) and E2E (End-to-End) Latency:** TPOT reflects the average time taking to generate a token (sometimes excluding the first token) while E2E latency reflects the total time taken for a request (or a batch of requests) from commited by users to when it completed. They have no constraints on the time interval between adjacent tokens.

- **TTFT (Time-to-First-Token) and TBT (Time-between-Tokens):** TTFT reflects the time taken for the generation of the first output token while TBT represents the fine-grained time interval between two adjacent tokens of a request. They further delve into each token generation process.

Based on these SLOs, some system level metrics have been proposed to measure the performance of the service:

- **SLO Attainment:** SLO attainment is used to describe the proportion of requests that meet the SLOs. Based on it, capacity is defined as the maximum request rate under the constraint of certain SLO attainment.

- **Goodput:** Goodput is defined as the number of completed requests that meet the SLOs per second in a service. It considers the trade-off between the resource utilization and user experience.

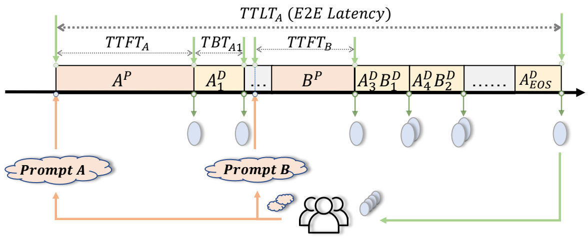


Figure 3: Exsiting SLOs of LLM Serving. Note that this figure ignores the difference between token generation from the LLM and its delivery to users.

2.4 Metric-Driven Optimization

Throughput-oriented optimization. Orca (Yu et al., 2022) introduces the continuous batching, dynamically constructing and processing batches, thereby fully leveraging the parallelism of GPUs. Building upon this, vLLM (Kwon et al., 2023) further incorporates paged attention, which notably enhances compuation throughput, and reduces operational costs. Consequently, it has been widely adopted and established itself as the SOTA framework for inference services.

SLO attainment-oriented optimization. Splitwise (Patel et al., 2023) and TetriInfer (Hu et al., 2024b) proposes splitting prefill and decode phases to separate device due to their different features of computing and memory access. Sarathi-Serve (Agrawal et al., 2024b) introduces chunked prefills and stall-free batching to mitigate the stall of generation. SCOOT (Cheng et al., 2024) propose an automatic paramter tuning system to find the optimal configuration for the system to meet the SLOs. These works improve SLO attainment defined on different metrics, enabling more requests to be served under SLO requirements.

Goodput-oriented optimization. By avoiding the interference between prefill and decode phases, DistServe (Zhong et al., 2024) achieves higher goodput under the same SLO requirements on TTFT and TPOT. That is, more requests that meet the SLOs can be served per second. In fact, there have been goodput-optimal works on DNNs before (Zhang et al., 2023).

In summary, despite the diverse metrics, certain projects such as Splitwise, Distserve, and TetriInfer have identified analogous optimization opportunities. However, the inability to directly compare the effectiveness of these optimizations across different measurement systems poses challenges in making informed optimization choices.

3 Revisiting the SLOs

We revisit the design of SLOs in recent works on LLM serving and demonstrate that existing SLOs are irrational, and propose a new SLO that is more aligned with user experience, focusing on the relationship between the information processing of the user and the delivery of information by the service.

3.1 A Framework of SLOs

To align various SLOs, we introduce a unified framework of SLOs that can be customized to represent the various requirements proposed in different workloads. We view the objective as setting the deadline for the generation time of each token, whereas exsiting SLOs only care about the generation time interval between adjacent tokens.

