HybridEP: Scaling Expert Parallelism to Cross-Datacenter Scenario via Hybrid Expert/Data Transmission

Weihao Yang†, Hao Huang†, Donglei Wu‡, Ningke Li†, Yanqi Pan†, Qiyang Zheng†,
Wen Xia†, Shiyi Li† and Qiang Wang†
Harbin institute of Technology, Shenzhen†, Guangzhou University‡

***Abstract*—Mixture-of-Experts (MoE) has become a popular architecture for scaling large models. However, the rapidly growing scale outpaces model training on a single DC, driving a shift toward a more flexible, cross-DC training paradigm. Under this paradigm, Expert Parallelism (EP), a core component of MoE, faces significant scalability issues due to the limited cross-DC bandwidth. Specifically, existing EP optimizations attempt to overlap data communication and computation, which has little benefit in low-bandwidth scenarios due to a much longer data communication time. Therefore, the trends of cross-DC EP scaling is fast becoming a critical roadblock to the continued growth of MoE models.**

To address this urgent challenge, we propose HybridEP, a modeling-guided framework to optimize EP under constrained bandwidth. Our key idea is to dynamically transform the spatial placement of experts to reduce data communication traffic and frequency, thereby minimizing EP's communication overheads. However, it is non-trivial to find the optimal solution because it complicates the original communication pattern by mixing data and expert communication. We therefore build a stream-based model to determine the optimal transmission proportion between experts and data. Guided by this model, we incorporate two techniques to implement HybridEP: (1) domain-based partition to construct the mapping between hybrid patterns and specific communication topology at GPU level, and (2) parameter-efficient migration to further refine this topology by reducing expert transmission overhead and enlarging the domain size. Combining all these designs, HybridEP can be considered as a more general EP with better scalability. Experimental results show that HybridEP outperforms existing state-of-the-art MoE training systems by up to 5.6× under constrained bandwidth. We further compare HybridEP and EP on large-scale simulations. HybridEP achieves up to 1.45× speedup with 1000 DCs under different bandwidths.

I. Introduction

Large language models (LLMs) [9], [10], [15], [25], [27] have achieved significant success in various tasks, such as translation [11], text generation [31], and question answering [56], driving the community to explore even larger model capacities for better performance. Mixture-of-Experts (MoE) has become increasingly popular to enable ultra-scale LLMs. It decouples computation from model size through sparse expert activation, leading to an easy expansion of model parameters to trillions with nearly constant cost [7], [20], [40], [45].

As the scaling law brings about larger pre-training scale, existing training methods in a single data center (DC) face severe challenges [8]. Therefore, recent architectural visions [18], [61] advocate for a more fluid and composable DC infrastructure, where computation, memory, and network resources can be pooled and recombined on demand, forming elastic clusters across smaller, interconnected DCs. Driven by this, some pioneers have achieved forward-looking results across DCs [14], [16] and even countries to scale model size to 10B-32B [24], [49], while maintaining fairly competitive model performance, lower costs, and more flexible resource utilization. This ongoing paradigm shift makes cross-DC training essential to build the Computing Power Network [52] and make computing power affordable to everyone [3]–[5].

Nevertheless, we find that constrained inter-DC bandwidth slows down MoE training, primarily due to Expert Parallelism (EP). EP is the core component of MoE that can account for more than 90% of training time under low bandwidth, as shown in Figure 2(b). Existing EP optimizations [20], [22], [28], [38], [42], [47], [57] target a single high-performance DC with the idea of overlapping computation and communication. However, they are impractical to scale EP across DCs, as it is impossible to fully overlap computation with the much longer communication time. What's worse, scaling EP across DCs becomes inevitable. Recent representative MoE models [12], [13], [17], [26], [39], [55] have demonstrated a rapidly unfolding trend of EP expansion, bringing an explosive growth of the burden for single-DC training. Therefore, efficiently scaling EP is a pressing challenge that must be addressed to sustain the future MoE development.

To this end, we propose HybridEP, a modeling-guided framework that can efficiently scale EP under constrained bandwidth. Our framework tries to answer a more fundamental question: *How can we structurally eliminate EP's communication overheads under constrained bandwidth, beyond simply overlapping or hiding them?* Our key insight is that **EP's overheads can be structurally mitigated by adjusting the spatial placement of experts.** Based on this, HybridEP migrates experts to change their spatial placements, thus altering the communication topology and reducing communication traffic and frequency. However, such a transformation introduces complex hybrid communication patterns between data and experts, and their proportion has a huge impact on the final effect. We therefore build a **Stream-Based Modeling** to decide the best proportion between data and experts. Our model adopts the divide-and-conquer approach and first it decouples MoE training into two distinct streams (i.e., computation and communication) and independently models them. Then it analyzes how the two streams overlap and derives a unified end-to-end performance model.

Following this, we implement HybridEP as a practical

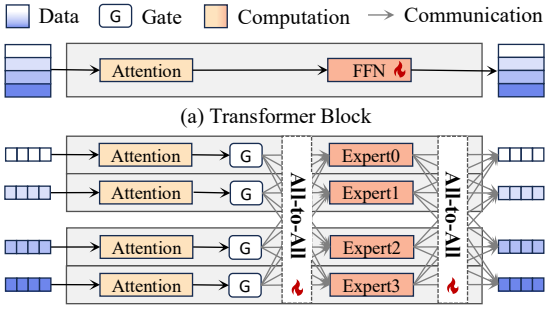
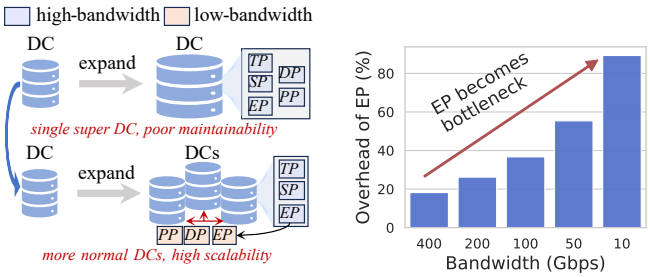


Fig. 1. **Structures of transformer and MoE block under real training configuration (2 nodes × 2 GPUs).** *Each MoE block replicates FFN to multiple experts, and only activates part of them (through gate network) to process data with constant computation cost. However, the frequent communication of All-to-All makes it the main bottleneck during training.*



(a) Trend of more DCs,larger EP. (b) The overhead ration of EP.

Fig. 2. **Analysis of parallelisms under common settings and the scalability of EP.** *(a) shows the trends of more DCs and larger EP, which requires more interconnections between DCs and scaling EP across DCs (b) shows the overhead ratio of EP under different bandwidths. This trend reveals a heavy overhead caused by EP when scaling under low bandwidth.*

framework to maximize training efficiency with two techniques: ① **Domain-Based Partition** to construct the hybrid communication topology. We introduce the expert domain to separate data and expert transmissions, and use a three-step mapping to construct the specific communication topology at the GPU level, ensuring compatibility with existing hierarchical hardware architectures. ② **Parameter-Efficient Migration** to optimize the communication topology. We further explore redundancy among expert and design a new expert representation to reduce communication traffic, as well as an asynchronous communicator to mitigate synchronization overhead. With the above designs, HybridEP can be considered as a more general EP with better scalability.

Experimental results suggest that HybridEP achieves the minimal training time compared to state-of-the-art MoE training systems. It achieves up to 5.68× speedup compared to Tutel [22], FasterMoE [20], and SmartMoE [57]. We further conduct a larger scale simulation to compare HybridEP and EP. With 1000 DCs connected, HybridEP achieves up to 1.45× speedup under different bandwidths.

In summary, we make the following contributions:

- We find that cross-DC MoE training is the ongoing paradigm to further expand LLM capacity, where EP becomes the main bottleneck to hinder training efficiency.
- We spatially reshape the placement of experts for efficient EP scaling under constrained bandwidth, which introduces a hybrid communication of data and experts.
- We implement HybridEP with *Stream-Based Modeling* to get the best proportion of hybrid patterns, *Domain-Based Partition* to construct the specific topology and *Parameter-Efficient Migration* to optimize it for better efficiency.
- Experiments suggest that HybridEP outperforms state-of-the-art works in both real scenario and simulation.

II. Background and Motivation

A. Parallelisms, MoE and its Communication Bottleneck

Common Parallelisms. There are five common parallelisms during distributed training, each with different properties:

Tensor Parallelism (TP): Splits weights of each layer across multiple GPUs, requiring extensive communication and large traffic during both forward and backward passes.

Sequence Parallelism (SP): Splits sequence across multiple GPUs [23], [34], which is typically applied at the end of pre-training for larger context window [19].

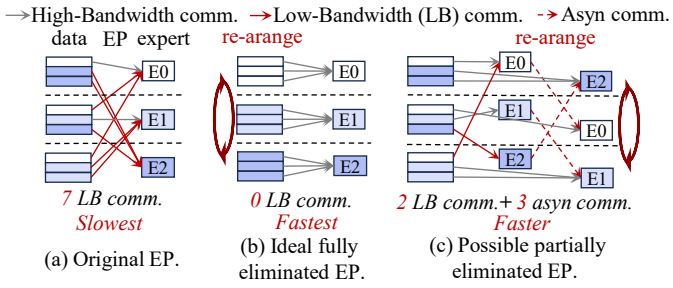
Pipeline Parallelism (PP): Divides the model layers into multiple stages, with each stage assigned to a different GPU. It uses point-to-point communication between stages to send/receive of activations and gradients.

