GRACE-MoE: Grouping and Replication with Locality-Aware Routing for Efficient Distributed MoE Inference

Anonymous authors
Paper under double-blind review

Abstract

Sparse Mixture of Experts (SMoE) performs conditional computation by selectively activating a subset of experts, thereby enabling scalable parameter growth in large language models (LLMs). However, the expanded parameter scale exceeds the memory capacity of a single device, necessitating distributed deployment for inference. This setup introduces two critical challenges: (1) *Communication Issue*: Transferring features to devices with activated experts leads to significant communication overhead. (2) *Computational Load Issue*: Skewed expert activation overloads certain GPUs, resulting in load imbalance across devices. Among these, communication overhead is identified as the main bottleneck in SMoE inference. Nevertheless, reducing communication between devices may exacerbate computational load imbalance, leading to device idleness and resource waste. Therefore, we present **GRACE-MoE**, short for **G**rouping and **R**eplic**a**tion with Lo**c**ality-Awa**r**e Routing for S**MoE** inference. **GRACE-MoE** is a co-optimization framework that jointly reduces communication overhead and alleviates computational load imbalance. Specifically, the framework comprises two key phases: ① *Grouping & Replication*: This phase groups experts based on their affinity to reduce cross-device communication. Additionally, dynamic replication is applied to address load skew, improving computational load balance across GPUs. ② *Routing*: This phase employs a locality-aware routing strategy with load prediction. It prioritizes local replicas to minimize communication overhead and balances requests across remote replicas when necessary. Experiments on diverse models and multi-node, multi-GPU environments demonstrate that **GRACE-MoE** efficiently reduces end-to-end inference latency, achieving up to **3.79**× speedup over state-of-the-art systems. Code for **GRACE-MoE** will be released upon acceptance.

1 Introduction

Large language models (LLMs) built on the Transformer architecture (Vaswani et al., 2017) demonstrate substantial performance gains as the number of parameters increases (Brown et al., 2020). However, scaling dense models by simply enlarging parameter counts incurs prohibitive computation and memory costs (Kaplan et al., 2020; Clark et al., 2022). The Sparse Mixture-of-Experts (SMoE) architecture mitigates this by partitioning parameters into experts and activating only a small subset per token, thereby enabling "large-parameter but small-computation" scaling (Shazeer et al., 2017). Recent SMoE systems, such as GShard (Lepikhin et al., 2020) and Switch Transformer (Fedus et al., 2022), have already reached trillion-parameter scales, underscoring this potential.

Unfortunately, the massive parameter scale of SMoE exceeds the memory and computation capacity of a single device[1], necessitating distributed deployment with expert parallelism, which is often combined with data parallelism (Lepikhin et al., 2020; Zhai et al., 2023). In this setting, experts within each MoE layer are partitioned across GPUs and coordinated through All-to-All communication. This design introduces two critical bottlenecks for inference: communication overhead and computational load[2] imbalance.

[1] In this paper, a device refers to an individual computing unit (i.e., a GPU), a node refers to a server that contains multiple GPUs, and cross-device communication covers both intra-node and inter-node cases.

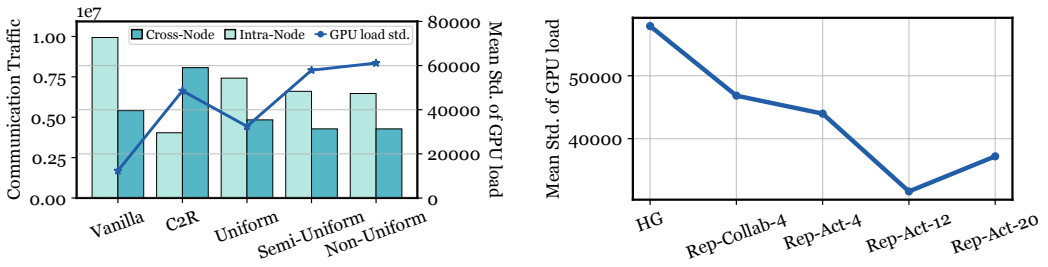
Each MoE layer requires two rounds of All-to-All communication, first dispatching tokens to experts and then aggregating results back. Executed repeatedly across layers, this process continuously amplifies communication latency, making communication the primary bottleneck in SMoE inference (He et al., 2022; Gale et al., 2023). In cross-node scenarios, where bandwidth is limited, All-to-All can account for over 70% of the time within a single MoE layer and around 40% of the overall end-to-end inference latency (Li et al., 2023a; Hwang et al., 2023). In parallel, the gating network naturally skews token routing, creating "hot" and "cold" experts. This phenomenon leads to imbalanced computational load, which overloads some GPUs while leaving others underutilized (Lewis et al., 2021; Clark et al., 2022; He et al., 2022). Overloaded GPUs slow down inference, while idle GPUs waste substantial computing resources. Since parallel inference performance is bounded by the slowest device, such imbalance further exacerbates communication tail latency (Go & Mahajan, 2025). More critically, prior work typically addresses only one of these two issues, yet optimizing one often worsens the other. For instance, reducing communication overhead usually aggravates computational load imbalance. This conflict remains unresolved, and most prior work has focused on single-node, multi-GPU settings. Scaling SMoE inference to multi-node environments remains an under-explored challenge, making a joint solution that simultaneously mitigates both issues particularly critical for cross-node scenarios.

