COBALT: LEVERAGING EXPERT CO-ACTIVATION FOR EFFICIENT DISTRIBUTED MOE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) has increasingly become a mainstream approach for scaling large language models, as it expands model capacity while keeping computation cost nearly constant. Training large-scale MoE models relies on Expert Parallelism (EP), which distributes expert replicas across GPUs and exchanges tokens through all-to-all communication. The efficiency of EP is often constrained by two system bottlenecks: cross-node token transfers are limited by inter-node bandwidth, while skewed expert workloads lead to imbalanced computation across GPUs. Prior work mitigates these bottlenecks based on per-expert workload statistics, but overlooks the fact that experts could share the communication.

In this work, we empirically present the observation that many pairs of experts are frequently co-activated by individual tokens. Motivated by this, we present COBALT, an efficient MoE training framework that leverages expert co-activation to reduce cross-node traffic and workload imbalance. COBALT adopts a two-stage expert layout planner that adapts expert layout to the evolving expert co-activation and workload conditions. It periodically co-locates frequently co-activated experts on the same node to reduce the cross-node communication, and performs per-step intra-node adjustment to rebalance the workloads. Subsequently, we develop a communication-aware task assignment method that routes tokens to fewer remote nodes based on the current expert layout. Experiments on 32 B200 GPUs show that COBALT achieves up to 1.53-2.41$\times$ (1.28-1.89$\times$ on average) of speedup compared to existing MoE training frameworks, while reducing cross-node token traffic by 75.74%-99.26%. Source code is available at this link.

1 INTRODUCTION

Mixture-of-Experts (MoE) has become the de facto architecture for scaling large language models (LLMs). By routing each token to a small subset of experts, it increases the model scale while keeping per-token computation nearly constant (Shazeer et al., 2017; Fedus et al., 2022). Currently, frontier open-source LLMs have been scaled to hundreds of billions or trillions of parameters through the MoE architecture (DeepSeek-AI, 2026; GLM-5 Team, 2026; Moonshot AI, 2026).

In a nutshell, for each token in an input sequence, the MoE architecture uses a gating module to select K experts for computation, and merges the results from these experts. Expert parallelism (EP) (Lepikhin et al., 2021; Fedus et al., 2022) is the standard way for the distributed training of large MoE models. As depicted in Figure 1, EP distributes experts across GPU ranks to amortize the memory and computation. Since the experts may be selected by different numbers of tokens, a common practice is to replicate experts with high workloads, while the placement of all experts' replicas jointly affects the workload balance across GPU ranks. Given each token's top-K expert selection, the task assignment process chooses one replica for each token-expert computation task. Then, EP uses all-to-all communication to dispatch tokens to their assigned replicas for computation and triggers another all-to-all communication to combine the results.

However, all-to-all communication of EP can dominate training time as it spans more GPUs. For example with Figure 2, our profiling results show that MoE communication accounts for 58%-83% of the training time across three models, and this portion increases as the EP degree grows. Similar results have also been reported in prior works (Zhai et al., 2023; Liu et al., 2026). Such a high communication cost can be largely attributed to the costly cross-node token transfers due to the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

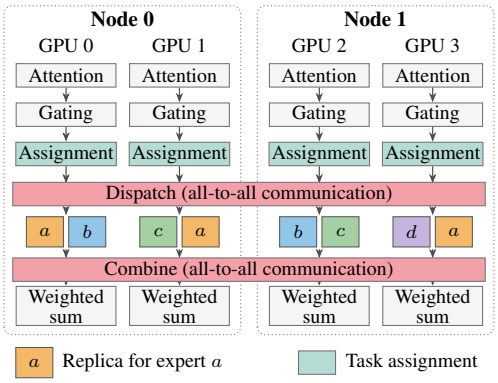

Figure 1: Illustration of expert parallelism and replication. In the example, experts a, b, c, d have 3, 2, 2, and 1 replica(s), respectively.

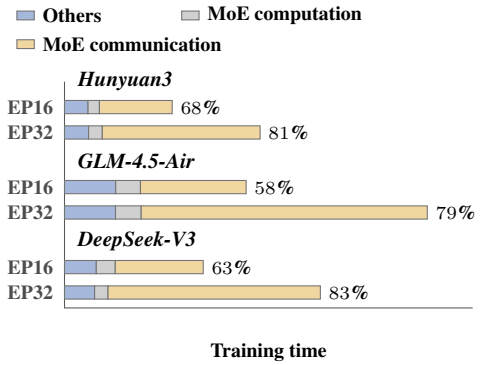

Figure 2: Time breakdown of training three MoE models with EP degrees of 16 and 32. The value next to each bar indicates the portion of MoE communication in training time.

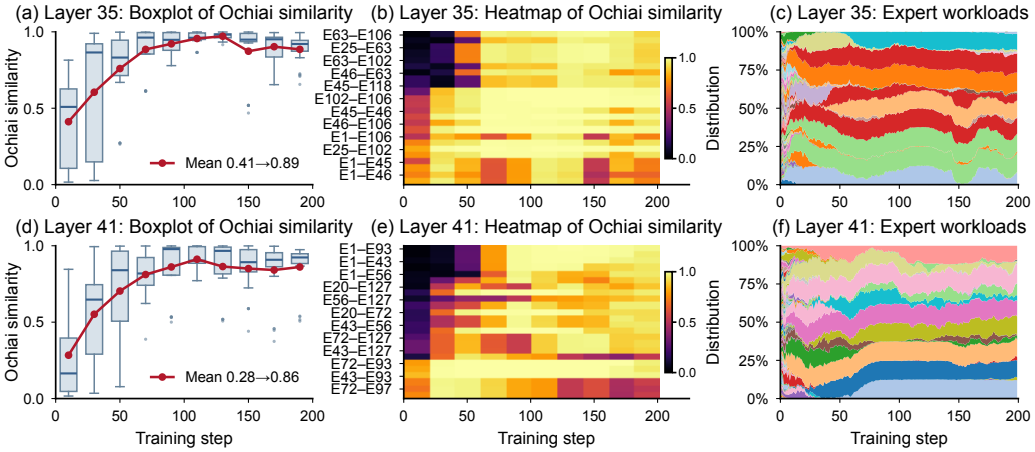

Figure 3: Observations when training GLM-4.5-Air. Statistics are collected on two model layers. (a)&(d) Ochiai similarity generally increases as training progresses. (b)&(e) The expert co-activation condition of individual expert pairs changes during training. (c)&(f) Distribution of expert workloads w.r.t. training steps.

bandwidth hierarchy. For instance, an NVIDIA B200 GPU server provides up to 1.8 TB/s of NVLink bandwidth for GPUs within the same node, whereas the inter-node InfiniBand connection provides at most 400 GB/s of bandwidth in aggregate (NVIDIA, 2026). This gap has driven communication libraries like DeepEP (Zhao et al., 2025) to deduplicate token transfers at the node level. Specifically, when two selected experts reside on different GPUs of the same remote node, the token is only sent across the inter-node connection once, and is forwarded via the intra-node link. Consequently, cross-node traffic depends on the number of distinct remote nodes selected by a token.

