

Machine Learning-Based Performance Modeling for Efficient Large Language Model Serving

Julianna Sereda

Department of Computer Science, George Mason University, Fairfax, VA, USA.
sereda1973@gmu.edu

Girindra Sehgal

Department of Electrical Engineering and Computer Science, University of Missouri,
Columbia, MO, USA.
girindramail@missouri.edu

Abstract

The rapid proliferation of large language models has created unprecedented demands on computing infrastructure, necessitating sophisticated performance modeling techniques to ensure efficient serving. This paper examines the role of machine learning-based performance modeling in optimizing large language model serving systems. We analyze the architectural trade-offs inherent in serving these models, including latency, throughput, memory management, and energy consumption. The paper discusses how supervised and reinforcement learning approaches can predict inference delays, resource utilization, and quality-of-service metrics, enabling dynamic scheduling and resource allocation. We explore data requirements, feature engineering, and model selection challenges, emphasizing the need for robust, generalizable models that can adapt to heterogeneous hardware and evolving workloads. The discussion extends to deployment considerations, including edge-cloud continuum, sustainability, and governance frameworks. We highlight the importance of fairness, transparency, and policy interventions to mitigate risks of algorithmic bias and environmental impact. Through case illustrations and cross-domain comparisons, we demonstrate that machine learning-based performance modeling is not merely a technical optimization but a socio-technical infrastructure requiring careful governance. The paper concludes with forward-looking perspectives on automated orchestration, federated learning for performance prediction, and the need for standardized benchmarks and auditing mechanisms.

Keywords

machine learning, large language models, performance modeling, inference serving, system architecture, sustainability, governance, fairness.

1. Introduction

The emergence of large language models (LLMs) has transformed natural language processing, enabling applications ranging from conversational agents to code generation and scientific discovery. However, the deployment of these models at scale presents formidable systems challenges. Unlike traditional deep learning models, LLMs are characterized by billions of parameters, autoregressive decoding, and variable-length sequences, leading to complex performance profiles that are difficult to predict and optimize. Efficient serving requires balancing multiple objectives: minimizing latency, maximizing throughput, reducing cost, and ensuring reliability. In this context, performance modeling becomes a critical

enabler for resource management, capacity planning, and adaptive scheduling. Traditional analytical models, while useful, often fail to capture the intricate interactions between hardware, software, and workload dynamics. Machine learning (ML) approaches offer a promising alternative by learning from empirical data to predict performance metrics under diverse conditions. This paper investigates the state of the art in ML-based performance modeling for LLM serving, with a focus on system-level implications. We examine architectural trade-offs, deployment strategies, sustainability concerns, and governance frameworks. Our analysis emphasizes that performance modeling is not a purely technical endeavor but a socio-technical infrastructure that shapes access, equity, and environmental impact. The paper is organized as follows. Section 2 reviews background and related work. Section 3 discusses system architecture and performance challenges. Section 4 presents ML approaches. Section 5 covers data and model selection. Section 6 addresses deployment and sustainability. Section 7 explores robustness, fairness, and governance. Section 8 provides case illustrations and cross-domain comparisons. Section 9 discusses policy implications and future directions. Section 10 concludes.

