Are We Scaling the Right Thing? A System Perspective on Test-Time Scaling

Youpeng Zhao[1*] **Jinpeng LV**[2] **Di Wu**[1] **Jun Wang**[1]

[1]University of Central Florida [2]Microsoft Research

{youpeng.zhao,di.wu,jun.wang}@ucf.edu
{jinpeng.lv}@microsoft.com

Abstract

Test-time scaling (TTS) has recently emerged as a promising direction to exploit the hidden reasoning capabilities of pre-trained large language models (LLMs). However, existing scaling methods narrowly focus on the compute-optimal Pareto-frontier, ignoring the simple fact that *compute-optimal is not always system-optimal*. In this work, we propose a system-driven perspective on TTS, analyzing how reasoning models scale against practical metrics, such as latency and cost-per-token. By evaluating the impact of popular optimizations such as tensor parallelism and speculative decoding, our preliminary analysis reveals the limitations of current methods and calls for a paradigm shift toward holistic, system-aware evaluations that capture the true essence of scaling laws at inference time.

1 Introduction

The prevalence of transformer-based large language models (LLMs) [41, 32] has brought the prospect of artificial general intelligence (AGI) within closer reach, transforming it from a distant dream into a practical pursuit. In recent years, we have witnessed a remarkable acceleration in the capabilities and accessibility of LLMs, driven by concurrent advances in model architecture [50, 40, 9], training techniques [1, 36], and hardware infrastructure [11, 30]. Powerful open-source models like LLaMA [40, 39], DeepSeek [9], and Qwen [46, 45] have democratized access to state-of-the-art language modeling, enabling a wide range of applications including intelligent assistants [2, 27], code generation [4], and reasoning agents [29, 12]. A key force behind such rapid progress has been the discovery and practice of scaling laws, which empirically show that the performance of LLMs improves predictably as a power-law function of three factors: *model size*, *dataset size*, and *computational budget* [18, 14]. These laws have provided a crucial road map for the industry, guiding the design of ever-larger models by allowing researchers and engineers to anticipate performance gains before training begins [11, 50].

However, as the LLM landscape gradually shifts from offline pre-training to online inference, these traditional scaling laws are revealing their limitations. First, the dominant user cases for LLMs now center on complex reasoning tasks [33, 6, 20] that involve multi-step problem-solving and planning. Albeit larger models excel at memorization and generalization on surface-level benchmarks [24, 21, 16], their performance gains on tasks requiring structured reasoning can be marginal without task-specific fine-tuning [44, 38, 3]. Second, as LLMs quickly transition from research prototypes to production systems, inference cost, instead of training cost, becomes the dominant long-term bottleneck. Serving large models at scale incurs significant computational, memory, and energy overhead, particularly in latency-sensitive applications [19, 53, 35]. The interplay

[*]Part of the work done during an internship at Microsoft Research.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Workshop on Efficient Reasoning.

between model performance and the practical costs of inference represents a critical trade-off that falls outside the purview of traditional scaling laws.

To address these real-world constraints, test-time scaling (TTS) has emerged as a new promising methodology [37, 15, 31]. Unlike pretraining scaling laws, TTS focuses on improving model performance during the inference process. This paradigm encompasses powerful techniques such as supervised fine-tuning (SFT) [25], specialized prompting strategies [42, 47], and reinforcement learning [10, 28], which unlock the latent reasoning abilities of LLM at a fraction of the pre-training cost. Despite its promise, the existing TTS paradigm remains narrowly focused on a Pareto frontier of performance vs. computation (i.e., FLOPs or the number of generated tokens). This ignores a critical reality of production systems: **compute-optimal does not always mean system-optimal**. On the one hand, previous research has shown that accelerating LLM inference demands extensive system-level optimization, such as efficient memory management [19, 53, 35, 52], fast GPU kernels [26, 8, 7, 48], and flexible decoding strategies [23, 5]. A truly system-optimal solution should ideally balance a wider array of practical metrics, including latency, memory, and even energy consumption. Moreover, the definition of compute-optimality fails to capture the hardware heterogeneity. Due to differences in memory hierarchy, interconnect bandwidth, and kernel-level scheduling, the same model may exhibit drastically different scaling behavior on various computing platforms [17, 13].