Although there exist various efforts to improve MoE communication and workload balance (He et al., 2022; Zhai et al., 2023; Liu et al., 2026; Nie et al., 2023), they are mainly driven by per-expert workload statistics. However, with node-level token transfer deduplication, an expert's workload alone cannot indicate whether it could share the cross-node communication with another expert. Particularly, the potential savings of cross-node traffic depend on whether two experts are simultaneously activated by many tokens. To investigate whether such potential savings can be realized, we analyze the expert selections from the perspective of expert pairs, and observe the *expert co-activation* phenomenon. Specifically, during the training of GLM-4.5-Air (GLM-4.5 Team, 2025), we track the set of active tokens for each expert in each step, and compute the Ochiai similarity[1] of every expert pair that has co-activation. Figure 3(a)(d) visualizes two layers' growth in expert co-activation: the average

[1]Given two sets A, B, their Ochiai similarity is $|A \cap B|/\sqrt{|A||B|}$, which is widely used to measure the similarity between sets. In our observation, a high Ochiai similarity of an expert pair indicates that the two experts are co-activated by many tokens.

Ochiai similarity rises from 0.41 to 0.89 and from 0.28 to 0.86, respectively. It indicates that expert co-activation exists in many expert pairs, and occurs more frequently as training progresses.

Inspired by this, we aim to leverage expert co-activation to accelerate expert parallelism training. However, doing so faces two essential challenges. First, as shown in Figure 3(c)(f), different experts vary substantially in terms of their workloads (i.e., the number of token-expert computation tasks). Simply placing frequently co-activated experts on one node may concentrate token-expert tasks there, causing stragglers. Thus, it calls for reducing cross-node traffic while maintaining workload balance. Second, the co-activated expert pairs change during training, as depicted in Figure 3(b)(e), and so does each expert's workload (Figure 3(c)(f)). As a result, we need to adaptively adjust the expert layout (including expert replication and replica placement) according to the evolving conditions, while ensuring the overhead does not harm overall training efficiency, which is unexplored yet.

To address these challenges, we present COBALT, an efficient distributed MoE training framework that utilizes the expert co-activation information to reduce cross-node communication while taking workload balance into account.

- To begin with, we formulate the joint optimization problem, which aims to deduce the expert layout and token-expert task assignment to minimize the communication cost of cross-node token transfers under the workload balance constraint.
- Subsequently, we dissect the problem to determine expert layout and task assignment accordingly. For expert layout, we propose a two-stage planner, which periodically updates the layout globally based on historical expert workloads and co-activation information, and performs per-step intra-node rebalancing to adjust the expert replicas within each node. Such a two-stage design adapts the expert layout to the evolving co-activation statistics while limiting the overhead. For task assignment, we develop a communication-aware task assignment method, which favors fewer remote nodes for each token given its top-K expert selections and the current expert layout.
- We implement COBALT and conduct experiments on 32 NVIDIA B200 GPUs. Empirical results show that COBALT outperforms existing works by up to 1.53-2.41$\times$ (1.28-1.89$\times$ on average), and shrinks the cross-node token transfers by 75.74%-99.26%.

In summary, we make three major contributions. *(New perspective)* To our knowledge, this is the first work to leverage expert co-activation for cross-node communication reduction and workload balance in distributed MoE training. *(New framework)* We develop COBALT, which features a two-stage expert layout planner and a communication-aware task assignment method to reduce cross-node communication while maintaining workload balance. *(State-of-the-art performance)* Empirical results on 32 NVIDIA B200 GPUs show that COBALT achieves up to 1.53-2.41$\times$ (1.28-1.89$\times$ on average) speedup compared to existing training frameworks.

2 RELATED WORK

Load-aware placement and replication. Existing systems adapt either data placement or expert placement to routing demand. FasterMoE (He et al., 2022), SmartMoE (Zhai et al., 2023), FlexMoE (Nie et al., 2023) and LAER-MoE (Liu et al., 2026) adjust expert replication and replica placement in response to load imbalance. SYMI (Skiadopoulos et al., 2026) and Themis (Qing et al., 2025) address where expert state resides and when it is materialized. These methods are driven by per-expert load statistics, and overlook finer-grained routing statistics. Another line of research exploits such finer-grained statistics. NetMoE (Liu et al., 2025) profiles the expert load of each training sample and rearranges sample placement between layers to optimize cross-node communication. HierMoE (Lin et al., 2026) implements node-level token transfer deduplication, and swaps expert positions to minimize the cross-node communication time. Occult (Luo et al., 2025) clusters experts with similar workloads onto one GPU rank via one-shot offline profiling, and then keeps the placement fixed. These methods reduce communication cost, but do not address the computation imbalance across GPUs. In contrast, COBALT exploits expert co-activation while accounting for workload balance across GPUs.

MoE gating and specialization. MoE architectures and gating modules shape which experts a token activates: DeepSeekMoE (Dai et al., 2024), Mixtral (Jiang et al., 2024a), and OLMoE (Muennighoff et al., 2025) develop expert specialization and gating designs; BASE (Lewis et al., 2021) and Expert

Choice (Zhou et al., 2022) balance expert selections. TA-MoE (Chen et al., 2022) and LocMoE (Li et al., 2024) introduce topology or locality into gating. Occult (Luo et al., 2025) changes the selected experts through modified fine-tuning. COBALT adapts expert layout and task assignment during training without changing the gating module's selected experts or weights.

Communication and kernel optimization. HierMoE (Lin et al., 2026) reduces redundant token transfers through hierarchical deduplication and uses expert swaps to balance communication load. DeepEP (Zhao et al., 2025) likewise supports node-level token deduplication. Lina (Li et al., 2023), Lancet (Jiang et al., 2024b), Comet (Zhang et al., 2025), and FLUX (Chang et al., 2024) schedule or overlap communication with computation, while FastMoE (He et al., 2021), Tutel (Hwang et al., 2023), MegaBlocks (Gale et al., 2023), and PIT (Zheng et al., 2023) improve MoE execution and sparse computation. COBALT builds on node-level token deduplication, and is orthogonal to the scheduling and kernel optimizations above.