In this paper, we propose **GRACE-MoE**, a hybrid framework that consists of two essential phases: ① *Grouping & Replication* and ② *Routing*, which are performed in the offline and online phases, respectively. During the offline phase, **GRACE-MoE** groups experts based on their affinity to reduce cross-device All-to-All communication and selectively replicate frequently activated experts from the most heavily loaded groups. The number of replicas is determined dynamically according to the load skew of the heaviest groups. For the online phase, we design a topology-aware routing strategy to determine which replica executes the computation. This strategy prioritizes local replicas, while weighted round-robin with load prediction distributes requests across remote replicas when no local replica is available. Together, these designs reconcile the conflicting objectives of communication efficiency and load balancing, enabling **GRACE-MoE** to jointly reduce communication overhead and alleviate computational load imbalance. Experiments on various MoE models and multi-node, multi-GPU setups demonstrate that **GRACE-MoE** significantly reduces communication latency, mitigates imbalance, and improves end-to-end inference without accuracy degradation. The main contributions of this work are summarized as follows:

- **Non-uniform expert grouping strategy**: We propose the first affinity-based non-uniform grouping scheme that co-locates highly co-activated experts, substantially reducing cross-device communication overhead compared with existing uniform grouping approaches.
- **Dynamic expert replication scheme**: We develop a dynamic mechanism that allocates replicas according to the load skew of the maximum load group in each layer, replicating only its busiest experts. This alleviates imbalance while avoiding redundant replication.
- **Topology-aware routing with prediction**: We design a lightweight routing algorithm that prioritizes local replicas on the same GPU or node as the tokens. When no local replica exists, weighted round-robin with load prediction distributes requests across remote replicas.
- **Joint optimization in multi-node environments**: **GRACE-MoE** is a comprehensive framework that co-optimizes communication overhead and computational load imbalance in large-scale multi-node, multi-GPU SMoE deployments, providing a practical solution for scalable distributed SMoE inference.

2 Related Work

Efficient SMoE Inference Systems. A wide range of systems have been proposed to accelerate SMoE inference, including SE-MoE (Shen et al., 2022), Janus (Liu et al., 2023), Lina (Li et al., 2023a), MC-SMoE (Li et al., 2023b), APTMoE (Wei et al., 2024), SGLang (Zheng et al., 2024), MoESys (Yu et al., 2024), Pre-gated MoE (Hwang et al., 2024), Klotski (Fang et al., 2025), CoServe (Suo et al., 2025) and many others. Among them, DeepSpeed-MoE (Rasley et al., 2020) reduces inference latency through architecture and operator optimizations, Tutel (Hwang et al., 2023) introduces adaptive parallelism and pipelining, and MegaBlocks (Gale et al., 2023) restructures computation into block-sparse matrix multiplications. In addition, general-purpose inference frameworks such as vLLM (Kwon et al., 2023) also provide support for distributed SMoE inference.



(a) Grouping strictness vs. communication traffic. (b) Replication type vs. computational load balance.

Figure 1: **Grouping strictness and replication strategies.** Experiments on OLMoE with 2 nodes $\times$ 2 GPUs/node, metrics reported in tokens. (a) Relaxing grouping strictness reduces communication compared to Vanilla. (b) Replicating highly activated experts alleviates load imbalance more effectively than replicating widely collaborative experts, relative to Hierarchical Grouping (HG).

Nevertheless, both communication overhead and computational load imbalance remain unresolved bottlenecks in expert-parallel inference.

Expert Placement and Routing. Existing studies on expert placement (e.g., grouping, replication) and routing have attempted to alleviate the bottlenecks in SMoE inference, including FasterMoE (He et al., 2022), FlexMoE (Nie et al., 2023), Prophet (Wang et al., 2023), ExFlow (Yao et al., 2024), Lazarus (Wu et al., 2024), MoETuner (Go & Mahajan, 2025), C2R (Zhang et al., 2025), Speculative MoE (Li et al., 2025), SwiftMoE (Skiadopoulos et al., 2025), EfficientMoE (Zeng et al., 2025), among others. However, these methods generally target either communication overhead or computational load imbalance, but rarely both simultaneously. For example, C2R (Zhang et al., 2025) reduces communication by grouping experts based on collaboration patterns and restricting the routing space, but this exacerbates load imbalance and degrades accuracy. Moreover, most of these works remain confined to single-node, multi-GPU settings, leaving the joint optimization of communication and load balancing in multi-node scenarios largely unexplored.

3 Observations

Motivated by the unresolved challenges of communication overhead and computational load imbalance in multi-node, multi-GPU scenarios (Section 1), we first conduct a systematic analysis of communication traffic, breaking it down into both intra-node and cross-node cases. Under the top-k routing pattern, experts in SMoE models are known to be not activated randomly and independently, but to exhibit clear co-activation characteristics, indicating the existence of affinity among experts. **?** further shows that such relationships are not fixed: some experts tend to be co-activated broadly with many others, while others are bound to only a few, and such distributions vary across layers. Inspired by this, we note that grouping experts strictly according to affinity naturally produces uneven group sizes.

To validate this insight, we design multiple grouping schemes ranging from uniform to fully non-uniform. As shown in Figure 1a, relaxing the uniformity constraint better exploits expert affinity and significantly decreases cross-device traffic. Results reveal a key observation: non-uniform grouping effectively reduces communication. Intuitively, flexible grouping co-locates high-affinity experts on the same GPU, whereas uniform grouping disrupts this natural affinity and limits optimization, which is also a key shortcoming of existing methods. Meanwhile, we observe that affinity-based grouping aggravates the inherent computational load imbalance of SMoE models, especially under non-uniform grouping. These observations provide important insights for our framework design.