Data Parallelism (DP): Replicates the full model on each GPU, where distinct batches are processed independently and gradients are synchronized across replicas during backward.

Expert Parallelism (EP): Distributes the experts in the MoE model across multiple GPUs. Specifically, the expert is expanded from the FFN – a basic module of Transformer structure [51], as shown in Figure 1. On this basis, MoE model further uses a gate network as the router, then uses two *All-to-All* (A2A) communications to change data to the part of activated experts for computation. However, large traffic and high frequency of EP makes it almost only deployed in HPC environments currently.

Necessity of Scaling EP across DCs. Firstly, considering the infrastructure, LLMs are pushing existing DCs to their limits. However, scaling up a single DC faces challenges like power limitations and increased vulnerability to outages [3]–[5]. Thus, recent reports suggest a more practical and resilient solution is to deploy multiple, smaller-scale DCs [8]. Secondly, considering the model, with the widespread application of MoE, some recent representative MoE models [12], [13], [17], [26], [39], [55] have shown a rapidly unfolding trend of expanding number of experts to hundreds or even thousands, exceeding the capacity of a single DC. Thus, combining the challenges of capacity expansion and the capacity limitations of a single DC, EP will inevitably expand to multiple DCs, as shown in Figure 2(a).

EP's Communication Bottleneck. As illustrated in Figure 2(b), EP causes extremely serious bottlenecks at relatively low bandwidths. Existing studies have also shown that under constrained bandwidth, EP almost occupies 50%-90% of the overall iteration time [20], [38], [47], becoming the main



(a) Original EP. (b) Ideal fully eliminated EP. (c) Possible partially eliminated EP.

Fig. 3. **An example of how the spatial placement of data and experts affect EP's efficiency.** *(a) shows the original placement of EP, where 7 low-bandwidth communications exist with slowest training speed. (b) shows the ideal case without any low-bandwidth communication through reshaping placements of data, which achieves the best efficiency. (c) shows a possible optimization in practice with only 4 low-bandwidth communications, achieved only by reshaping the placement of experts. This achieves a good compromise between ideal case and original EP.*

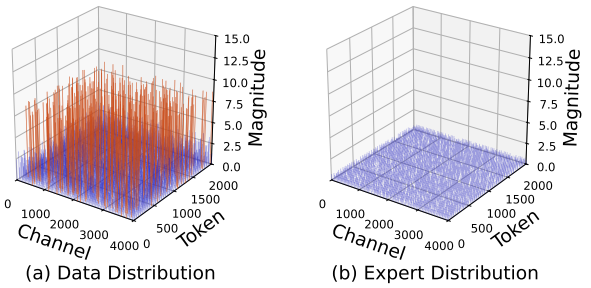


Fig. 4. **Compressibility analysis of data and experts.** *The distribution of data has large magnitude and many outliers (red part), while the distribution of expert weight is relative small and flat, leading to a higher compressibility.*

bottleneck in improving training efficiency. Unlike bandwidth-friendly parallelisms such as DP and PP, the communication-heavy design makes EP particularly sensitive to network conditions. It relies on the data-to-expert routing pattern (i.e., sending each data to correspond experts via A2A), requiring frequent communication for synchronization across devices and the traffic expands proportionally to the number of activated experts. Therefore, EP is not designed to scale across DCs and there is an urgent need for efficient EP scaling.

B. Motivation and Challenges

In Figure 3, we use a simple example to show how the placement of data and experts affects EP's communication overheads and motivates HybridEP.

Case of Original EP. In Figure 3(a), the basic logic of original EP operates like sending data to the corresponding experts decided by the routing result. When experts are placed across different nodes connected by low bandwidth, this introduces multiple slow communications (e.g., 7 in our example), which becomes the main bottleneck and degrades EP's efficiency.

Case of Ideal Fully-Eliminated EP. Ideally, if we know in advance which expert the data will be routed to, we can re-arrange before training and place data sent to the same expert together with high-bandwidth interconnections, avoiding the huge low-bandwidth communication latency, as shown in Figure 3(b). This essentially moves EP to the beginning of training, reducing its frequency and traffic and achieving the fastest speed. However, the problem is that the data routing results are dynamic and determined in real time, so it is almost impossible to know the routing results in advance and re-arrange the data before training. Meanwhile, since data-to-expert assignments vary across layers, it is unrealistic to assume the perfect placement before training. Thus, it is almost impossible to achieve this.

Case of Possible Patially-Eliminated EP. Considering that the basic logic of EP is to assign data with corresponding experts, we can optimize EP efficiency by changing the spatial placement of experts, as shown in the Figure 3(c). After experts process their data, we re-arange the placement of experts and then continue the computation. Although it can not achieve the ideal case, it still reduces the number of low-bandwidth communications (e.g., 5 in our example) for better efficiency. Note that we distinguish the transmission of data and expert, because we find that the expert has two rather good advantages for lightweight transmission compared to data, which has not been fully exploited by previous works to optimize the efficiency of EP. Specifically, ① *Better compressibility.* Expert weights typically exhibit compact and concentrated distributions, with fewer outliers compared to data, as shown in Figure 4, which is also confirmed by prior works [21], [33], [36], [54]. This allows experts to be compressed more aggressively without degrading accuracy to reduce traffic. ② *Asynchronous communication potential.* Unlike data that are tightly coupled with every computational operator, experts only participate in computation intermittently. This allows to pre-transmit experts independently ahead of EP, reducing synchronization overhead. Together, these points underscore that experts are inherently suited for lightweight, early, and stable communication. This forms the foundation for HybridEP.

Why this works? The key reason is that experts are *sparsely activated*, which is also the biggest difference between EP and the remaining parallelisms. Only in EP, data are processed by a small number of experts (part of model parameters), while in other parallel modes, all data need to be processed by all model parameters. Analogously to Figure 3(c), if EP does not have the property of sparse activation, that is assuming each data needs to be calculated with all experts, then no matter how the expert placement is changed, the number of low-bandwidth EP communications will not change. Therefore, only parallelisms with sparse activation properties like EP can improve efficiency through changing expert placement.

Challenges. The above analysis provides a vision of efficient EP scaling, introducing a hybrid transmission of both experts and data. However, in a real training scenario, it is still non-trivial to maximize its potential, which can be summarized into three main challenges:

- ① *How to determine the best proportion of data-expert hybrid communication patterns?* Essentially, experts communicate in All-Gather (AG) pattern, as each device collects experts from others (details in §III). We need to balance this with the A2A pattern for data.
- ② *How to construct the specific communication topology*

TABLE I
FREQUENTLY USED NOTATIONS IN MODELING.

Name	Description
D	Data size of a GPU.
P_E	Expert size with # parameters.
C	Computation throughput of a GPU.
B	Communication bandwidth between GPUs.
V^x	Communication volume of x. x is either A2A or AG.
Lat_x^y	y's latency of operation x (either comp. or comm.).
G	Number of GPUs in the cluster.
G^x	GPU set x. x is either A2A or AG.
$\|G^x\|$	Number of GPUs in GPU set G^x.

and ensure compatibility with existing architecture? Even if the proportion of two communications is determined, the most basic communication granularity is GPU, not DC. Thus a specific topology needs to be built to align the communication granularity.

- ③ *How to effectively exploit high compressiblity and asynchronicity for lightweight expert migration?* Although expert transmission can be lightweight, we still need to design corresponding compression and communication mechanisms to achieve theoretical efficiency.

To this end, we propose HybridEP to address the above challenges. First, we design a stream-based modeling to address the first challenge and provide a high-level guidance for the overall design. Following this, we introduce two techniques to unlock its potential. We address the second challenge by determining the hybrid communication topology and compatibility with existing hierarchical architecture through the division of expert domains. Then we address the third challenge by designing an expert compression algorithm and an asynchronous communicator for lightweight expert migration.

III. FOUNDATION: STREAM-BASED MODELING

The modeling process is shown in Figure 5(a). We first split MoE training into computation and communication streams (§III-A, §III-B). Then, we model the overlap between two streams (§III-C) and jointly consider computation, communication, and overlap to minimize training latency (§III-D, §III-E). We summarize frequently used notations in Table I. Our modeling assumes that each GPU has the same expert number, each expert has the same size, and the gate network activates experts evenly. For simplicity, we first assume that there is only one GPU in each DC (using GPU to represent the DC) to align the communication granularity, which does not affect the accuracy of our model.

A. Computation Modeling

GeMM Modeling. The computation latency mainly comes from General Matrix Multiplication (GeMM) operations. Following prior works [20], [32], we use a linear model to estimate the latency. Given two matrices to be multiplied, with size (L, H) and (H, M) respectively, the latency of a single GeMM operation can be expressed as:

$$Lat_{comp}^{GeMM} = \frac{LMH}{C}, \quad (1)$$

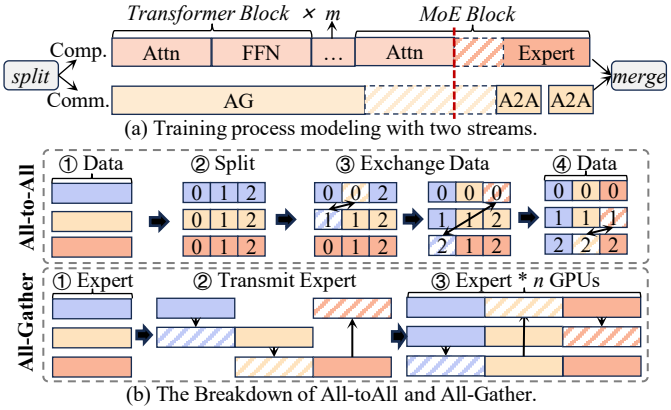


Fig. 5. **The modeling of training process and the communication breakdown of A2A and AG.** *(a) A shows a modeling process using the divide-and-conquer approach. The training process is first split into independent modeling of computing and communication streams, and then their overlapping relationships are considered for merging. (b) shows that the traffic of A2A remains unchanged (i.e., O(1)), while the traffic of AG is multiplied by number of GPUs (i.e., O(n)).*

where C represents the average computation throughput of GPU. Note C will be reduced if GeMM is too small to utilize GPU power. However, this will not affect overall modeling effectiveness due to its small overhead, confirmed by [20].

