SEM-MOE: SEMANTIC-AWARE MODEL-DATA COLLABORATIVE SCHEDULING FOR EFFICIENT MOE INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Prevailing LLM (Large Language Model) serving engines employ expert parallelism (EP) to implement multi-device inference of massive Mixture-of-Experts (MoE) models. However, the efficiency of expert parallel inference is largely bounded by inter-device communication, as EP embraces expensive all-to-all collectives to route tokens to the remote experts if not collocating on the same GPU/NPU device. Nevertheless, state-of-the-art schemes treat expert device-placement and request (or token) device-scheduling as separate concerns, triggering excessive communication between them and compromising inference efficiency

This paper proposes Sem-MoE, a novel **model-data** collaborative scheduling framework to minimize the steep communication costs in EP-centric MoE serving. Sem-MoE maximally collocates experts and their activating tokens onto the same device using proactively modeled activation likelihood between them and introduces three key techniques: (1) Offline model scheduling, which preliminarily clusters and collocates experts onto devices based on their co-activation tendencies for certain classes of input. (2) Online inter-request data scheduling for Attention-DP setups, which proactively rebatches incoming requests onto the device that hosts experts most likely and frequently activated by the corresponding requests. (3) Online intra-request data scheduling for Attention-TP setups, which seamlessly fuses a token reshuffling procedure into the original inference pipeline and proactively reschedules tokens to devices to reduce dispersed remote routing. We build Sem-MoE into a prevailing LLM serving engine SGLANG. Experiments show our collaborative scheduling approach can effectively reduce the all-to-all communication volume in EP and achieve superior inference throughput compared to existing solutions.

1 INTRODUCTION

The democratization of large language models (LLMs) has been largely driven by continuous model scaling. Over the past five years, the parameter count of the largest trained LLMs has increased by three orders of magnitude, posing significant challenges to the scalability and economic viability of both training and inference under modern AI hardware constraints.

To mitigate these challenges, the Mixture-of-Experts (MoE) architecture Fedus et al. (2022); Artetxe et al. (2022); Jiang et al. (2024) has been introduced. Unlike dense models, MoE models sparsely activate one or more expert sub-networks per input, enabling training of trillion-parameter models without compromising accuracy, while maintaining a sub-linear increase in computational cost. This approach has gained widespread adoption in recent industrial-strength LLMs, including DeepSeek-V3 DeepSeek-AI (2024b)/DeepSeek-R1 DeepSeek-AI (2025), GPT-OSS OpenAI et al. (2025), the Qwen3-Series Yang et al. (2025), and Kimi-K2 Team et al. (2025).

However, at inference time, massive MoE models still require substantial GPU/NPU[1] resources to compute, store, and load both expert and attention parameters. To achieve scalability and meet latency requirements, existing inference frameworks deploy multi-dimensional parallelism strategies that

[1]We use GPU and NPU interchangeably in this paper.

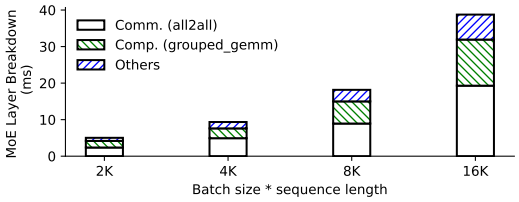


Figure 1: Latency breakdown for DeepSeek-V2-Lite inference over a single MoE layer. Hardware: 8-GPU (96GB) server with fast inter-GPU network (900GB/s).

distribute experts and attention blocks across interconnected devices. An efficient parallelization scheme must effectively partition input tokens and model parameters, maximize resource utilization, and minimize communication overhead.

To address the memory demands of large-scale MoE deployment and leverage aggregate memory bandwidth, modern inference engines such as SGLang Zheng et al. (2024) and vLLM Kwon et al. (2023) employ expert parallelism (EP), whereby experts are distributed across devices. Attention layers are typically parallelized via data parallelism (DP) or tensor parallelism (TP). While EP enables parallel computation of experts across GPUs, it introduces significant communication overhead: intermediate activations must be *dispatched* from the gating module on a source GPU to the destination GPUs hosting the routed experts, and later *combined* back after expert computation. These operations often result in cluster-wide any-to-any token shuffling, typically implemented via two `all2all` collective operations (e.g., NCCL's `all2all`).

Our analysis reveals that the inference performance of MoE models remains severely constrained by these costly `all2all` operations. For instance, a preliminary experiment running SGLang on the DeepSeek-V2-Lite model with 8 GPUs shows that EP communication accounts for up to 59.2% of the forward-pass latency in the MoE layers, respectively—even on high-speed interconnects (see Figure 1). This bottleneck is further exacerbated on slower interconnects such as PCI-e or Ethernet. Therefore, systematically reducing EP communication has become a critical task for improving the efficiency and scalability of MoE inference.

In this paper, we demonstrate that the communication overhead of EP can be substantially reduced through a novel **semantic-aware model–data collaborative scheduling** approach. This method forecasts expert routing paths for both requests and individual tokens, and proactively co-schedules tokens and experts to eliminate redundant communication. We present **Sem-MoE**, a framework that implements this idea via two key techniques:

First, Sem-MoE performs *offline model scheduling* to reduce expert dispersion. Experts that are frequently activated together are clustered and placed on the same device or server based on predicted token-expert affinities. This grouping is performed periodically offline to avoid runtime overhead.

Second, Sem-MoE employs *online data scheduling* to align input tokens with their corresponding expert groups. This includes: (1) *Inter-request scheduling* for DP-based attention: dynamically batching requests to maximize expert affinity and minimize cross-device transfers. (2) *Intra-request scheduling* for TP-based attention: proactively shuffling token activations during the TP communication phase. Specifically, Sem-MoE replaces the standard post-attention `allreduce` with a `shuffled-reduce-scatter` and a deferred `shuffled-allgather`, effectively merging proactive token routing with necessary data transformation.

By integrating collaborative model-data scheduling, Sem-MoE significantly reduces communication volume and improves inference throughput, as demonstrated through extensive experiments implemented on top of SGLang.

We list Sem-MoE's contributions as follows.

1. We conduct a comprehensive data analysis and reveal a significant *context-independent correlation* between tokens and experts in large-scale MoE models, which provides a foundational insight for optimizing expert placement and token routing.

2. We design and implement an efficient *model-data collaborative scheduling algorithm* that leverages the observed token–expert affinity. Our scheduler improves local activation rate by **15.4%** compared to baseline methods, substantially reducing unnecessary cross-device communication.
3. We implement **Sem-MoE** on top of the state-of-the-art inference engine SGLang and perform extensive evaluations. The results demonstrate that Sem-MoE achieves a throughput improvement of up to **2.78x** under specific SLOs in Attention-DP scenarios and up to **24.9%** latency reduction under Attention-TP setups, validating the practical effectiveness of our approach.

2 BACKGROUND

Mixture-of-Experts The Mixture-of-Experts (MoE) architecture is a conditional computation paradigm designed to scale model capacity without a proportional increase in computational cost [1]. Unlike dense models, where all parameters are activated for every input, an MoE model consists of a multitude of expert sub-networks (typically Feed-Forward Networks, FFNs) and a gating network (or router). For each input token, the gating network predicts a sparse combination of experts (e.g., the top-k experts) to which the token is dispatched. Only the selected experts are activated for computation. The most common gating function is the Top-K Gating, which selects the k experts with the highest scores. This design enables models to possess a vast number of parameters (e.g., trillions) while keeping the FLOPs per token roughly constant, as only a small, fixed number of experts (e.g., $k = 2$) are active per token. This has made MoE the de facto standard for building state-of-the-art large language models, such as the DeepSeek series DeepSeek-AI (2024a;b; 2025), the GPT-OSS series OpenAI et al. (2025), and the Qwen series Qwen-Team (2024); Yang et al. (2025).