2. Background and Related Work

Research on LLM serving has progressed rapidly, driven by the need to deploy models like GPT-3, LLaMA, and their variants in production environments. Early systems such as Orca introduced iteration-level scheduling to improve throughput by batching requests at the granularity of decoding steps [1]. However, memory management remained a bottleneck until the advent of PagedAttention, which virtualizes key-value caches to reduce fragmentation and enable higher concurrency [2]. These innovations highlight the tight coupling between memory, computation, and scheduling. Parallel efforts have focused on optimizing transformer inference through techniques like kernel fusion, quantization, and speculative decoding. For instance, methods for efficiently scaling transformer inference have explored model parallelism and pipelining to distribute computation across accelerators [3]. While these systems achieve impressive performance, they often rely on heuristic configurations that may not generalize across workloads. The need for predictive models has thus become apparent. Traditional performance modeling in computer systems relies on queueing theory, analytical models, or simulation. In the context of deep learning serving, systems like Clipper employed online learning for model selection and batching [11]. However, LLMs introduce non-linearities and stateful dependencies that challenge such approaches. Machine learning-based performance modeling has emerged as a response, using regression, neural networks, and reinforcement learning to predict latency, throughput, and resource utilization. For example, FrugalGPT demonstrated that LLM cascades can reduce cost while maintaining quality, but such strategies require accurate predictions of each model's performance [4]. Similarly, FlashAttention optimized attention computation by being IO-aware, yet its performance depends on sequence length and hardware characteristics [5]. Speculative decoding and its variants, such as SpecInfer, accelerate inference by predicting multiple tokens, but their effectiveness hinges on the accuracy of draft models and acceptance rates [6,7]. These works underscore the value of data-driven performance models that can adapt to dynamic conditions. Recent advances in distributed inference have further complicated performance modeling, as communication overhead and synchronization costs become significant [8,9]. Consequently, a holistic approach that integrates ML-based prediction with system-level scheduling is essential. This paper builds on these foundations to explore how ML can be leveraged for efficient LLM serving.

3. System Architecture and Performance Challenges

The architecture of an LLM serving system comprises several interacting components: a request queue, a scheduler, a batching mechanism, a memory manager, and execution engines on heterogeneous hardware. Each component introduces trade-offs that affect end-to-end performance. For instance, larger batch sizes improve throughput by amortizing weight loading and kernel launch overheads, but they increase latency due to longer processing times and potential queueing delays. Conversely, smaller batches reduce latency but underutilize hardware. This tension is exacerbated by the autoregressive nature of LLM decoding, where each token generation depends on previous tokens, creating sequential dependencies. Memory management is another critical challenge. The key-value cache grows linearly with sequence length and batch size, often exceeding GPU memory. Techniques like PagedAttention alleviate this by paging cache blocks, but they introduce address translation overhead and fragmentation risks [2]. Model parallelism, including tensor and pipeline parallelism, distributes the model across devices, but communication costs can dominate, especially for large models [3]. The scheduler must decide how to allocate requests to devices, when to batch, and whether to preempt or migrate tasks. These decisions are interdependent and highly dynamic, making static policies suboptimal. Performance modeling aims to predict the outcomes of such decisions. Traditional analytical models often assume Poisson arrivals and exponential service times, which do not hold for LLM workloads characterized by burstiness and heavy-tailed sequence lengths. ML-based models can learn from historical data to capture these patterns. However, they require features that reflect system state, workload characteristics, and hardware conditions. The challenge lies in building models that are both accurate and computationally lightweight, as the modeling overhead must not outweigh the benefits. Furthermore, the architecture must support online adaptation, as workload shifts and hardware failures occur. This requires a feedback loop where performance predictions inform scheduling, and actual measurements update the models. Such closed-loop systems raise questions about stability, convergence, and robustness, which we discuss later. In summary, the performance challenges of LLM serving are multifaceted, spanning memory, computation, communication, and scheduling. ML-based performance modeling offers a pathway to navigate these trade-offs, but it must be integrated into the system architecture in a principled manner.

4. Machine Learning Approaches for Performance Modeling

Machine learning approaches for performance modeling can be categorized into supervised, unsupervised, and reinforcement learning. Supervised learning is the most common, where models are trained on labeled data consisting of input features and observed performance metrics. Features may include request arrival rate, sequence length distribution, batch size, GPU utilization, memory usage, and hardware specifications. Target metrics include latency, throughput, energy consumption, and quality-of-service violations. Regression algorithms such as linear regression, random forests, gradient boosting, and neural networks have been applied. For instance, recent work has applied machine learning algorithms to predict LLM inference delay, demonstrating that models like random forest and XGBoost can achieve low prediction errors under varying loads [10]. Such predictions enable proactive scheduling, allowing the system to preemptively scale resources or adjust batch sizes. Unsupervised learning can be used for clustering workloads or detecting anomalies. Reinforcement learning (RL) is particularly suited for sequential decision-making in scheduling. An RL agent can learn a policy that maps system states to actions such as batch size selection, request routing,

