Characterizing Scaling Performance of LLM Inference on GPU-based Supercomputers

Prajwal Singhania*‡, Emir Gencer*‡, Siddharth Singh†, Abhinav Bhatele*
*Department of Computer Science, University of Maryland, College Park, USA †NVIDIA, Inc., Santa Clara, USA
E-mail: prajwal@umd.edu, egencer@umd.edu, bhatele@cs.umd.edu

***Abstract*—Large language models (LLMs) are increasingly important for scientific computing and agentic coding workflows. With growing model sizes, a single node on current-generation HPC systems no longer suffices for LLM inference. Inference must span slower scale-out interconnects and combine different parallelization strategies. This combinatorial space of strategies is large, and each configuration places different demands on the hardware. Which configuration performs best at a given scale, workload, and model architecture remains unclear. To address this, we empirically sweep hundreds of parallel deployment configurations across models and workloads on two leadership-class systems. We analyze scaling trends and find optimal configurations for different workloads and performance metrics, aiming to provide actionable insights for HPC practitioners.**

I. Introduction and Motivation

Large language models (LLMs) have become core components of software development and scientific research workflows in the past few years. As open-weight LLMs become more capable and widely used, deploying them on existing HPC systems is increasingly important. With growing parameter counts, their memory requirements can exceed the HBM capacity of a single node on current-generation HPC systems (typically four to eight GPUs connected by NVLink). Serving these models thus requires parallelizing models and requests across multiple nodes over a heterogeneous network fabric.

The combinatorial space of parallel deployment configurations grows rapidly with GPU count, and each configuration uses compute, memory, and communication resources differently. Consequently, the best configuration shifts with scale, workload, and model architecture. Prior studies of inference are confined to small GPU counts, at most eight GPUs [1]–[3]. Scaling trends over large GPU counts therefore remain poorly understood yet crucial for efficient LLM deployments. We characterize the scaling behavior of parallel LLM inference configurations across GPU counts, model architectures, workloads, and performance metrics on leadership-class systems.

II. Background: Model Parallelism

Inference Phases. LLM inference has two phases: a compute-bound *prefill* phase that processes the prompt in parallel to produce the first output token and a memory-bandwidth-bound *decode* phase that generates subsequent tokens autoregressively.

‡Equal contribution.

Data Parallelism (DP) distributes requests across groups of GPUs. For dense models, DP replicas are independent and require no inter-replica communication.

Tensor Parallelism (TP) shards each layer across GPUs, parallelizing its matrix multiplications and reducing the per-GPU parameter and KV-cache memory footprint. Each layer ends with a costly all-reduce on the critical path.

Pipeline Parallelism (PP) assigns contiguous layer subsets to GPUs and communicates activations point-to-point, only at stage boundaries. Pipelining requests maintains high GPU utilization and hides communication overhead but can underutilize GPUs at low request counts.

Expert Parallelism (EP) shards MoE expert layers across GPUs, assigning each GPU a distinct subset of expert parameters. Because EP shards only expert layers, it is used in conjunction with DP or TP to shard non-expert layers. When combined with DP, replicas route tokens to different experts via all-to-all communication.

III. Experimental Setup

TABLE I
Workload characteristics. Input (ISL) and output (OSL) sequence lengths are given in tokens.

Workload	Concurrency	Mean ISL	Mean OSL
Prefill-heavy	64, 256, 1024	8203 ± 1182	1022 ± 150
Decode-heavy	64, 256, 1024	1025 ± 148	8179 ± 1194

We benchmark vLLM 0.15.2 [4] on two open-weight models: Llama-3.1-70B [5] and GPT-OSS-120B [6]. We evaluate two synthetic 1,000-request traces (Table I): the decode-heavy trace represents reasoning and long rollouts, and the prefill-heavy trace represents summarization and single-turn long-context agentic workflows.

We run on two HPC systems: Perlmutter [7] (four A100-40GB GPUs per node) and Alps [8] (four GH200-96GB GPUs per node). Both use NVLink within nodes and Slingshot-11 between nodes. We conduct strong scaling, keeping the workload and model fixed. We report output throughput, P50 TTFT (time to first token), and P50 TPOT (time per output token), averaging each result over three trials.