Computation Stream Modeling. Assume that there are m transformer blocks before a MoE block, The computation latency can be expressed as

$$\begin{aligned} Lat_{comp} &= mLat_{comp}^{TF} + Lat_{comp}^{MoE} \\ &= (m+1)Lat_{comp}^{Att} + mLat_{comp}^{FFN} + nLat_{comp}^{Ep}, \end{aligned} \quad (2)$$

where TF, MoE represents the transformer and MoE block. Att, FFN, Ep represents the computation process of attention, FFN, and expert, which consists of multiple GeMM operations. Here we can consider their latency as constant. n is the number of experts on one GPU, thus expert computation latency is repeated by n times. For brevity, we consider $(m+1)Lat_{comp}^{Att} + mLat_{comp}^{FFN}$ as *Pre-Expert*, denoted Lat_{comp}^{PE}.

B. Communication Modeling

All-to-All Communication Modeling. Figure 5(a) shows A2A communication details [50]. Specifically, given G GPUs, the data D on each GPU will be split into G chunks of size $\frac{D}{G}$. Then, $G-1$ chunks will be sent to other GPUs through A2A, while 1 chunk remains on the local GPU. Therefore, for GPU set G^{A2A} that participates in A2A communication, the overall traffic is expressed as:

$$V^{A2A} = \frac{D}{|G^{A2A}|} * (|G^{A2A}| - 1), Lat_{comm}^{A2A} = \frac{V^{A2A}}{B}, \quad (3)$$

where B is bandwidth. While D, B are constants, Lat_{comm}^{A2A} remains almost constant with increased $|G^{A2A}|$.

All-Gather Communication Modeling. Figure 5(b) shows AG communication details. Specifically, the expert parameters P_E on each GPU will be sent to other $G-1$ GPUs through

AG. Therefore, for GPU set G^{AG} that participates in AG, communication traffic is expressed as:

$$V^{AG} = P_E * (|G^{AG}| - 1), Lat^{AG}_{comm} = \frac{V^{AG}}{B}. \quad (4)$$

Therefore, Lat^{AG}_{comm} increases linearly with $|G^{AG}|$.

Relationships between Two Communications. A2A can be seamlessly transformed into AG. If an expert has been obtained through AG, then the corresponding data chunk is not necessary to be transmitted through A2A. When the i-th GPU G_i uses AG to collect expert P_E from G_j, the A2A's traffic changes from $D * \frac{G-1}{G}$ to $D * \frac{G-1}{G} - \frac{D}{G}$, while the AG's traffic changes from 0 to P_E. Therefore, when A2A's traffic decreases by $\frac{D}{G}$, AG's traffic increases by P_E.

Communication Stream Modeling. Its latency comes from both A2A and AG, which can be expressed as:

$$Lat_{comm} = Lat^{AG}_{comm} + 2Lat^{A2A}_{comm}. \quad (5)$$

where A2A performs twice before and after expert computation, and AG only performs once as experts do not need to be sent back to their original GPUs.

C. Overlap Modeling

Overlap Modeling of Two Streams. The overlap time between computation and communication comes from three parts, as shown in Figure 5(a) split by the red dotted line. Specifically, ① pre-expert computation (i.e., Lat^{PE}_{comp}) and AG; ② expert computation (Lat^{Ep}_{comp}) and AG; ③ expert computation (Lat^{Ep}_{comp}) and A2A; Note that the pre-expert computation cannot overlap with A2A because the data depend on the pre-expert results. Therefore, overlap can be written as:

$$Lat_{ovlp} = Lat^{PE,AG}_{ovlp} + Lat^{Ep,AG}_{ovlp} + Lat^{Ep,A2A}_{ovlp}, \quad (6)$$

Note that case ② and ③ have been optimized by previous works [35], [46], therefore expert computation is fully overlap with AG and A2A ($Lat^{Ep,AG}_{ovlp} + Lat^{Ep,A2A}_{ovlp} = nLat^{Ep}_{comp}$). For case ①, the overlap time is $min(Lat^{PE}_{comp}, Lat^{AG}_{comm})$. Therefore, the final overlap time can be expressed as:

$$Lat_{ovlp} = min(Lat^{PE}_{comp}, Lat^{AG}_{comm}) + nLat^{EP}_{comp}, \quad (7)$$

D. Problem Formulation

To minimize the latency, we have the following definition:

Definition 1. *Given a cluster with G GPUs ($G > 1$), p_i is the proportion of data chunks (which leave from G_i) that are transmitted through A2A, while $1-p_i$ is the proportion of data chunks (which leave from G_i) that are transformed into expert and transmitted through AG, where $p_i \in \{\frac{0}{G-1}, \cdots, \frac{G-1}{G-1}\}$.*

When $p_i = \frac{G-1}{G-1}$, there is only A2A; when $p_i = \frac{0}{G-1}$, there is only AG. The training latency can be expressed as:

$$\min_{p_i} \quad Lat_{final}(p_i) = Lat_{comp} + Lat_{comm} - Lat_{ovlp} \quad (8)$$
$$\text{s.t.} \quad p_i \in \{\frac{0}{G-1}, \cdots, \frac{G-1}{G-1}\}, Eq\ 2, Eq\ 5, Eq\ 7.$$

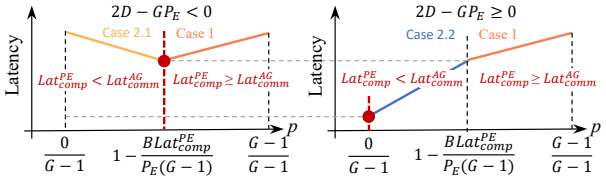


Fig. 6. **Visualization of Eq 10's solution.** *Two red dots indicate the optimal p with minimal latency under two cases.*

Note that each GPU has its own Eq 8, and they should be synchronized. Therefore, system latency is the maximal latency of all GPUs, which can be expressed as:

$$Lat_{all} = \max_{0 \leq i < G} \{\min Lat_{final}(p_i)\} \quad (9)$$

Finally, Eq 9 depends solely on parameter p_i, and our goal is to minimize Lat_{all} by choosing the optimal p_i.

E. Problem Solution

For simplicity, we assume that all the p_i are the same. Thus, Eq 9 can be simplified to an easy-to-solve format:

$$Lat_{all} = \min Lat_{final}(p)$$
$$= \begin{cases} \min(Lat^{PE}_{comp} + 2Lat^{A2A}_{comm}), \text{if } Lat^{PE}_{comp} \geq Lat^{AG}_{comm} \\ \min(Lat^{AG}_{comm} + 2Lat^{A2A}_{comm}), \text{if } Lat^{PE}_{comp} < Lat^{AG}_{comm}. \end{cases} \quad (10)$$

The final solution can be organized into two cases.

Case 1:when $Lat^{PE}_{comp} \geq Lat^{AG}_{comm}$, Eq 10 is simplified as:

$$\begin{cases} Lat_{all} & = Lat^{PE}_{comp} + \frac{2D(G-1)}{GB}p \\ \frac{G-1}{G-1} & \geq p \geq \frac{P_E(G-1) - BLat^{PE}_{comp}}{P_E(G-1)} \end{cases} \quad (11)$$

Note that Lat^{PE}_{comp}, D, B are positive constants. Thus, to minimize Lat_{final}, we need to configure the minimum p.

Case 2:when $Lat^{PE}_{comp} < Lat^{AG}_{comm}$, Eq 10 is simplified as

$$\begin{cases} Lat_{all} & = p\frac{(G-1)(2D-GP_E)}{BG} + \frac{(G-1)P_E}{B} \\ \frac{0}{G-1} & \leq p < \frac{P_E(G-1) - BLat^{PE}_{comp}}{P_E(G-1)} \end{cases} \quad (12)$$

Note that the sign of $\frac{(G-1)(2D-GP_E)}{BG}$ has two cases for minimal Lat_{final}. ① When $2D - GP_E < 0$, we configure the maximum p, denoted **Case 2.1**. ② When $2D - GP_E \geq 0$, we configure the minimum p, denoted **Case 2.2**.

Summary. Our model find the best proportion for minimal latency (i.e., p), which can be summarized into two cases, as shown in Figure 6. Specifically, when $2D - GP_E < 0$, the variation of overall latency consists of Case 1 and Case 2.1. Therefore, the optimal p is configured to $1 - \frac{BLat^{PE}_{comp}}{P_E(G-1)}$, where we use both AG and A2A. When $2D - GP_E \geq 0$, the variation of overall latency consists of Case 1 and Case 2.2. Therefore, the optimal p is configured to 0, where we only use AG. Note that when $p = 1$, HybridEP degenerates into the standard EP, indicating that EP is a special case of our framework.