or resource allocation. The reward function typically combines latency, throughput, and cost. However, RL requires extensive exploration, which can degrade performance during training. Therefore, simulation or offline training with historical data is often used. Another approach is meta-learning, where models learn to adapt quickly to new workloads or hardware. Transfer learning can also help by leveraging models trained on similar tasks. Despite these advances, several challenges remain. First, the space of possible configurations is vast, and data collection is expensive. Second, performance models must generalize across different model architectures, batch sizes, and hardware. Third, the models must be interpretable enough to diagnose anomalies and build trust. Fourth, the computational cost of running sophisticated ML models for every scheduling decision may be prohibitive. Therefore, lightweight surrogate models or hybrid approaches that combine analytical and ML models are often preferred. For example, one can use an analytical model to provide a baseline and an ML model to correct for residuals. This hybrid approach can reduce training data requirements and improve robustness. Overall, ML-based performance modeling is a rapidly evolving field with significant potential for LLM serving.

5. Data, Features, and Model Selection

The effectiveness of ML-based performance modeling hinges on the quality and relevance of data. Data collection for LLM serving involves logging request metadata, system metrics, and hardware counters. Request metadata includes prompt length, generation length, arrival time, and priority. System metrics include queue lengths, batch sizes, memory usage, and cache hit rates. Hardware counters provide insights into GPU utilization, memory bandwidth, and power consumption. However, collecting such data at scale raises privacy and security concerns, as prompts may contain sensitive information. Therefore, anonymization and aggregation are necessary. Feature engineering is critical. Raw metrics must be transformed into meaningful features. For example, sequence length distribution can be summarized by mean, variance, and percentiles. Temporal features such as time of day and request rate trends can capture diurnal patterns. Hardware features include GPU model, memory capacity, and interconnect bandwidth. The choice of features affects model accuracy and interpretability. Model selection depends on the trade-off between accuracy, inference latency, and interpretability. Simple models like linear regression are fast and interpretable but may underfit complex non-linear relationships. Tree-based models like random forests and gradient boosting offer a good balance, handling non-linearities and interactions while remaining relatively fast. Neural networks can capture highly complex patterns but require more data and computational resources. For real-time scheduling, the inference latency of the performance model itself must be minimal, often in the order of microseconds to milliseconds. Therefore, model compression techniques such as pruning, quantization, and knowledge distillation are employed. Additionally, online learning algorithms can update models incrementally as new data arrives, adapting to concept drift. However, online learning introduces challenges related to stability and catastrophic forgetting. Cross-validation and holdout sets are used to evaluate model performance, but they must account for temporal dependencies. Standard metrics include mean absolute error, root mean squared error, and R-squared. For scheduling, the cost of misprediction may be asymmetric; underestimating latency can lead to SLA violations, while overestimating may cause over-provisioning. Therefore, quantile regression or custom loss functions may be more appropriate. Finally, data governance is essential. Data must be collected ethically, with consent and transparency, and stored securely. These considerations link performance modeling to broader policy and governance frameworks. For instance, work on serving DNNs like clockwork emphasizes the

importance of performance predictability from the bottom up, which can inform feature selection [12]. Similarly, INFaaS automates model-less inference serving, highlighting the need for adaptive models that can handle diverse workloads [13]. Techniques such as POS provide practical approaches to estimate inference latency, which can be used as baselines for ML models [14]. Compiler-based optimizations like TVM also benefit from accurate performance models to guide auto-tuning [15].