Definition. We define the deadline of the i-th output token of a request as d_i, while t_i is the actual generation time of the i-th output token. Therefore, the SLO constraints can be formulated as:

$$\forall i, t_i \leq d_i. \tag{1}$$

Customization of existing SLOs. The framework can be customized to represent the various requirements proposed in different works by adjusting the deadline of each token. The details customization of existing SLOs are following:

- **TTFT and TBT.**

$$d_i = \begin{cases} TTFT, & i = 1, \\ t_{i-1} + TBT, & i > 1. \end{cases} \tag{2}$$

  Note that the deadline of the i-th token is determined by the generation time of the previous token, which, as we will show, is not aligned with user experience.

- **End-to-end latency.**

$$d_i = E2E, \tag{3}$$

  where $E2E$ is the time of end-to-end latency. Obviously, if the last token is generated before the end-to-end latency, the request meets the SLO. As aforementioned, the end-to-end latency is a very loose constraint, which is not aligned with user experience all the time.

3.2 Optimization on Existing SLOs

Due to the prefill-prioritizing principle for improving throughput in vLLM, the decode phase of the

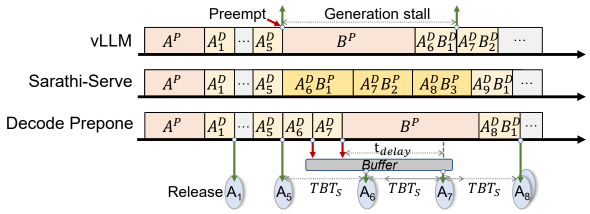


Figure 4: An illustration of iteration scheduling strategies.

following tokens for request A will be stalled until the prefill phase of request B is finished, which results in a generation stall, i.e., a large TBT between A_5^D and A_6^D. Therefore, Sarathi-Serve splits the prefill phase of B into multiple chunks (B_1^P, B_2^P, B_3^P) and fuses them with the decode phases of request A in the same batch. Specifically, one prefill chunk of request B will attach decoding one token of request A, like $A_6^D B_1^P$, $A_7^D B_2^P$ and $A_8^D B_3^P$. Assuming the prefill stage of B is split into n_c chunks, the stall time of A is approximately reduced to about $\frac{1}{n_c}$ of the original. By this way, the stall time is smoothed, resulting in a smaller TBT. However, we observe that the absolute latency from decode tokens of request B (B_1^D, B_2^D....) will not benefit from the optimization. Further, our concern arises that this slicing approach, by introducing frequent assessments of the KV cache, may inadvertently lead to an increase in overall latency rather than a decrease.

To summarize, the chunked-prefills smooths the TBT by slicing the prefill phase and fusing them with the decode phases of other requests. This provides an insight that instead of slicing, can we manually schedule the prefill and decode phases and achieve better performance?

3.3 A Naive Imitation of Sarathi-Serve

We propose a naive imitation strategy, called *decode prepone*, which can achieve a comparable effect to chunked prefills on TBT by simply scheduling without slicing. As shown in Figure 4, specifically, the next n decode tokens for request A (A_6^D and A_7^D) are preponed to be generated before the prefill of request B starts. Meanwhile, instead of directly outputting these tokens of request A, which can result in large TBT between n-th token (A_7^D) to $n+1$-th token (A_8^D), we smoothly output these tokens during the prefill phase of request B.

To achieve smooth output, we take an intuitive approach by assigning a t_{delay} to the output timing of each preponed token. As shown in Figure 4, even though A_6^D and A_7^D have completed their decoding, they are scheduled to be released sequentially after the t_{delay} interval, while ensuring their output time will not exceed the completion time of B's prefill phase. This strategy smooths the overall output flow while maintaining overall latency and mitigating excessive TBT concerns. Besides, it can also be adopted to trade TTFT for TBT/TPOT by delaying the delivery of the first token.

3.4 A New Request-level SLO Defination

Before delving into the details of the new SLO, we first introduce a *output delay* trick that can be used to imporve the SLO attainment on TTFT and TBT to highlight the issue of exsiting metrics.