MoE Training Systems. There has been extensive research on optimizing systems of MoE training systems, including FastMoE He et al. (2021), FasterMoE He et al. (2022), TA-MoE Chen et al. (2022), SmartMoE Zhai et al. (2023), and FlexMoE Nie et al. (2023). However such optimizations can not directly translate to inference scenarios as inference is workload-sensitive and strongly emphasizes latency over throughput.

MoE Inference Systems. Integrated serving engines such as DeepSpeed-MII Holmes et al. (2024), TensorRT-LLM NVIDIA, vLLM Kwon et al. (2023), and SGLang Zheng et al. (2024) have holistic optimization for LLM inference that spans serving schedulers (e.g., continuous batching), dedicated high-performance kernels, efficient parallelization, quantization, and elaborate compiler passes for graph-level optimizations. Built upon these general holistic optimizations for LLM inference, DeepSpeed-MoE Rajbhandari et al. (2022); Singh et al. (2023) and Tutel Hwang et al. (2022) specifically optimize MoE models' computation and communication. Following the design paradigm of DeepSpeed-MoE, popular industry and open-sourced inference engines like vLLM and SGLang also adopted expert parallelism deployment. Sem-MoE specializes in optimizing MoE parallelization (particularly EP) and inherits holistic optimizations from prior work.

MoE Load-balancing and Experts Re-grouping. Lina Li et al. (2023) probes the variation of expert hotness and allots non-uniform expert replicas to achieve load-balanced expert computation. Similar studies Huang et al. (2023) exist to pursue expert load balancing and mitigate other sources of MoE computing inefficiencies. EPS-MoE Qian et al. (2025) optimizes the computation of MoE FeedForward Network (FFN) modules by dynamically selecting the best backend implementation of GroupGemm and DenseGemm. DeepSeek also adopts EPLB (expert-parallelism load balancing) in its real-world deployment DeepSeek-AI (2024b). ExFlow Yao et al. (2024) exploits the affinity between experts across adjacent layers to reduce remote routing and collocate closely related experts. Exflow only considers the model scheduling for MoE models, and requires a heavy `allgather` before the execution of each model layer, which significantly incurs memory pressure and extra communication overhead. MoETuner Go & Mahajan (2025) optimizes the MoE model serving by finding an optimal expert placement strategy to minimize inter-device communication.

MoE Offloading and Prefetching. Existing prediction-based work on MoE inference primarily focuses on prefetching offloaded experts and strategically saving GPU memories Yi et al. (2023); Xue et al. (2024); Zhong et al. (2024), though offloading can extend inference latency and is rarely used in latency-critical serving scenarios. In contrast, Sem-MoE focuses on the speculative reduction of communication overheads and exposes no risks to compromise latency. The work also constructs

probabilistic models to predict the token-expert routing paths, while Sem-MoE's features modelling more comprehensive MoE information, i.e., intra-layer and inter-layer expert affinity and token-expert affinity, compared to prior work. Pre-gated MoE Hwang et al. (2024) modifies the MoE model architecture to predict the experts to route at the next layer. Sem-MoE requires no modification to MoE architecture.

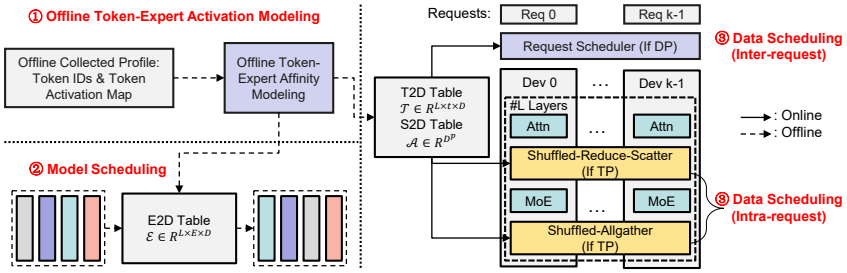


Figure 2: The workflow of Sem-MoE.

3 METHODOLOGY

3.1 SEMANTIC-AWARE MOE (SEM-MOE): OVERVIEW

Figure 2 illustrates the overall workflow of Sem-MoE, where solid and dashed lines represent online and offline operations, respectively. The process begins with Sem-MoE collecting token activation profiles, which include token identifiers and token-expert activation frequencies (Step ① in Figure 2). Based on these profiles, Sem-MoE probabilistically models the token-expert routing likelihood and formulates a balanced token-expert co-clustering problem to generate scheduling hints. These hints are materialized as lightweight lookup tables: a token-to-expert-group table $\mathcal{T}$[2], an expert-group-sequence-to-expert-group table $\mathcal{A}$, and an expert grouping table $\mathcal{E}$.

These scheduling tables drive the subsequent collaborative model-data scheduling. In the model scheduling phase (Step ②), Sem-MoE utilizes the expert-to-device table $\mathcal{E}$ to reconfigure the placement of experts across all layers. In the data scheduling phase (Step ③), different policies are applied depending on the parallelism strategy of the attention layers:

- For attention layers deployed with Data Parallelism (DP), Sem-MoE employs *inter-request* data scheduling. This policy reorders incoming requests according to the token-to-device table $\mathcal{T}$ to maximize request-expert-group affinity, thereby reducing the `all2all` communication overhead across DP domains.
- For attention layers partitioned via Tensor Parallelism (TP), Sem-MoE adopts *intra-request* data scheduling. This technique proactively shuffles tokens during the post-attention `reduce-scatter` operation, directing them to devices predicted to host their target experts in advance, thus minimizing potential token redistribution in subsequent MoE layers.

Figure 3 provides a concrete example of Sem-MoE's operation. In the baseline of Case 1 (top row), requests are distributed across DP ranks for independent attention computation. After expert assignment, tokens are dispatched to their respective expert devices via `all2all` operations. With Sem-MoE's inter-request scheduling, requests are intelligently mapped to DP ranks to enhance data locality, reducing `all2all` volume. This effect is further amplified by complementary model scheduling that optimizes expert placement. In Case 2, which involves TP for attention, tokens require reduction before dispatch and gathering after combination. Sem-MoE's intra-request scheduling predicts expert routes prior to the gating module, allowing tokens to be shuffled and scattered via a customized `shuffled-reduce-scatter (SRS)` operator. Combined with model scheduling, this approach achieves a higher local activation rate, significantly cutting down `all2all` traffic.

The inference acceleration achieved by Sem-MoE stems from the increased local activation rate enabled by collaborative model-data scheduling. Let G denote the number of devices, B the

[2]We use expert group and expert cluster interchangeably.