6. Deployment, Infrastructure, and Sustainability

Deploying LLM serving systems involves choices about infrastructure, including cloud, edge, and hybrid environments. Cloud deployments offer scalability and elasticity but incur monetary and environmental costs. Edge deployments reduce latency and bandwidth usage but are constrained by limited compute and memory. Hybrid approaches can balance these trade-offs, for example, by offloading simple requests to edge devices and complex ones to the cloud. Performance modeling must account for heterogeneous hardware and network conditions. The energy consumption of LLM serving is significant. Training large models already has a substantial carbon footprint, but inference at scale can surpass training in total energy use due to continuous operation. Therefore, sustainability is a critical consideration. ML-based performance modeling can help by predicting energy consumption and enabling energy-aware scheduling. For instance, a scheduler can shift workloads to regions with cleaner energy or defer non-urgent requests to times of low carbon intensity. However, this requires accurate predictions of both performance and energy. Techniques such as dynamic voltage and frequency scaling, power capping, and model quantization can reduce energy, but they affect latency and quality. Performance models can help navigate these trade-offs. Infrastructure also includes networking. Distributed serving across multiple nodes introduces communication overhead, which can be modeled using ML to predict bandwidth and latency. Furthermore, the lifecycle of hardware matters; older GPUs may be less efficient but still usable. Performance models can inform decisions about when to upgrade or retire hardware. From a sustainability perspective, extending hardware lifetime and improving utilization are key. ML-based performance modeling can facilitate these goals by enabling better packing of requests and reducing idle time. However, there is a risk of rebound effects: efficiency improvements may lead to increased usage, offsetting energy savings. Therefore, policy interventions such as carbon pricing or caps may be necessary. The environmental impact of LLM serving has been analyzed in works on sustainable AI, which call for holistic metrics that include carbon footprint [16]. Strubell et al. provide early policy considerations for deep learning in NLP, emphasizing the need for energy-aware practices [17]. Concerns about the risks of large models, including environmental and social costs, are raised by Bender et al. [18] and Bommasani et al. [19]. In summary, deployment and sustainability are intertwined with performance modeling, requiring a holistic approach that considers economic, environmental, and social factors.

7. Robustness, Fairness, and Governance

Robustness is a paramount concern for ML-based performance models. Models trained on historical data may fail when workloads shift, hardware changes, or adversarial inputs are introduced. For example, a sudden surge in requests with unusually long sequences could cause the model to under-predict latency, leading to cascading failures. Therefore, models must be continuously monitored and updated. Uncertainty quantification techniques, such as Bayesian neural networks or ensemble methods, can provide confidence intervals for predictions, allowing the scheduler to adopt conservative strategies when uncertainty is high.

Adversarial robustness is also important; malicious actors might craft inputs to manipulate performance predictions and disrupt service. Fairness is another critical dimension. In LLM serving, fairness concerns arise in resource allocation. If the scheduler prioritizes requests based on predicted performance, it may systematically disadvantage certain user groups or applications. For instance, users with shorter prompts might be favored, while those with longer, more complex requests experience higher latency. This could exacerbate digital divides. Therefore, fairness metrics must be defined and incorporated into scheduling objectives. This may involve enforcing max-min fairness, proportional fairness, or prioritizing requests from underserved communities. Governance frameworks are needed to ensure that performance modeling aligns with societal values. This includes transparency in how models are trained and used, accountability for decisions, and mechanisms for redress. Regulatory bodies may require auditing of scheduling algorithms to detect bias. Standards for data collection and model reporting can facilitate oversight. Furthermore, the concentration of LLM serving in a few large cloud providers raises concerns about market power and access. Policies that promote interoperability, open standards, and public infrastructure could mitigate these risks. International cooperation is also essential, as LLM serving has global environmental and social impacts. Finally, the hidden technical debt in machine learning systems, as discussed by Sculley et al., reminds us that performance models themselves require maintenance and can degrade over time [20]. In conclusion, robustness, fairness, and governance are not afterthoughts but integral to the design and deployment of ML-based performance modeling for LLM serving.