Output delay trick. Output delay is a tactic where tokens are released until the TBT deadline is reached, rather than immediately upon generation. Implementing output delay can be done by adding an intermediate buffer layer between the inference engine and the client, allowing looser constraints on the delivery of subsequent tokens.

Delaying the delivery of generated tokens to users can improve metrics, which is counterintuitive in fact. Essentially, it is because the premature delivery of tokens inadvertently imposes additional latency constraints on the subsequent tokens. Thus, there is an urgent need to devise a novel SLO that not only protects the user experience but also refrains from penalizing the early delivery of tokens.

Intuition. In fact, users do not frequently notice the lag of the last word during the generation process. We argue that generation stalls are not necessarily harmful to user experience, as long as the delivery of tokens is aligned with the user's reading speed. Given the limitations of TBT in setting the time interval between adjacent tokens, we shift the focus of the SLO to the actual user experience. For instance, we can set the constraint of each request according to the response delay that users can tolerate and the speed of processing output information, such as reading the output of the chatbot, understanding the summary of long text, listening, etc.

Definition: Porting the new SLO to the framework, we have

$$d_i = V \times i, \tag{4}$$

where V is the output information processing speed of the user, and i is the index of the output words. d_i constraints the deadline of the i-th token, after which the user will perceive a pause in the output stream.

4 Revisiting the System level Metrics

Note that SLOs are only concerned with the user experience at request level. However, in the system view, the service provider is more concerned about the overall performance of the service. Specifically, the throughput of the service is a key metric, directly related to the capacity and efficiency of the service. Combining SLOs and throughput, the goodput is a metric that can reflect the throughput of the service that successfully meets the SLOs.

4.1 Existing Strategy

A common practice is the most urgent request-first strategy, based on the intuition that the request nearest to its deadline is the most important and should be processed first.

In addition to this greedy strategy, goodput-based scheduling is also a dominant strategy. Reviewing the definition of goodput as equation 5:

$$\text{Goodput} = \frac{\sum_{r\in R} \mathbb{1}(\forall i, t_i \leq d_i) \cdot n_r}{T}, \quad (5)$$

where R is the set of requests, $\mathbb{1}(\cdot)$ is the indicator function, T is the time interval of serving the requests in R, and n_r is the number of tokens that the request r generates. We observe that if a request does not meet the SLOs, its goodput is assigned a value of 0. This approach, when optimizing for goodput, often leads to abandoning requests that cannot meet the SLOs. In LLM serving, however, this is an unacceptable outcome for users. While latency undoubtedly degrades the user experience, abandoning a request altogether poses an even greater threat.

4.2 Smooth Goodput

Given the shortcomings of the existing goodput metric, a new metric must comprehensively consider the contribution of each request, even if it slightly exceeds the SLO requirements. In such cases, users have to wait for the subsequent token to be generated, after they have finished reading all the previously delivered tokens.

Streaming service and user experience. Unlike models with a single forward inference process, interactive LLM applications are typically deployed as streaming services due to the autoregressive nature of LLMs. Research (Egger et al., 2012) on web based streaming services has shown that the waiting time of users is a key factor affecting user experience.

Therefore, we introduce the concept of user wait time, namely *user idle latency*, to measure the user experience. The user idle latency is cumulative duration during which a user is idle and waiting for new tokens to be generated due to the lower generation speed. Formally, the user idle latency l of a request r is defined as:

$$l_r = \max_{i=1}^{n}(t_i - d_i), \quad (6)$$

where t_i is the time when the i-th token is generated, d_i is the deadline time of the i-th token delivered to the user, and n is the number of output tokens in the request r.