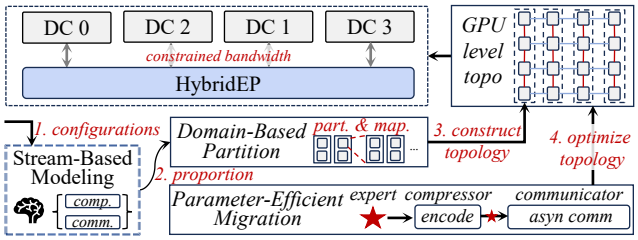


Fig. 7. **HybridEP overview.** *After modeling decides the proportion of transmitting data and expert, HybridEP uses the domain-based partition to construct specific GPU communication topology. Moreover, the parameter-efficient migration reduces the overhead for a better partition.*

IV. Design and Implementation

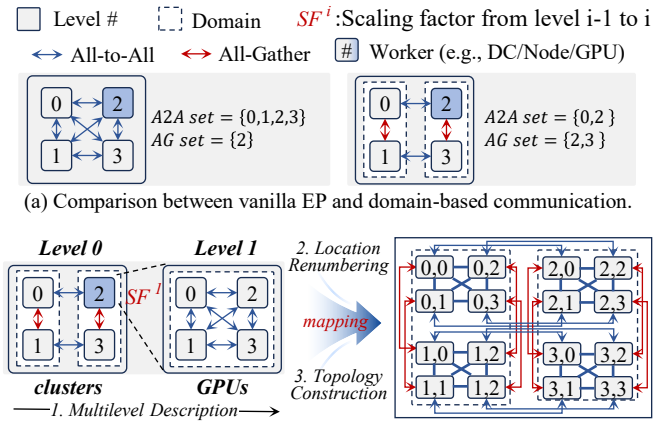
The overview of HybridEP is shown in Figure 7. Before training, HybridEP first takes the environmental configurations as input and uses the modeling to find the best proportion of transmitting data and experts. Oriented by this, HybridEP then introduces *domain-based partition* to partition GPUs for A2A and AG communication (§IV-A), which constructs the communication topology at GPU level. Moreover, HybridEP designs parameter-efficient migration to optimize the determined communication topology with a better partition (§IV-B).

A. Domain-Based Partition

Expert Domain and the Domain-Based Communication. The *Expert Domain* is a set of DCs that only uses AG communication within it. The size of expert domain is defined as the number of DCs in it, denoted as S_{ED}. HybridEP assume that each domain has the same size. Figure 8(a) right side shows an example, we set $S_{ED} = 2$ and sequentially group every 2 DCs into the same domain. With the help of expert domain, we have the following domain-based communication rule: AG will only occur for intra-domain communication, and A2A will only occur for inter-domain communication. Such a simple rule can effectively separate the two communication patterns for better management.

Necessity of Scaling to Multilevel. In actual scenario, the training environment often consists of hierarchical architectures, and the basic communication granularity is GPU. Thus, although how to communicate between DCs is determined, the specific behavior of each GPU is still unclear. Aligning with the GPU granularity is a critical step for real training scenarios. To bridge this gap, HybridEP first abstracts the hierarchical structure into *Multilevel Description*, handling the complex and changeable environments in reality. Then, it renumbers the global GPU number via *Location Renumbering* to adapt to the multilevel. Finally, it performs the *Topology Construction* algorithm to determine the specific topology at GPU level. The workflow is illustrated in Figure 8(b).

Multilevel Description. We first define that *Worker* is a physical entity (e.g., DC, node, or GPU). Normally, we consider GPU as the smallest granularity of a worker. *Level* is a set of workers that are connected with homogeneous bandwidth. Thus, we expand the definition of expert domain size at level l as the number of workers in the domain, denoted as S^l_{ED}. To describe the relationship between different levels,



(a) Comparison between vanilla EP and domain-based communication.

(b) Mapping multi-level partition to the specific topology via three steps.

Fig. 8. **Domain-based communication and the topology construction at multilevel.** *(a) shows how expert domain affects communication, which splits communications into the in-domain AG and the cross-domain A2A. (b) shows the mapping between topology and multilevel partition through three key steps: Multilevel Description, Location Renumbering and Topology Construction.*

we use the *scaling factor* SF^i to indicate that a worker at level $i-1$ can be expanded to level i with SF^i sub-workers. Note that we set SF^0 to the total number of workers at level 0. Take Figure 8(b) as an example, given an environment with 4 DCs and each with 4 GPUs, it is split into two levels with $SF^0 = 4, SF^1 = 4$ and the domain size at each level is $S^0_{ED} = 2, S^1_{ED} = 4$, respectively.

Location Renumbering. To clarify detailed communication rules, we first renumber the locations for each GPU for multilevel architecture. Specifically, we follow Pytorch [41] to allocate a global index m to each GPU. Then, given a $L-1$ level partition, we renumber the global index m into multilevel locations $(x_0, x_1, \cdots, x_{L-1})$. With the scaling factor list $[SF^0, \cdots, SF^{L-1}]$, the renumbering function $f : m \mapsto (x_0, x_1, \cdots, x_{L-1})$ can be expressed as:

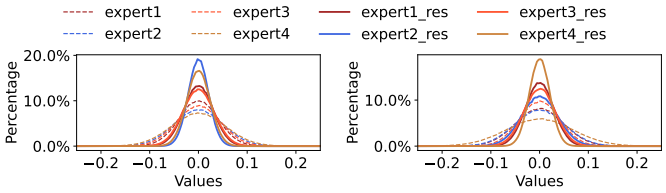
$$\begin{aligned} f(m) =& (x_0, x_1, \cdots, x_{L-1}) \\ x_i =& \frac{m}{\prod_{j=i+1}^{L-1} SF^j} \mod SF^i, i \in \{0, 1, \cdots, L-2\} \\ x_{L-1} =& m \mod SF^{L-1} \end{aligned} \tag{13}$$

Therefore, GPU m's level-i worker number can be obtained by $f(m)[i]$. Moreover, with the expert domain size S^i_{ED}, GPU m's level-i domain can be obtained by $\frac{f(m)[i]}{S^i_{ED}}$.

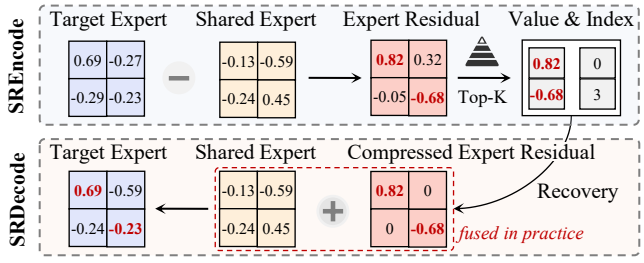
Topology Construction. The related pseudocode is shown in Algorithm 1. Specifically, given two GPUs with global index m and n, we decide which type of communication is required at different levels. We first obtain their multilevel locations by $f(m)$ and $f(n)$. To limit the inefficiency caused by multiple communications, we limit the range of GPUs that can communicate with each other. Specifically, only when $f(m)[l] \neq f(n)[l]$ and the indices of subsequent layers are the same, two GPUs can communicate with each other. At each level, the communications between GPUs follow the domain-based communication rule.

Algorithm 1 Communication Topology Construction

Input: GPU m, GPU n, current level l, scaling factor SF^l, expert domain size S^l_{ED}
Output: Communication type (None or AG or A2A)
$Loc_m \leftarrow f(m)$
$Loc_n \leftarrow f(n)$
$W_m, W_n \leftarrow Loc_m[l], Loc_n[l]$
$ED_m, off_m \leftarrow \frac{W_m}{S^l_{ED}}, W_m \mod S^l_{ED}$
$ED_n, off_n \leftarrow \frac{W_n}{S^l_{ED}}, W_n \mod S^l_{ED}$
if $Loc_D[l+1:] == Loc_E[l+1:]$ **then**
  if $ED_n == ED_m$ and $off_n \neq off_m$ **then**
    return AG
  end if
  if $ED_D \neq ED_E$ and $off_D == off_E$ **then**
    return A2A
  end if
end if
return None



(a) Both two weight matrices of experts has redundancy.

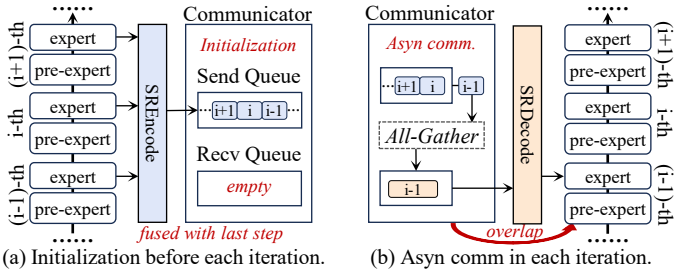


(b) Two phases of compressing and decompressing experts.

Fig. 9. **Redundancy amoung experts and the workflow of SR-Based Expert Compression.** *In SRDecode, we fuse the recovery and the addition operation in practice for better efficiency.*

B. Parameter-Efficient Migration

How Lightweight Migration Optimizes Communication Topology. Essentially, the lightweight migration reduces the size of P_E, leading to a larger expert domain which achieves better efficiency. Specifically, as shown in Figure 9, a smaller P_E can lead to a smaller p mainly due to two aspects: 1. When $2D < GP_E$, the corresponding p of the optimal point (i.e., the red dot) will decrease. 2. It allows more training configurations to be converted from $2D < GP_E$ to $2D \geq GP_E$. A smaller p indicates a larger domain of experts, which changes the constructed communication topology. Furthermore, the larger domain, the better efficiency can achieve. This is because Eq 11 and Eq 12 show that the overall latency decreases after p decreases theoretically. Thus, we regard parameter-efficient migration as a process of optimizing the communication topology by expanding the domain of experts, which aims to further improve the efficiency.