8. Case Illustrations and Cross-Domain Comparisons

To ground the discussion, consider a case where a cloud provider serves a 70-billion-parameter LLM to thousands of concurrent users. The system must handle a mix of short chat queries and long document summarization tasks. Without ML-based performance modeling, the provider might use static batching and simple round-robin scheduling, leading to high tail latency for long requests and low utilization during off-peak hours. By deploying an ML model that predicts latency and throughput based on request characteristics and current load, the provider can dynamically adjust batch sizes, prioritize urgent requests, and route tasks to appropriate hardware. The model might be trained on logs from the past month, using features such as prompt length, generation length, time of day, and GPU utilization. The scheduler uses predictions to minimize a weighted sum of latency and energy. This case illustrates the potential benefits but also the challenges: data collection, model maintenance, and the need to balance multiple objectives. Cross-domain comparisons can offer valuable insights. In database systems, query optimizers use cost models to choose execution plans. These models are often analytical but increasingly incorporate ML to handle complex workloads. Similarly, web serving systems use load balancers that predict server response times. However, LLM serving is unique due to the stateful, autoregressive nature of decoding and the massive memory requirements. Lessons can also be drawn from cloud computing, where auto-scaling algorithms use ML to predict demand. These systems emphasize the importance of feedback loops and the risks of oscillation. Another comparison is with network traffic engineering, where ML predicts traffic matrices to optimize routing. In all these domains, the key is to integrate ML models into control loops carefully, ensuring stability and robustness. For LLM serving, the control loop operates at multiple timescales: fast scheduling decisions (milliseconds) and slower capacity planning (hours to days). ML models must be designed for these timescales, with lightweight models for fast decisions and more complex models for planning. This hierarchical approach can improve both efficiency and reliability.

9. Policy Implications and Future Directions

The integration of ML-based performance modeling into LLM serving has significant policy implications. First, transparency and accountability are essential. Users and regulators should be able to understand how scheduling decisions are made, especially when they affect access to services. This may require explainable AI techniques or at least documentation of model training and validation. Second, privacy must be protected. Performance data can reveal sensitive information about user behavior, so data minimization and anonymization are necessary. Third, environmental policies should encourage sustainable LLM serving. Governments could mandate carbon reporting for data centers, incentivize renewable energy use, and fund research into energy-efficient algorithms. Fourth, competition policy should address the dominance of large cloud providers. Open standards and interoperability could lower barriers to entry, fostering innovation. Fifth, international cooperation is needed to prevent a race to the bottom in environmental and labor standards. Future research directions include the development of federated learning for performance modeling, where models are trained across multiple organizations without sharing raw data. This could improve generalization while preserving privacy. Another direction is the use of digital twins, where a simulated replica of the serving system is used to train and test ML models safely. Automated machine learning (AutoML) could democratize performance modeling, allowing smaller operators to deploy sophisticated models without deep expertise. Additionally, we need standardized benchmarks for LLM serving that include not only latency and throughput but also energy, fairness, and robustness metrics. Such benchmarks would enable fair comparisons and drive progress. Finally, we must consider the human factor. System operators need tools that provide actionable insights, not just predictions. Visualization and what-if analysis can help them understand trade-offs. Training programs should equip engineers with the skills to build and govern ML-based performance models responsibly. In conclusion, the future of LLM serving depends on a synergistic combination of technical innovation and thoughtful policy.