4 Method

We propose **GRACE-MoE**, a hybrid optimization framework built upon profiling of routing behaviors. During profiling, per-layer expert selections are recorded to derive expert affinity matrices and load statistics. Guided by these results, the framework integrates offline non-uniform hierarchical

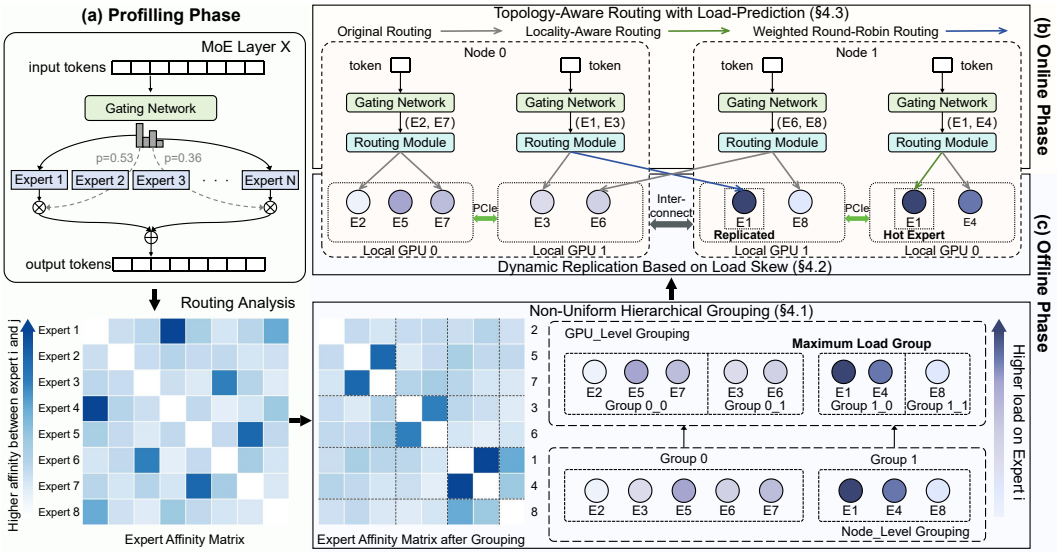


Figure 2: Overview of **GRACE-MoE**. (a) Profiling expert selections to build affinity matrices. (b) Grouping high-affinity experts on the same device and dynamically replicating hot experts to balance computational load. (c) Adaptive routing reduces communication by prioritizing local replicas and balances requests via weighted round-robin with load prediction across remote replicas.

expert grouping (Section 4.1) and dynamic replication based on load skew (Section 4.2) with online locality-aware routing incorporating load prediction (Section 4.3). This comprehensive design effectively reduces communication overhead while improving computational load balance in multi-node, multi-GPU SMoE inference, as illustrated in Figure 2.

4.1 Expert Grouping: Communication-Centric Optimization

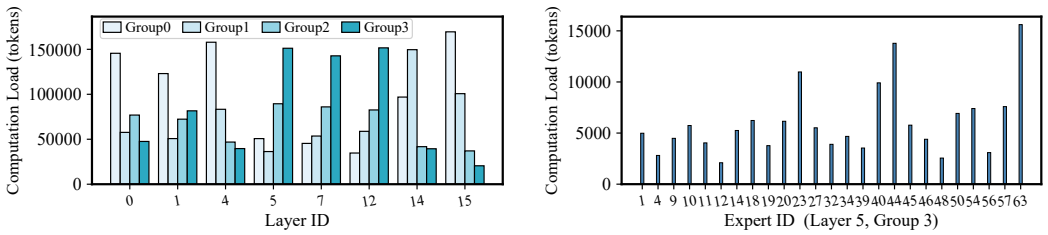
The objective of expert grouping is to colocate high-affinity experts on the same GPU to reduce cross-device communication. We build on spectral clustering to design a hierarchical grouping scheme tailored for multi-node, multi-GPU topologies.

Non-Uniform Grouping of Experts Based on Intra-Layer Affinity. Spectral clustering naturally produces groups with dense intra-connections and sparse inter-connections, which aligns well with our communication-centric objective. As observed in Section 3, affinity-based grouping tends to form uneven group sizes but better captures co-activation patterns, thereby reducing communication. We therefore apply spectral clustering on the expert affinity matrix to generate fully non-uniform groups, with sizes determined solely by affinity.

Although fully non-uniform grouping reduces communication, it leads to computational load imbalance that is even more severe than in the uniform scheme. To mitigate this, we propose controlled non-uniform grouping, regulated by a non-uniformity ratio r that bounds group-size deviations. Given an ideal group size $E = \frac{n}{D}$, where n is the number of experts per layer and D is the number of groups, actual sizes are restricted to $[E - \delta, E + \delta]$, where $\delta = E \cdot r$. The choice of r is critical: too small a value splits high-affinity experts and increases communication, while too large a value creates substantial group size disparity and worsens load imbalance. We model the selection of r as an optimization problem that balances affinity utilization against grouping non-uniformity. We define intra-group affinity utilization $U(r)$ and size deviation $S(r)$ as

$$U(r) = \frac{\sum_{C \in \mathcal{C}(r)} \sum_{\substack{i<j \\ i,j \in C}} A_{i,j}}{\sum_{i<j} A_{i,j}}, \quad S(r) = \sqrt{\frac{1}{D} \sum_{d=1}^{D} \left(|C_d| - E\right)^2}. \tag{1}$$

where r is the candidate ratio, $\mathcal{C}(r) = \{C_1, \ldots, C_D\}$ denotes the grouping with D groups, and $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the affinity matrix, with $A_{i,j}$ denoting the affinity between experts i and j. By plotting



(a) Group-level load analysis across layers.

(b) Per-expert load within the heaviest group.

Figure 3: **Computational load distribution after hierarchical grouping.** (a) Sampled layers show that affinity clustering concentrates load only on a few groups. (b) In Layer 5, per-expert loads in the heaviest group reveal that overload comes from a few frequently activated experts, not all.

$(S(r), U(r))$, we select the knee point as the optimal r, preserving affinity while avoiding excessive size gaps. The validity of this choice is empirically confirmed in Appendix **??**. After determining r, we refine fully non-uniform grouping by reassigning experts with the lowest intra-group affinity to candidate groups with higher affinity, yielding a scheme with controlled non-uniformity. Details of the algorithm are provided in Appendix A.2.

