Parallelism Strategies and Concurrency Effects for Mixture-of-Experts Inference on GPU Systems

Ananya Hegde[1], Akshata Kumble[1], and Ravi Gupta[2].*
*Corresponding Author: hegde.ana@northeastern.edu

[1]Northeastern University [2]AI Researcher

***Abstract*—Mixture-of-Experts (MoE) architectures reduce inference cost by activating only a sparse subset of parameters per token. However, when these models exceed single-GPU memory, practitioners must distribute them across devices using tensor parallelism (TP), expert parallelism (EP), or a hybrid of both, and there is little empirical guidance on which strategy to choose. We present a controlled comparison of these strategies for MoE inference serving on NVIDIA A100 GPUs using vLLM. Across multiple models and workload settings, we find that EP achieves 85–102% of TP throughput depending on expert count, matching or exceeding TP for models with 60–64 experts, while naive TP+EP hybrid placement underperforms both alternatives. We also show that a diffusion-based MoE language model is substantially slower than its dense counterpart, highlighting the need for multi-GPU expert distribution in this emerging class. Overall, our results suggest that request concurrency has a larger impact on throughput than parallelism choice in many cases, and that EP is a strong default for large-expert MoE models.**

***Keywords*—mixture-of-experts, inference optimization, tensor parallelism, expert parallelism, vLLM, diffusion language models, concurrency scaling, benchmarking, cost efficiency**

I. Introduction

Mixture-of-Experts (MoE) models scale language models efficiently by activating only a sparse subset of parameters per token [1]. Models like Mixtral 8×7B [2] and DeepSeek-V3 [3] have demonstrated that MoE architectures can match or exceed dense models at a fraction of the per-token compute cost. However, sparsity reduces computation but not memory—all expert weights must reside on accessible devices—and the gating network's dynamic routing decisions create irregular, data-dependent communication patterns. As a result, MoE inference can be up to 15× slower than FLOP-equivalent dense models [4].

When a model exceeds single-GPU capacity, the operator must choose a distribution strategy. Tensor parallelism (TP) shards weight matrices across GPUs with all-reduce synchronization at each layer [5]. Expert parallelism (EP) assigns whole experts to different devices with all-to-all token routing. A hybrid (TP+EP) combines both across a larger GPU pool. Each implies different communication overhead, memory footprint, and sensitivity to workload characteristics. Prior systems work—including DeepSpeed-MoE [6], MegaBlocks [7], and vLLM [8]—has optimized MoE computation for training or improved serving infrastructure, but no study has systematically compared these placement strategies for inference across models with varying expert counts.

This gap matters because expert count fundamentally changes the communication-to-computation ratio. A model with 8 experts routes tokens to 2 large feed-forward blocks; a model with 64 experts [9] routes to 8 much smaller ones. We address this gap with three contributions:

1) A controlled comparison of TP, EP, and TP+EP hybrid under vLLM across three autoregressive MoE models (8–64 experts) on NVIDIA A100 GPUs, sweeping seven concurrency levels and three workload profiles.
2) Evidence via one-way ANOVA that request concurrency is the dominant throughput factor, explaining 51.7% of throughput variance compared to 6.6% for strategy and 3.8% for model choice.
3) Single-GPU characterization of two diffusion language models [10], [11], revealing a 9.8× throughput gap between dense and sparse MoE diffusion architectures.

As a feasibility study, we also train a RandomForest classifier on architecture and workload features to predict the optimal placement strategy, achieving 92.3% accuracy on a stratified 80/20 split—matching the majority-class baseline, suggesting that a broader training set is needed for meaningful prediction.

Section II reviews MoE fundamentals and prior systems work. Section IV describes our hardware, models, and workload configurations. Section V presents throughput, latency, and cost-efficiency results. Section VI analyzes the interaction between expert count and concurrency, discusses diffusion MoE deployment, throughput–latency trade-offs, the throughput predictor feasibility study, and limitations. Section VII summarizes findings and outlines future work.