Inference scheduling. Semantic Parallelism (Li et al., 2026) groups experts according to their activation patterns and schedules requests for MoE inference. Libra (Yang et al., 2026) predicts expert activations to hint replication and token assignment. These systems exploit activation patterns for inference placement and scheduling. COBALT targets training, where model updates can change co-activation patterns and expert replication additionally requires correct gradient aggregation, optimizer updates, and parameter consistency.

3 METHOD

As discussed in Section 1, there exists the expert co-activation phenomenon during the training of MoE models. COBALT leverages this to reduce cross-node communication while balancing computation across GPUs. Table 4 summarizes the frequently used notations.

3.1 PROBLEM FORMULATION AND DESIGN OVERVIEW

System decisions. We consider one MoE layer with E experts distributed over G GPU ranks on N nodes, with G/N ranks per node. For each token t, the set of K selected experts is denoted as $\mathcal{D}_t$. COBALT preserves these selections, and determines where the resulting token–expert computation tasks execute. Specifically, there are two decisions to make. *(1) Expert layout* comprises the *expert replication* that determines how many replicas each expert has, and the *replica placement* that determines the GPU rank holding each replica. *(2) Task assignment* determines the GPU rank that each token–expert computation task is assigned to. In a nutshell, the expert layout defines the available ranks for each task, and task assignment selects one of those ranks. Thus, they jointly affect communication and load balance: sending a token's tasks to fewer nodes saves network transmission but concentrates workload on these nodes, whereas distributing them facilitates workload balance but causes higher cross-node communication.

Figure 4: Frequently used notations.

E	Number of experts in an MoE layer.
K	Number of experts selected per token.
G	Number of ranks for EP (i.e., EP degree).
N	Number of nodes for EP.
$\mathtt{N}(r)$	Node containing rank r.
T	The number of tokens per training step.
s_t	Source rank of token t.
$\mathcal{D}_t$	The set of selected experts of token t.
$\mathcal{P}_r$	The set of experts with a replica on rank r.
r_{te}	The rank assigned token-expert task (t, e).
w_{er}	Workload of expert e's replica on rank r.
ρ	Budget to control the maximum number of replicas per rank.
ε	Tolerance factor of workload imbalance.

Problem formulation. Formally, the expert layout is represented by $\mathcal{P} = \{\mathcal{P}_r\}$, where $\mathcal{P}_r$ is the set of experts who have a replica on rank r. These sets describe both expert replication and replica placement: expert e has $\sum_r \mathbb{I}[e \in \mathcal{P}_r]$ replicas in total. To avoid out-of-memory errors, we enforce a replica budget $\rho \geq 0$, which allows at most $(1+\rho)E/G$ replica slots per rank.[2] For task assignment, we denote r_{te} as the rank that executes task (t, e), where $e \in \mathcal{D}_t$ and $e \in \mathcal{P}_{r_{te}}$. We formulate the problem as searching for the expert layout and task assignment that minimize communication while taking workload imbalance into account.

[2] ρ controls the number of replica slots per rank. Every token–expert computation task executes once, so a higher ρ (more replica slots) allows an expert's work to be divided across ranks while consuming more memory.

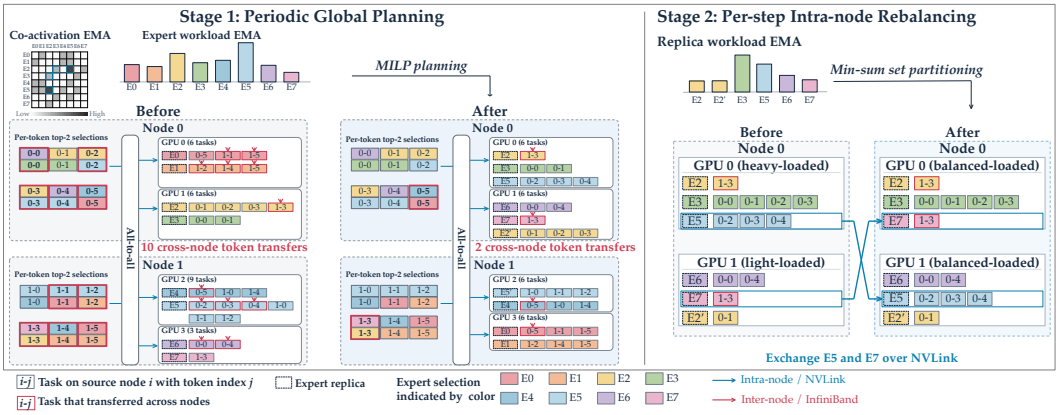

Figure 5: Overview of COBALT's two-stage expert layout planning.

For communication, we focus on cross-node communication cost as inter-node bandwidth is smaller than intra-node bandwidth. It is common practice to deduplicate token transmission at the node level (Zhao et al., 2025): if two ranks on the same node need to receive the same token, then it is done by one cross-node transmission followed by an intra-node transmission (a.k.a. intra-node forwarding). Thus, the communication cost can be modeled as the total number of cross-node token transfers, i.e.,

$$V = \sum_{t=1}^{T} \left| \bigcup_{e \in \mathcal{D}_t} \{\mathtt{N}(r_{te})\} \setminus \{\mathtt{N}(s_t)\} \right|, \tag{1}$$

where T is the number of tokens, $\mathtt{N}(r)$ denotes the node that rank r resides, $\bigcup_{e \in \mathcal{D}_t} \{\mathtt{N}(r_{te})\}$ is the set of destination nodes for token t, and s_t is the source rank for token t.

For workload imbalance, we model the workload of the replica of expert e on rank r as the number of tasks assigned to it, i.e., $w_{er} = \sum_{t:e \in \mathcal{D}_t} \mathbb{I}[r_{te} = r]$. Since every task executes once, we have $\sum_r \sum_e w_{er} = KT$, and the average per-rank workload is KT/G. To restrict workload imbalance, we bound each rank's load by $(1+\varepsilon)$ times the average, where $\varepsilon \geq 0$ controls the allowed imbalance.

Putting them together, we formulate the following problem:

$$\begin{aligned} \arg\min_{\{\mathcal{P}_r\},\{r_{te}\}} \quad & V \\ \text{s.t.} \quad & \textstyle\sum_r \mathbb{I}[e \in \mathcal{P}_r] \geq 1,\ \forall e, \qquad |\mathcal{P}_r| \leq (1+\rho)E/G,\ \forall r, \\ & e \in \mathcal{P}_{r_{te}},\ \forall t,\ e \in \mathcal{D}_t, \qquad \textstyle\sum_{e \in \mathcal{P}_r} w_{er} \leq (1+\varepsilon)KT/G,\ \forall r. \end{aligned} \tag{2}$$

The first two constraints ensure every expert has at least one replica while the number of replicas per rank does not exceed the budget. The third constraint ensures that each task is assigned to an available expert replica, and the last constraint is for workload balance.