Hierarchical Grouping for Distributed Expert Placement. In multi-node, multi-GPU scenarios, we adopt a hierarchical grouping strategy. At the inter-node level, experts within each layer are divided into N large groups mapped to nodes. Since inter-node communication is much more expensive, we apply fully non-uniform grouping to maximize intra-node affinity and minimize cross-node traffic. Within each node, these groups are further partitioned into G smaller groups mapped to individual GPUs, where controlled non-uniform grouping is applied to balance group size while preserving affinity. This two-level strategy achieves communication optimization across the topology: affinity is maximized within GPUs, weaker across GPUs in the same node, and rare across nodes. As a result, communication overhead is significantly reduced.

4.2 Expert Replication: Computational Load Balance-Centric Optimization

The affinity-based expert grouping scheme reduces communication but also aggravates the inherent computational load[2] imbalance of SMoE models. High-affinity experts are frequently co-activated, and when grouped together, they tend to overload their hosting GPU. To mitigate this imbalance while preserving the communication benefits of grouping, we propose dynamic expert replication.

Selection of Experts for Replication. We compare two candidates: highly activated experts and widely collaborative experts, with a copy of each selected expert placed on every GPU. Figure 1b shows that replicating the former balances load better, whereas the latter is less effective since wide collaboration does not necessarily imply selection by most tokens. Thus, we replicate only highly activated experts. Further tests show that replicating only a few experts provides limited relief, while a moderate number effectively reduces imbalance. However, excessive replication reverses the trend and increases load skew. We attribute this to redundant replication, which degrades the system toward data parallelism and disrupts affinity-based grouping, while also incurring unnecessary memory overhead. Hence, the replication scope must be carefully constrained. As illustrated in Figure 3, after grouping, only a few groups in each layer handle the majority of tokens, and the overload mainly stems from a small number of frequently activated experts. We therefore replicate only these experts within the heaviest group rather than the entire group, preserving intra-group affinity and communication benefits while avoiding redundancy and wasted resources.

Dynamic Replica Allocation Based on Load Skew. Since expert activation distributions and grouping results vary across layers, the computational load skew of the heaviest group also differs. Therefore, we propose a dynamic replica allocation strategy driven by load skew. After generating the expert groups in each layer, profiling data are used to calculate the load W_i of each group, yielding the maximum $W_{\max}$ and mean load $\overline{W}$. The computational load skew factor (ρ) is defined as

[2]The computational load is the number of tokens assigned to an expert, or the total over a group or GPU.

$W_{\max}/\overline{W}$, and the number of replicas is determined by Eq. (2).

$$n_{\text{replica}} = \min\left(\max\left(1, \lfloor \rho \rfloor\right),\ n_{\text{gpu}} - 1\right). \tag{2}$$

Within the heaviest group, experts are ranked by individual load, and those whose cumulative load exceeds $W_{\max} \cdot \frac{n_{\text{replica}}}{1+n_{\text{replica}}}$ are identified as hot experts. These experts are then replicated onto the n_{replica} most underutilized GPUs. The original primary replicas remain, while additional replicas serve only as secondary copies, keeping the grouping structure intact. This mechanism effectively redistributes the workload of hotspot GPUs while maintaining communication efficiency, significantly mitigating the imbalance amplified by grouping.

4.3 Routing Policy: Co-Optimizes Communication and Computational load

After replication, multiple expert instances exist, and the system must decide which replica executes computation. The routing policy should balance two objectives: minimizing cross-device communication and balancing computation load. We explore two complementary strategies.

Weighted Round-Robin with Load Prediction. After replication, each duplicated expert has $n_{\text{replica}} + 1$ instances distributed across different GPUs, and routing must decide which instance processes incoming tokens. To guide this decision, we leverage the pre-replication load statistics from Section 4.2 and predict the post-replication computational load of GPUs. Let $W_{\max}$ denote the pre-replication load of the heaviest group and W_{r} the total load of its replicated experts. Assuming this load is evenly split across all $n_{\text{replica}} + 1$ instances, the per-instance load is $W_{\text{p}} = W_{\max}/(n_{\text{replica}} + 1)$. The updated loads are then:

$$W'_{\max} = W_{\max} - W_{\text{r}} + W_{\text{p}}, \quad W'_{\text{i}} = W_{\text{i}} + W_{\text{p}} \tag{3}$$

where W_{i} is the pre-replication load of a replica-hosting GPU. Based on these predictions, routing weights are assigned inversely proportional to the predicted loads, and tokens are dispatched via weighted random round-robin. This alleviates overload on hotspot GPUs by directing more tokens to less loaded GPUs. However, randomness can trigger unnecessary cross-device communication, routing tokens to remote GPUs even when local replicas exist. This limits effectiveness under high concurrency, especially in multi-node scenarios.

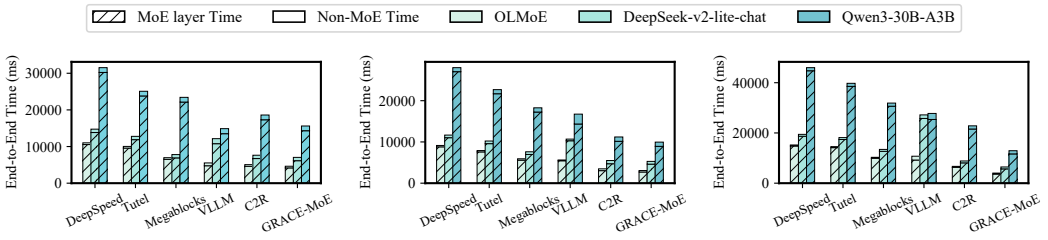
Topology-Aware Routing with Locality Preference. In distributed clusters, communication overhead exhibits a clear hierarchy: intra-GPU communication overhead is cheapest, followed by intra-node communication across GPUs, while inter-node communication is the most expensive. This motivates a routing policy that prioritizes replicas based on locality. The scheme follows a hierarchical locality-first policy: (i) If a replica exists on the same GPU as the token, it is selected. (ii) Otherwise, a replica on another GPU within the same node is chosen. (iii) Only if no intra-node replica is available, the token is routed to a cross-node replica. Within each tier, if multiple replicas are available, weighted round-robin with load prediction is applied to balance computational load. While sacrificing some load balance, it significantly reduces communication overhead, which is the dominant bottleneck in large-scale inference, thereby achieving a practical trade-off between communication and computation. The details of our routing policies are provided in Appendix A.3.

