Characterizing Scaling Performance of LLM Inference on GPU-based Supercomputers

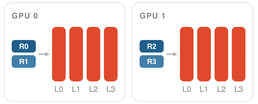
Prajwal Singhania[1]*, Emir Gencer[1]*, Siddharth Singh[2], Abhinav Bhatele[1]
[1]Department of Computer Science, University of Maryland [2]NVIDIA Inc.

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.

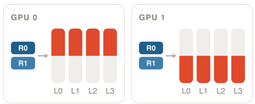
Background

Data Parallelism (DP)



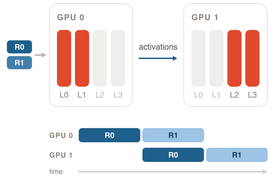
- Full model replica on each GPU but different request batches.
- No communication between replicas because inference is pure forward pass.

Tensor Parallelism (TP)



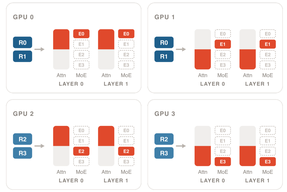
- Each model layer sharded across GPUs with identical batches for each GPU.
- All-reduce communication of activations at the end of every layer.

Pipeline Parallelism (PP)



- Different GPUs hold different layers of the model with micro-batching to pipeline request execution.
- Point-to-point communication of activations between GPUs.

Expert Parallelism (EP)



- For expert layer, each GPU holds a different subset of experts.
- Non-expert layers can be TP-sharded (All-reduce communication) or DP-sharded (All-to-all communication).

* Equal contribution by authors

Experimental Methodology

- We benchmark vLLM 0.15.2 [1] using its benchmarking CLI tool, strong-scaling each model and workload pair across GPU counts.
- All experiments use two HPC machines – Perlmutter (4xA100-40GB per node) and Alps (4xGH200-96GB per node).
- Figures show the best parallel setting per workload & GPU count for the analysed metric. Results are reported as average across 3 trails.
- ***Note***. For MoE models, DP only replicates the non-expert layers. Experts are sharded across all TP×DP ranks, so EP=TP×DP.

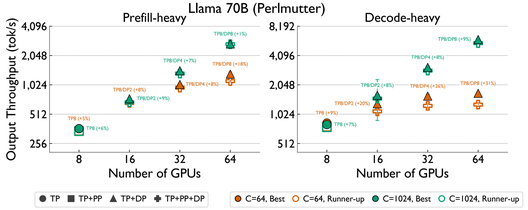
Table 1: Models and parallelism configurations.

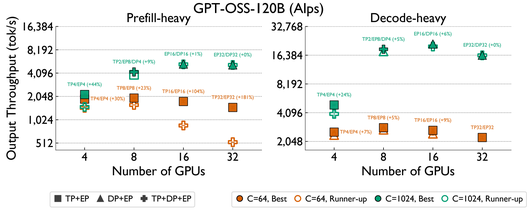
Model	GPUs	Parallelism
Llama-3.1-70B (Dense)	8, 16, 32, 64	TP×PP×DP
GPT-OSS-120B (MoE)	4, 8, 16, 32	EP=TP×DP (PP=1)

Table 2: Workload characteristics over **1000 requests**.

Workload	Max Concurrency	Mean I.S.L.	Mean O.S.L.
Prefill-heavy	64, 256, 1024	8203 ± 1182	1022 ± 150
Decode-heavy	64, 256, 1024	1025 ± 148	8179 ± 1194

Output Throughput Analysis

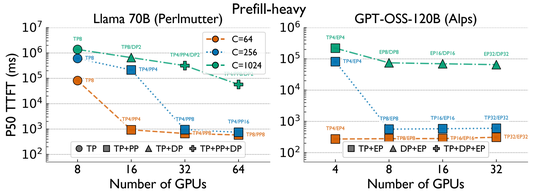




- DP scaling is throughput-optimal with linear scaling at high concurrency but sublinear at low concurrency (smaller batches).
- For decode-heavy workload, sublinear scaling is more prominent.
- Cross-node TP8 outperforms TP4PP2 for both workloads.

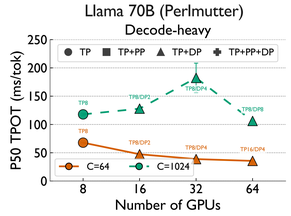
- Within a node (4 GPUs), TP4EP4 is optimal across workloads.
- EP+DP scaling is non-monotonic due to all-to-all communication.
- At low concurrency, TP+EP beats runner-up config significantly for prefill-heavy & marginally for decode-heavy workloads.

P50 TTFT Analysis

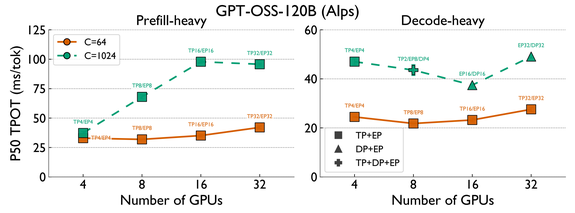


- Figure on the left shows the P50 TTFT optimal configuration.
- Median TTFT is capacity-thresholded: adding GPUs reduces queueing but only upto a certain limit.
- High PP configs offer lower P50 TTFT due to reduced queueing.
- GPT-OSS favors TP+EP at low concurrency, but DP+EP at high concurrency.

P50 TPOT Analysis



- At low concurrency, P50 TPOT decreases with GPU count.
- At high concurrency, there is little or non-monotonic P50 TPOT improvement.
- TP+DP is TPOT optimal.



- For prefill-heavy workloads, TP+EP is optimal but TPOT worsens with scale.
- For decode-heavy workloads, TPOT initially improves, then degrades; TP+EP is preferred at low concurrency while DP-based configs at higher concurrency.

Conclusion and Future Work

Dense Models

- DP maximizes throughput; scaling is sublinear at low concurrency.
- Latency-optimal ≠ throughput-optimal - TP+DP for TPOT, PP for TTFT.

MoE Models

- TP+EP is optimal within a node and at low concurrencies beyond a node.
- DP-based configs are optimal for throughput and TPOT at high concurrencies.
- MoE DP is not communication-free; scaling is non-monotonic.

Future Work. Extend to AMD GPUs and in-depth explanation of the trends observed.

Acknowledgements

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.
This work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID lp98 on Alps. OpenAI
Claude Code and Codex were used as AI coding assistants and as writing assistants for proof-reading and editing text; No section was fully AI generated and all content of this submission and reviewed by the authors.
Claude Design was used for generating the parallelism schematics in the Background section of the poster; fully reviewed by the authors.

References

[1] Kwon, Woosuk, et al. "Efficient memory management for large language model serving with pagedattention." Proceedings of the 29th symposium on operating systems principles. 2023.