Design overview. Solving Equation (2) gives us the optimal expert layout and task assignment given the top-K selections of one training step. However, each step's top-K selections become available only after gating. Computing and applying a new expert layout for every step would require frequent cross-node transfers of expert parameters. We therefore separate expert layout planning from task assignment. First, the *two-stage expert layout planner* (Section 3.2) uses historical statistics for periodic global updates and rebalances GPUs within each node at every training step. Then, the *communication-aware task assignment* (Section 3.3) then assigns the current tasks over the current expert layout, balancing GPU loads while favoring shared token transfers. Lastly, Section 3.4 describes how these decisions are applied during training.

3.2 TWO-STAGE EXPERT LAYOUT PLANNER

As mentioned in Section 1, the co-activation evolves during training, which indicates that the optimal expert layout is not fixed. However, adjusting the expert layout globally requires moving parameters across the inter-node links, incurring high communication cost. In contrast, exchanging replicas between ranks on the same node can utilize the faster intra-node links. This motivates us to design a two-stage expert layout planner. As depicted in Figure 5, it periodically updates both expert replication and replica placement to re-new the expert layout across all nodes, and adjusts replica placement within each node for each training step. By doing so, we can limit cross-node expert parameter transfer overhead while allowing per-step load rebalancing.

Stage 1: Periodic global planning. To adapt to the changes in co-activation, we periodically re-plan the expert layout (including expert replication and replica placement) while taking the expected workload balance into account. To capture the workload assigned to each expert as well as the co-activation phenomenon, the planner uses two statistics from previous training steps: $\hat{w}_e$ is the exponential moving average (EMA) of expert e's assigned computation task count. $\hat{c}_{ij}$ is the EMA of the number of tokens selecting both experts i and j, representing how frequently they are co-activated.

Since we do not have the per-token information (i.e., top-K selections $\mathcal{D}_t$ and source rank s_t) for periodic planning, we transform Equation (2) for expert layout planning. For one thing, we introduce auxiliary decision variables w'_{er} to denote the expected workload assigned to the replica of expert e on rank r. For another, we turn the objective from minimizing cross-node communication to maximizing the total co-activation. Based on these, we re-formulate the following problem:

$$\begin{aligned} \arg\max_{\{\mathcal{P}_r\},\,\{w'_{er}\}} \quad & \textstyle\sum_n \sum_{i<j} \hat{c}_{ij}\, \mathbb{I}[i \in \mathcal{P}^{(n)} \wedge j \in \mathcal{P}^{(n)}] & \\ \text{s.t.} \quad & \textstyle\sum_r \mathbb{I}[e \in \mathcal{P}_r] \geq 1,\ \forall e, & |\mathcal{P}_r| \leq (1+\rho)E/G,\ \forall r \\ & \textstyle\sum_r w'_{er} = \hat{w}_e,\ \forall e & w'_{er} = 0 \quad \text{if } e \notin \mathcal{P}_r, \\ & \textstyle\sum_{e \in \mathcal{P}_r} w'_{er} \leq (1+\varepsilon)KT/G,\ \forall r & \end{aligned} \tag{3}$$

Here, $\mathcal{P}^{(n)} = \cup_{r:\mathtt{N}(r)=n} \mathcal{P}_r$ denotes the set of experts that have replica(s) residing on node n. The first two constraints are the same as Equation (2). The third and fourth constraints ensure all computation tasks are properly assigned. And the last constraint prevents workload imbalance.

Equation (3) can be re-written as a mixed-integer linear programming (MILP) problem. (Due to the space constraint, we leave the detailed MILP formulation in Appendix B.) Therefore, it can be efficiently solved via existing libraries like SCIP (Achterberg, 2009). Moreover, the solving can be overlapped with the training process. The solution of Equation (3) consists of the expert layout ($\{\mathcal{P}_r\}$) and the expected workload division across ranks ($\{w'_{er}\}$). The former will be used to update the expert layout, while the latter will be utilized in the subsequent stage.

Stage 2: Per-step intra-node rebalancing. Between global planning, COBALT supports adjusting replica placement within each node to achieve load rebalancing. Similarly, we estimate each replica's workload by tracking the EMA of the number of token-expert computation tasks executed by that replica. Unlike global planning, the EMA here is at the replica level rather than expert level, and is initialized from the solution of periodic planning (i.e., w'_{er} of Equation (3)) and updated after each training step. Subsequently, for each node, we need to partition the replicas among the ranks, while minimizing the highest workload among the ranks.

Undoubtedly, this is a variant of the min-sum set partitioning problem, with two additional constraints that two replicas of the same expert cannot reside on one rank, and that each rank contains at most $(1+\rho)E/G$ replicas. We solve this problem via a greedy bin-packing approach, which sorts the replicas by the EMA workloads and partitions them into sets greedily. Although it may not achieve the optimal replica placement, such a heuristic approach is extremely fast and can be overlapped with the model computation (detailed in Section 3.4), which is necessary since it is done per-step.

3.3 COMMUNICATION-AWARE TASK ASSIGNMENT

Ideally speaking, we could keep the expert layout ($\{\mathcal{P}_r\}$) fixed and solve Equation (2) to deduce the current batch's task assignments ($\{r_{te}\}$). However, the top-K selections become available only after gating, placing this solving on the critical path. Moreover, even with a fixed layout, the problem retains KT decision variables, which makes the solving prohibitively time-consuming. Therefore, we seek a heuristic task assignment approach that reduces communication based on the current expert layout, rather than directly solving Equation (2) for every training step.

Our key observation is that a token's cross-node communication cost depends on its destination nodes, while choosing ranks within those nodes does not change the number of cross-node input transfers. We assign tasks in two steps: first selecting the fewest remote nodes needed to serve each token, and then sampling a replica within each assigned node. The routine is shown in Algorithm 1.

Step 1: Selecting execution nodes. Let $\mathcal{R}_{e,n} = \{r \in \mathcal{R}_n : e \in \mathcal{P}_r\}$ be the ranks holding expert e on node n, and let $\mathcal{N}_e = \{n : \mathcal{R}_{e,n} \neq \varnothing\}$ be the nodes holding expert e. For token t, selected experts that are currently on the source node will be executed there, requiring no cross-node input transfer.