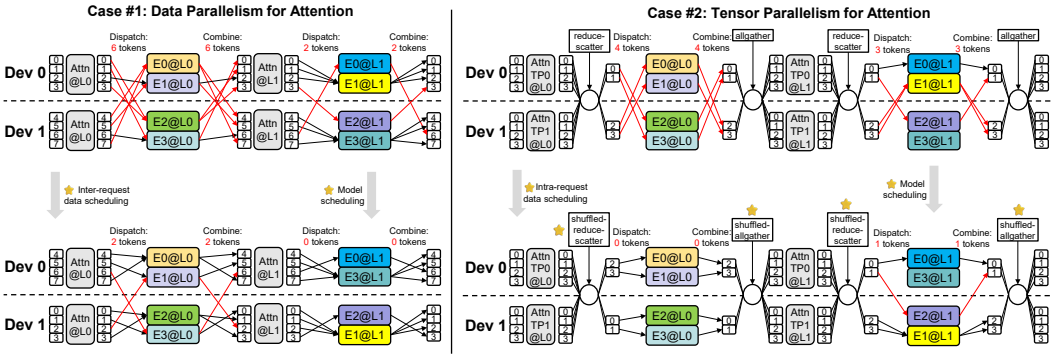


Figure 3: An illustrating example of Sem-MoE. In Case #1 (Attention with DP), Sem-MoE reschedules requests and adjusts expert placement in Layer 1, reducing the number of remotely activated tokens (Remote activated tokens refer to the total number of tokens dispatched to and combined from remote devices) from 16 to 4. In Case #2 (Attention with TP), token rescheduling via `shuffled-reduce-scatter` and expert repositioning in Layer 1 reduce remote token activations from 12 to 2.

global batch size, S the sequence length, and k the number of experts activated per token. The communication volume of an `all2all` operation is given by $\frac{\alpha k B S}{G}$, where α represents the fraction of non-local activations. By maximizing the local activation rate (i.e., minimizing α), Sem-MoE effectively trims communication overhead. The subsequent sections detail the offline modeling and online scheduling algorithms.

3.2 Predicting Expert Routing Path

The routing choice of each MoE layer is given by the gating function: $G_L(h_{L,j}) = \texttt{top-k}(\texttt{softmax}(\mathbf{W}_{L,g} h_{L,j} + \mathbf{b}_{L,g}))$. Accurate prediction of token-expert routing patterns in advance is fundamental to Sem-MoE's scheduling optimization.

Context-Independent Token Activation Prediction. We observe that despite the theoretical dependence of expert routing on contextual semantics (as expressed by the gating function $G_L(h_{L,j})$), in practice, tokens exhibit strong *context-independent* affinities to specific experts. This enables effective prediction based solely on token identity. Through offline profiling on datasets such as *Sharegpt* using models including DeepSeek-V2-lite and Qwen3-30B-A3B (see Appendix Figure 6), we construct a **token-to-expert activation table** $\mathbf{T}^{(L)} \in \mathbb{N}^{t \times N^{(L)}}$ for each MoE layer L, where $\mathbf{T}^{(L)}_{j,k}$ counts how frequently token x_j activates expert $E^{(L)}_k$. The corresponding routing probability is: $\Pr(E^{(L)}_k | x_j) = \mathbf{T}^{(L)}_{j,k} / \sum_{k=1}^{N^{(L)}} \mathbf{T}^{(L)}_{j,k}$. For efficient online inference, these probabilities are tabulated in a **token-to-expert confidence table** $\boldsymbol{\mathcal{C}}_p \in \mathbb{R}^{t \times N}$. Out-of-vocabulary tokens are handled via nearest-neighbor matching in the embedding space.

The token-level predictions form the basis for scheduling in both Attention-DP and Attention-TP scenarios. For **Attention-DP**, the affinity of an entire request to an expert group is derived by aggregating the predictions of its constituent tokens, enabling request-level scheduling. For **Attention-TP**, the fine-grained token-level predictions are directly utilized, and are further refined by modeling inter-layer dependencies, as discussed in Section 3.3. This predictive framework provides the essential guidance for Sem-MoE's collaborative scheduling optimization detailed next.

3.3 Model-Data Collaborative Scheduling

We illustrate how such expert-routing forecasting models can guide the co-dispatching of tokens and experts. Sem-MoE formulates the model-data co-scheduling problem as a 0-1 integer programming (ILP) -based co-clustering problem.

From the offline profiling, we obtain the number of deduplicated (un-deduplicated) tokens t (S), the number of experts per layer N, the number of clusters E (also EP degree), the token j frequency $\boldsymbol{a}_j$,

and the activation probability $\boldsymbol{\mathcal{C}}_{p,jk}$ that token j activates expert k. The decision integer variables are set as the routing $\boldsymbol{R}_{ij} \in \{0,1\}$ of token j to cluster i, and the placement $\boldsymbol{C}_{ij} \in \{0,1\}$ of expert j to cluster i.

We aim to minimize an objective function $\mathcal{L} = \theta \sum_{i=1}^{E} \left| \sum_{j=1}^{t} (\boldsymbol{R}_{ij}\boldsymbol{a}_j) - \frac{S}{E} \right| + (1 - \theta) \sum_{i_1 \neq i_2} \left(\sum_{j=1}^{t} \sum_{k=1}^{N} (\boldsymbol{R}_{i_1 j}\boldsymbol{C}_{i_2 k}\boldsymbol{\mathcal{C}}_{p,jk}\boldsymbol{a}_j) \right)$, where the left part is to ensure that the token frequencies of different clusters as even as possible to promote load balancing among EP ranks, and the right part is to minimize the `all2all` communication overhead caused by remote activation (i.e., the summation of all the activations of tokens and experts belonging to different clusters), a factor $\theta \in (0,1)$ controlling the percentage of two sub-objectives. We further require that each token belongs to only one class, each expert belongs to only one class, and the number of experts in each class is equal by adding hard constraints $\sum_{i=1}^{E} \boldsymbol{R}_{ij} = 1,$ for $j = 1 \ldots t$, $\sum_{i=1}^{E} \boldsymbol{C}_{ij} = 1,$ for $j = 1 \ldots N$, and $\sum_{j=1}^{N} \boldsymbol{C}_{ij} = \frac{N}{E},\ i = 1 \ldots t$.

The above ILP problem is difficult to solve directly using LP solvers, given a large number of intermediate variables introduced in the linearization process. Sem-MoE provides an alternating optimization algorithm. It can quickly obtain a feasible solution while ensuring load balancing. The detailed co-clustering algorithm can be referred to in § B in the Appendix. The solution can then be applied to offline model scheduling and online inter-/intra-request data scheduling.

Model scheduling. Before deployment, Sem-MoE adjusts the expert placement layout according to the solved $\boldsymbol{C}$, placing expert j to device k if $\boldsymbol{C}_{jk} = 1$. Accordingly, Sem-MoE shuffles the column of the gate matrix, thereby realizing a transparent expert re-distribution.

Data scheduling: Attention-DP Scenarios. In Attention-DP setups, where requests are processed independently across DP ranks, scheduling operates at the *request granularity*. We use the variable $\boldsymbol{S}_{\boldsymbol{r}} \in [\![E]\!]$ to denote the cluster to which request $\boldsymbol{r}$ needs to be scheduled. Once the scheduling of tokens is determined, the scheduling of the request $\boldsymbol{r}$ can be determined by aggregating the routing result of its tokens, $\boldsymbol{S}_{\boldsymbol{r}} = \arg\max_{j \in [\![E]\!]} \sum_{i \in \boldsymbol{r}} \boldsymbol{R}_{ij}$. Meanwhile, to achieve runtime load balance, Sem-MoE realizes a workload-aware balanced request scheduling algorithm. For continual E requests, Sem-MoE guarantees these requests are distributed to all E ranks, such that the loads of all ranks would not skew in decoding stage. Detailed algorithm could be referred to in Algorithm 2 in § B. This request-level scheduling minimizes cross-device communication by collocating entire requests with their most likely expert group (i.e., DP rank).

Data Scheduling: Attention-TP Scenarios. In Attention-TP setups, the attention computation itself is distributed, requiring fine-grained, *token-level* scheduling. Here, we enhance the basic token-expert prediction with **inter-layer expert-expert affinity**. We observe that expert selections exhibit Markovian dependencies across layers: the experts chosen at layer L depend on selections at previous layers. We model this using an n-gram device transition model: $\Pr(D_k^{(L)} | D^{(L-1)}, \ldots, D^{(L-n)})$ where $D^{(l)} \in \{1, \ldots, Q\}$, where $D^{(l)} \in 1, \ldots, Q$ denotes the device index of the expert selected at layer l. These transitions are stored in an **expert-group-sequence-to-expert-group confidence table** $\mathcal{A}_p$ (we use 2-gram in practice). Together with the token routing matrix $\boldsymbol{R}$, we can achieve more accurate proactive scheduling during the TP communication phase. The detailed algorithm can be found in Algorithm 3.