In this work, we propose to investigate TTS from a system-driven perspective, where the focus is shifted from theoretical computation to practical system measurements. Specifically, we aim to explore the scaling behavior of reasoning models to system metrics, namely, latency and cost-per-token, and discover the lack of system consideration of existing TTS. Furthermore, we evaluate how prevalent optimization strategies, such as tensor parallelism and speculative decoding, would help achieve better system-oriented TTS. Our goal is to conduct a preliminary analysis that provides a more comprehensive understanding of TTS's true performance. Ultimately, we aim to call for a paradigm shift in how the AI and system communities evaluate reasoning strategies, moving beyond accuracy alone to embrace the holistic, system-level costs of intelligence.

2 Related Work

Scaling Laws. Scaling laws first emerged as a powerful paradigm for neural networks in [18], where it shows that the language model performance, as measured by its test loss, improves predictably as a power-law with increases in model size (parameters), dataset size, and training compute. Their initial findings suggest that scaling model size is the most critical factor for achieving better performance. Later, Chinchilla conducts a more extensive analysis and concludes that for a given compute budget, optimal performance is achieved not by training the largest possible model, but by training a comparatively smaller model on significantly more data [14]. These principles have become an essential recipe in LLM pre-training, guiding the development of many subsequent popular state-of-the-art models [50, 11, 22, 46].

Test-time Scaling (TTS). Due to their ubiquitous performance on diverse language tasks, LLMs have become the de facto backend model executors for many popular applications [27, 2]. With the growing demands of real-world LLM-based applications, achieving proper scaling at inference time has become increasingly important. Test-time scaling (TTS) methods aim to improve the LLM performance at relatively low costs on complex reasoning tasks such as mathematical reasoning [42, 25]. Notably, chain-of-thought (CoT) [42] aims to leverage specialized instructions to guide pre-trained LLMs to generate better answers, and S1 [25] curates high-quality datasets and employs supervised fine-tuning (SFT) to achieve better TTS scaling. Nonetheless, existing methods generally consider achieving compute-optimal instead of system-optimal. Kinetics explores TTS scaling from sparse attention, but its system metrics are limited to per-layer latency [34]. Our work aims to reveal TTS scaling from a more comprehensive system angle.

LLM Systems. The proliferation of LLM-based applications has driven the development of numerous system-level designs to enhance inference efficiency. System advancements encompass kernel-level optimizations [8, 7, 48], efficient memory management [19, 53, 35, 52], and flexible scheduling strategies [49, 43, 51]. Notably, the open-sourced vLLM [19] and SGLang [53] have become the default serving engines for production-level system backends. However, the workload characteristics

of long sequence reasoning have not been fully understood in these systems. Our work aims to take the initial step in bridging the gap.

3 Analysis

3.1 Experimental Setup

Models and Tasks. We choose two types of popular reasoning models for preliminary analysis, DeepSeek-R1-Distilled-Qwen [10, 46] and S1.1 [25], with three model size configurations, namely 1.5B, 7B, and 14B. We use the MATH500 dataset [20] and report the zero-shot accuracy results, and set the output length from 1K to 16K.

System Setups. We use a private cluster with 4 NVIDIA GH200-96GB GPUs for evaluations, and build a serving engine using vLLM v.0.7.3 with PyTorch 2.6.0 and CUDA 12.6. All evaluations are conducted using the FlashAttention v.2.7.2 backend. For system-level metrics, we report both the average end-to-end latency per request and the cost per token measured as the number of GPUs used times latency, and divided by the total generated tokens.

Optimization Strategies. To study the effect of existing optimization methods of system-level TTS, we select two popular methods, namely speculative decoding with a simple N-gram predictor model [23] and tensor parallelism [36]. We set the number of speculative tokens to 5 and the minimum and maximum lookup to 1 and 4, respectively.