5 Experiments

5.1 Experimental Setup

Models and Datasets. Models: We evaluate **GRACE-MoE** on three representative MoE models, including OLMoE (Muennighoff et al., 2024), DeepSeek-v2-lite-chat (Liu et al., 2024), and Qwen3-30B-A3B (Yang et al., 2025). Datasets: We use WikiText-2-v1 (Merity et al., 2016), MATH (Hendrycks et al., 2021), and the GitHub subset of The Pile (Gao et al., 2020), which together cover text generation, code generation and mathematical reasoning. Table 1 provides additional details of the model architectures.

Baselines and Evaluation Metrics. Baselines: We compare against widely used SMoE inference systems, including DeepSpeed (Rasley et al., 2020), Tutel (Hwang et al., 2023), Megablocks (Gale et al., 2023), vLLM (Kwon et al., 2023), and the placement optimization method C2R (Zhang et al.,



(a) 1 node × 8 GPUs/node (b) 2 nodes × 2 GPUs/node (c) 2 nodes × 4 GPUs/node

Figure 4: **End-to-end inference latency and MoE layer time.** Evaluation of **GRACE-MoE** and all baselines across three models with batch size = 128, prefill length = 64, and decode length = 16.

2025). Metrics: We report communication efficiency, computational load balance, and overall inference efficiency, measured by All-to-All time, communication traffic, GPU idle time, mean per-layer standard deviations of GPU computational load, MoE layer time, and end-to-end inference latency.

Implementation Details. Hardware: Experiments are conducted in a multi-node, multi-GPU environment, with each server equipped with 8 × NVIDIA RTX 3090 (24GB) or 8 × NVIDIA RTX 4090 (48GB). The 3090 GPUs are interconnected through PCIe 1.0 ×16 channels (4 GB/s per direction), whereas the 4090 GPUs use PCIe 4.0 ×16 channels (63 GB/s per direction). Inter-node communication is supported by a 25 Gb Ethernet connection. Software: We implemented **GRACE-MoE** on Megablocks (Gale et al., 2023) with PyTorch 2.5 (Paszke et al., 2019) and Triton 3.0 (Tillet et al., 2019), enabling inference under combined data and expert parallelism. During communication, for tokens routed to multiple experts on the same device are transmitted only a single copy. Within the target device, tokens are distributed hierarchically: duplicated among activated experts if the target is a GPU, or first across GPUs in the node and then within each GPU if the target is a node. Parameters and activations are stored in BFloat16 precision during inference. The workload is configured with batch size 128, prefill length 64, and decode length 16.

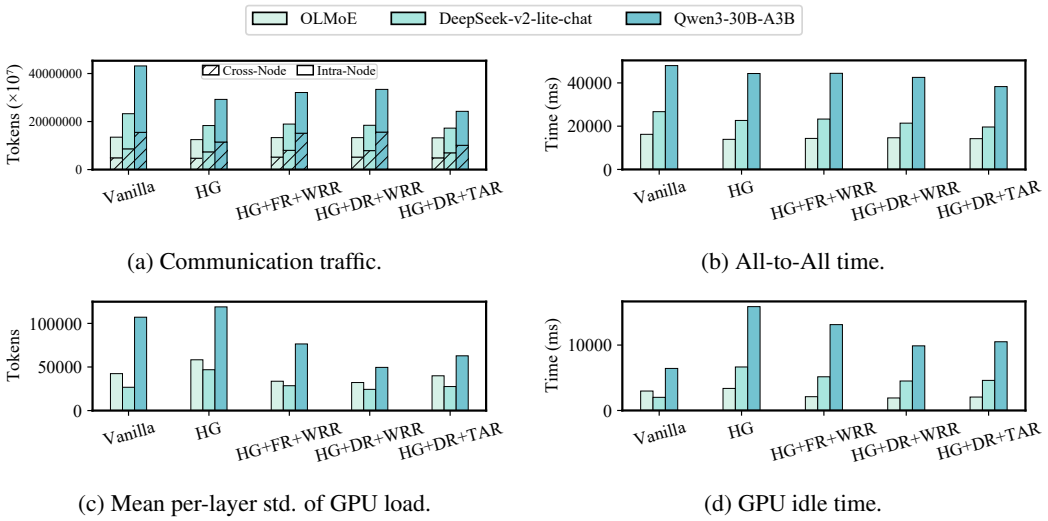
5.2 End-to-End Performance

We evaluate **GRACE-MoE** on the WikiText-2-v1 (Merity et al., 2016) dataset across the three representative MoE models introduced in Section 5.1. Experiments are conducted on one server with 8×RTX 3090 GPUs and two servers with 8×RTX 4090 GPUs each, covering single-node (8 GPUs) and multi-node setups (2 nodes×2 GPUs/node and 2 nodes×4 GPUs/node). As shown in Figure 4, **GRACE-MoE** consistently outperforms all baselines across models and cluster configurations. It reduces MoE layer time by 2.1–75.5%, shortens end-to-end latency up by 33066 ms, and achieves overall speedups of 0.95–3.79×. While gains are evident even in single-node setups, the benefits are especially pronounced in multi-node scenarios. As nodes and GPUs increases, baseline systems suffer steep latency growth due to rising communication overhead. In contrast, **GRACE-MoE** reduces MoE layer time by up to 75.5% and end-to-end latency by up to 73.6%, effectively suppressing this trend and demonstrating strong scalability. Overall, by integrating non-uniform hierarchical grouping, dynamic replication, and locality-aware routing, **GRACE-MoE** jointly optimizes communication overhead and computational load balance, leading to consistently improved end-to-end inference performance.