Definition: The smooth goodput is defined as the service benefit per unit of time. The benefit of a request is defined by two factors: the number of tokens that the request generates and the read latency of the request. Formally, we have:

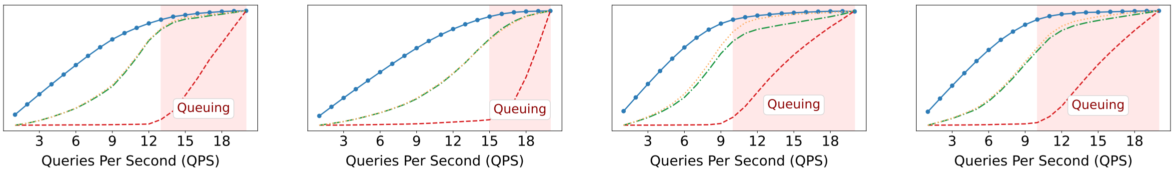
$$\text{benefit}(r) = n_r - \alpha \cdot f(l_r), \quad (7)$$

where n_r is the number of tokens that the request r generates, $f(\cdot)$ is a function that maps the user idle latency to the percentage of the benefit that the request can generate, and α is a weight. For interactive applications with stringent latency requirements, a higher value of α should be chosen to ensure that idle latency is minimized. In practical deployments, the parameters of the benefit function can be calibrated using historical workload data, including request latency metrics and user behaviors (e.g., cancellations and complaints), to better align the service characteristics with the benefit calculation.

The smooth goodput is defined as:

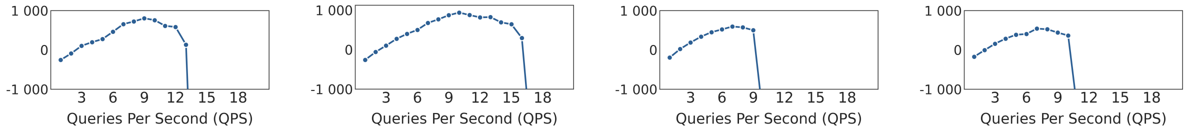
$$\text{smooth goodput} = \frac{\sum_{r\in R} \text{benefit}(r)}{T}, \quad (8)$$

where T is the time interval of serving the requests committed by the users denoted by R. We notice that Andes (Liu et al., 2024a) also considers the benefit of the requests that miss the SLOs. However, they consider the average token slowdown to the deadline in SLOs, while we consider the maximum token slowdown, i.e., the user idle latency. In practice, once the slowdown has occurred, catching up later does not improve the user experience as the user has already experienced the delay. The maximum slowdown represents the furthest deviation from the deadline within the entire request, which corresponds to the total time the user spends waiting for token generation. Therefore, smooth goodput is more reasonable in this context.



(a) LLaMA-3.1-8B. (b) LLaMA-3.1-8B with CP. (c) Qwen2.5-14B. (d) Qwen2.5-14B with cp.

Figure 5: Evaluate with existing metrics.



(a) LLaMA-3.1-8B. (b) LLaMA-3.1-8B with CP. (c) Qwen2.5-14B. (d) Qwen2.5-14B with CP.

Figure 6: Evaluate with smooth goodput.

5 Evaluation

In this section, we re-evaluate different scheduling strategies under the unified metric framework we propose. Then we analyze the results and summarize the challenges of LLM servings. By comparing with the existing metrics, we demonstrate the advantages of smooth goodput.

5.1 Experiment Setup

Settings. We conduct our experiments on a server equipped with an NVIDIA A100-SXM4-80GB GPU, running Debian GNU/Linux 12 and CUDA 12.2. We use LLaMA-3.1-8B-instruct (Grattafiori et al., 2024) and Qwen2-7B (Yang et al., 2024) as base models in the experiments. All of our code development is based on vLLM 0.6.3, and the versions of all required packages are consistent with the requirements of it.

Workloads. For workload, we use ShareGPT as the simulation of the conversations with chatbots, and LooGLE (Li et al., 2024) as the simulation of longer conversations. We set the arrival times of requests to follow the Poisson distribution or processed real-world trace with the average rate set as the parameter to simulate the arrival of requests. We also conduct the real-world trace experiments to evaluate the performance under real-world scenarios.

