Performance Analysis of Fused MoE Inference in vLLM on 4×A100 GPUs

by: Timothy Adeyemi, Temi Adebowale, Abrar Amin, and Stefanie Rivera-Osorio

Abstract

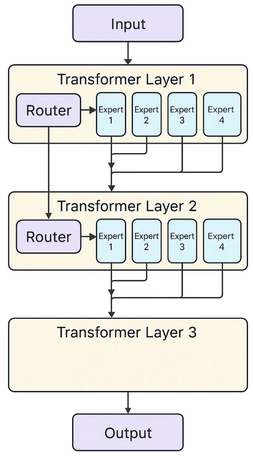
Mixture-of-Experts (MoE) models are models that run only a few expert sub-networks for each token it receives. Although cheap to serve, a major drawback with these types of models are that they split their work unevenly across GPUs. We study how vLLM's fused MoE kernels share that work and spend their time when serving Qwen3-30B-A3B on a 4×A100 node. To do this, we measure expert load on each GPU and each layer, break GPU time down by kernel, and test how configuration and model layout affect throughput and latency. We've found imbalance that looks mild across a whole run but averages 1.39× per layer, with one GPU doing up to 1.75× its fair share in the worst layer.

Background

MoE models like Qwen3-30B-A3B hold 30B parameters but **use only** 3B per token. Each of Qwen's 48 layers has 128 experts, and these experts are spread across the four GPUs (32/GPU). It also has a router per layer (using topkGating) that picks 8 of them for every token. That makes the model computationally cheap to run, but each GPU's workload depends on where the router sends tokens. In our setup, each of the four GPUs hold 32 experts per layer. Every MoE layer is a synchronization point (tensor parallel allReduce), so the busiest GPU becomes the bottleneck for when we receive a response.

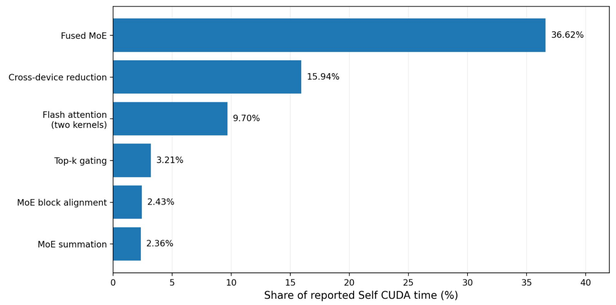
Goal: Identify performance bottlenecks in vLLM inference for an MOE model on four A100 GPUs and evaluate how request scheduling and expert load balancing affect throughput and latency.

Experimental Setup: We ran Qwen3-30B-A3B with vLLM 0.33 on four NVIDIA A100 GPUs 40GB (NVLink) on Perlmutter. Each request used a 512-token random prompt and requested 128 output tokens. We used 4-way tensor parallelism and expert parallelism, with 32 experts per layer on each GPU. vLLM ran the expert computations using Triton's fused_moe kernel.



Outcomes

Figure A. GPU kernel time breakdown



Running the selected experts took **36.6%** of profiled time on one GPU, the largest share of any kernel. **Choosing** those experts took **3.2%**.

Combining results across GPUs took 15.9%, and attention, took 9.7%. The other smaller bars , grouping tokens for each expert (2.4%), and summing the experts' outputs for each token (2.4%).

Figure B. Per-layer GPU routing imbalance

Across the full run, the **busiest GPU** received **8% more** expert **assignments** than the four-GPU average.

Within layer 36, that gap reached 71%.

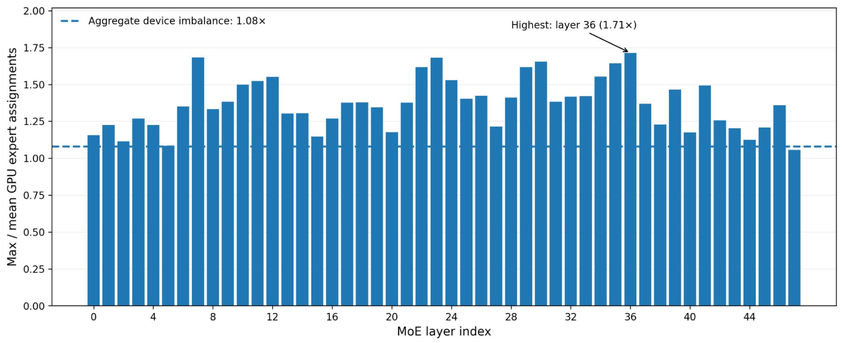


Table 1. Serving performance by request load

Incoming load	Throughput requests/s	Median TTFT ms	p99 ITL ms	Mean TPOT ms
4 requests/s	3.93	78	58	11.1
8 requests/s	7.88	84	63	16.5
Burst	24.76	4,279	137	17.3

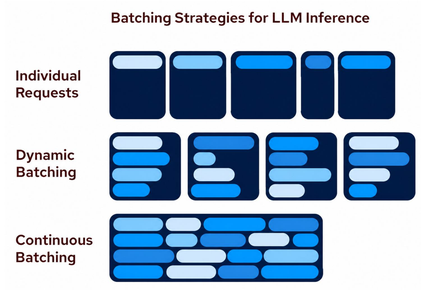
The **burst** delivered the **highest throughput** at 24.76 requests/s, but the **median wait** for the **first token rose** from 78 ms at light load **to 4.28 s** during the burst.

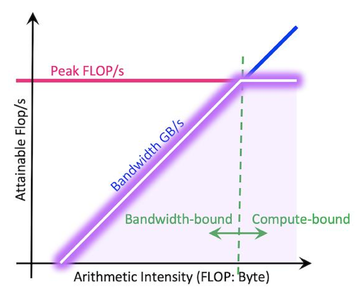
Where We're Going...

November 9-13: Run controlled comparisons. Tim and Temi will lead backend experiment comparisons using the same prompts, inputs, precision, and hardware. We will test out different scheduler/batch limits and request rates, then individually test chunked prefill, CUDA Graphs, and load balancing. Warms ups will be done for each setup, and reports will document latency distribution and throughput as well as record these configurations.

November 16-20: Investigate and explain performance differences. Abrar and Stefanie will lead trace analysis to identify time allocated to attention, routing, GPU communication, and expert computation. This will allow us to relate uneven GPU load imbalance to latency and account for operations running simultaneously. This will contribute directly to our bottleneck analysis to see if computation is limited by calculation speed or memory bandwidth, with communication delays analyzed separately.

November 23-27: Package results. Stefanie will lead plots and report consolidation. Temi will finalize experiment setup and reproducibility instructions. Tim and Abrar will validate all scripts and trace artifacts. All members will then work together to establish conclusions and limitations. Our planned deliverables will include profiling traces, benchmarking and analysis code, information sheets, comparative plots, and a pull request to CCL-bench.





Key Findings

1. **The biggest imbalance appears within individual layers.**
   - In Figure B, the busiest GPU handled 8% more expert assignments than average across the whole run. In the most uneven layer (layer 36), one GPU handled 71% more than average. Different GPUs were busiest in different layers, so the totals looked more even.
2. **Burst traffic increased throughput, but users waited longer for answers to start.**
   - Compared with light traffic, the system completed about 6× as many requests per second (3.9 → 24.8), while the median wait for the first token grew about 55× (78 ms → 4.28 s).