The remaining experts form $\mathcal{U}_t = \{e \in \mathcal{D}_t : \mathtt{N}(s_t) \notin \mathcal{N}_e\}$ (line 3). To minimize the cross-node communication, we only need to solve a minimum set-cover problem, i.e., finding the smallest set of remote nodes that covers the remaining experts. We solve this problem by enumerating node subsets with bitmasks. Specifically, for each $e \in \mathcal{U}_t$, an $(N-1)$-bit mask $m_{t,e}$ is able to mark all remote nodes holding replicas of expert e (line 4). A candidate mask b is feasible exactly when $b \,\&\, m_{t,e} \neq 0, \forall e \in \mathcal{U}_t$, and the feasible mask with the smallest population count is the best solution to the minimum set-cover problem (lines 5-8). Finally, for each $e \in \mathcal{U}_t$, we randomly select an execution node n_{te} from the nodes indicated by the best feasible mask (lines 9-10).

Step 2: Selecting execution ranks. For each task (t, e), denote n_{te} as its execution node selected in Step 1. We uniformly sample an execution rank r_{te} from $\mathcal{R}_{e,n_{te}}$ (lines 14-15). Thus, replicas of the same expert within a node receive equal workloads in expectation. This exactly matches our per-step layout adjustment: intra-node rebalancing distributes the replica workloads across ranks, while this sampling distributes incoming tasks among the available replicas.

Algorithm 1 Communication-aware task assignment.

Input: Per-token selected experts $\mathcal{D}_t$ and source rank s_t, per-expert residing ranks $\mathcal{R}_{e,n}$ and nodes $\mathcal{N}_e$
Output: Execution rank r_{te} for each task (t, e)

1: **for** each token t **in parallel do**
2: *// Step 1: Select execution nodes.*
3: $\mathcal{U}_t \leftarrow \{e \in \mathcal{D}_t : \mathtt{N}(s_t) \notin \mathcal{N}_e\}$
4: $m_{t,e} \leftarrow \mathrm{RemoteMask}(\mathcal{N}_e, \mathtt{N}(s_t)), \forall e \in \mathcal{U}_t$
5: $b^\star \leftarrow 2^{N-1} - 1$ *// Start from all remote nodes*
6: **for** $b = 0, 1, \cdots, 2^{N-1} - 2$ **do**
7: **if** $b \& m_{t,e} \neq 0, \forall e \in \mathcal{U}_t$ **and** $\mathrm{BitCount}(b) < \mathrm{BitCount}(b^\star)$ **then**
8: $b^\star \leftarrow b$ *// Fewer remote nodes but feasible*
9: **for** each $e \in \mathcal{U}_t$ **do**
10: $n_{te} \leftarrow \mathrm{RandomNode}(b^\star \,\&\, m_{t,e})$
11: **for** each $e \in \mathcal{D}_t \setminus \mathcal{U}_t$ **do**
12: $n_{te} \leftarrow \mathtt{N}(s_t)$ *// Execute on the local node*
13: *// Step 2: Select execution ranks.*
14: **for** each $e \in \mathcal{D}_t$ **do**
15: $r_{te} \leftarrow \mathrm{UniformChoice}(\mathcal{R}_{e,n_{te}})$
16: **return** $\{r_{te} : e \in \mathcal{D}_t\}$

It is noteworthy that our task assignment does not explicitly enforce a workload balance constraint. However, this constraint is already considered during periodic expert layout planning and per-step intra-node rebalancing (Section 3.2). And the uniform sampling in task assignment maintains the balance.

The complexity of Algorithm 1 is $O(T2^{N-1})$. In practice, the EP degree is not high, so 2^{N-1} can be viewed as constant (e.g., EP degree of $G = 32$ gives $2^{N-1} = 8$ since each node contains 8 GPUs). Additionally, we can parallelize the task assignment for the T tokens. Consequently, the time cost of our task assignment process is negligible.

3.4 Implementation

We implement COBALT on top of PyTorch. To deduplicate token transmission and improve communication efficiency, we adopt DeepEP (Zhao et al., 2025) as the backend for the all-to-all communication. Both the MILP problem (Eq. (3)) of periodic global planning and the min-sum set partitioning problem of per-step intra-node rebalancing are solved on the CPU, whereas we implement a GPU kernel for the task assignment (Algorithm 1) using Triton. By default, we trigger the global expert layout update every 10 training steps. The solving of expert layout is overlapped with model computation, while the global update necessitates pausing the training process and transmitting the expert model parameters of across nodes. However, the overhead is worthwhile given its benefits. For each training step, we perform the intra-node load balancing immediately after parameter prefetching so that the adjustment can be fully overlapped by model computation.

4 Experiments

4.1 Experimental Setup

Setup. To evaluate the effectiveness of COBALT, we conduct experiments on 4 NVIDIA B200 GPU servers (32 GPUs in total) with 1.8 TB/s intra-node NVLink and 400GB/s inter-node InfiniBand connections. We compare COBALT with three baselines. The first is a standard PyTorch implementation that applies EP to the MoE-FFN module and Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023) to the other modules. We refer to this baseline as EP+FSDP. The other two baselines are SmartMoE (Zhai et al., 2023), and LAER-MoE (Liu et al., 2026). To be fair, all four frameworks use DeepEP (Zhao et al., 2025) for the all-to-all communication in EP. For the evaluation workloads, we

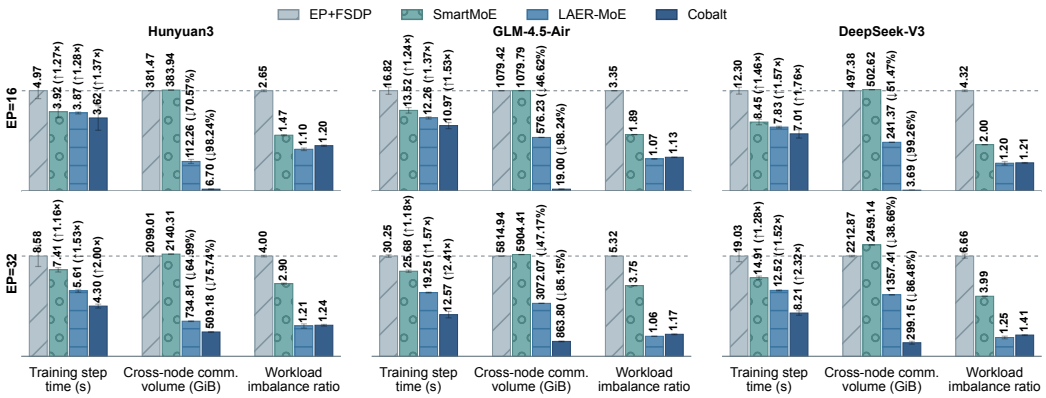

Figure 6: End-to-end comparison. We evaluate with 16 and 32 GPUs, respectively. For training step time, we provide the speedup compared to EP+FSDP on each bar. For cross-node communication volume, we provide the relative reduction compared to EP+FSDP.