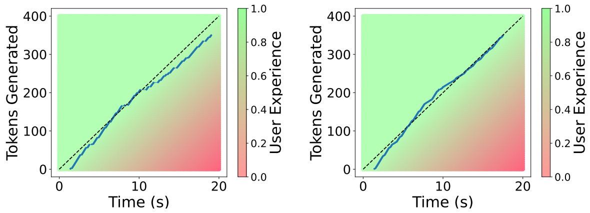
Metrics. We use the smooth goodput to evaluate the performance of LLM serving. We set $\alpha = 5$ in our experiments, with a default information consumption speed of 20 tokens per second. As a comparison, we also use the existing SLOs and system level metrics as introduced in Section 2.

5.2 Analysis with Existing Metrics and Smooth Goodput at the Service Level

We first analyze the performance of different strategies using existing metrics, highlighting the statistical regularities of vLLM under varying request rates and examining the underlying causes. Subsequently, we introduce smooth goodput under the same scheduling strategy to reveal new insights that existing metrics fail to capture.

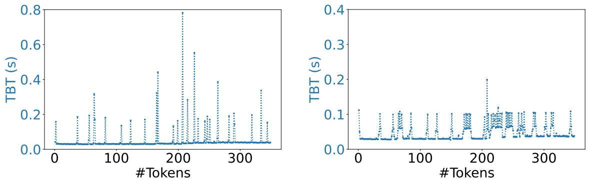
Figure 5 illustrates the performance of vLLM at different request rates using the ShareGPT dataset, which features relatively short prompts and responses. These existing metrics provide a comprehensive view of service performance. In the unsaturated stage, as the request rate increases, resources are utilized more efficiently, leading to increasing throughput. Meanwhile, more requests in the batch results in longer batch processing times and consequently higher TBT and TPOT. Once the system reaches its capacity, further increasing in request rate causes more requests in queue, significantly increasing TTFT. However, no balanced point can be found obviously between throughput and user experience using existing metrics, since the metrics are not designed to consider the trade-off between them.

Next, we evaluate using smooth goodput under the same experiments. We set the information consumption speed to 5 tokens per second and $\alpha = 10$. As shown in Figure 6, in the unsaturated stage, smooth goodput increases with the request rate, as the benefits from increased throughput outweigh the costs. However, as the number of requests continues to rise, the benefits decrease due to high user



(a) With chunked-prefills off. (b) With chunked-prefills on.

Figure 7: Token delivery timeline of vLLM.



(a) With chunked-prefills off. (b) With chunked-prefills on.

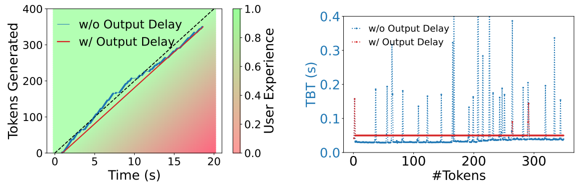
Figure 8: The TBT metrics of vLLM.

idle time, leading to a decrease in smooth goodput. Chunked prefills reaches the peak smooth goodput at a higher request rate than vLLM since it combines prefill and decode phases to fully utilize the GPU's parallelism, accommodating more requests before queuing. This highlights the importance of considering the balance between throughput and user experience in LLM serving systems.

5.3 Analysis with SLOs at Request Level

We conduct experiments to demonstrate that our new SLOs can measure the benefit of each request. We verify this with prompts averaging 1600 tokens in length. From the service logs of the two strategies, we select the same request under the same trace for comparison. Figures 7 and 8 describe the token generation process of the request with and without the chunked prefills technology. The chunked prefills implemented in vLLM significantly reduce the number of generation stalls, providing a smoother token generation process. However, analysis of the data reveals that many token generation stalls caused by prefill preemption go unnoticed by users because some tokens have already been delivered to them. At this point, users are busy processing the information and may not even notice the generation stall, provided that a sufficient amount of tokens has already been delivered.