II. Background and Related Work

A. Mixture-of-Experts Inference

MoE layers replace the dense feed-forward network in a transformer [22] block with N expert FFNs and a learned router [1]. The concept dates back to early work on adaptive expert mixtures [21].

$$\text{MoE}(\mathbf{h}) = \sum_{i \in \text{top-}k} g_i \cdot \text{FFN}_i(\mathbf{h}), \tag{1}$$

where $g_i = \text{softmax}(\mathbf{W}_r \mathbf{h})_i$. This reduces per-token computation while preserving total capacity [12], but inactive experts

still occupy memory and create overhead when placement is poorly matched to hardware.

B. Parallelism Strategies

Tensor Parallelism (TP) shards each weight matrix across P devices with all-reduce synchronization after partial matrix multiplications [5]. For MoE layers, this communication occurs for every selected expert. **Expert Parallelism (EP)** assigns complete experts to individual devices and exchanges routed tokens via all-to-all collectives, as introduced in GShard [19] and formalized in DeepSpeed-MoE [6]. **Hybrid TP+EP** distributes experts across EP groups while applying TP within each group, introducing both all-to-all and all-reduce overhead.

C. Prior Work

MoE systems research has focused primarily on training. DeepSpeed-MoE [6] and Tutel [13] introduced infrastructure for scaling sparse models and reducing routing overhead. MegaBlocks [7] reformulated MoE computation as block-sparse operations to eliminate token dropping. On the serving side, Orca [18] introduced continuous batching for autoregressive inference, and vLLM [8] builds on this with PagedAttention-based memory management. However, none of these systems directly compare TP, EP, and hybrid placement for inference across models with varying expert counts.

D. Diffusion Language Models

LLaDA [10] generates text by iteratively denoising masked tokens over T forward passes. LLaDA-MoE [11] extends this with sparse expert layers, making routing overhead especially costly since it is incurred at every denoising step rather than once per output token.

III. Architecture

Fig. 1 shows the end-to-end serving pipeline used in our study. Requests are tokenized, routed to a sparse top-k expert subset, merged by weighted aggregation, and then scheduled for GPU execution. A runtime monitor feeds latency and throughput signals back to the scheduler so queueing and batching decisions can adapt to workload intensity.

IV. Experimental Setup

This section describes the hardware, models, parallelism strategies, workloads, and metrics used in our evaluation.

A. Hardware

Our experiments span two GPU configurations. Single-GPU baselines and two-GPU multi-GPU runs (TP-only and EP-only) use NVIDIA A100-SXM4-80GB GPUs. Four-GPU hybrid (TP+EP) runs use NVIDIA A100-40GB GPUs. The use of different GPU variants across strategies is a confound we address in Section VI; we partially account for it by reporting per-GPU throughput in addition to absolute throughput. All runs use CUDA 12.3.0 and vLLM 0.8+ [8] in online serving mode.

TABLE I: Model characteristics.

Model	Tot.	Act.	Exp.	k	Type
Mixtral-8x7B	46.7B	12.9B	8	2	AR MoE
Qwen-MoE-A2.7B	14.3B	2.7B	60	4	AR MoE
OLMoE-1B-7B	6.9B	1.3B	64	8	AR MoE
LLaDA-8B	8.0B	8.0B	–	–	Dense
LLaDA-MoE-7B	7.0B	1.4B	64	8	Diff. MoE

TABLE II: Parallelism strategies evaluated. TP = tensor parallel degree, EP = expert parallelism enabled.

Strategy	TP	EP	GPUs	GPU Type
Baseline	1	No	1	A100-80/40GB
TP-only	2	No	2	A100-80GB
EP-only	1	Yes	2	A100-80GB
TP+EP hybrid	2	Yes	4	A100-40GB

B. Models

We evaluate five models spanning three autoregressive MoE architectures and two diffusion language models. Table I summarizes their characteristics.