(a) Initialization before each iteration. (b) Asyn comm in each iteration.

Fig. 10. **Tow stage of asynchronous communicator.** *(a) shows the initialization stage, which is fused with the last optimizer step. Each MoE layer sequentially sends their experts processed by SREncode to Send Queue. (b) shows the asyn comm stage, which is overlapped with pre-expert computation. The communication results of each MoE layer are stored in Recv Queue and processed by SRDecode for subsequent computation.*

Redundancy Among Experts In addition to the better compressibility of experts shown in Figure 4, we further explored the redundancy among experts to improve compression ratio. We find that the main differences among experts are concentrated in a small number of parameters. It suggests that different experts may learn similar knowledge from data, which is also reported in other related work [12]. As shown in Figure 9(a), after averaging expert weights and subtracting them from the original weights, the result's distribution (with suffix "res") is more concentrated than the originals. This indicates that the residuals are sparse and the key differences between experts focused on a few parameters.

SR-Based Expert Compression. We are motivated to divide experts into shared and residual parts, which learn redundant knowledge and specific knowledge separately. Specifically, the shared expert is shared by all GPUs and is initialized by averaging all experts. At each training iteration, it will be synchronized with asynchronous All-Reduce in the backward propagation phase. Our expert compression has two phases, as shown in Figure 9(b). In the *encode phase*, the compressor first obtains the expert residual by subtracting the target expert and the shared expert. Then, it compresses expert residual through Top-k. The compressed expert residual is saved in the value-index format to transmit to other GPUs. In the *decode phase*, the compressor first recovers the compressed expert residual. Then, it restores the target expert by adding up the shared expert and the residual expert. Note that in practice, we fused the above two steps of decode phase for less overhead.

The Mechanism of Asynchronous Communicator. We use an asynchronous communicator to achieve the theoretical effect of our modeling as much as possible. To fully combine the asynchronous characteristics with SR compression without too much extra overhead, we divide the behavior of the asynchronous communicator into two stages like SR compression. As illustrated in Figure 10, the communicator considers the model as a stack of (pre-expert, expert) pair, and has a Send Queue and a Recv Queue. In *Initialization* stage, all experts in the model are sequentially processed by SREncode, and the compressed results will be delivered to the Send Queue. The

TABLE II
CONFIGURATIONS OF MODELS.

Model	Dataset	E	H	P_E	#Layers
Llama-Tiny	PennTreebank [2]	32	512	2.1M	12
Mistral-Small	WikiText2 [37]	32	768	4.7M	12
GPT-Medium	OpenWebText-10k [1]	32	1024	8.4M	12
GPT-Large	WikiText103 [37]	32	1024	8.4M	16

Recv Queue is set to be empty. Note that this process happens before each iteration begins, we fuse the SREncode with the update process (optimizer step) of the last iteration for less overhead. In *Asyn-comm* stage, the Send Queue sequentially pop expert residuals for AG communication. The Recv Queue receives the corresponding results and send to SRDecode for the subsequent expert computation. Note that this communication process is parallel to the pre-expert computation process of the model so we can overlap them. Moreover, we fused the SRDecode with expert computation for better efficiency.

V. EVALUATION

Our experiments aim to answer the following questions:

- Is our stream-based modeling accurately estimating computation, communication, and determining the best proportion with minimal latency? (§V-B)
- What is the end-to-end speedup of HybridEP with different data/expert size? (§V-C)
- How much does the domain-based partition and parameter-efficient migration contribute to the final effect? (§V-D)
- Does efficient parameter migration affect training accuracy and what's the impact of its compression/decompression process on computation? (§V-E)
- Does HybridEP has better communication traffic and frequency characteristics compared with EP? (§V-F)
- As a more general framework, does HybridEP have better scalability than EP on larger scale? (§V-G)

A. Experiment Setup

Testbed. We conduct experiments on three clusters consisting of different number of DCs. Due to the limitations of the actual environment, we regard a single node as a DC, which is internally connected by PCIe3.0 x16 (128 Gbps), and DCs are connected by a low bandwidth of Ethernet (10 Gbps). Specifically, we have ① **Cluster-S:** a cluster with 8 × NVIDIA A800 GPUs in a single DC. ② **Cluster-M:** a cluster with 16 × NVIDIA A800 GPUs on 2 DCs. ③ **Cluster-L:** a cluster with 32 × NVIDIA A800 GPUs on 4 DCs. Note that we use Cluster-S to verify the effectiveness of our modeling without considering hierarchical architecture, while using Cluster-M and Cluster-L to verify the effectiveness of HybridEP in real-world training tasks. Moreover, HybridEP is built based on Tutel [22] and Pytorch v.1.12.1, and the experiment environment is under Ubuntu-18.04, CUDA-11.3, cuDNN-7.6, and NCCL-2.10.

TABLE III
CONFIGURATIONS OF MOE LAYERS.

Parameter	Candidate Values
K	$\{1, 2, 4\}$
B	$\{8, 16, 32\}$
L	$\{128, 256, 512\}$
H	$\{512, 768, 1024\}$
M	$\{768, 1024, 1536, 2048, 3072, 4096\}$

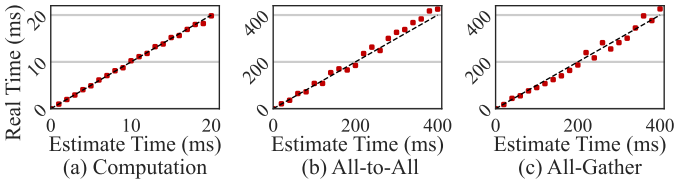


Fig. 11. **Latency Verification of Comp. and Comm..** *Since the estimate computation, A2A, AG latency (red markers) are close to the real latency (black line), our stream-based modeling can effectively model system latency.*

Configurations of Models, Datasets, and Compared Methods. We summarize tested models and datasets in Table II. Specifically, ① We use Llama-Tiny [19] for PennTreebank [2] dataset, which is one of the most known and used corpus for the evaluation of models for sequence labeling. ② We use Mistral-Small [26] for wikitext2 [37] dataset, which is a collection of over 100 million tokens extracted from the set of verified Good and Featured articles on Wikipedia; ③ We use GPT-Medium [6] for OpenWebText-10k, which is an open-source replication of the WebText dataset from OpenAI [1]; ④ We use GPT-Large [6] for WikiText103 [37], which is similar to wikitext2 but much larger. Note that we only built a smaller version for training based on the above model structure, not the original one. We compare HybridEP with Tutel [22], FasterMoE [20] and SmartMoE [58]. These MoE-specific optimized systems focus on dimensions of data transmission, expert transmission, and pipeline, which are commonly used in HPC environment. Note that we do not compare to some training systems [42], [48] because they also make some other optimizations besides MoE, which is also adopted by many works [20], [38], [47], [57], [58].

Extra Configurations. We use Adam optimizer for all experiments with a learning rate of $1e-4$ and Pytorch DDP for backward propagation, which can efficiently synchronize gradients of model parameters using ll-Reduce. Note that we do not use Zero Optimizer [43] for the non-MoE part and also the pipeline parallelism due to the potential network bandwidth conflicts, which may affect our model's accuracy. Moreover, all configurations will be adjusted within Table III to meet different experiment requirements. Specifically, K is the number of activated experts, B is the batch size, L is the sequence length, and H, M are experts' two dimensions.

B. Modeling Verification

To verify modeling effectiveness, ① we first verify whether it can accurately estimate the computation and communication latency, ② we then verify whether it can find the optimal proportion of A2A and AG (p in Figure 6).

TABLE IV
CONFIGURATIONS OF MODELING VERIFICATION.

Case	p	G	B	Lat_{comp}^{PE}	D	P_E
Mix-1	0.75	8	128 Gbps	0.049 ms	8 MB	4.7 MB
Mix-2	0.5	8	128 Gbps	0.049 ms	8 MB	2.35 MB
AG-only-1	0	8	128 Gbps	0.099 ms	3 MB	0.094 MB
AG-only-2	0	8	128 Gbps	0.099 ms	3 MB	0.047 MB

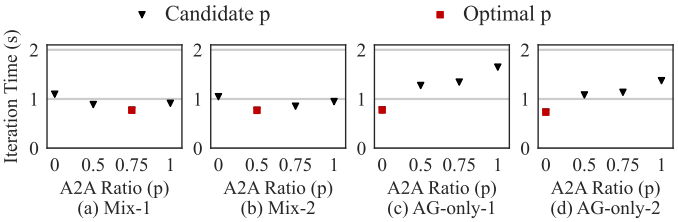


Fig. 12. **Modeling Verification.** *Results suggest that our modeling can find the optimal p (red marker) with the least iteration time among candidate configurations (black marker).*

Verification of Estimated Computation and Communication Latency. We adjust the sizes of the data traffic and expert size to test the accuracy of our model, as shown in Figure 11. Results suggest that the estimated latency is close to the real latency. However, they are fluctuating because our experiment platform is shared by multiple users with unstable network bandwidth. Nevertheless, such small fluctuations do not affect the effectiveness of our model.