Output Delaying Trick. We verify the effectiveness of the output delay trick to support our argument on SLOs. As shown in Figure 9a, we implement the output delay trick by buffering tokens and outputting them at a relatively slower rate. This trick is independent of any framework's scheduling strategy and can be implemented on both the server and client sides. Compared to no delay, the output delay trick effectively reduces the tail TBT without affecting the service throughput, as shown in Figure 9b. It delays the delivery of most tokens to the user but achieves better performance in existing metrics. This smooths the TBT to nearly a constant value (the information consumption rate of users) but does not reduce user idle time at all. This indicates that the total time users spend waiting has not improved, and therefore users may still complain about the service. This is also why we believe that existing metrics cannot measure user experience well.



(a) Token delivery timeline. (b) The TBT metric.

Figure 9: Illustration of the output delay trick.

6 Conclusion and Future Work

In this paper, we propose a metric framework to evaluate the performance of LLM serving. We show that existing metrics fail to capture user experience and demonstrate the correlation between user experience and output delivery speed in streaming LLM serving. We introduce smooth goodput to measure service benefit per unit time, considering both service efficiency and user experience. Using this framework, we re-evaluate performance under multiple workloads, demonstrating its capability in analyzing service performance. We hope this framework can provide a unified standard for evaluating LLM serving performance and foster research in LLM serving optimization.

For future work, we observe that the latest slow-thinking models (OpenAI et al., 2024; DeepSeek-AI et al., 2025) undergo a lengthy thought process before delivering tokens to users, which motivates us to explore semantic-aware SLOs, e.g., assigning looser SLOs to requests carrying more information. Additionally, models with different sizes and abilities may produce different output throughput and quality, where considering the optimal balance between throughput and user experience is a promising direction.

Limitations

While we propose a unified metric framework for evaluating LLM serving, designed to reflect the essence of user experience in streaming scenarios such as chatbots and text translation, it is important to note that current services also include offline and non-streaming delivery scenarios. Our metrics can accommodate these workloads but will degrade to resemble existing throughput and E2E latency metrics, as these scenarios do not require consideration of token delivery timelines.

References

Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav Gulavani, Ramachandran Ramjee, and Alexey Tumanov. 2024a. Vidur: A large-scale simulation framework for llm inference. *Preprint*, arXiv:2405.05465.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. 2024b. Taming {Throughput-Latency} tradeoff in {LLM} inference with {Sarathi-Serve}. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 117–134.

Marc Brysbaert. 2019. How many words do we read per minute? a review and meta-analysis of reading rate. *Journal of memory and language*, 109:104047.

Ke Cheng, Zhi Wang, Wen Hu, Tiannuo Yang, Jianguo Li, and Sheng Zhang. 2024. Towards slo-optimized llm serving via automatic inference engine tuning. *Preprint*, arXiv:2408.04323.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Xin Luna Dong, Seungwhan Moon, Yifan Ethan Xu, Kshitiz Malik, and Zhou Yu. 2023. Towards next-generation intelligent assistants leveraging llm techniques. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5792–5793.

Sebastian Egger, Tobias Hossfeld, Raimund Schatz, and Markus Fiedler. 2012. Waiting times in quality of experience for web based services. In *2012 Fourth international workshop on quality of multimedia experience*, pages 86–96. IEEE.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. 2020. Serving {DNNs} like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 443–462.

Cunchen Hu, Heyang Huang, Liangliang Xu, Xusheng Chen, Jiang Xu, Shuang Chen, Hao Feng, Chenxi Wang, Sa Wang, Yungang Bao, Ninghui Sun, and Yizhou Shan. 2024a. Inference without interference: Disaggregate llm inference for mixed downstream workloads. *Preprint*, arXiv:2401.11181.