The three autoregressive models were chosen to span a wide range of expert counts: Mixtral-8x7B-Instruct-v0.1 [2] uses 8 large experts with top-2 routing, Qwen1.5-MoE-A2.7B [14] uses 60 experts with top-4 routing, and OLMoE-1B-7B-0924 [9] uses 64 experts with top-8 routing. This variation allows us to examine how expert count and granularity affect the relative performance of placement strategies. The two LLaDA models [10], [11] serve as a dense–sparse pair for single-GPU diffusion characterization: LLaDA-8B is a dense diffusion language model, while LLaDA-MoE-7B applies a sparse MoE architecture to the same diffusion framework.

C. Parallelism Strategies

We evaluate four configurations, summarized in Table II.

In TP-only, each layer's weight matrices are sharded across two GPUs with all-reduce synchronization. In EP-only, experts are distributed across two GPUs with all-to-all token routing; attention and non-expert layers remain unsharded. The hybrid strategy combines both: attention layers are tensor-parallelized across two GPUs while experts are distributed across the full four-GPU pool. Multi-GPU strategies are evaluated on the three autoregressive models only; LLaDA models are evaluated in the single-GPU baseline configuration.

D. Workloads and Concurrency

For the autoregressive models, we define three workload profiles that vary the ratio of input to output tokens, shown in Table III.

Each workload is swept across seven concurrency levels: 1, 4, 8, 16, 32, 64, and 128 simultaneous requests. This yields $3 \times 7 \times 3 \times 3 = 189$ multi-GPU benchmark configurations (3 models $\times$ 7 concurrency levels $\times$ 3 workloads $\times$ 3 strategies). Inputs are randomly generated (`--dataset-name random`) and sampling is greedy (temperature$= 0$) to ensure deterministic decoding.

For the LLaDA diffusion models, we sweep diffusion steps $\{32, 64, 128\}$ and generation lengths $\{64, 128, 256\}$ at concurrency 1 on a single GPU, yielding 6 configurations per model