5.3 Component Analysis

In multi-node, multi-GPU SMoE inference, reducing communication overhead often aggravates computational load imbalance, while mitigating imbalance may increase communication. To study this trade-off, we evaluate the three models from Section 5.1 on a 2 nodes × 2 GPUs/node setup with RTX 4090 GPUs using WikiText-2-v1 (Merity et al., 2016), focusing on communication reduction, computational load balance, and their joint optimization.

Research Question 1: How to Reduce Communication Overhead? In multi-node, multi-GPU scenarios, All-to-All communication overhead often becomes the primary bottleneck. To mitigate



(a) Communication traffic.

(b) All-to-All time.

(c) Mean per-layer std. of GPU load.

(d) GPU idle time.

Figure 5: **Component analysis.** Grouping, replication and routing schemes are compared across three models under a 2 node × 2 GPUs/node setup on the WikiText dataset. Abbreviations: Vanilla (Average Grouping), HG (Hierarchical Grouping), FR (Fixed-Count Replication), DR (Dynamic-Count Replication), WRR (Weighted Round-Robin with Load Prediction), TAR (Topology-Aware Routing with Locality Preference).

it, we propose non-uniform hierarchical grouping. As shown in Figure 5, compared with Vanilla, this scheme reduces All-to-All time by 14.3%, 15.2%, and 7.6% on the three models, while cross-node communication is cut by 3.3%, 14.6%, 26.0% and intra-node cross-GPU traffic by 10.0%, 25.1%, 35.8%. This demonstrates that non-uniform hierarchical grouping effectively captures expert affinity. By co-locating experts with high affinity on the same device, this strategy minimizes cross-device transfers in token dispatching and result aggregation, thereby delivering superior communication efficiency in multi-node, multi-GPU inference.

Research Question 2: How to Mitigate Computational Load Imbalance? As illustrated in Figure 5, the non-uniform hierarchical grouping scheme increases both GPU idle time and computational load imbalance, while achieving the lowest communication overhead. This occurs because high affinity often coincides with frequent co-activation, so clustering such experts overloads the hosting GPU. Although it moderates non-uniformity within nodes and partly alleviates skew, GPU idle time still rises by 0.13×, 2.31×, and 1.47×, and the average per-layer standard deviations of GPU load grows by 37.5%, 75.2%, and 11.1%, leaving some devices persistently underutilized. To address this, we design a dynamic expert replication based on load skew, and a routing policy which dispatch tokens via weighted round-robin with load prediction. Compared with hierarchical grouping without replication, this strategy lowers idle time by 43.3%, 32.3%, 37.8% and reduces average load deviation by 44.8%, 48.0%, 58.3%, significantly improving utilization. For comparison, we also evaluate a simpler fixed-replica scheme that assigns one replica of the overloaded experts in the heaviest group of each layer to the least-loaded GPU, which reduces GPU idle time by 42.2%, 39.1%, 35.7% but yields only limited improvements. Overall, our dynamic scheme achieves the lowest GPU idle time and the best computational load balance by dynamically allocating replica counts according to load skew and routing tokens through load-aware weighted polling, underscoring the effectiveness of adaptive adjustment.

Research Question 3: How to Achieve Joint Optimization of Communication Overhead and Computational Load Balance? Results in Figure 5 demonstrate that dynamic replication with weighted round-robin routing efficiently mitigates computational load imbalance but its randomized nature introduces unnecessary cross-device transfers. On average across the three models, it increases cross-node and intra-node communication by 17.7% and 2.8%, offsetting part of the communication gains from grouping optimization. To overcome this limitation, we propose topology-aware routing with locality preference. Compared with pure weighted round-robin, this strategy reduces All-to-All time by 2.8%, 8.3%, and 10.0%, lowers cross-node communication by 6.3%,

Algorithm 1 Controlled Non-uniform Grouping

Require: Affinity $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, number of groups D, non-uniformity ratio r, number of experts N_e

$E \leftarrow \lfloor N_e / D \rfloor$; $\quad \delta \leftarrow \max(1, \mathrm{round}(E \cdot r))$
$num_{\min} \leftarrow \max(1, E - \delta)$; $\quad num_{\max} \leftarrow E + \delta$
Initialize grouping $L \leftarrow \{L_1, \ldots, L_D\}$ as D empty lists.
Initial clusters $\{C_d\}_{d=1}^{D} \leftarrow$ SPECTRALCLUSTERING$(\boldsymbol{A}, D)$; $\quad \Omega \leftarrow \emptyset$
for $d = 1$ to D **do**
    if $|C_d| > num_{\max}$ **then**
        keep top-$num_{\max}$ by $\sum_{j \in C_d} \boldsymbol{A}_{ij}$ as L_d; push rest to Ω
    else
        $L_d \leftarrow C_d$
    end if
end for
for each $e \in \Omega$ **do**
    $d^\star \leftarrow \arg\max_{d:\, |L_d| < num_{\max}}$ INTRASCORE$(\boldsymbol{A}, L_d \cup \{e\})$
    $L_{d^\star} \leftarrow L_{d^\star} \cup \{e\}$
end for
$need[d] \leftarrow \max(0,\, num_{\min} - |L_d|)$ for $d = 1..D$; $\quad S \leftarrow \sum_{d=1}^{D} need[d]$
if $S > 0$ **then**
    Collect weakest affinity experts from $|L_d| > num_{\min}$
    Reassign them to needy groups maximizing INTRASCORE
end if
return $\{L_d\}_{d=1}^{D}$