3.4 Implementation and System Optimization

Sem-MoE is implemented as a plug-in module for the SOTA LLM inference engine SGLang. Our system comprises approximately 5,000 lines of Python code, along with several custom Triton OpenAI kernels for high-performance communication operations.

To support affinity-aware scheduling in the Attention-DP scenario, we extend SGLang's request scheduler to incorporate token–expert affinity information derived from our prediction models. This enables the runtime to batch requests with similar expert activation patterns onto the same device, minimizing cross-device communication.

For the Attention-TP scenario, we implement two fused communication primitives: `shuffled-reduce-scatter (SRS)`, and `shuffled-allgather (SAG)`. These kernels integrate speculative token shuffling—based on predicted expert routes—into standard

`reduce-scatter` and `allgather` collectives. The shuffling logic relies on an optimized `argsort` kernel, which outperforms the native PyTorch implementation by **25%**. The overall overhead of embedding shuffling into the ring-based communication schedule is negligible, measured at approximately **1%**. Furthermore, for efficient `all2all`, Sem-MoE integrates frontier MoE communication libraries deepep Zhao et al. (2025).

By combining offline expert reorganization with online token- and request-level scheduling, Sem-MoE achieves significant reductions in `all2all` communication volume, leading to improved end-to-end inference throughput in both DP and TP configurations.

4 EXPERIMENT

4.1 EXPERIMENTAL ENVIRONMENTS

We evaluate Sem-MoE on an 8-GPU server, representing commercial GPU servers unified for both training and inference, which are configured with 96GB-HBM per GPU and fast homogeneous interconnects. GPUs inside a server can communicate with each other at a premium bandwidth (900GBps). The server is equiped with two 44-core Intel CPUs and 2TB DDR5 memory.

4.2 MODELS, DATASETS AND WORKLOAD TRACES

Models. We choose two types of typical MoE models for evaluation, i.e., Qwen3-30B-A3B with 128 experts per layer and DeepSeek-V2-Lite with 64 routed experts per layer. Both Qwen3-30B-A3B and DeepSeek-V2-Lite are prevailing open-sourced MoE models.

Datasets. We use the following three representative datasets *MMLU* Hendrycks et al. (2021b;a), *lmsys-chat-1m* Zheng et al. (2023), *ShareGPT-Vicuna-unfiltered* Datasets (2023). In the experiments, we only focus on the prompt parts of these datasets. These datasets contain data from different domains, representing real-world user request patterns, and can effectively evaluate the affinity of requests and tokens from different domains for experts.

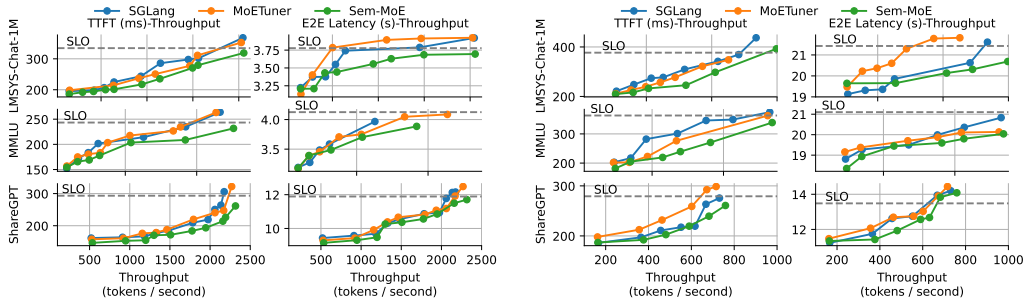
4.3 BASELINES AND PERFORMANCE METRICS

We select SGLang and MoETuner as the baselines to compare, representing the SOTA LLM inference engine and SOTA MoE model scheduling technique.

SGLang: SGLang Zheng et al. (2024) is a prevailing open-source LLM inference framework, incorporating numerous optimizations, including but not limited to continuous batching, paged attention, flash attention, radix-attention, advanced quantization, etc. SGLang declares optimizations for MoE models with high-performance, fused triton kernels OpenAI, supporting DP and TP for attention layers and EP for MoE layers. SGLang is the SOTA open-sourced LLM inference engine, which we set as a strong baseline to compare.

MoETuner: MoETuner Go & Mahajan (2025) is an optimization framework that enhances MoE model serving performance by finding an optimal expert placement strategy. It addresses critical bottlenecks in expert parallelism, namely imbalanced token processing loads across GPUs and skewed inter-GPU communication, which lead to significant tail latency. The core of MoETuner is an Integer Linear Programming (ILP) formulation that leverages predictable token routing dependencies across layers. It jointly optimizes expert-to-GPU assignments to balance computational workloads and minimize communication costs, thereby reducing end-to-end execution time. We embed MoETuner into SGLang as a comparable baseline.

Metrics: To mitigate performance fluctuating, we set the input length and output length fixed. Then we vary the request rate from 10 to 175 req/s, and observe the following metrics. **Throughput** is the number of tokens (tokens/s) that an inference system can process per unit time. **TTFT (Time to First Token)** measures the duration between the request's arrival and the first token's generation time. **E2E (End-to-End) Latency** measures the duration between the request's arrival and the last token's generation time. Following prior work Wu et al. (2024b;a), we set the latency SLO as 5$\times$ of the latency under the lightest input load (minimal request rate). For each dataset, we use 20% of the



(a) DeepSeek-V2-Lite. (b) Qwen3-30B-A3B.

Figure 4: Attention-DP Scenario: Inference throughput under TTFT and E2E latency SLOs.

Models	Input Length	p99 TTFT (ms)			Median E2E Latency (ms)		
		SGLang	MoETuner	Sem-MoE	SGLang	MoETuner	Sem-MoE
DeepSeek-V2-Lite	256	84.87	80.96	**75.63**	617.87	604.46	**603.10**
	512	98.10	92.17	**88.69**	609.81	602.02	**599.89**
	1024	111.19	119.76	**100.74**	608.96	606.54	**604.45**
Qwen3-30B-A3B	256	85.72	87.24	**74.46**	758.60	763.97	**750.26**
	512	104.76	101.86	**83.87**	766.80	759.41	**759.16**
	1024	107.73	107.36	**103.69**	769.83	764.16	**762.30**

Table 1: Attention-TP Scenario: TTFT and E2E latency under different

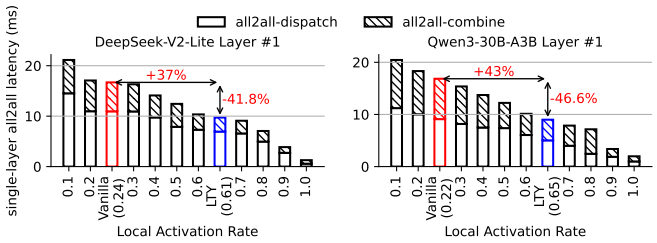
data to train the token activation prediction model and generate the expert placement table, and the remaining 80% is used to sample experimental requests.