Verification of the Optimal p. We then adjust the training configurations to verify whether our modeling can find the optimal proportion p of A2A and AG in different cases, as shown in Table IV. Note that one node has 8 GPUs in our configuration. Therefore, the candidates p are $0, 0.5, 0.75, 1$, which indicates that the expert domain size is $8, 4, 2, 1$, respectively. The results are shown in Figure 12, where the optimal p has the lowest average iteration latency among 4 candidate p, demonstrating the effectiveness of our model. Specifically, Mix-1 and Mix-2 represent Case 2.1 in Figure 6 (i.e., $2D - GP_E < 0$), therefore HybridEP communicates through both A2A and AG, and our modeling finds the optimal proportion of A2A data (i.e., $p = 0.5, 0.25$). Moreover, AG-only-1 and AG-only-2 represent Case 2.2 in Figure 6 (i.e., $2D - GP_E \geq 0$), therefore HybridEP should communicate only through AG (i.e., $p = 0$) for the lowest iteration latency.

C. End-to-end Speedup

We test HybridEP in Cluster-M and Cluster-L with different MoE configurations (Table III) in two scenarios. Specifically, ① different data traffic ranging from 6 MB to 192 MB; ② different expert size ranging from 32 MB to 2 MB.

Different Data Traffic. We change data traffic from 6 to 192 MB and fix expert size to 0.36 MB. The results are shown in Table V, where HybridEP achieves an average speedup of up to 5.60 ×. Specifically, with larger data traffic, lower bandwidth, and more connected DCs, The communication bottleneck of EP becomes more and more obvious. However, HybridEP finds the appropriate proportion of A2A and AG (p in Figure 6), thus achieving significant speedup.

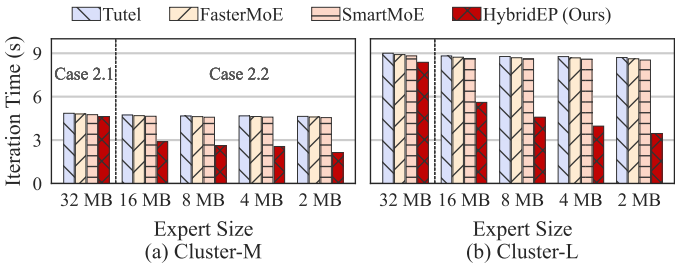


Fig. 13. **Average Iteration Time under Different Expert Sizes.** *Results suggest that as the expert size decreases, the computation cost decreases, HybridEP's iteration latency decreases. However, iteration latency of compared methods is nearly unchanged, despite the decreased computation overhead (i.e., $\frac{1}{16}$, expert size decreases from 32 MB to 2 MB).*

Different Expert Size. We change expert size from 32 MB to 2 MB and fix the data traffic to 16 MB. Therefore, computation cost decreases as expert sizes decrease, and we do not use the SR expert compression for better observation. The results are shown in Figure 13, where HybridEP achieves a speedup ranging from 1.18 × to 2.57 ×. Specifically, as the expert size decreases, HybridEP can transmit more experts with small traffic, thus enlarging the expert domain size and reducing EP's overhead. Thus, the acceleration effect of case 2.1 is not as significant as that of case 2.2. However, Case 2.1 can be transformed into Case 2.2 for higher speed with SR compression to change the condition to $2D - GP_E \geq 0$.

D. Ablation Study

In this section, we evaluate how domain-based partition (baseline) and parameter-efficient migration contributes to the overall speedup with different data traffic and expert size.

Configurations and Results Analysis. In Table VI, Data&Expert represent the size of data and expert. The remaining two items correspond to the two designs of HybridEP (+*Migration* equals to HybridEP). For 24&8MB configuration, our modeling suggests that Cluster-S has $p = 0.5$ (i.e., $S_{ED}^0 = 4$), while Cluster-M and Cluster-L has two levels, denote as $S_{ED}^0 = 2, S_{ED}^1 = 2$ and $S_{ED}^0 = 4, S_{ED}^1 = 1$. For 48&2MB configuration, p is 0 for all clusters. +*Migration* adds parameter-efficient migration (i.e., HybridEP). Table VI suggests that +*Migration* (i.e., HybridEP) achieves a speedup of 1.25× to 2.82×, compared to the baseline *Partition*. Larger data traffic and smaller expert size contribute to faster training speed. Note that the S_{ED} in our experiments includes all DCs, so the more DCs that are interconnected, the more significant the speedup. However, this may not be always true in practice, more details in §V-G.

E. Analysis on Migration

In this section, we first evaluate whether the SR expert compressor affects accuracy. Then we conduct the time breakdown experiments to demonstrate how the two phases of SR compression fused with other operations for less overhead. Our configurations are shown in Table II.

Configurations of SR-Based Expert Compression. HybridEP has a hyperparameter (compression ratio, CR) to control SR-based expert compression considering model accuracy

TABLE V
AVERAGE ITERATION TIME (IN SECONDS) AND AVERAGE SPEEDUP (×) UNDER DIFFERENT DATA TRAFFIC SIZES.

Method \ Data	Cluster-M						Cluster-L					
	6 MB	12 MB	24 MB	48 MB	96 MB	192 MB	6 MB	12 MB	24 MB	48 MB	96 MB	192 MB
Tutel	2.52 s	4.26 s	5.82 s	7.62 s	12.65 s	20.35 s	3.74 s	7.30 s	10.69 s	13.54 s	18.59 s	28.46 s
FasterMoE	2.58 s	4.37 s	5.90 s	7.81 s	12.80 s	20.82 s	3.86 s	7.50 s	11.09 s	13.88 s	19.32 s	29.43 s
SmartMoE	2.59 s	4.34 s	5.97 s	7.80 s	12.68 s	20.91 s	3.82 s	7.46 s	10.94 s	14.08 s	19.25 s	29.53 s
HybridEP (Ours)	2.48 s	2.63 s	2.74 s	2.82 s	3.01 s	3.78 s	3.49 s	3.53 s	3.54 s	3.85 s	4.24 s	5.20 s
Avg. Speedup	**1.03×**	**1.64×**	**2.15×**	**2.75×**	**4.22×**	**5.47×**	**1.09×**	**2.10×**	**3.08×**	**3.59×**	**4.49×**	**5.60×**

TABLE VI
ABLATION STUDY.

Cluster	Data&Expert	Partition	+ Migration
Cluster–S	24&8 MB	0.76 s	0.61 s
Cluster–M	24&8 MB	3.41 s	2.54 s
Cluster–L	24&8 MB	6.12 s	3.48 s
Cluster–S	48&2 MB	1.06 s	0.74 s
Cluster–M	48&2 MB	6.21 s	2.81 s
Cluster–L	48&2 MB	10.89 s	3.86 s

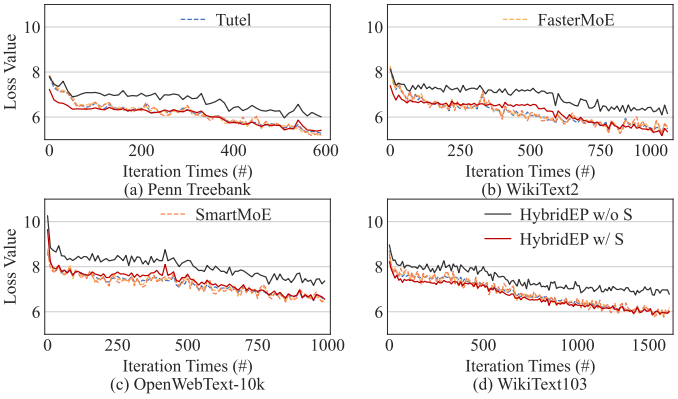


Fig. 14. **Loss Analysis**. *HybridEP's expert compression ratio is 50 ×. Moreover, w/o S indicates that HybridEP directly compress experts, while w/ S indicates that HybridEP compress experts with shared expert (§IV-B).*

and compressibility. We use *HybridEP w/ S* to represent the compression with shared expert, and use *HybridEP w/o S* to represent the naive method that directly compresses the expert through Top-k. Our goal is to find the maximum CR without affecting model performance.

Results of Accuracy Analysis. Figure 14 suggests that the loss value of *HybridEP w/ S* is close to that of the compared methods (i.e., Tutel, FasterMoE, and SmartMoE). Therefore, our proposed SR-based compression algorithm can retain both a high compression ratio (i.e., 50 ×, we do not display other results due to the page limit) and high accuracy. In contrast, *HybridEP w/o S*'s loss value is quite higher than compared methods, which indicates that the shared expert in our design plays an important role in accuracy maintenance.

Time Breakdown Analysis. As shown in Figure 15, as the expert size increases, the time overhead of both SREncode and SRDecode increases. When integrated with other computations, the overheads can be further reduced by up to 30% and 45%, respectively. Although they are not completely eliminated, it is not significant compared to the communication and remains within acceptable limits. However, designing more efficient expert compression is still worth exploring.