11.6%, and 35.2%, and decreases intra-node cross-GPU communication average by 6.7%, while GPU idle time and average load deviation rise only marginally by 5.25% and 21.4% on average. This topology-aware scheme prioritizes local replicas on the same GPU as the token, and resorts to weighted round-robin with load prediction only when no local replica exists. By reducing cross-device communication while maintaining reasonable computational load balance, it achieves a more favorable trade-off. Although absolute load uniformity is slightly compromised, communication overhead remains the dominant bottleneck in practical multi-node, multi-GPU inference, making this trade-off well suited for high-performance inference. As shown in Figure 4, under the 2-node $\times$ 2-GPU/node setup, the proposed strategy reduces end-to-end latency by up to 66.5%, 54.7%, and 64.5% across the three models, achieving maximum speedups of 2.98 $\times$, 2.21 $\times$, and 2.81 $\times$, respectively, compared to baselines.

6 CONCLUSION

This paper introduces **GRACE-MoE**, a framework designed to tackle the dual challenges of communication overhead and computational load imbalance in distributed SMoE inference. Its core components include non-uniform hierarchical grouping based on affinity, dynamic replication driven by load skew, and routing strategies combining topology awareness with load prediction. Without compromising model accuracy, it improves end-to-end inference efficiency and demonstrates strong scalability, offering a practical solution for large-scale SMoE deployment.

While **GRACE-MoE** is effective, two directions remain for future work. First, its reliance on expert replication increases memory demands, which may limit deployment on memory-constrained GPUs. Techniques such as model splitting or expert sharing could avoid this. Second, our evaluation has been confined to academic hardware setups and moderate models. Extending **GRACE-MoE** to larger models and industrial-scale clusters will further validate its efficiency and scalability.

7 ETHICS STATEMENT

This work does not involve human subjects, sensitive data, or other ethical issues. To the best of our knowledge, it does not raise any concerns as outlined in the ICLR Code of Ethics.

8 Reproducibility Statement

We have taken measures to ensure the reproducibility of our results. Section 4 of the main paper provides a detailed description of **GRACE-MoE** and the design of each module. Section 5.1 further specifies the models, datasets, baselines, evaluation metrics, hardware and software environment used in our experiments, as well as the implementation details of the framework. Additional information, including model architecture details and the corresponding GPU hardware configurations, is provided in Appendix A.4. The information included in the paper and supplementary materials should be sufficient for an independent researcher to reproduce our work. We plan to make the full source code publicly available to facilitate reproducibility.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International conference on machine learning*, pp. 4057–4086. PMLR, 2022.

Zhiyuan Fang, Yuegui Huang, Zicong Hong, Yufeng Lyu, Wuhui Chen, Yue Yu, Fan Yu, and Zibin Zheng. Klotski: Efficient mixture-of-expert inference via expert-aware multi-batch pipeline. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 574–588, 2025.

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.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Seokjin Go and Divya Mahajan. Moetuner: Optimized mixture of expert serving with balanced expert placement and token routing. *arXiv preprint arXiv:2502.06643*, 2025.

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*, pp. 120–134, 2022.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 5:269–287, 2023.

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. IEEE, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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*, pp. 611–626, 2023.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pp. 6265–6274. PMLR, 2021.

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, 2023a.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy. *arXiv preprint arXiv:2310.01334*, 2023b.

Yan Li, Pengfei Zheng, Shuang Chen, Zewei Xu, Yuanhao Lai, Yunfei Du, and Zhengang Wang. Speculative moe: Communication efficient parallel moe inference with speculative token and expert pre-scheduling. *arXiv preprint arXiv:2503.04398*, 2025.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Juncai Liu, Jessie Hui Wang, and Yimin Jiang. Janus: A unified distributed training framework for sparse mixture-of-experts models. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pp. 486–498, 2023.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

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. *Proceedings of the ACM on Management of Data*, 1(1):1–19, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3505–3506, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Liang Shen, Zhihua Wu, WeiBao Gong, Hongxiang Hao, Yangfan Bai, HuaChao Wu, Xinxuan Wu, Jiang Bian, Haoyi Xiong, Dianhai Yu, et al. Se-moe: A scalable and efficient mixture-of-experts distributed training and inference system. *arXiv e-prints*, pp. arXiv–2205, 2022.

Athinagoras Skiadopoulos, Mark Zhao, Swapnil Gandhi, Thomas Norrie, Shrijeet Mukherjee, and Christos Kozyrakis. Accelerating mixture-of-experts training with adaptive expert replication. *arXiv preprint arXiv:2504.19925*, 2025.

Jiashun Suo, Xiaojian Liao, Limin Xiao, Li Ruan, Jinquan Wang, Xiao Su, and Zhisheng Huo. Coserve: Efficient collaboration-of-experts (coe) model inference with limited memory. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 178–191, 2025.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wei Wang, Zhiquan Lai, Shengwei Li, Weijie Liu, Keshi Ge, Yujie Liu, Ao Shen, and Dongsheng Li. Prophet: Fine-grained load balancing for parallel training of large-scale moe models. In *2023 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 82–94. IEEE, 2023.

Yuanxin Wei, Jiangsu Du, Jiazhi Jiang, Xiao Shi, Xianwei Zhang, Dan Huang, Nong Xiao, and Yutong Lu. Aptmoe: Affinity-aware pipeline tuning for moe models on bandwidth-constrained gpu nodes. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14. IEEE, 2024.

Yongji Wu, Wenjie Qu, Tianyang Tao, Zhuang Wang, Wei Bai, Zhuohao Li, Yuan Tian, Jiaheng Zhang, Matthew Lentz, and Danyang Zhuo. Lazarus: Resilient and elastic training of mixture-of-experts models with adaptive expert placement. *arXiv preprint arXiv:2407.04656*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jinghan Yao, Quentin Anthony, Aamir Shafi, Hari Subramoni, and Dhabaleswar K DK Panda. Exploiting inter-layer expert affinity for accelerating mixture-of-experts model inference. In *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 915–925. IEEE, 2024.