IV. Results

We now present the results of our LLM inference performance study.

A. Output Throughput Analysis

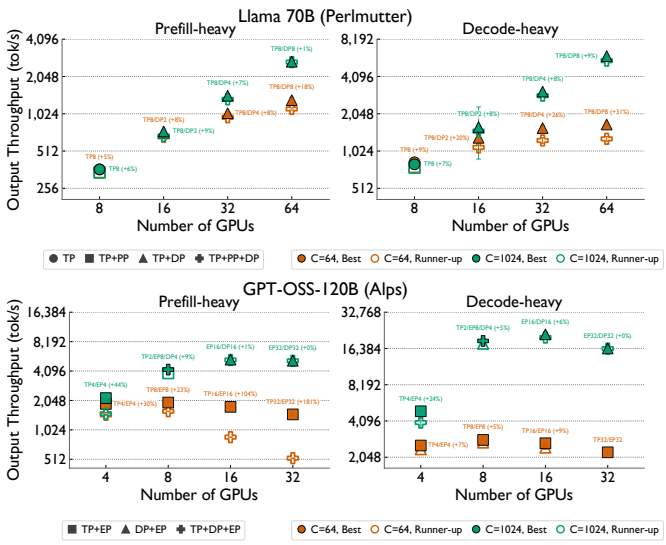


Fig. 1. Output-token throughput for Llama-3.1-70B on Perlmutter and GPT-OSS-120B on Alps at maximum concurrencies of 64 and 1024. At each GPU count, the runner-up is the best configuration using a different combination of parallelism strategies than the winner. Annotations report the winning configuration and its percentage improvement over the runner-up.

Figure 1 compares the output throughput across models and workloads. For Llama-3.1-70B (top row), DP is the preferred strategy for scaling throughput: beyond the eight-GPU minimum required to fit the model, both the best and runner-up configurations add DP replicas. At high concurrency (C=1024), throughput scales nearly linearly (∼7.3× from eight to 64 GPUs), whereas scaling is sublinear at C=64. This is because splitting fewer requests across replicas leads to smaller per-replica batch sizes and underutilized GPU compute. Among model-parallel strategies, TP8-based configurations outperform PP-based configurations at low concurrency, while the gap narrows at high concurrency.

For GPT-OSS-120B (bottom row), TP4/EP4 is optimal within a node at both concurrencies. At low concurrency, TP+EP remains optimal but throughput degrades with GPU count. At high concurrency, adding DP initially improves throughput before degrading it. Unlike dense-model DP, MoE DP is not communication-free: vLLM shards experts across TP×DP ranks, requiring all-to-all token routing that grows more expensive with scale.

B. TTFT Analysis

Figure 2 presents the P50 TTFT for the best configuration on the prefill-heavy workload; the decode-heavy workload exhibits the same trends and is omitted for space. For both models, we observe a capacity threshold: as the GPU count increases, TTFT drops sharply until sufficient serving capacity is reached, after which it plateaus. The threshold shifts to larger GPU counts at higher concurrency. This is because TTFT is affected by queueing delays, which increase with more concurrent requests being served. For the dense model,

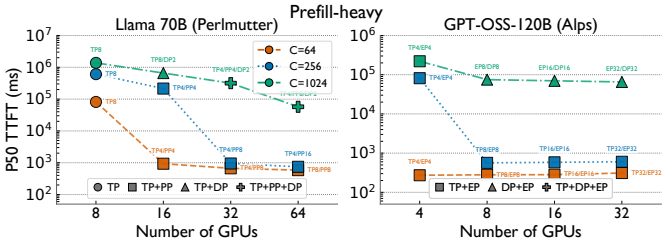


Fig. 2. P50 TTFT for the prefill-heavy workload. Each point is the P50 TTFT-minimizing configuration at that concurrency and GPU count.

TTFT-optimal configurations use high degrees of pipeline parallelism, as pipelining model stages reduces queueing delays.