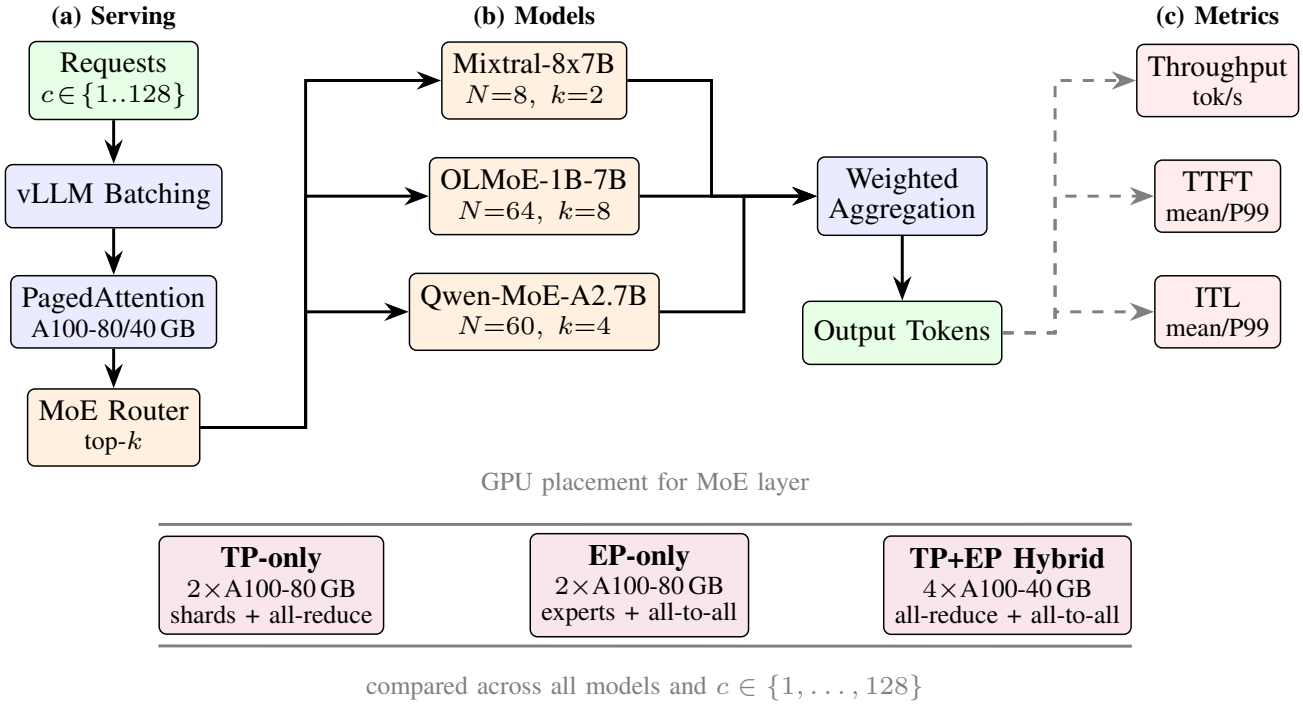

Fig. 1: End-to-end evaluation pipeline. **(a)** Serving stack with vLLM continuous batching and PagedAttention. **(b)** Three MoE models with varying expert counts. **(c)** Metrics collected per run. Bottom: three placement strategies evaluated.

TABLE III: Workload profiles for multi-GPU experiments.

Workload	In	Out	Prompts
Decode-heavy	128	128	100
Prefill-heavy	512	256	50
Balanced	1024	512	20

(12 total). These runs use a custom inference engine built on HuggingFace Transformers rather than vLLM, as vLLM does not natively support diffusion-based generation.

E. Metrics

We report four primary metrics:

- **Throughput** (tokens/s): total output tokens generated per second across all concurrent requests.
- **Time-to-first-token (TTFT)**: latency from request submission to the first generated token, reflecting prefill and scheduling overhead.
- **Inter-token latency (ITL)**: average time between consecutive generated tokens during decoding.
- **Per-GPU throughput** (tokens/s/GPU): throughput normalized by the number of GPUs used, as a measure of cost efficiency.

V. Results

We present results across five dimensions: strategy comparison, concurrency scaling, latency trade-offs, cost efficiency, and diffusion MoE characterization. All multi-GPU results cover three autoregressive MoE models across three workload profiles and seven concurrency levels (189 configurations total). Unless otherwise noted, tables report means across all configurations while figures highlight specific concurrency levels where strategy differences are most pronounced.

TABLE IV: Mean throughput (tok/s) by model and strategy, averaged across all workloads and concurrency levels. EP/TP shows EP throughput as a percentage of TP.

Model	TP	EP	Hybrid	EP/TP
Mixtral-8x7B	672.4	574.0	257.7	85.4%
OLMoE-1B-7B	1016.6	952.7	508.3	93.7%
Qwen-MoE-A2.7B	668.0	684.2	357.5	102.4%

A. Strategy Comparison

Table IV summarizes mean throughput across all workloads and concurrency levels for each model–strategy pair. EP achieves 85.4% of TP throughput for Mixtral (8 experts), 93.7% for OLMoE (64 experts), and exceeds TP by 2.4% for Qwen (60 experts). The trend is consistent: as expert count increases, EP closes the gap with TP and eventually surpasses it. This is expected, since EP distributes whole experts across devices, and models with many small experts benefit more from this distribution than models with few large experts, where TP's weight sharding is more efficient.

Hybrid (TP+EP) placement underperforms both alternatives across all models, achieving only 38–54% of TP throughput. We note that hybrid runs used 4×A100-40GB while TP and EP used 2×A100-80GB; however, even after normalizing for GPU count (Section V-D), hybrid remains the worst-performing strategy.

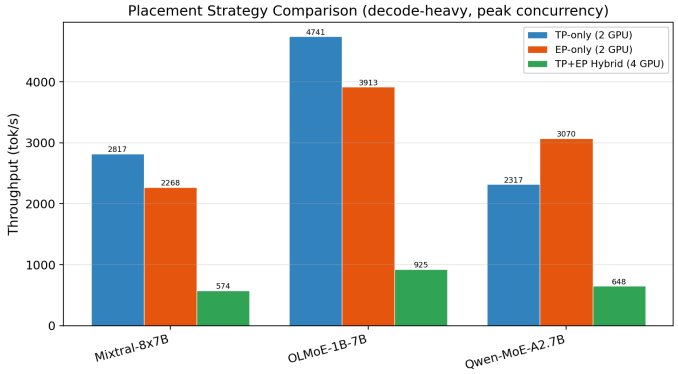

Fig. 2: Throughput by strategy at concurrency 128, decode-heavy workload. EP exceeds TP for Qwen (60 experts) while hybrid underperforms both alternatives across all models.