4.4 End-to-End Inference Performance

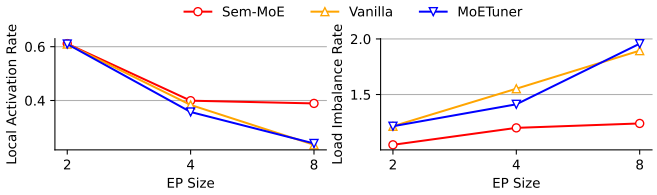
Figure 4 shows the end-to-end performance evaluation of Sem-MoE and two baselines.

Attention-DP Scenario. For the attention-DP scenario, requests are scheduled to different DP ranks for attention and synchronized at MoE layer. The latency of each layer is determined by the slowest DP(EP) rank. Thus we use token throughput to measure the overall performance in attention-DP. We draw a latency-throughput curve to measure the highest throughput a system can achieve under pre-defined SLOs, shown in Figure 4. Data points near the bottom-right corner are better. For Deepseek-V2-Lite, Sem-MoE achieves throughput improvements of 31% and 221% against SGLang with DeepEP under TTFT and end-to-end latency SLO constraints, and 32% and 278% against MoETuner, respectively. For Qwen3-30B-A3B, Sem-MoE's throughput improvement peaks at 98% and 11% against SGLang under SLO constraints, while also achieving gains of 35% and 32% against MoETuner. As the request rate increases continually, baselines stock a bunch of unprocessed requests, yielding a steeper curve, resulting in the above high throughput improvement (221% and 278%). The results demonstrate that Sem-MoE can obtain certain performance gains by scheduling the requests across different DP ranks and co-placing the experts in appropriate devices.

Attention-TP Scenario. For the attention-TP scenario, there is no scheduler for a single inference instance, as different TP ranks receive the same input. Meanwhile, latency becomes a main concern in TP settings. Therefore, we set the request rate to 1 req/s and vary the input sequence length to observe the TTFT and end-to-end latency directly as shown in Table 1. For Deepseek-V2-Lite, Sem-MoE outperforms the best baseline in TTFT by 12.21%, 10.60%, and 18.89% under input lengths of 256, 512, and 1024. For Qwen3-30B-A3B, the performance optimization ratio is 17.16%, 24.90%, 3.80%. Thanks to the load-balance and inter-layer communicaiton optimizing effect, MoETuner gains performance improvement in some cases, yet may slow down in the other cases. Model-data collaborative scheduling can bring holistic performance boosting in all tested scenarios. The speedup of TTFT also translates to the shrinking of end-to-end inference latency, just as shown in Table 1. We would delve into the execution of MoE layers to analyze the rationale behind the breakdown of the inference speedup.



(a) Local activation rate against overall EP overhead.



(b) Local activation rate and load-imbalance rate of Sem-MoE and baselines.

Figure 5: Breakdown Evaluation

4.5 A Detailed Look at EP Communication Reduction

In Figure 5(a), we show the local activation rate and the resulting latency of a single MoE layer under the attention-TP scenario. Local activation means the tokens' activation is routed to an expert collocated on the same GPU device, and thus, remote routing and its associated EP communication can be skipped. Results show that, compared with the vanilla placement, Sem-MoE can increase LAR by 37% and 43% for DeepSeek-V2-Lite and Qwen3-30B-A3B, which translates to 41.8%/46.6% latency reduction of the belonging expert layer. Besides Vanilla and Sem-MoE, other bars in the figure are measured by mocking the routing module of SGLang and skipping the delays in communication to fabricate hypothetical baselines just for reference. Note that a 100% LAR may not be achieved in theory, as different tokens can contradict each other to group their own hot experts, but GPU memory is limited.

4.6 Algorithm Evaluation

The model-data collaborative scheduling algorithm needs to find balanced co-clusters of tokens and experts, with experts having a maximal likelihood of being gated (routed) from tokens within the same cluster, and minimal likelihood across clusters. An additional regularizer is load balancing that ensures hot and cold experts are relatively evenly distributed. Sem-MoE adopts Algorithm 1 to approximately solve the problem and is evaluated against two baselines, the vanilla scheduling policy (original expert placement policy and round-robin scheduling) and MoETuner. Figure 5(b) shows the averaged local activation rate (ratio of tokens computed at the local device) and load imbalance rate (maximum load divided by the median load) of all the MoE layers in DeepSeek-V2-Lite. Sem-MoE achieves the best local activation rate with balanced expert clusters outperforming the best baseline by 15.4% and 36.7% under EP8 setting.

5 Conclusion

The communication overhead of expert parallelism renders a significant bottleneck in serving large-scale MoE models. We present Sem-MoE, which can proactively and losslessly trim EP's `all2all` communication volume via model-data collaborative scheduling, leveraging the intrinsic affinity between model experts and input tokens. Experiments show that Sem-MoE can significantly reduce communication overhead and boost inference throughput under differently specified SLO constraints, both in attention-DP and attention-TP scenarios.

REFERENCES

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, Giri Anantharaman, Xian Li, Shuohui Chen, Halil Akin, Mandeep Baines, Louis Martin, Xing Zhou, Punit Singh Koura, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Mona Diab, Zornitsa Kozareva, and Ves Stoyanov. Efficient large scale language modeling with mixtures of experts, 2022. URL https://arxiv.org/abs/2112.10684.

Chang Chen, Min Li, Zhihua Wu, Dianhai Yu, and Chao Yang. Ta-moe: Topology-aware large scale mixture-of-expert training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22173–22186. Curran Associates, Inc., 2022.

Hugging Face Datasets. Sharegpt vicuna unfiltered dataset., 2023. URL https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024a.

DeepSeek-AI. Deepseek-v3 technical report, 2024b. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL http://jmlr.org/papers/v23/21-0998.html.

Seokjin Go and Divya Mahajan. Moetuner: Optimized mixture of expert serving with balanced expert placement and token routing, 2025. URL https://arxiv.org/abs/2502.06643.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. Fastmoe: A fast mixture-of-expert training system, 2021. URL https://arxiv.org/abs/2103.13262.

Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. Fastermoe: modeling and optimizing training of large-scale dynamic pre-trained models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '22, pp. 120–134, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392044. doi: 10.1145/3503221.3508418. URL https://doi.org/10.1145/3503221.3508418.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.

Connor Holmes, Masahiro Tanaka, Michael Wyatt, Ammar Ahmad Awan, Jeff Rasley, Samyam Rajbhandari, Reza Yazdani Aminabadi, Heyang Qin, Arash Bakhtiari, Lev Kurilenko, and Yuxiong He. Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference, 2024.

Haiyang Huang, Newsha Ardalani, Anna Sun, Liu Ke, Hsien-Hsin S. Lee, Anjali Sridhar, Shruti Bhosale, Carole-Jean Wu, and Benjamin Lee. Towards moe deployment: Mitigating inefficiencies in mixture-of-expert (moe) inference, 2023. URL https://arxiv.org/abs/2303.06182.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, Joe Chau, Peng Cheng, Fan Yang, Mao Yang, and Yongqiang Xiong. Tutel: Adaptive mixture-of-experts at scale. *CoRR*, abs/2206.03382, June 2022. URL https://arxiv.org/pdf/2206.03382.pdf.

Ranggi Hwang, Jianyu Wei, Shijie Cao, Changho Hwang, Xiaohu Tang, Ting Cao, and Mao Yang. Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 1018–1031, 2024. doi: 10.1109/ISCA59077.2024.00078.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Jiamin Li, Yimin Jiang, Yibo Zhu, Cong Wang, and Hong Xu. Accelerating distributed MoE training and inference with lina. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pp. 945–959, Boston, MA, July 2023. USENIX Association. ISBN 978-1-939133-35-9. URL https://www.usenix.org/conference/atc23/presentation/li-jiamin.