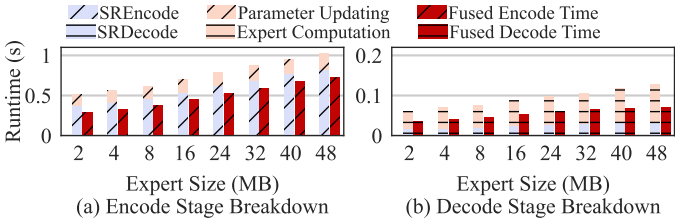


Fig. 15. **Time breakdown of Parameter-Efficient Migration's Tow Phases.** *Under different expert sizes, (a) shows the effect of SREncode fused with the parameter update of last iteration, which can reduce overhead by 30%. (b) shows the effect of SRDecode fused with multiple expert computations, which can reduce overhead by 45%.*

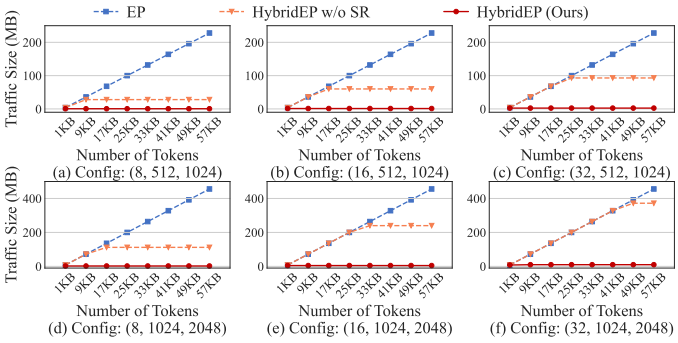


Fig. 16. **Traffic Scalability Analysis**. *HybridEP has less communication traffic under constrained bandwidth, leading to a better scalability. The configuration is a triplet, representing the size of EP and the tow dimensions of expert weights (H & M).*

F. EP vs. HybridEP: Characteristic Comparison

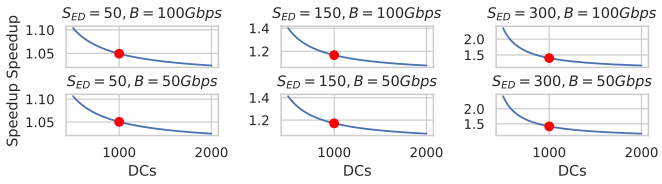
In this section, we show the comparison between HybridEP and EP in terms of communication traffic and frequency under different configurations.

Traffic Analysis. As shown in Figure 16, the traffic of original EP grows linearly with the number of tokens during each training iteration. In contrast, HybridEP introduces a more fixed and input-independent traffic with limited upper bound. When the number of tokens is small, HybridEP's traffic is almost the same as EP's. However, when the number of tokens increases significantly, EP becomes a huge communication bottleneck, while HybridEP guarantees a fixed traffic via only transmitting experts. This makes HybridEP more predictable and stable, which is especially advantageous in low-bandwidth or burst-sensitive environments.

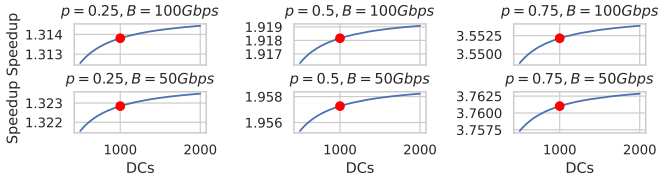
Frequency Analysis. We use the sum of all GPU-to-GPU communications as frequency. The comparison is shown in Table VII. Note that $S_{ED} = 1$ represents the original EP. As the expert domain expands, the A2A communication frequency

TABLE VII
COMMUNICATION FREQUENCY WITH DIFFERENT EP SIZE.

EP Size	Comm. Type	Expert Domain Size (S_{ED})					
		1 (EP)	2	4	8	16	32
8	A2A	56	24	8	0	-	-
	AG	0	8	24	56	-	-
16	A2A	240	112	48	16	0	-
	AG	0	16	48	112	240	-
32	A2A	992	480	224	96	32	0
	AG	0	32	96	224	480	992



(a) Fixed S_{ED} and dynamic p.



(b) Fixed p and dynamic S_{ED}.

Fig. 17. **Speedup of HybridEP on Large Scale Simulation.**

decreases quadratically, while the AG frequency increases accordingly. This can be seen as a gradual shift of A2A communication to AG. However, due to the more asynchronous nature of AG and its ability to significantly reduce traffic via compression, HybridEP achieves higher efficiency.

G. EP vs. HybridEP: Large Scale Simulation

In this section, we conduct the performance simulation with SimAI [53] due to limited environments. We test on a large scale to verify HybridEP's effectiveness under general settings.

Results of Simulation. We evaluate the effectiveness under different bandwidth from two cases, as shown in Figure 17. We first fix the expert domain size S_{ED} and expand the number of DCs. This essentially reduces the proportion p determined by HybridEP's model, resulting in a smaller acceleration effect. Given 1000 DCs (red dot), HybridEP achieves a speedup of 1.05× to 1.45×. Then, we fix the proportion p and expand the number of DCs. This essentially increases the size of domain, resulting in an improvement in the acceleration effect. Given 1000 DCs, HybridEP achieves 1.31× to 3.76× speedup. Furthermore, in both cases, the lower the bandwidth, the greater the speedup. Note that in practice, the first case is the most common because S_{ED} is fixed due to the fixed training configurations. Thus, the speedup decreases as the number of DCs increases. How to expand in the second case still remains extremely challenging, which is widely recognized.

VI. DISCUSSION

Storage Overhead of SR-Based Expert Compression. The additional storage overhead introduced by our proposed expert compression algorithm can be handled. Specifically, it consists of expert residual and shared expert. ① The expert residual consumes little GPU memory due to its high compressibility. ② The shared experts compete with local experts for GPU memory, which can be solved by offloading local experts to CPU memory while keeping shared experts in GPU memory. Offloading local experts to CPU memory is an effective strategy, which has been well studied (e.g., ZeRO-Offload [44]) and can be directly integrated into HybridEP.

Backward Propagation in MoE Training. The backward phase has the unique All-Reduce communication to synchronize model parameters, which competes with other types of communications and thus affects our modeling. Nevertheless, our modeling is still effective for backward propagation because the All-Reduce communication traffic is relative to the model size, and its latency can be regarded as a constant when model configurations are determined. Therefore, our modeling can handle backward propagation by simply adding a constant.

VII. RELATED WORKS

We introduce works that are orthogonal (or related) to our study, which mainly focus within the high performance cluster.

Optimizations on the Gate Network. Our modeling assume that the gate network activates experts evenly, and many works focused on how to achieve this. For example, Lewis et al. [29] proposed the BASE layer with token-to-expert allocation schema. Zhou et al. [60] proposed to allow experts to choose tokens. HybridEP can integrate them.

Optimizations on A2A Communication. To reduce A2A time, existing works focus on improving bandwidth utilization and reducing communication volume. For example, Hetu-MoE [38] proposed a hierarchical A2A algorithm to reduce communication rounds of inter-node communications; [22], [42] proposed the 2D-hierarchical A2A algorithms to better utilize high-speed intra-node links; Zhou et al. [59] used ZFP compression to reduce the A2A traffic.

Optimizations on Comp. & Comm. Overlap. HybridEP combines prefetch and pipeline to fully overlap computation and communication, while existing studies try to optimize one of them as much as possible. For example, [22], [30], [46] try to find the optimal pipeline degree to fully overlap expert computation and A2A communication, while Janus [35] tries to increase overlap time by pre-fetching experts.

VIII. CONCLUSION

This paper presents **HybridEP**, a modeling-guided framework that can efficiently scale EP under constrained bandwidth. Through modeling, the appropriate strategy depends solely on the proportion of data and expert transmission traffic. Based on the modeling, we then introduce two techniques: domain-based partition and parameter-efficient migration to make our modeling practical, efficient, and scalable. Experiments suggest that HybridEP outperforms state-of-the-art MoE systems by up to 5.6×. On large scale simulation with over 1000 DCs under different bandwidths, HybridEP achieves up to 1.45× speedup compared to the original EP.

REFERENCES

[1] "openwebtext-10k," https://huggingface.co/datasets/stas/openwebtext-10k, 2019.

[2] "Penn Treebank," https://paperswithcode.com/dataset/penn-treebank, 2020.

[3] "Google turns to nuclear to power AI data centres," https://www.bbc.co.uk/news/articles/c748gn94k95o, 2024.

[4] "Microsoft Azure CTO: US data centers will soon hit size limits," https://www.semafor.com/article/10/11/2024/microsoft-azure-cto-us-data-centers-will-soon-hit-limits-of-energy-grid, 2024.

[5] "US utilities see uptick in data center deals, signaling booming demand," https://www.reuters.com/business/energy/us-utilities-signal-booming-demand-data-centers-ai-takes-root-2024-08-12/, 2025.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020.

[7] S. Cao, S. Liu, T. Griggs, P. Schafhalter, X. Liu, Y. Sheng, J. E. Gonzalez, M. Zaharia, and I. Stoica, "Moe-lightning: High-throughput moe inference on memory-constrained gpus," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2025, p. 715–730.

[8] T. Chen, A. Kubicek, L. Huang, and T. Hoefler, "Crosspipe: Towards optimal pipeline schedules for cross-datacenter training," in *Proceedings of the 2025 USENIX Annual Technical Conference, USENIX ATC 2025, Boston, MA, USA, July 7-9, 2025*, D. Altinbüken and R. Stutsman, Eds. USENIX Association, 2025, pp. 1089–1108.

[9] Y. Chen, A. F. AbouElhamayed, X. Dai, Y. Wang, M. Andronic, G. A. Constantinides, and M. S. Abdelfattah, "Bitmod: Bit-serial mixture-of-datatype llm acceleration," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 1082–1097.

[10] M. Cho, K. A. Vahid, Q. Fu, S. Adya, C. C. D. Mundo, M. Rastegari, D. Naik, and P. Zatloukal, "edkm: An efficient and accurate train-time weight clustering for large language models," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 323–323.

[11] R. Dabre, C. Chu, and A. Kunchukuttan, "A survey of multilingual neural machine translation," *ACM Comput. Surv.*, vol. 53, no. 5, pp. 99:1–99:38, 2021.

[12] D. Dai, C. Deng, C. Zhao, R. X. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu, Z. Xie, Y. K. Li, P. Huang, F. Luo, C. Ruan, Z. Sui, and W. Liang, "DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models," *CoRR*, vol. abs/2401.06066, 2024.