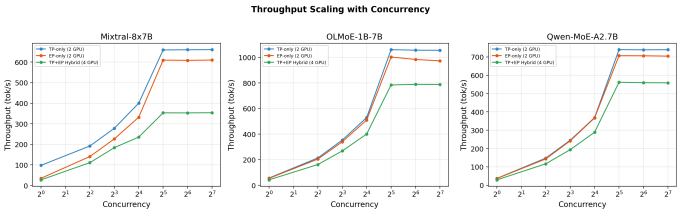

Fig. 3: Throughput vs. request concurrency for each model under the decode-heavy workload. All strategies scale near-linearly up to concurrency 32–64; hybrid saturates earliest.

Table IV shows that EP achieves 85–102% of TP throughput on average across all configurations. However, the aggregate masks an important interaction between expert count and concurrency. At peak concurrency (Fig. 2), Qwen's 60 experts give EP a 32.5% throughput advantage over TP (3070 vs. 2317 tok/s), while Mixtral's 8 experts still favor TP. Hybrid delivers only 20–28% of TP throughput despite using twice as many GPUs.

B. Concurrency Scaling

Fig. 3 shows throughput as a function of concurrency for the decode-heavy workload. All strategies exhibit near-linear scaling from concurrency 1 through 32–64, after which growth rates diverge. TP and EP continue scaling through concurrency 128 for most models, while hybrid saturates earlier—particularly for Qwen, where hybrid throughput plateaus at concurrency 32.

A notable result is Qwen at concurrency 128: EP achieves 3070 tok/s versus TP's 2317 tok/s, a 32.5% advantage. This is the only configuration in our evaluation where EP substantially outperforms TP, and it occurs at the highest concurrency for a model with 60 experts—suggesting that EP's all-to-all routing scales better than TP's all-reduce synchronization under heavy load when expert count is large.

To quantify the relative importance of each experimental factor, we conduct one-way ANOVA tests on throughput. All three factors—concurrency, strategy, and model—are statistically significant at $p < 0.05$:

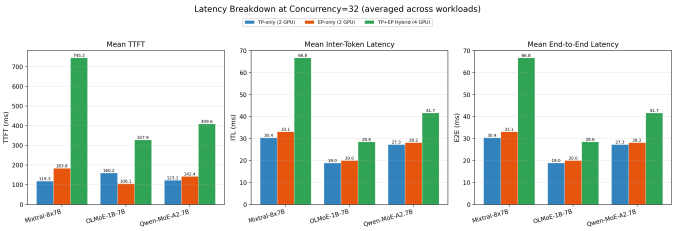

Fig. 4: Mean TTFT, ITL, and end-to-end latency at concurrency 32, averaged across workloads. Hybrid placement incurs 2.2–6.3× higher TTFT than TP.

TABLE V: Mean throughput per GPU (tok/s/GPU) by model and strategy.

Model	TP	EP	Hybrid
Mixtral-8x7B	336.2	287.0	64.4
OLMoE-1B-7B	508.3	476.3	127.1
Qwen-MoE-A2.7B	334.0	342.1	89.4

- Concurrency: $F = 32.48$, $p < 0.001$, explaining 51.7% of throughput variance.
- Strategy: $F = 6.57$, $p = 0.002$, explaining 6.6% of variance.
- Model: $F = 3.70$, $p = 0.027$, explaining 3.8% of variance.

Concurrency dominates, accounting for over seven times more variance than strategy and thirteen times more than model choice. This suggests that practitioners deploying MoE models should prioritize concurrency tuning—through batch size, request queuing, or continuous batching—before investing effort in strategy selection.