Xiaonan Nie, Xupeng Miao, Zilong Wang, Zichao Yang, Jilong Xue, Lingxiao Ma, Gang Cao, and Bin Cui. Flexmoe: Scaling large-scale sparse pre-trained model training via dynamic device placement. *Proc. ACM Manag. Data*, 1(1), May 2023. doi: 10.1145/3588964. URL https://doi.org/10.1145/3588964.

NVIDIA. Tensorrt-llm. [Online]. Accessed 23 Oct 2024, https://github.com/NVIDIA/TensorRT-LLM.

OpenAI. Applied ai experiments and examples for pytorch. [Online]. Accessed 23 Oct 2024, https://github.com/pytorch-labs/applied-ai/tree/main.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCallum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Yulei Qian, Fengcun Li, Xiangyang Ji, Xiaoyu Zhao, Jianchao Tan, Kefeng Zhang, and Xunliang Cai. Eps-moe: Expert pipeline scheduler for cost-efficient moe inference, 2025. URL https://arxiv.org/abs/2410.12247.

Qwen-Team. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024. URL https://qwenlm.github.io/blog/qwen-moe/.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18332–18346. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.html.

Siddharth Singh, Olatunji Ruwase, Ammar Ahmad Awan, Samyam Rajbhandari, Yuxiong He, and Abhinav Bhatele. A hybrid tensor-expert-data parallelism approach to optimize mixture-of-experts training. In *Proceedings of the 37th International Conference on Supercomputing*, ICS '23, pp. 203–214, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700569. doi: 10.1145/3577193.3593704. URL https://doi.org/10.1145/3577193.3593704.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Bingyang Wu, Shengyu Liu, Yinmin Zhong, Peng Sun, Xuanzhe Liu, and Xin Jin. Loongserve: Efficiently serving long-context large language models with elastic sequence parallelism, 2024a. URL https://arxiv.org/abs/2404.09526.

Bingyang Wu, Yinmin Zhong, Zili Zhang, Shengyu Liu, Fangyue Liu, Yuanhang Sun, Gang Huang, Xuanzhe Liu, and Xin Jin. Fast distributed inference serving for large language models, 2024b. URL https://arxiv.org/abs/2305.05920.

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. Moe-infinity: Offloading-efficient moe model serving, 2024. URL https://arxiv.org/abs/2401.14361.

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, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang

Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Jinghan Yao, Quentin Anthony, Aamir Shafi, Hari Subramoni, Dhabaleswar K., and Panda. Exploiting inter-layer expert affinity for accelerating mixture-of-experts model inference, 2024.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. Edgemoe: Fast on-device inference of moe-based large language models, 2023. URL https://arxiv.org/abs/2308.14352.

Mingshu Zhai, Jiaao He, Zixuan Ma, Zan Zong, Runqing Zhang, and Jidong Zhai. SmartMoE: Efficiently training Sparsely-Activated models through combining offline and online parallelization. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pp. 961–975, Boston, MA, July 2023. USENIX Association. ISBN 978-1-939133-35-9. URL https://www.usenix.org/conference/atc23/presentation/zhai.

Chenggang Zhao, Shangyan Zhou, Liyue Zhang, Chengqi Deng, Zhean Xu, Yuxuan Liu, Kuai Yu, Jiashi Li, and Liang Zhao. Deepep: an efficient expert-parallel communication library. https://github.com/deepseek-ai/DeepEP, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024. URL https://arxiv.org/abs/2312.07104.

Shuzhang Zhong, Ling Liang, Yuan Wang, Runsheng Wang, Ru Huang, and Meng Li. Adapmoe: Adaptive sensitivity-based expert gating and management for efficient moe inference, 2024. URL https://arxiv.org/abs/2408.10284.

A Empirical Study of Token-expert Affinity

A.1 Impact of Request Semantics on Expert Activation

To better illustrate the impact of request semantics on expert activation, we selected requests from several different types of topics in the MMLU dataset and profiled the expert activations in the 24th layer of Qwen3-30B-A3B, as shown in Figure 6(a). After performing t-SNE dimensionality reduction, it can be observed that requests from similar topics exhibit similarity in activated experts. Requests from the math-related topics of abstract algebra and college mathematics activate similar experts, whereas requests from humanities topics, such as philosophy and professional law, are relatively distant from the math-related ones in the reduced-dimensional space. Sem-MoE leverages this semantic affinity between requests to perform co-scheduling of data and models for requests under the Attention-DP scenario.

A.2 Intra-layer bi-clustered token-expert Conjugacy

Within each MoE layer, each expert module in an LLM layer is trained to process a particular semantic domain of tokens. Tokens and experts exhibit high affinity in different dimensions. We profile the intermediate activation of the gating module in each MoE layer in the DeepSeek-V2-Lite. One important observation shows that strong bi-clustered conjugacy between tokens and experts, as shown in Figure 6. That is, experts are likely to be activated by a certain sub-group of tokens with high semantic affinity, while they are not likely to be activated by other general tokens in the vocabulary. And from the tokens' perspective, it is true that semantically similar tokens are likely to activate a certain sub-group of experts. This is the preliminary motivation for model-data collaborative scheduling.

Inter-layer expert-expert affinity The left picture of Figure 6(c) shows the activation correlation of the 4th and 5th layer of the Mixtral-8x7B model. The x-/y-axis represents the expert groups of a layer. For Mixtral-8x7B, each token is routed to 2 out of 8 experts at each layer. Thus, the number of expert groups at each layer is $\binom{8}{2} = 28$. When tokens choose some concrete experts at the fourth layer, they tend to choose a rather fixed set of experts at the next layer with high probability. We name this phenomenon *inter-layer expert-expert affinity*.

Simple conditional probability model for token activation path The above examples illustrate the tabularized relationship between tokens and experts. We argure that, simple conditional probability model can work to predict the activation path of tokens by combining the above intra-layer conjugacy and/inter-layer affinity. We first calculate the kurtosis[3] of each token's activation map. The right picture of Figure 6(b) shows most of the kurtosis values are higher than 8, indicating that the to-route experts for each token concentrate on a narrowed set, regardless of the context. We further use partial (25%) of the profile dataset to calculate each tokens' most routed top-k experts. Then the static top-k experts are used to predict the tokens activation using the left part (75%) of the profile dataset, achieving a 96.3% precision and a 78.8% F1 score. The right part of Figure 6(c) predicts the next-layer-activation via looking back the prior layers. As we know more about the previous activation sequence at layer L, we can predict the activation at layer $L + 1$ with higher confidence (about 70% when looking back 5 layers).

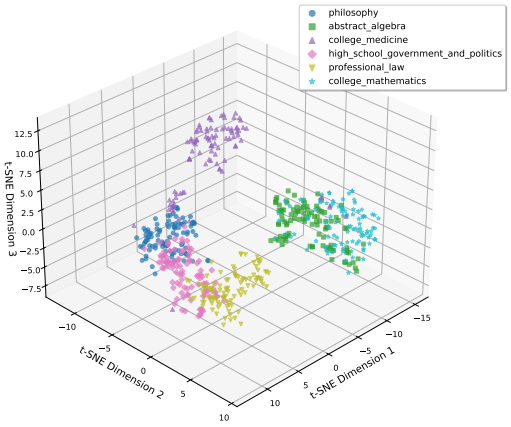
Expert pre-grouping and token re-batching Leveraging the intrinsic conjugacy between tokens and experts in MoE models to achieve token-expert co-dispatching, may help reduce communication volume and boost distributed parallel inference. First, similar experts can be pre-grouped together at deployment time based on pre-profiled and modeled affinity. Second, on the fly, individual tokens can be re-shuffled and re-batched to the GPU devices that host the expert groups whose member experts have largest modeled conjugacy with the token. Such co-scheduling aims to avoid scattered, cross-device token-expert activations, or equivalently, maximize the probability of intro-device, local activations. Figure 6(d) shows an micro-benchmark experiment, testing the `all2all` latency under different local activation rate using the *nccl* `all2allv` API. With the local activation rate (α) varying from 0.2 to 0.9, the latency decreases gradually, showing the performance gains with improved expert-token co-scheduling.