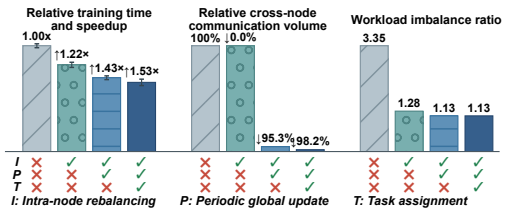

Figure 7: Ablation studies (GLM-4.5-Air, EP=16).

Table 1: Overhead of layout planning and adjustment in one training step (GLM-4.5-Air).

	Planning (seconds)		Adjustment (seconds)	
	Global	Intra-node	Global	Intra-node
EP16	1.21±0.43	0.25±0.12	2.04±0.06	0.03±0.01
EP32	2.61±1.25	0.37±0.18	2.41±0.14	0.05±0.02

consider model architectures from Hunyuan3 (Tencent Hy Team, 2026), GLM-4.5-Air (GLM-4.5 Team, 2025), and DeepSeek-V3 (DeepSeek-AI, 2024). Due to the memory constraint, we restrict the number of layers to be 16, 45, and 12 for the three models, comprising 60B, 106B, and 140B model parameters, respectively. Since our goal is to improve system efficiency, we conduct the training on the C4 dataset (Raffel et al., 2020), which is widely used in LLM training, and set the training sequence length to 8192. In our experiments, all frameworks start from the same checkpoint of each model, which is obtained after 200 steps of training from scratch. By default, we set $\rho = 0.5$ and $\varepsilon = 0.15$. More details of the hardware and workload configurations are provided in Appendix A.

Metrics. We mainly focus on training step time and the corresponding speedup. The training step time is averaged over 50 steps and includes the time cost of expert layout adjustment. We also report two metrics to interpret the efficiency: the cross-node communication volume caused by token transfers (excluding those caused by FSDP and expert layout update), and the workload imbalance ratio (computed as the maximum workload of a single rank divided by the average workload of all ranks). We make five runs for each experiment and report the mean and standard deviation.

4.2 Experimental Results

End-to-end comparison. We first evaluate the end-to-end training efficiency. As shown in Figure 6, Cobalt consistently achieves the best efficiency, outperforming EP+FSDP, SmartMoE, and LAER-MoE by up to 2.41×, 2.04×, and 1.53× (1.89×, 1.52×, and 1.28× on average), respectively. Moreover, it can be observed that the speedup of EP=32 is higher than that of EP=16, which is reasonable because a larger EP size often leads to more cross-node communication.

To interpret the performance, we record the cross-node communication volume and workload imbalance ratio. The gain of Cobalt stems from the reduction in both cross-node communication and workload imbalance. To be specific, Cobalt significantly reduces cross-node communication volume by 75.74-99.26% compared to EP+FSDP and consistently achieves the lowest level of workload imbalance. LAER-MoE mainly optimizes workload balance across ranks but does not address the cross-node communication bottleneck. In contrast, Cobalt accounts for both, reducing cross-node communication volume by 30.71-78.29% compared to LAER-MoE under a comparable level of workload ratio. SmartMoE requires each expert either to have only one replica on a single rank or to be fully replicated across all ranks, so its performance falls behind ours, especially for EP=32.

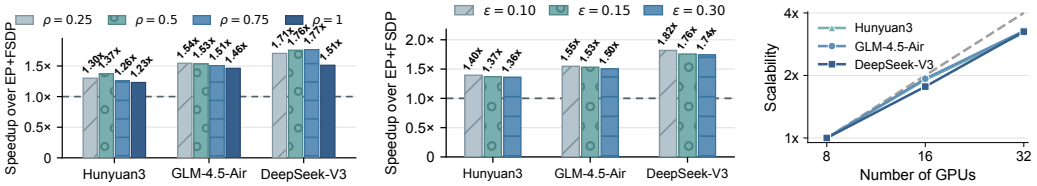

Figure 8: Left & Middle: Sensitivity w.r.t. varying ρ and ε, respectively (GLM-4.5-Air, EP=16). Right: Scalability in terms of training throughput.

Ablation studies. We assess the effectiveness of the proposed techniques in COBALT. The results are provided in Figure 7. To begin with, our intra-node rebalancing contributes to workload balancing. It lowers the imbalance ratio from 3.35 to 1.28, yielding 1.22× of speedup in training time. Then, our periodic global update significantly shrinks the cross-node communication volume by 95.3%, and it also helps decrease imbalance ratio to 1.13, resulting in a speedup of 1.43×. Lastly, enabling our communication-aware task assignment can further reduce the cross-node communication by 98.2%. Putting them together, COBALTachieves a speedup of 1.53× compared to the EP+FSDP baseline.

Overhead of proposed techniques. We measure the overhead of our two-stage expert layout planner in Table 1. In short, the overhead is minor for both planning and adjustment. Intra-node rebalancing takes negligible time and can be overlapped with previous layer's model computation. The global layout planning takes relatively longer time and increases w.r.t. the EP degree, yet it is still minor compared to the training time. Although the global layout adjustment cannot be overlapped with model computation, it is triggered periodically, so the impact on overall training efficiency is limited. In addition, since our communication-aware task assignment is fully parallelized in a GPU kernel, its overhead stays within 1 millisecond, which is omittable.

Sensitivity. There are two hyper-parameters in COBALT: ρ controls the maximum replicas per rank and ε controls the expected workload balance ratio. We evaluate the sensitivity w.r.t. them individually. Figure 8 provides the results with varying ρ and ε. First, COBALT consistently outperforms EP+FSDP with different ρ. When $\rho = 1$, the performance of COBALT drops slightly, which is due to the increased overhead of model gradient synchronization. However, with a moderate ρ between 0.25 and 0.75, COBALT's performance is relatively stable, demonstrating its robustness to ρ. Second, COBALT also has a robust performance w.r.t. ε, showcasing less than 5% of differences in training step time when ε changes. Thus, it does not require laborious hyper-parameter tuning.

Scalability. Finally, we assess the scalability of COBALT by varying the number of GPUs to be 8, 16, and 32. As shown in Figure 8, COBALT shows good scalability, improving the training throughput by 3.20-3.27× when increasing the number of GPUs from 8 to 32. When scaling to more GPUs, it is common to increase the data parallelism degree rather than EP, so we can apply the proposed techniques within each data parallelism group. As a result, expanding to a larger scale would not harm COBALT's effectiveness in reducing the cross-node communication and workload imbalance.