10. Conclusion

This paper has examined machine learning-based performance modeling for efficient large language model serving. We have argued that as LLMs become increasingly integrated into critical infrastructure, the ability to predict and optimize their performance is essential. ML approaches offer a powerful means to navigate the complex trade-offs between latency, throughput, cost, and energy. However, they also introduce new challenges related to data, robustness, fairness, and governance. We have discussed the architectural foundations of LLM serving, the landscape of ML techniques, and the importance of deployment and sustainability considerations. Through case illustrations and cross-domain comparisons, we have shown that performance modeling is a socio-technical system that requires careful design and oversight. We have highlighted the need for policy interventions to ensure transparency, privacy, environmental sustainability, and equitable access. Looking forward, we envision a future where performance models are continuously updated, privacy-preserving, and integrated into adaptive control loops. Standardized benchmarks and auditing mechanisms will be crucial for accountability. Ultimately, the goal is not just to serve LLMs faster or cheaper, but to do so in a manner that is robust, fair, and sustainable. By adopting a holistic perspective, researchers and practitioners can contribute to a responsible and efficient LLM serving ecosystem.

References

1. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., ... & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. In Proceedings of the 29th Symposium on Operating Systems Principles (pp. 611-626). ACM.
2. Yu, G. I., Jeong, J. S., Kim, G. W., Kim, S., & Chun, B. G. (2022). Orca: A distributed serving system for Transformer-based generative models. In 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22) (pp. 521-538). USENIX Association.
3. Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., ... & Dean, J. (2023). Efficiently scaling transformer inference. Proceedings of Machine Learning and Systems, 5, 606-624.
4. Chen, L., Zaharia, M., & Zou, J. (2023). FrugalGPT: How to use large language models while reducing cost and improving performance. arXiv preprint arXiv:2305.05176.
5. Dao, T., Fu, D. Y., Ermon, S., Rudra, A., & Ré, C. (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. Advances in Neural Information Processing Systems, 35, 16344-16359.
6. Leviathan, Y., Kalman, M., & Matias, Y. (2023). Fast inference from transformers via speculative decoding. In International Conference on Machine Learning (pp. 19274-19286). PMLR.
7. Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., ... & Jia, Z. (2023). SpecInfer: Accelerating generative large language model serving with tree-based speculative inference and verification. In Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (pp. 1-16). ACM.
8. Agrawal, A., Kedia, N., Panwar, A., Mohan, J., Kwatra, N., Gulavani, B. S., ... & Ramjee, R. (2024). Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24) (pp. 117-134). USENIX Association.
9. Zhong, Y., Liu, S., Chen, J., Hu, J., Zhu, Y., Liu, X., ... & Li, D. (2024). Distributed inference and serving of large language models: A survey. IEEE Transactions on Parallel and Distributed Systems, 35(8), 1423-1441.
10. Zhang, J., & Gao, H. (2026, March). LLM inference delay prediction based on machine learning algorithms. In 2026 International Conference on Generative Artificial Intelligence and Information Security (GAIIS) (pp. 37-41). IEEE.
11. Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., & Stoica, I. (2017). Clipper: A low-latency online prediction serving system. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17) (pp. 613-627). USENIX Association.
12. Gujarati, A., Karimi, R., Alzayat, S., Hao, W., Kaufmann, A., Vigfusson, Y., & Mace, J. (2020). Serving DNNs like clockwork: Performance predictability from the bottom up. In 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20) (pp. 443-462). USENIX Association.

13. Romero, F., Li, Q., Yadwadkar, N. J., & Kozyrakis, C. (2021). INFaaS: Automated model-less inference serving. In 2021 USENIX Annual Technical Conference (USENIX ATC 21) (pp. 397-411). USENIX Association.
14. Zhang, Y., Wang, Y., & Li, H. (2019). POS: A practical approach to estimate the inference latency of DNNs. arXiv preprint arXiv:1907.01098.
15. Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., Guestrin, C., & Krishnamurthy, A. (2018). TVM: An automated end-to-end optimizing compiler for deep learning. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18) (pp. 578-594). USENIX Association.
16. Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green AI. *Communications of the ACM*, 63(12), 54-63.
17. Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3645-3650). Association for Computational Linguistics.
18. Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (pp. 610-623). ACM.
19. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... & Liang, P. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.
20. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, 28, 2503-2511.