B Algorithm for the model-data co-scheduling Solver

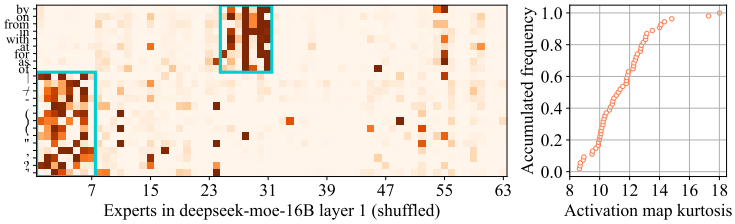
Algorithm 1 describes the model-data co-scheduling alternating optimization algorithm in Sem-MoE. The algorithm achieves optimal performance by alternating between optimizing the scheduling of requests and the placement of experts. Initially, requests are clustered based on their affinity to experts to determine their scheduling (line 44). In each iteration, the algorithm alternates between optimizing expert placement with fixed requests and request scheduling with fixed experts. For optimizing expert placement, experts are first sorted by their hotness in descending order. Given the current request scheduling and expert placement, the affinities between experts and the cluster's experts/requests are computed and aggregated via weights α_e and β_e, which is adjusted by the cluster's current load to derive a final affinity score (lines 11-13). The expert is assigned to the highest-scoring cluster, with saturated clusters masked (line 14). The algorithm then performs ft_steps fine-tuning rounds, randomly selects two clusters and swaps their experts if it improves the affinity score (lines 20-25). Request scheduling optimization is similar to expert placement. The req-req affinity and req-expert affinity for each cluster are calculated, aggregated to obtain an affinity score, and the request is scheduled to the cluster with the highest score (lines 28-42).

By now, the token-device scheduling table $\mathcal{T}$, token-device scheduling confidence table $\mathcal{T}_p$, and expert-device scheduling table $\mathcal{E}$ are generated. After the scheduling table $\mathcal{E}$ is constructed, the experts at each layer need to be rearranged according to the scheduling table during online inference service deployment. In addition, the Sem-MoE rearranges the gating module by column to implement transparent expert shuffle. The semantics of other layers are not affected. The rearranged experts are highly boxed, so that the token activation at each layer is de-cohesive, and the redundant network communication overhead caused by dispersive activation is reduced.

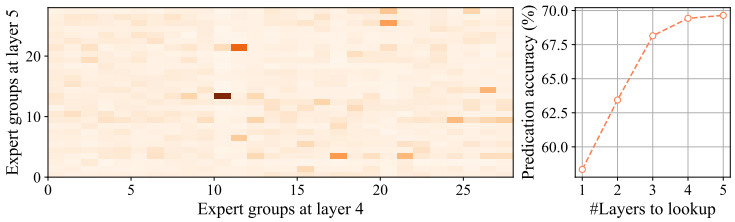
[3]Kurtosis is a measure of the tailedness of a distribution. High Kurtosis indicates a token favors several fixed experts during multiple occurrences.



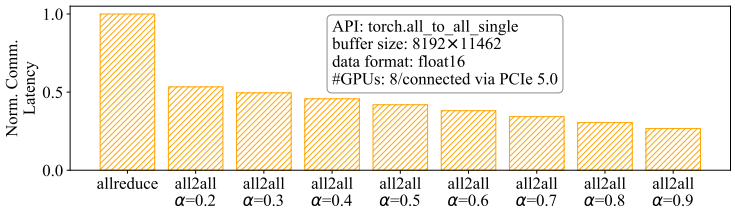
(a) Example of expert activations for requests in different topics (24th layer of Qwen3-30B-A3B profiled using the MMLU dataset), with t-SNE dimensionality reduction.



(b) Example of intra-layer token-expert activation map (1st MoE layer of DeepSeek-V2-Lite profiled using the Sharegpt dataset). Darker color in the map indicates higher activation frequency or stronger correlation.



(c) Example of inter-layer expert-expert correlation map (4th/5th MoE layer of Mixtral-8x7B profiled using the LongBench dataset). Darker color in the map indicates stronger correlation.



(d) Example of perfromance of `allreduce` and `all2all` under different local activation ratio (α).

Figure 6: Conjugacy illustration and collective communication micro-benchmark.

Algorithm 1: Alternating-based data-model co-scheduling algorithm

input: n_steps: number of iteration steps;
$\boldsymbol{\mathcal{C}}_p$: the token-2-expert confidence table; $\boldsymbol{a}$: the token frequency;
$\boldsymbol{r}$: requests list; K: number of requests;
N: number of experts per layer; E: number of co-clusters;
t: number of tokens;
output: $\mathcal{E}$: expert labels; $\mathcal{T}$: token labels;
$\mathcal{T}_p$: confidence of tokens choosing specific experts

$p\_matrix\_ep\_opt \leftarrow$ `zeros(`N, E`)` / E
$p\_matrix\_req\_opt \leftarrow$ `zeros(`K, E`)` / E
Function `expert_place(`$\boldsymbol{\mathcal{C}}_p$, p_matrix_req, α_e, β_e`)`**:**
    $loads \leftarrow$ compute per expert load by ($\boldsymbol{\mathcal{C}}_p$)
    `sort_by_load(`$\boldsymbol{e}$, $loads$`)`
    $mask$, $cnter \leftarrow$ `ones(`E`)`, `zeros(`E`)`
    $EAfE$, $EAfR \leftarrow$ `zeros(`N, E`)`, `zeros(`N, E`)`
    $p\_matrix\_ep \leftarrow$ `zeros(`N, E`)` / E
    $loads\_cls \leftarrow$ `zeros(`E`)`
    for e *in* $\boldsymbol{e}$ **do**
        $EAfE[e] \leftarrow$ compute expert-expert affinity by ($mask$, p_matrix_ep, $\boldsymbol{\mathcal{C}}_p$)
        $EAfR[e]$ compute req-expert affinity by ($mask$, p_matrix_req, $\boldsymbol{\mathcal{C}}_p$)
        $aff\_score \leftarrow \alpha_r * EAfE[e] + \beta_r * EAfR[e] - \gamma_e * loads\_cls$
        $cls_e \leftarrow \arg\max_{cls}\, aff\_score$
        $p\_matrix\_ep[e][cls_e] \leftarrow 1$
        $cnter[cls_e] \leftarrow cnter[cls_e] + 1$
        if $cnter[cls_e] >= N/E$ **then**
            $maks[cls_e] \leftarrow 0$
        $loads\_cls[cls_e] \leftarrow loads\_cls[cls_e] + loads[e]$
    repeat
        $cls_1, cls_2 \leftarrow$ randomly select a cluster in $[\![E]\!]$
        $e_1, e_2 \leftarrow$ randomly select experts in $p\_matrix\_ep[:][cls_1]$ and $p\_matrix\_ep[:][cls_2]$
        if `aff_gain(`e_1, e_2, cls_1, cls_2`)` > 0 **then**
            `swap(`e_1, e_2, p_matrix_ep`)`
    until *iterating for* ft_steps *steps*
    return p_matrix_eq