Cunchen Hu, Heyang Huang, Liangliang Xu, Xusheng Chen, Jiang Xu, Shuang Chen, Hao Feng, Chenxi Wang, Sa Wang, Yungang Bao, et al. 2024b. Inference without interference: Disaggregate llm inference for mixed downstream workloads. *arXiv preprint arXiv:2401.11181*.

Suhas Jayaram Subramanya, Daiyaan Arfeen, Shouxu Lin, Aurick Qiao, Zhihao Jia, and Gregory R Ganger. 2023. Sia: Heterogeneity-aware, goodput-optimized ml-cluster scheduling. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 642–657.

Andreas Kosmas Kakolyris, Dimosthenis Masouros, Sotirios Xydis, and Dimitrios Soudris. 2024. Slo-aware gpu dvfs for energy-efficient llm inference serving. *IEEE Computer Architecture Letters*.

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*, pages 611–626.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2024. Loogle: Can long-context language models understand long contexts? *Preprint*, arXiv:2311.04939.

Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E Gonzalez, et al. 2023. {AlpaServe}: Statistical multiplexing with model parallelism for deep learning serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pages 663–679.

Jiachen Liu, Zhiyu Wu, Jae-Won Chung, Fan Lai, Myungjin Lee, and Mosharaf Chowdhury. 2024a. Andes: Defining and enhancing quality-of-experience in llm-based text streaming services. *arXiv preprint arXiv:2404.16283*.

Xiaoxuan Liu, Cade Daniel, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Xiangxi Mo, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. 2024b. Optimizing speculative decoding for serving large language models using goodput. *arXiv preprint arXiv:2406.14066*.

Sara Montagna, Stefano Ferretti, Lorenz Cuno Klopfenstein, Antonio Florio, and Martino Francesco Pengo. 2023. Data decentralisation of llm-based chatbot systems in chronic disease self-management. In *Proceedings of the 2023 ACM Conference on Information Technology for Social Good*, pages 205–212.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina

Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI. 2024. Chatgpt. `https://chat.openai.com`. Accessed: 2024-08-09.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2023. Splitwise: Efficient generative llm inference using phase splitting. *Power*, 400(700W):1–75.

Archit Patke, Dhemath Reddy, Saurabh Jha, Haoran Qiu, Christian Pinto, Shengkun Cui, Chandra Narayanaswami, Zbigniew Kalbarczyk, and Ravishankar Iyer. 2024. One queue is all you need: Resolving head-of-line blocking in large language model serving. *Preprint*, arXiv:2407.00047.

Yongxia Xia Skadberg and James R Kimmel. 2004. Visitors' flow experience while browsing a web site: its measurement, contributing factors and consequences. *Computers in human behavior*, 20(3):403–422.

Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. 2024. Dynamollm: Designing llm inference clusters for performance and energy efficiency. *arXiv preprint arXiv:2408.00741*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Minh Duc Vu, Han Wang, Zhuang Li, Jieshan Chen, Shengdong Zhao, Zhenchang Xing, and Chunyang Chen. 2024. Gptvoicetasker: Llm-powered virtual assistant for smartphone. *arXiv preprint arXiv:2401.14268*.

Yuxin Wang, Yuhan Chen, Zeyu Li, Xueze Kang, Zhenheng Tang, Xin He, Rui Guo, Xin Wang, Qiang Wang, Amelie Chi Zhou, and Xiaowen Chu. 2024. Burstgpt: A real-world workload dataset to optimize llm serving systems. *Preprint*, arXiv:2401.17644.

Harald Weinreich, Hartmut Obendorf, Eelco Herder, and Matthias Mayer. 2008. Not quite the average: An empirical study of web use. *ACM Transactions on the Web (TWEB)*, 2(1):1–31.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538.

Hong Zhang, Yupeng Tang, Anurag Khandelwal, and Ion Stoica. 2023. {SHEPHERD}: Serving {DNNs} in the wild. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 787–808.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. *arXiv preprint arXiv:2401.09670*.