5 CONCLUSION

This work presents COBALT, which leverages expert co-activation to coordinate expert layout and token-expert task assignment to reduce cross-node traffic while balancing GPU workloads. Its two-stage planner uses historical co-activation and workload statistics for periodic global updates, then rebalances replicas within each node at every step. For each token, communication-aware task assignment selects the fewest remote nodes needed to serve its experts and distributes tasks among their replicas. Experimental results with three MoE models on 32 NVIDIA B200 GPUs show that COBALT achieves up to 1.53-2.41× (1.28-1.89× on average) of speedup over existing training frameworks, and substantially reduces cross-node token-transfer volume by 75.74–99.26%.

Limitations and possible future work. Due to the fact that expert selections are only available after gating, the expert layout is solved based on exponential moving average statistics rather than ad hoc statistics, and the task assignment is deduced heuristically. These may lead to sub-optimal solutions. Hence, a possible future direction is to refine the problem solving. Besides, because of the high expense of hardware resources, our evaluation is conducted on 32 GPUs. How COBALT behaves when it is combined with other parallelism strategies on larger clusters is worth exploring.

Reproducibility Statement

Section 3 specifies the cost model, the planner, the execution algorithm, and the framework implementation. Section 4.1 describes the experimental setup, and Appendix A gives the detailed configurations. Our source code is also available at `https://anonymous.4open.science/r/Cobalt`.

AI Use Statement

We used AI tools for language polishing, grammar checking, and code debugging. All research design, technical content, data analysis, interpretations, and conclusions have been independently developed, reviewed, and verified by the authors, who take full responsibility for the accuracy and integrity of the work.

References

Tobias Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. doi: 10.1007/s12532-008-0001-1.

Li-Wen Chang, Wenlei Bao, Qi Hou, Chengquan Jiang, Ningxin Zheng, Yinmin Zhong, Xuanrun Zhang, Zuquan Song, Chengji Yao, Ziheng Jiang, Haibin Lin, Xin Jin, and Xin Liu. FLUX: Fast Software-based Communication Overlap On GPUs Through Kernel Fusion. *arXiv preprint arXiv:2406.06858*, 2024. URL https://arxiv.org/abs/2406.06858.

Chang Chen, Min Li, Zhihua Wu, Dianhai Yu, and Chao Yang. TA-MoE: Topology-aware large scale mixture-of-expert training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22173–22186, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8b465dd58ac50e1b0b22894fd581f62f-Abstract-Conference.html.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. *arXiv preprint arXiv:2401.06066*, 2024. URL https://arxiv.org/abs/2401.06066.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

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 https://www.jmlr.org/papers/v23/21-0998.html.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems*, volume 5, pp. 288–304, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/5a54f79333768effe7e8927bcccffe40-Abstract-mlsys2023.html.

GLM-4.5 Team. GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models. *arXiv preprint arXiv:2508.06471*, 2025. URL https://arxiv.org/abs/2508.06471.

GLM-5 Team. GLM-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026. URL https://arxiv.org/abs/2602.15763.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. FastMoE: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*, 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*, pp. 120–134, 2022. doi: 10.1145/3503221.3508418.

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. In *Proceedings of Machine Learning and Systems*, volume 5, pp. 269–287, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/5616d34cf8ff73942cfd5aa922842556-Abstract-mlsys2023.html.

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. *arXiv preprint arXiv:2401.04088*, 2024a. URL https://arxiv.org/abs/2401.04088.

Chenyu Jiang, Ye Tian, Zhen Jia, Shuai Zheng, Chuan Wu, and Yida Wang. Lancet: Accelerating mixture-of-experts training via whole graph computation-communication overlapping. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 74–86, 2024b. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/339caf45a6fa281cae8adc6465343464-Abstract-Conference.html.

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. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.16668.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE Layers: Simplifying Training of Large, Sparse Models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6265–6274. PMLR, 2021. URL https://proceedings.mlr.press/v139/lewis21a.html.

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, 2023. URL https://www.usenix.org/conference/atc23/presentation/li-jiamin.

Jing Li, Zhijie Sun, Xuan He, Li Zeng, Yi Lin, Entong Li, Binfan Zheng, Rongqian Zhao, and Xin Chen. LocMoE: A low-overhead MoE for large language model training. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 6377–6387, 2024. doi: 10.24963/ijcai.2024/705. URL https://www.ijcai.org/proceedings/2024/705.

Yan Li, Zhenyu Zhang, Zhengang Wang, Pengfei Chen, and Pengfei Zheng. Semantic parallelism: Redefining efficient MoE inference via model-data co-scheduling. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/f0552f14388d95b19740dee809f5cad1-Abstract-Conference.html.

Wenxiang Lin, Xinglin Pan, Lin Zhang, Shaohuai Shi, Xuan Wang, and Xiaowen Chu. HierMoE: Accelerating MoE training with hierarchical token deduplication and expert swap. In *IEEE INFOCOM 2026 – IEEE Conference on Computer Communications*, pp. 1–10, 2026. doi: 10.1109/INFOCOM59046.2026.11571570.

Xinyi Liu, Yujie Wang, Fangcheng Fu, Xupeng Miao, Shenhan Zhu, Xiaonan Nie, and Bin Cui. NetMoE: Accelerating MoE training through dynamic sample placement. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e0c256700465c158de71081b4cf5e8c3-Abstract-Conference.html.

Xinyi Liu, Yujie Wang, Fangcheng Fu, Xuefeng Xiao, Huixia Li, Jiashi Li, and Bin Cui. LAER-MoE: Load-adaptive expert re-layout for efficient mixture-of-experts training. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 1055–1072, 2026. doi: 10.1145/3779212.3790180.

Shuqing Luo, Pingzhi Li, Jie Peng, Yang Zhao, Yu Cao, Yu Cheng, and Tianlong Chen. Occult: Optimizing collaborative communications across experts for accelerated parallel MoE training and inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41235–41253. PMLR, 2025. URL https://proceedings.mlr.press/v267/luo25f.html.

Moonshot AI. Kimi K3: Open frontier intelligence. Official model release, 2026. URL https://github.com/MoonshotAI/Kimi-K3.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannah Hajishirzi. OLMoE: Open Mixture-of-Experts Language Models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/9b224ace8963c9385ad5e2b5c9039b97-Abstract-Conference.html.

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. doi: 10.1145/3588964.

NVIDIA. *NVIDIA DGX B200 User Guide: Introduction to NVIDIA DGX B200 Systems*, 2026. URL https://docs.nvidia.com/dgx/dgxb200-user-guide/introduction-to-dgxb200.html. Accessed September 16, 2026.

Yuhao Qing, Guichao Zhu, Lintian Lei, Fanxin Li, Shixiong Zhao, Zekai Sun, Xiuxian Guan, Xusheng Chen, Dong Huang, Ping Luo, Yiming Qiu, and Heming Cui. Themis: Efficient sparse model training through fully sharded sparse data parallelism. *arXiv preprint arXiv:2502.02581v2*, 2025. URL https://arxiv.org/abs/2502.02581v2.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://www.jmlr.org/papers/v21/20-074.html.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1701.06538.