C. Latency Trade-offs

Fig. 4 shows mean TTFT, ITL, and end-to-end latency at concurrency 32, averaged across all three workload profiles. Hybrid placement incurs substantially higher TTFT, ranging from 2.2× (OLMoE) to 6.3× (Mixtral) compared to TP. EP adds moderate TTFT overhead for Mixtral (164 vs. 119 ms, 37% higher) but remains comparable to TP for OLMoE and Qwen. ITL and end-to-end latency follow the same pattern: hybrid is consistently worst, while EP and TP are near-identical for high-expert-count models. For latency-sensitive applications, these results argue against hybrid placement. EP is a reasonable alternative to TP: its TTFT penalty is modest for high-expert-count models and its ITL is nearly identical to TP for OLMoE and Qwen.

D. Cost Efficiency

To account for the different GPU counts across strategies, we report throughput normalized per GPU (Table V). TP and EP achieve similar per-GPU efficiency across all models, with Qwen slightly favoring EP (342 vs. 334 tok/s/GPU). Hybrid is dramatically less efficient: it uses twice the GPUs yet delivers 19–25% of TP's per-GPU throughput.

We emphasize that the hybrid comparison involves a hardware confound (A100-40GB vs. A100-80GB), so the per-GPU numbers are not directly comparable. However, the magnitude of the gap—hybrid achieves roughly one-fifth to one-quarter of

TABLE VI: EP throughput as a percentage of TP, by model and workload.

Model	Decode	Prefill	Balanced
Mixtral-8x7B	82.4%	88.2%	86.9%
OLMoE-1B-7B	88.1%	100.9%	94.1%
Qwen-MoE-A2.7B	111.4%	96.5%	96.3%

TABLE VII: Single-GPU throughput (tok/s) for diffusion language models. Ratio is dense/MoE.

Configuration	Dense	MoE	Ratio
Steps = 32	99.1	10.1	9.8×
Steps = 64	49.5	4.9	10.1×
Steps = 128	24.3	2.5	9.7×
Gen len = 64	24.8	2.6	9.5×
Gen len = 128	50.6	5.0	10.1×
Gen len = 256	96.1	9.9	9.7×

TP's per-GPU throughput—suggests that the hybrid penalty is real and not solely attributable to the smaller GPU memory.

E. Workload Sensitivity

We examine whether the workload profile (decode-heavy, prefill-heavy, balanced) affects the relative ranking of strategies. Table VI shows the EP/TP throughput ratio for each model–workload combination.

The ranking is generally stable: TP leads or ties EP for Mixtral across all workloads, while Qwen and OLMoE show near-parity. The strongest workload effect appears for Qwen under decode-heavy conditions, where EP averages 111.4% of TP—driven primarily by Qwen's 32.5% EP advantage at concurrency 128. Prefill-heavy workloads narrow the TP–EP gap for Mixtral (from 82.4% to 88.2%), consistent with the expectation that longer prefill sequences amortize EP's routing overhead. Overall, workload choice has a smaller effect on strategy ranking than concurrency level.

F. Diffusion MoE Characterization

Table VII presents single-GPU throughput for the dense LLaDA-8B and sparse LLaDA-MoE models across diffusion step counts and generation lengths. The dense model is consistently 9.5–10.1× faster than its MoE counterpart, with a mean ratio of 9.8×.

The throughput gap is remarkably stable across configurations, suggesting it arises from the architectural overhead of routing through many small experts at every diffusion step rather than from a specific bottleneck in step count or sequence length. At 2.5–10.1 tok/s, LLaDA-MoE is far below practical serving thresholds, motivating future work on multi-GPU expert distribution for diffusion MoE architectures. Both models were run on a custom inference engine using HuggingFace Transformers, as vLLM does not currently support diffusion-based generation.

VI. Analysis and Discussion

A. Expert Count and Concurrency Interact

Our results show that neither expert count nor concurrency alone determines the optimal strategy—rather, their interaction drives the outcome. For Mixtral (8 experts, top-2), TP outperforms EP across all concurrency levels and workloads, with EP averaging 85.4% of TP throughput. For high-expert models, the picture reverses. OLMoE (64 experts, top-8) shows near-parity at most concurrency levels (EP at 93.7% of TP), and Qwen (60 experts, top-4) averages 102.4% of TP overall—with EP exceeding TP by 32.5% at concurrency 128.