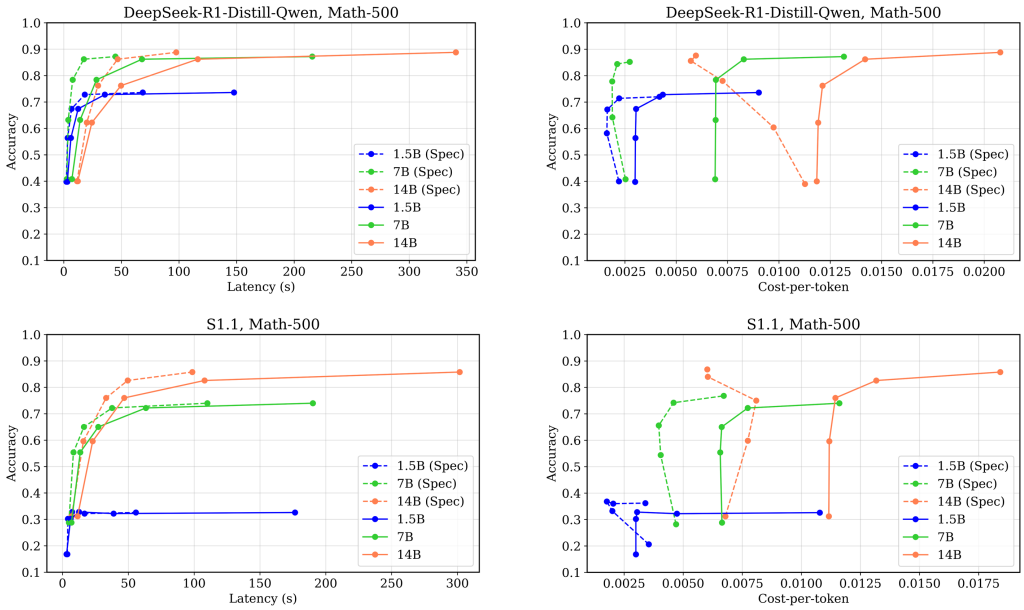

Figure 1: System-level scaling behavior comparison of different models applying speculative decoding on the MATH500 dataset.

3.2 Results

Figure 1 and 2 demonstrate the preliminary results. In terms of latency scaling, we have three key observations. First, the accuracy performance improves with more time spent on the reasoning process, but the gains gradually diminish and flatten out after a certain threshold. This is in contrast to prior compute-oriented scaling, where the accuracy is linear with respect to the number of generated tokens. Second, even with a simple N-gram model, speculative decoding has consistently shown improvement in latency over the naive greedy decoding strategy, revealing its great potential for application to larger models. Third, naive scaling of the inference to more GPUs with tensor parallelism does not yield proportionally better results. For instance, in the case of DeepSeek-R1-Distilled-Qwen-14B, scaling