Athinagoras Skiadopoulos, Mark Zhao, Swapnil Gandhi, Thomas Norrie, Shrijeet Mukherjee, and Christos Kozyrakis. SYMI: Efficient mixture-of-experts training via model and optimizer state decoupling. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*, pp. 75–92, 2026. URL https://www.usenix.org/conference/nsdi26/presentation/skiadopoulos.

Tencent Hy Team. Hy3. Official model release, 2026. URL https://github.com/Tencent-Hunyuan/Hy3.

Jaehoon Yang, Yushin Kim, Seokwon Moon, Yeonhong Park, and Jae W. Lee. Libra: Effective yet efficient load balancing for large-scale MoE inference. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/9ff1ac9a659085fed0735362cafe5e53-Abstract-Conference.html.

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. URL https://www.usenix.org/conference/atc23/presentation/zhai.

Shulai Zhang, Ningxin Zheng, Haibin Lin, Ziheng Jiang, Wenlei Bao, Chengquan Jiang, Qi Hou, Weihao Cui, Size Zheng, Li-Wen Chang, Quan Chen, and Xin Liu. COMET: Fine-grained computation-communication overlapping for mixture-of-experts. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/e27ea0cd50b798ff8942caf9203f0992-Abstract-Conference.html.

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. GitHub, 2025. URL https://github.com/deepseek-ai/DeepEP.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023. URL https://arxiv.org/abs/2304.11277.

Ningxin Zheng, Huiqiang Jiang, Quanlu Zhang, Zhenhua Han, Lingxiao Ma, Yuqing Yang, Fan Yang, Chengruidong Zhang, Lili Qiu, Mao Yang, and Lidong Zhou. PIT: Optimization of Dynamic Sparse Deep Learning Models via Permutation Invariant Transformation. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 331–347, 2023. doi: 10.1145/3600006.3613139. URL https://doi.org/10.1145/3600006.3613139.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. Mixture-of-Experts with Expert Choice Routing. In *Advances in Neural Information Processing Systems*, volume 35, pp. 7103–7114, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/2f00ecd787b432c1d36f3de9800728eb-Abstract-Conference.html.

A More Experimental Details

Hardware environments. All experiments are conducted on 4 GPU servers, with each server containing 8 NVIDIA B200 GPUs, totaling 32 GPUs. Each GPU is equipped with 192 GB of HBM3e memory that provides up to 8 TB/s memory access bandwidth. GPUs within each node are interconnected by NVLink with 1.8 TB/s of bandwidth. For cross-node communication, each node is equipped with 8 ConnectX-7 InfiniBand NICs, providing up to 400 GB/s of aggregate bandwidth.

Additional workload configurations. Table 2 lists more details about the evaluated models, and the default training configurations used in our experiments.

Table 2: Detailed model and training configurations.

Model	Hidden Dim	FFN Dim	Num. Experts	Selected Experts	Attn. heads	Num. Layers	Total Params & Activated Params per Token
Hunyuan3	4096	1536	192	8	64	16	60B-A5B
GLM-4.5-Air	4096	1408	128	8	96	45	106B-A12B
DeepSeek-V3	7168	2048	256	8	128	12	140B-A8B

Configuration	EP16	EP32
Sequence length	8192	8192
Micro-batch size	2	2
Local batch	4	4
Global batch	64	128

B MILP Reformulation

In Section 3.2, we mention that Equation (3) can be re-written as a mixed-integer linear programming (MILP) problem. Below we explain how it is reformulated.

Equation (3) chooses both the expert layout and how to divide each expert's estimated work among its replicas. We express these decisions by introducing auxiliary variables: $x_{er} \in \{0,1\}$ indicates whether rank r holds a replica of expert e, and $l_{er} \geq 0$ represents the work assigned to that replica. Thus, $\mathcal{P}_r = \{e : x_{er} = 1\}$ and l_{er} represents w'_{er} in Equation (3). Let $\mathcal{R}_n = \{r : \mathtt{N}(r) = n\}$ be the ranks on node n. The variable $u_{en} \in [0,1]$ indicates whether node n holds at least one replica of expert e. For each expert pair $i < j$ and node n, the variable $z_{ijn} \in [0,1]$ indicates whether both experts have replicas on that node. The resulting mixed-integer linear programing problem is

$$
\begin{aligned}
\underset{\{x_{er}\},\{u_{en}\},\{z_{ijn}\},\{l_{er}\}}{\arg\max} \quad & \textstyle\sum_n \sum_{i<j} \hat{c}_{ij} z_{ijn} \\
\text{s.t.} \quad & x_{er} \leq u_{en},\ \forall e,\ n,\ r \in \mathcal{R}_n, && u_{en} \leq \textstyle\sum_{r\in\mathcal{R}_n} x_{er},\ \forall e,\ n, \\
& z_{ijn} \leq u_{in}, \quad z_{ijn} \leq u_{jn},\ \forall i<j,\ n, && z_{ijn} \geq u_{in} + u_{jn} - 1,\ \forall i<j,\ n, \\
& x_{er} \in \{0,1\},\ \forall e,\ r, && u_{en} \in [0,1],\ \forall e,\ n, \\
& z_{ijn} \in [0,1],\ \forall i<j,\ n, && \textstyle\sum_r x_{er} \geq 1,\ \forall e, \\
& \textstyle\sum_e x_{er} \leq (1+\rho)E/G,\ \forall r, && \textstyle\sum_r l_{er} = \hat{w}_e,\ \forall e, \\
& 0 \leq l_{er} \leq \hat{w}_e x_{er},\ \forall e,\ r, && \textstyle\sum_e l_{er} \leq (1+\varepsilon)KT/G,\ \forall r,
\end{aligned}
\tag{4}
$$

The first two constraints force $u_{en} = 1$ exactly when node n holds a replica of expert e. The next two constraints then force $z_{ijn} = 1$ exactly when node n holds replicas of both experts. Because x_{er} is binary, both u_{en} and z_{ijn} need only continuous bounds. Therefore, it counts $\hat{c}_{ij}$ once for each node holding both experts, even if their replicas reside on different ranks of that node, exactly as in Equation (3). The remaining constraints enforce expert coverage and the per-rank replica budget, conserve each expert's estimated work, and prevent a rank without its replica from receiving that work. The last workload inequality caps each rank's planned work at the same threshold as Equation (3). Hence, every feasible solution maps to a feasible placement and workload division in the original planning problem, and vice versa, with the same objective value. Only the EG placement variables x_{er} need to be integer. Consequently, Equation (4) is an MILP reformulation of Equation (3).