Dianhai Yu, Liang Shen, Hongxiang Hao, Weibao Gong, Huachao Wu, Jiang Bian, Lirong Dai, and Haoyi Xiong. Moesys: A distributed and efficient mixture-of-experts training and inference system for internet services. *IEEE Transactions on Services Computing*, 17(5):2626–2639, 2024.

Yan Zeng, Chengchuang Huang, Yipeng Mei, Lifu Zhang, Teng Su, Wei Ye, Wenqi Shi, and Shengnan Wang. Efficientmoe: Optimizing mixture-of-experts model training with adaptive load balance. *IEEE Transactions on Parallel and Distributed Systems*, 2025.

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, 2023.

Mohan Zhang, Pingzhi Li, Jie Peng, Mufan Qiu, and Tianlong Chen. Advancing MoE efficiency: A collaboration-constrained routing (C2R) strategy for better expert parallelism design. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6815–6825, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.347. URL https://aclanthology.org/2025.naacl-long.347/.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37: 62557–62583, 2024.

A APPENDIX

A.1 LLM USAGE STATEMENT

In preparing this paper, a large language model (LLM) was used solely for grammar checking and language polishing. The design and implementation of all algorithms and experiments, as well as the analysis of results and conclusions, were independently conducted and written by the authors.

A.2 ALGORITHM FOR CONTROLLED NON-UNIFORM GROUPING

For completeness, we provide the detailed pseudocode of our grouping scheme. Algorithm 2 defines the intra-group affinity score used to evaluate candidate assignments, while Algorithm 3 gives the full procedure for controlled non-uniform grouping with non-uniformity ratio r.

Algorithm 2 Calculate Affinity Score Intra Group

function INTRASCORE(Affinity $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, list of experts S)
  return $\sum_{i \in S} \sum_{j \in S} \boldsymbol{A}_{ij}$
end function

Algorithm 3 Controlled Non-uniform Grouping

Require: Affinity $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, number of groups D, non-uniformity ratio r, number of experts N_e

$E \leftarrow \lfloor N_e / D \rfloor$; $\quad \delta \leftarrow \max(1, \mathrm{round}(E \cdot r))$
$num_{\min} \leftarrow \max(1, E - \delta)$; $\quad num_{\max} \leftarrow E + \delta$
Initialize grouping $L \leftarrow \{L_1, \ldots, L_D\}$ as D empty lists.
Initial clusters $\{C_d\}_{d=1}^{D} \leftarrow$ SPECTRALCLUSTERING$(\boldsymbol{A}, D)$; $\quad \Omega \leftarrow \emptyset$
for $d = 1$ to D **do**
  if $|C_d| > num_{\max}$ **then**
    keep top-$num_{\max}$ by $\sum_{j \in C_d} \boldsymbol{A}_{ij}$ as L_d; push rest to Ω
  else
    $L_d \leftarrow C_d$
  end if
end for
for each $e \in \Omega$ **do**
  $d^\star \leftarrow \arg\max_{d:\, |L_d| < num_{\max}}$ INTRASCORE$(\boldsymbol{A}, L_d \cup \{e\})$
  $L_{d^\star} \leftarrow L_{d^\star} \cup \{e\}$
end for
$need[d] \leftarrow \max(0,\, num_{\min} - |L_d|)$ for $d = 1..D$; $\quad S \leftarrow \sum_{d=1}^{D} need[d]$
if $S > 0$ **then**
  Collect weakest affinity experts from $|L_d| > num_{\min}$
  Reassign them to needy groups maximizing INTRASCORE
end if
return $\{L_d\}_{d=1}^{D}$

A.3 ALGORITHM FOR TOPOLOGY-AWARE ROUTING WITH LOAD PREDICTION

We present two routing policies for replica assignment. Algorithm 4 specifies the weighted polling strategy, while Algorithm 5 incorporates it into a topology-aware routing policy. Together, these ensure that replicas are selected with minimal cross-device communication overhead while maintaining balanced computational load.

A.4 EXPERIMENT CONFIGURATIONS

Details of the model architectures used for evaluation are summarized in Table 1.

Algorithm 4 Weighted Round-Robin with Load Prediction

function CHOOSEBYPOLLINGWEIGHT(polling_weights)
    gpus ← keys(polling_weights), weights ← values(polling_weights)
    selected_gpu_id ← WeightedRandomChoice(gpus, weights)
    return selected_gpu_id
end function

Algorithm 5 Topology-Aware Routing with Locality Preference

Require: token_gpu_id, token_node_id ▷ GPU and Node ID that the token belongs to
Require: replica_gpus ▷ List of GPU IDs where replicas reside
Require: polling_weights: {gpu_id → weight}
local_gpu_replicas ← $\{ g \in \text{replica\_gpus} \mid g = \text{token\_gpu\_id} \}$
local_node_replicas ← $\{ g \in \text{replica\_gpus} \mid \text{Node}(g) = \text{token\_node\_id} \}$
if local_gpu_replicas $\neq \emptyset$ **then**
    return token_gpu_id
else if local_node_replicas $\neq \emptyset$ **then**
    local_weights ← polling_weights restricted to local_node_replicas
    return CHOOSEBYPOLLINGWEIGHT(local_weights)
else
    return CHOOSEBYPOLLINGWEIGHT(polling_weights)
end if

Table 1: Model architecture details used in experiments.

Model	Top_k	Experts	MoE Layers	Params
OLMoE	8	64	16	6.92B
DeepSeek-v2-lite-chat	6	64	26	15.7B
Qwen3-30B-A3B	8	128	48	30.5B