C. TPOT Analysis

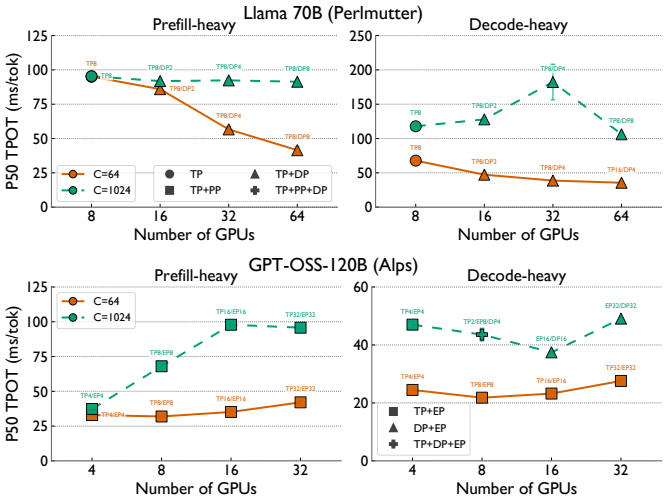


Fig. 3. P50 TPOT across workloads and models. Each point is the P50 TPOT-minimizing configuration at that concurrency and GPU count.

Figure 3 reports P50 TPOT for the best configuration across GPU counts. For Llama-3.1-70B (top row), TP+DP configurations are optimal. At low concurrency, TPOT decreases with scale for both workloads. At high concurrency, TPOT remains relatively stable for the prefill-heavy workload, while it varies non-monotonically for the decode-heavy workload. For GPT-OSS-120B (bottom row), TP+EP configurations are generally optimal, but TPOT does not improve with scale in most cases. For the prefill-heavy workload, TPOT consistently increases with scale. For the decode-heavy workload, TPOT decreases up to a certain GPU count and then increases.

V. Conclusion and Future Work

Optimal parallel configurations depend on model architecture, workload, and performance objective. Dense models favor DP for throughput, TP+DP for TPOT, and PP for TTFT, while MoE models generally favor TP+EP at low concurrency and DP+EP at high concurrency. MoE models also exhibit non-monotonic scaling due to communication overheads introduced by EP. We plan to extend this study to AMD GPU-based systems and further analyze these scaling trends to understand the factors that drive them.

ACKNOWLEDGMENT

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility, operated under Contract No. DE-AC02-05CH11231 using NERSC award DDR-ERCAP0034262 and ALCC-ERCAP0034775. The authors acknowledge the University of Maryland supercomputing resources made available for conducting the research reported in this paper. This work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID lp98 on Alps. OpenAI Codex [9] and Anthropic Claude Code [10] were used for coding and writing assistance, as well as proofreading and editing the abstract and poster text. The authors reviewed and take full responsibility for all content of this submission.

REFERENCES

[1] H. Wang, X. Xiao, M. Yan, Z. Zhu, D. Han, D. Wang, W. Li, X. Ye, C. Hu, H. Chen, and G. Sun, "A systematic characterization of LLM inference on GPUs," 2025. [Online]. Available: https://arxiv.org/abs/2512.01644

[2] M. Arif, A. Maurya, S. Vazhkudai, and B. Nicolae, "Understanding inference scaling for LLMs: Bottlenecks, trade-offs, and performance principles," in *2026 ACM/IEEE 53rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2026, pp. 1070–1083.

[3] L. Xu, K. K. Suresh, Q. Anthony, N. Alnaasan, and D. K. Panda, "Characterizing communication patterns in distributed large language model inference," in *2025 IEEE Symposium on High-Performance Interconnects (HOTI)*. IEEE, 2025, pp. 1–11.

[4] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[5] A. Grattafiori, A. Dubey, A. Jauhri *et al.*, "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[6] OpenAI, "gpt-oss-120b & gpt-oss-20b model card," 2025. [Online]. Available: https://arxiv.org/abs/2508.10925

[7] NERSC, "Perlmutter system architecture," https://docs.nersc.gov/systems/perlmutter/architecture/.

[8] CSCS, "Alps system architecture," https://www.cscs.ch/computers/alps.

[9] OpenAI, "Codex," 2026, accessed 2026-04-07. [Online]. Available: https://openai.com/codex/

[10] Anthropic, "Claude code," 2026, accessed 2026-04-07. [Online]. Available: https://www.anthropic.com/product/claude-code