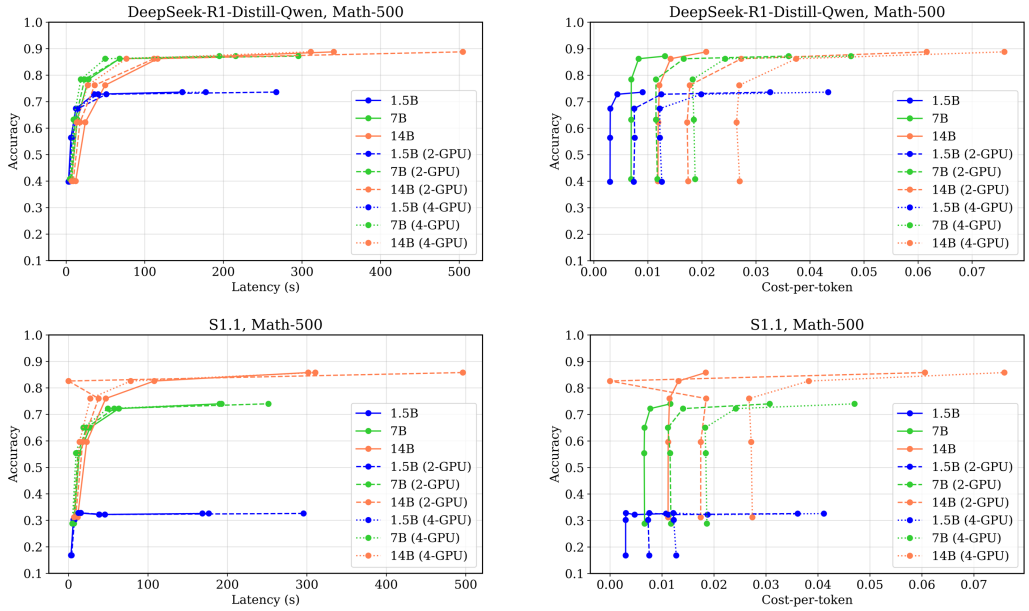

Figure 2: System-level scaling behavior comparison of different models applying tensor parallelism with different numbers of GPUs on the MATH500 dataset.

to 4 GPUs only improves the latency by an average of 1.7×; for 1.5B models, multi-GPU results are even worse than single-GPU. Such a phenomenon can be potentially attributed to the workload nature of reasoning tasks, emphasizing long sequences instead of large batch sizes. Therefore, the inference is bottlenecked by the intra-GPU synchronization.

In regard to cost-per-token analysis, two key insights emerge from the results. First, speculative decoding has shown better performance than tensor parallelism in reducing token costs, due to its latency speedup. However, the scaling behavior is much more unpredictable than baseline methods in the case of larger 14B models. This could be attributed to the randomness of the speculative model, where it cannot guarantee the same number of correct tokens at each step. Second, due to the poor latency performance, tensor parallelism results in higher token costs than baseline methods. This echoes the fact that compute-optimal does not equal system-optimal.

4 Conclusion and Discussion

In this work, we introduce a new system-oriented perspective on investigating test-time scaling (TTS). Our key insight is that compute-optimal is not naturally equivalent to system-optimal, where the latter is often more important in real-world deployment with limited compute, memory, and energy. Our results show the potential of speculative decoding and the limitations of tensor parallelism for system-oriented TTS.

While our work only shows results for limited workload scenarios, the underlying takeaway that reaching the true TTS Pareto frontier demands consideration of holistic system-level costs holds for different models and hardware. By advocating for a shift from traditional accuracy vs. computation to system-aware evaluations, this work aims to provide a new angle to validate the practicality of existing methods in real-world scenarios. This would encourage researchers to innovate on model architectures and inference strategies that are not just more accurate, but also resource-efficient, leading to a new class of LLMs optimized for low latency and cost. Ultimately, this work calls for a more mature and pragmatic approach in developing TTS methods, which focuses on both the algorithm advancements and the corresponding suitable system designs.

5 Acknowledgment

This work was sponsored in part by the U.S. National Science Foundation (NSF) under Grants 2426368 and 2400014. This work used DeltaAI at UIUC NCSA through allocation CIS250367 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. NSF grants 2138259, 2138286, 2138307, 2137603, and 2138296.

References

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

[2] Anthropic. Introducting claude, 2023.

[3] Antonis Antoniades, Xinyi Wang, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. Generalization v.s. memorization: Tracing language models' capabilities back to pretraining data. *ArXiv*, abs/2407.14985, 2024.

[4] Anysphere. Cursor: The ai code editor, 2023.

[5] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In *Neural Information Processing Systems*, 2024.

[6] Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021.

[7] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *ArXiv*, abs/2307.08691, 2023.

[8] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[9] DeepSeek-AI. Deepseek-v3 technical report. *ArXiv*, abs/2412.19437, 2024.

[10] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, abs/2501.12948, 2025.

[11] Nolan Dey, Gurpreet Singh Gosal, Zhiming Chen, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, and Joel Hestness. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. 2023.

[12] Google. Gemini, 2023.