The communication structure of each strategy explains this pattern. Under TP, the all-reduce volume per MoE layer is [5]

$$2 \times \text{hidden\_dim} \times \text{batch\_tokens} \times \frac{P-1}{P}, \tag{2}$$

which is independent of expert count N. Under EP, the all-to-all volume scales with the routing factor [15]:

$$k \times \text{hidden\_dim} \times \text{batch\_tokens} \times \frac{P-1}{P}. \tag{3}$$

For Mixtral ($k=2$), EP communication is modest; for OLMoE ($k = 8$), it is four times larger. Yet OLMoE's experts are individually much smaller, so each device processes its local experts faster, offsetting the communication cost. At high concurrency, larger batch sizes amortize EP's routing overhead, allowing the benefit of distributing 64 experts across devices to dominate. This interaction between expert granularity and batch size echoes findings from GShard [15], which observed that MoE scaling efficiency depends on the ratio of computation to communication per expert. It is also consistent with DeepSeek-MoE [20], which showed that fine-grained expert segmentation improves specialization but increases communication cost under expert parallelism.

These results suggest a practical heuristic: for models with few large experts ($N \leq 8$), prefer TP; for models with many small experts ($N \geq 60$), EP is competitive at moderate concurrency and preferable at high concurrency; hybrid TP+EP should be avoided without careful tuning, as it underperformed in all 63 configurations we tested.

B. Diffusion MoE Deployment

The 9.8× throughput gap between dense and sparse diffusion models reflects a structural mismatch: in autoregressive models, routing cost is paid once per token [1], while in diffusion models it is paid across T denoising steps [10], amplifying overhead by a factor of T. The gap is stable across step counts and generation lengths, confirming it is architectural. Future mitigation strategies include step-adaptive routing and expert caching across steps—analogous to KV-cache reuse [8] and the temporal-locality buffering demonstrated by Huang et al. [4].

C. Throughput–Latency Trade-offs

No single strategy is optimal across all operating points. At concurrency 1, all strategies deliver similar throughput and single-GPU execution minimizes TTFT. As concurrency increases, TP and EP pull ahead, with TP generally providing the best latency profile: hybrid TTFT is 2.2–6.3× worse than TP, while EP's TTFT penalty ranges from negligible (OLMoE, Qwen) to moderate (Mixtral, 37% higher). The importance of concurrency tuning aligns with the continuous batching

paradigm introduced by Orca [18], which showed that iteration-level scheduling can dramatically improve GPU utilization—our ANOVA results quantify this effect for MoE placement specifically.

For latency-sensitive deployments, TP remains the safest default. For throughput-sensitive deployments at high concurrency with many-expert models, EP can deliver both higher throughput and acceptable latency—as demonstrated by Qwen at concurrency 128, where EP achieves 32.5% higher throughput with comparable ITL. This trade-off surface suggests that production serving systems could benefit from concurrency-aware strategy switching, similar to the adaptive parallelism approaches explored by Tutel [13].

D. Throughput Prediction Feasibility

We trained a RandomForestClassifier on architecture and workload features to predict the optimal placement strategy. Using a stratified 80/20 split, the classifier achieved 92.3% accuracy—effectively matching the 92.1% majority-class baseline of always predicting TP. With only three models, the classifier defaults to TP rather than learning meaningful decision boundaries. A broader training set spanning more architectures and GPU configurations would be needed for a useful predictor [17].