Function `request_schedule(`$\boldsymbol{\mathcal{C}}_p$, p_matrix_ep, α_r, β_r`)`**:**
    `sort_by_len(`$\boldsymbol{r}$`)`
    $mask$, $cnter \leftarrow$ `ones(`E`)`, `zeros(`E`)`
    $RAfR$, $RAfE \leftarrow$ `zeros(`K, E`)`, `zeros(`K, E`)`
    $p\_matrix\_req \leftarrow$ `zeros(`K, E`)` / E
    for r *in* $\boldsymbol{r}$ **do**
        $RAfR[r] \leftarrow$ compute req-req affinity by ($mask$, p_matrix_req, $\boldsymbol{\mathcal{C}}_p$)
        $RAfE[r] \leftarrow$ compute req-expert affinity by ($mask$, p_matrix_ep, $\boldsymbol{\mathcal{C}}_p$)
        $aff\_score \leftarrow \alpha_r * RAfR[r] + \beta_r * RAfE[r]$
        $cls_r \leftarrow \arg\max_{cls}\, aff\_score$
        $p\_matrix\_req[r][cls_r] \leftarrow 1$
        $cnter[cls_r] \leftarrow cnter[cls_r] + 1$
        if $cnter[cls_r] >= K/E$ **then**
            $maks[cls_r] \leftarrow 0$
    return p_matrix_req

$p\_matrix\_req \leftarrow$ cluster based on expert affinity
repeat
    $p\_matrix\_ep \leftarrow$ `expert_place(`$\boldsymbol{\mathcal{C}}_p$, p_matrix_req, α_e, β_e`)`
    $p\_matrix\_req \leftarrow$ `request_schedule(`$\boldsymbol{\mathcal{C}}_p$, p_matrix_ep, α_r, β_r`)`
    $scores \leftarrow$ summation the max load and communication cost given p_matrix_eq and p_matrix_req
    $better\_scheduling \leftarrow$ samples with scores
    update $p\_matrix\_ep\_opt$ and $p\_matrix\_req\_opt$
until *iterating for* n_steps *steps*

$\mathcal{E} \leftarrow$ `argmax(`$p\_matrix\_ep\_opt$, *axis*=1`)`
$p\_matrix\_tk\_opt \leftarrow$ count the tokens per req in $p\_matrix\_req\_opt$
$\mathcal{T}, \mathcal{T}_p \leftarrow$ `argmax_with_values(`$p\_matrix\_tk\_opt$, *axis*=1`)`

B.1 Modeling Inter-layer Activation Conjugacy

Leveraging the conditional probability model described in § A.2, we use a simple probability-based first-order Marcov chain to model the inter-layer activation conjugacy. To reduce the combination space, we model the activation device sequence rather than the activation expert sequence, because we only care about the device-level token rebatching. When looking back l layers, we construct a table shaped like $[E^l, E]$, where the row of the table indicates the sequence of devices selected at the previous l layers and the column indicates the probability of activating the E devices in the current layer. Like the § B shows, we also calculate the activation sequence to device table $\mathcal{A}$ and the confidence table $\mathcal{A}_p$. In practice, we set the number of looking-back layers as 2.

Algorithm 2: Online request scheduling based on fast lookup

input: $\mathcal{R} \in \mathbb{N}^n$: Input requests; $\mathcal{T}$: token-to-expert-cluster Schedule Table; E: number of DP size

```
1  dev_mask ← ones(E)
2  Function get_dp_rank(R, T):
3      dev_score ← sum(T[R, :], dim = 0)
4      dev_score[ dev_mask] ← − inf
5      dev_id ← argmax(dev_score)
6      dev_mask[dev_id] ← False
7      if dev_mask all are False then
8          dev_mask ← ones(E)
9      return dev_id
10 dev_id ← get_dp_rank(R, T)
11 schedule(R, dev_id)
```

B.2 Speculative Token Shuffling on the Fly Based on Fast Lookup

To reduce the combination space, we model the activation device sequence rather than the activation expert sequence, because we only care about the device-level token rebatching. We implement a fast online token re-batching mechanism based on fast looking-up tables in both Attention-DP and Attention-TP (Algorithm 2 & Algorithm 3).

Data Scheduling: Attention-DP Scenarios. The algorithm 2 queries the token-to-expert-cluster scheduling table $\mathcal{T}$ based on the token IDs appearing in the request $\mathcal{R}$, and aggregates the results to obtain a score for each device for that request (line 3). Then $\mathcal{R}$ is scheduled to the device with the max valid score (line 5). To prevent requests biased toward a subset of experts, which could skew the load during the decoding phase, we introduce a dev_mask. The device is masked after it is allocated (line 4-5). Once a round of allocation is completed and all devices are masked, the dev_mask is reset and enters a new round (line 7-8). This ensures that Sem-MoEachieves expert affinity while maintaining load balance across devices.

Data Scheduling: Attention-TP Scenarios. The algorithm 3 queries the token-to-expert-cluster scheduling table $\mathcal{T}$ and expert-cluster-sequence-to-expert-cluster table $\mathcal{S}$, together with their confidences first. Then, the table with higher confidence is adopted to obtain the device ID list to which the current batch token needs to be shuffle (line 2). The algorithm performs the argsort operation to obtain the shuffle indicators (line 3) of the token. Then, the final shuffle indicators are obtained by grouping, aligning, and concatenation, and the token is shuffled (line 4 to line 7). After rebatching is complete, Sem-MoE calls the `reduce-scatter` operation. After MoE computing is complete, Sem-MoE runs the `allgather` operation to collect tokens. Finally, the order of tokens are shuffled back based on the previously calculated shuffle indicators (lines 14-18).

The both algorithm do not involve complex load calculation and decision-making. They are directly completed by querying tables. The runtime overhead mainly involves large token matrix shuffling, which we optimize via high-performance kernels. The memory occupation of the scheduling tables is negligible. For example, for DeepSeek-V2, the memory space that the token-to-device table $\mathcal{T}$ occupies is $\frac{102400 \times 60 \times 2}{1024^2} \approx 11.72MB$ (assuming the data format is `int16`).

Algorithm 3: Online token re-batching based on fast lookup

input: $\mathcal{B} \in \mathbb{N}^n$: Input token IDs; $\mathcal{T}$: token-to-expert-cluster Schedule Table;
$\mathcal{A}$: expert-cluster-sequence-to-expert-cluster Schedule Table

Function `rebatch_tokens` $(\mathcal{B}, \mathcal{T})$ **:**
- $dev\_ids \leftarrow$ `cond` $(\mathcal{T}_p[\mathcal{B}] > \mathcal{A}_p[\mathcal{B}], \mathcal{T}[\mathcal{B}], \mathcal{A}[\mathcal{B}])$
- $shf\_indices \leftarrow$ `argsort` (dev_ids)
- $g\_shf\_indices \leftarrow$ `group_by_key` $(shf\_indices)$
- $g\_shf\_indices \leftarrow$ `align` $(g\_shf\_indices)$
- $shf\_indices \leftarrow$ `concat` $(g\_shf\_indices)$
- $\mathcal{B} \leftarrow \mathcal{B}[shf\_indices]$
- **return** $shf\_indices$

Function `resume_tokens` $(\mathcal{B}, shf\_indices)$ **:**
- $r\_shf\_indices \leftarrow$ `argsort` $(shf\_indices)$
- $\mathcal{B} \leftarrow \mathcal{B}[r\_shf\_indices]$

$shf\_indices \leftarrow$ `rebatch_tokens` $(\mathcal{B}, \mathcal{T})$
$\mathcal{B}_{local} \leftarrow$ `reduce_scatter` $(\mathcal{B})$
executing MoE layer
$\mathcal{B} \leftarrow$ `allgather` $(\mathcal{B}_{local})$
`resume_tokens` $(\mathcal{B}, shf\_indices)$