[13] Congjie He, Yeqi Huang, Pei Mu, Ziming Miao, Jilong Xue, Lingxiao Ma, Fan Yang, and Luo Mai. Waferllm: Large language model inference at wafer scale. In *USENIX Symposium on Operating Systems Design and Implementation*, 2025.

[14] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. Training compute-optimal large language models. *ArXiv*, abs/2203.15556, 2022.

[15] Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Jia Xu, Linjian Mo, and Min Zhang. A survey of test-time compute: From intuitive inference to deliberate reasoning. 2025.

[16] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

[17] Norman P. Jouppi, George Kurian, Sheng Li, Peter C. Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Cliff Young, Xiaoping Zhou, Zongwei Zhou, and David A. Patterson. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023.

[18] Jared Kaplan, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020.

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

[20] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *ArXiv*, abs/2305.20050, 2023.

[21] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *ArXiv*, abs/1609.07843, 2016.

[22] Meta. Introducing llama 3.1: Our most capable models to date, 2024.

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

[24] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2018.

[25] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Fei-Fei Li, Hanna Hajishirzi, Luke S. Zettlemoyer, Percy Liang, Emmanuel J. Candes, and Tatsunori Hashimoto. s1: Simple test-time scaling. *ArXiv*, abs/2501.19393, 2025.

[26] NVIDIA. Fastertransformer, 2019.

[27] OpenAI. Introducting chatgpt, 2022.

[28] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[29] OpenAI. Introducing chatgpt agent: bridging research and action, 2023.

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

[31] Yuxiao Qu, Matthew Y. R. Yang, Amrith Rajagopal Setlur, Lewis Tunstall, Edward Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-tuning. *ArXiv*, abs/2503.07572, 2025.

[32] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[33] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *ArXiv*, abs/2311.12022, 2023.

[34] Ranajoy Sadhukhan, Zhuo Chen, Haizhong Zheng, Yang Zhou, Emma Strubell, and Beidi Chen. Kinetics: Rethinking test-time scaling laws. *ArXiv*, abs/2506.05333, 2025.

[35] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark W. Barrett, Joseph Gonzalez, Percy Liang, Christopher Ré, Ioan Cristian Stoica, and Ce Zhang. High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094—-31116. PMLR, 2023.

[36] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *ArXiv*, abs/1909.08053, 2019.

[37] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *ArXiv*, abs/2408.03314, 2024.

[38] Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. *ArXiv*, abs/2205.10770, 2022.

[39] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023.

[40] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023.

[41] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

[43] Bingyang Wu, Yinmin Zhong, Zili Zhang, Gang Huang, Xuanzhe Liu, and Xin Jin. Fast distributed inference serving for large language models. *arXiv preprint arXiv:2305.05920*, 2023.

[44] Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. *ArXiv*, abs/2410.23123, 2024.

[45] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Jingren Zhou, Junyan Lin, Kai Dang, Keqin Bao, Ke-Pei Yang, Le Yu, Li-Chun Deng, Mei Li, Min Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shi-Qiang Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *ArXiv*, abs/2505.09388, 2025.

[46] 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, Ke-Yang Chen, Kexin Yang, Mei Li, Min 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, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. Qwen2 technical report. *ArXiv*, abs/2407.10671, 2024.

[47] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601, 2023.

[48] Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for llm inference serving. *ArXiv*, abs/2501.01005, 2025.

[49] Gyeong-In Yu and Joo Seong Jeong. Orca: A distributed serving system for transformer-based generative models. In *USENIX Symposium on Operating Systems Design and Implementation*, 2022.

[50] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022.

[51] Youpeng Zhao and Jun Wang. Alise: Accelerating large language model serving with speculative scheduling. *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024.

[52] Youpeng Zhao, Di Wu, and Jun Wang. Alisa: Accelerating large language model inference via sparsity-aware kv caching. *ArXiv*, abs/2403.17312, 2024.

[53] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph Gonzalez, Clark W. Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. In *Neural Information Processing Systems*, 2023.