E. Limitations

Several limitations constrain our findings. First, hybrid TP+EP inherently requires more GPUs than either strategy alone: with TP=2 and EP=2, the minimum GPU count is TP$\times$EP $= 4$, compared to 2 for TP-only or EP-only. Our hybrid runs used 4$\times$A100-40GB while TP and EP used 2$\times$A100-80GB, introducing a difference in both GPU count and per-device memory. We partially account for this through per-GPU normalization (Table V), but the memory difference may independently affect performance through reduced batch capacity or KV cache size. The hybrid results should be interpreted as a lower bound on what the strategy could achieve on matched hardware. Second, our evaluation covers only three autoregressive MoE models; results may differ for architectures with shared experts [3], [20] or different routing algorithms [12]. Third, all runs used random inputs with greedy decoding, which may not capture routing patterns of production workloads [22]. Fourth, the LLaDA models [10], [11] were evaluated on a single GPU only, so we cannot draw conclusions about multi-GPU placement for diffusion MoE.

VII. Conclusion

We compared TP, EP, and hybrid placement for MoE inference across three models with 8–64 experts on NVIDIA A100 GPUs. EP achieves 85–102% of TP throughput and exceeds TP at high concurrency for many-expert models, while naive hybrid under-performs both in all configurations. Concurrency explains 51.7% of throughput variance—far more than strategy or model choice—suggesting practitioners should tune batching before selecting a placement strategy. We additionally find diffusion MoE models 9.8$\times$ slower than dense counterparts, motivating future multi-GPU work for this emerging class. Future work should extend this comparison to architectures with shared experts [3], [20], validate hybrid results on uniform hardware, and investigate live routing profiling under realistic workloads [22]. We also plan to collect GPU execution traces using profiling tools such as NVIDIA Nsight to measure compute occupancy and memory bandwidth utilization, and use these profiles to train a more robust CPU-based placement predictor that can characterize whether a configuration is memory-bound or compute-bound and recommend a strategy accordingly—repurposing idle CPU cycles during GPU-heavy inference for intelligent scheduling without impacting throughput.

References

[1] N. Shazeer *et al.*, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *Proc. ICLR*, 2017.

[2] A. Q. Jiang *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[3] DeepSeek-AI, "DeepSeek-V3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.

[4] H. Huang *et al.*, "Toward efficient inference for mixture of experts," in *Proc. NeurIPS*, 2024.

[5] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[6] S. Rajbhandari *et al.*, "DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale," in *Proc. ICML*, 2022, pp. 18332–18346.

[7] T. Gale, D. Narayanan, C. Young, and M. Zaharia, "MegaBlocks: Efficient sparse training with mixture-of-experts," in *Proc. MLSys*, 2023.

[8] W. Kwon *et al.*, "Efficient memory management for large language model serving with PagedAttention," in *Proc. SOSP*, 2023.

[9] N. Muennighoff *et al.*, "OLMoE: Open mixture-of-experts language models," *arXiv preprint arXiv:2409.02060*, 2024.

[10] S. Nie *et al.*, "Large language diffusion models," *arXiv preprint arXiv:2502.09992*, 2025.

[11] Y. Zhu *et al.*, "LLaDA-MoE: A sparse MoE diffusion language model," *arXiv preprint arXiv:2509.24389*, 2025.

[12] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, no. 120, pp. 1–40, 2022.

[13] C. Hwang *et al.*, "Tutel: Adaptive mixture-of-experts at scale," in *Proc. MLSys*, 2023.

[14] Qwen Team, "Qwen1.5-MoE: A small but mighty open source mixture-of-experts model," 2024. [Online].

[15] D. Lepikhin *et al.*, "GShard: Scaling giant models with conditional computation and automatic sharding," *arXiv preprint arXiv:2006.16668*, 2020.

[16] T. Dao *et al.*, "FlashAttention-2: Faster attention with better parallelism and work partitioning," in *Proc. ICLR*, 2024.

[17] D. Narayanan *et al.*, "Efficient large-scale language model training on GPU clusters using Megatron-LM," in *Proc. SC*, 2021.

[18] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, "Orca: A distributed serving system for transformer-based generative models," in *Proc. OSDI*, 2022, pp. 521–538.

[19] D. Dai *et al.*, "DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models," in *Proc. ACL*, 2024, pp. 1280–1297.

[20] L. Zheng *et al.*, "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," in *Proc. NeurIPS*, 2023.

[21] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[22] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.