[13] DeepSeek-AI, A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J. L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Wang, J. Chen, J. Chen, J. Yuan, J. Qiu, J. Li, J. Song, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Xu, L. Xia, L. Zhao, L. Wang, L. Zhang, M. Li, M. Wang, M. Zhang, M. Zhang, M. Tang, M. Li, N. Tian, P. Huang, P. Wang, P. Zhang, Q. Wang, Q. Zhu, Q. Chen, Q. Du, R. J. Chen, R. L. Jin, R. Ge, R. Zhang, R. Pan, R. Wang, R. Xu, R. Zhang, R. Chen, S. S. Li, S. Lu, S. Zhou, S. Chen, S. Wu, S. Ye, S. Ye, S. Ma, S. Wang, S. Zhou, S. Yu, S. Zhou, S. Pan, T. Wang, T. Yun, T. Pei, T. Sun, W. L. Xiao, and W. Zeng, "Deepseek-v3 technical report," *CoRR*, vol. abs/2412.19437, 2024.

[14] M. Diskin, A. Bukhtiyarov, M. Ryabinin, L. Saulnier, q. lhoest, A. Sinitsin, D. Popov, D. V. Pyrkin, M. Kashirin, A. Borzunov, A. Villanova del Moral, D. Mazur, I. Kobelev, Y. Jernite, T. Wolf, and G. Pekhimenko, "Distributed Deep Learning In Open Collaborations," in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021, pp. 7879–7897.

[15] J. Dong, B. Luo, J. Zhang, P. Zhang, F. Feng, Y. Zhu, A. Liu, Z. Chen, Y. Shi, H. Jiao, G. Lu, Y. Guan, E. Zhai, W. Xiao, H. Zhao, M. Yuan, S. Yang, X. Li, J. Wang, R. Men, J. Zhang, C. Zhou, D. Cai, Y. Xie, and B. Fu, "Enhancing large-scale ai training efficiency: The c4 solution for real-time anomaly detection and communication optimization," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 1246–1258.

[16] A. Douillard, Q. Feng, A. A. Rusu, R. Chhaparia, Y. Donchev, A. Kuncoro, M. Ranzato, A. Szlam, and J. Shen, "DiLoCo: Distributed Low-Communication Training of Language Models," 2024. [Online]. Available: https://arxiv.org/abs/2311.08105

[17] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, pp. 120:1–120:39, 2022.

[18] S. S. Gill, M. Xu, C. Ottaviani, P. Patros, R. Bahsoon, A. Shaghaghi, M. Golec, V. Stankovski, H. Wu, A. Abraham, M. Singh, H. Mehta, S. K. Ghosh, T. Baker, A. K. Parlikad, H. Lutfiyya, S. S. Kanhere, R. Sakellariou, S. Dustdar, O. F. Rana, I. Brandic, and S. Uhlig, "AI for next generation computing: Emerging trends and future directions," *Internet Things*, vol. 19, p. 100514, 2022.

[19] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. van der Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. de Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand, A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardi, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C.-H. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E.-T. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, Guangyi, Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I.-E. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J.-B. Gaya, J. Marcus, J. Tang, J. Chan,

J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A, L. Silva, L. Bell, L. Zhang, L. Guo, L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev, N. Dong, N. Cheng, O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng, S. Cho, S. Virk, S. Subramanian, S. Choudhury, S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar, V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, Yu, Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma, "The llama 3 herd of models," *CoRR*, vol. abs/2407.21783, 2024.

[20] J. He, J. Zhai, T. Antunes, H. Wang, F. Luo, S. Shi, and Q. Li, "FasterMoE: modeling and optimizing training of large-scale dynamic pre-trained models," in *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2022, pp. 120–134.

[21] C. Hooper, S. Kim, H. Mohammadzadeh, M. W. Mahoney, Y. S. Shao, K. Keutzer, and A. Gholami, "Kvquant: Towards 10 million context length LLM inference with KV cache quantization," in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, Eds., 2024.

[22] C. Hwang, W. Cui, Y. Xiong, Z. Yang, Z. Liu, H. Hu, Z. Wang, R. Salas, J. Jose, P. Ram, H. Chau, P. Cheng, F. Yang, M. Yang, and Y. Xiong, "Tutel: Adaptive Mixture-of-Experts at Scale," in *Proceedings of the Sixth Conference on Machine Learning and Systems (MLSys)*.

[23] S. A. Jacobs, M. Tanaka, C. Zhang, M. Zhang, S. L. Song, S. Rajbhandari, and Y. He, "Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models," *CoRR*, vol. abs/2309.14509, 2023.

[24] S. Jaghouar, J. M. Ong, M. Basra, F. Obeid, J. Straube, M. Keiblinger, E. Bakouch, L. Atkins, M. Panahi, C. Goddard, M. Ryabinin, and J. Hagemann, "INTELLECT-1 technical report," *CoRR*, vol. abs/2412.01152, 2024.

[25] H. Jang, J. Song, J. Jung, J. Park, Y. Kim, and J. Lee, "Smart-infinity: Fast large language model training using near-storage processing on a real system," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024, pp. 345–360.

[26] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de Las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mixtral of experts," *CoRR*, vol. abs/2401.04088, 2024.

[27] S. Laskar, P. Majhi, S. Kim, F. Mahmud, A. Muzahid, and E. J. Kim, "Enhancing collective communication in mcm accelerators for deep learning training," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024, pp. 1–16.

[28] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[29] M. Lewis, S. Bhosale, T. Dettmers, N. Goyal, and L. Zettlemoyer, "Base layers: Simplifying training of large, sparse models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6265–6274.

[30] J. Li, Y. Jiang, Y. Zhu, C. Wang, and H. Xu, "Accelerating distributed moe training and inference with lina," in *Proc. USENIX ATC 2023*, 2023.

[31] J. Li, T. Tang, W. X. Zhao, J. Nie, and J. Wen, "Pre-trained language models for text generation: A survey," *ACM Comput. Surv.*, vol. 56, no. 9, pp. 230:1–230:39, 2024.

[32] Y. Li, Y. Sun, and A. Jog, "Path forward beyond simulators: Fast and accurate gpu execution time prediction for dnn workloads," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23, 2023, p. 380–394.

[33] J. Lin, J. Tang, H. Tang, S. Yang, G. Xiao, and S. Han, "AWQ: activation-aware weight quantization for on-device LLM compression and acceleration," *GetMobile Mob. Comput. Commun.*, vol. 28, no. 4, pp. 12–17, 2024.

[34] H. Liu, M. Zaharia, and P. Abbeel, "Ring attention with blockwise transformers for near-infinite context," *CoRR*, vol. abs/2310.01889, 2023.

[35] J. Liu, J. H. Wang, and Y. Jiang, "Janus: A unified distributed training framework for sparse mixture-of-experts models," in *Proceedings of the ACM SIGCOMM 2023 Conference, ACM SIGCOMM 2023, New York, NY, USA, 10-14 September 2023*, 2023, pp. 486–498.

[36] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu, "KIVI: A tuning-free asymmetric 2bit quantization for KV cache," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[37] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *Proc. ICLR'17*, 2017.

[38] X. Miao, X. Nie, H. Zhang, T. Zhao, and B. Cui, "Hetu: a highly efficient automatic parallel distributed deep learning system," *Sci. China Inf. Sci.*, vol. 66, no. 1, 2023.

[39] MiniMax, A. Li, B. Gong, B. Yang, B. Shan, C. Liu, C. Zhu, C. Zhang, C. Guo, D. Chen, D. Li, E. Jiao, G. Li, G. Zhang, H. Sun, H. Dong, J. Zhu, J. Zhuang, J. Song, J. Zhu, J. Han, J. Li, J. Xie, J. Xu, J. Yan, K. Zhang, K. Xiao, K. Kang, L. Han, L. Wang, L. Yu, L. Feng, L. Zheng, L. Chai, L. Xing, M. Ju, M. Chi, M. Zhang, P. Huang, P. Niu, P. Li, P. Zhao, Q. Yang, Q. Xu, Q. Wang, Q. Wang, Q. Li, R. Leng, S. Shi, S. Yu, S. Li, S. Zhu, T. Huang, T. Liang, W. Sun, W. Sun, W. Cheng, W. Li, X. Song, X. Su, X. Han, X. Zhang, X. Hou, X. Min, X. Zou, X. Shen, Y. Gong, Y. Zhu, Y. Zhou, Y. Zhong, Y. Hu, Y. Fan, Y. Yu, Y. Yang, Y. Li, Y. Huang, Y. Li, Y. Huang, Y. Xu, Y. Mao, Z. Li, Z. Li, Z. Tao, Z. Ying, Z. Cong, Z. Qin, Z. Fan, Z. Yu, Z. Jiang, and Z. Wu, "Minimax-01: Scaling foundation models with lightning attention," *CoRR*, vol. abs/2501.08313, 2025.

[40] X. Pan, W. Lin, L. Zhang, S. Shi, Z. Tang, R. Wang, B. Li, and X. Chu, "Fsmoe: A flexible and scalable training system for sparse mixture-of-experts models," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2025, p. 524–539.

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 2019, pp. 8024–8035.

[42] S. Rajbhandari, C. Li, Z. Yao, M. Zhang, R. Y. Aminabadi, A. A. Awan, J. Rasley, and Y. He, "DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale," in *Proceedings of the 39th International Conference on Machine Learning (ICML)*, vol. 162, 2022, pp. 18 332–18 346.

[43] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020, pp. 1–16.

[44] J. Ren, S. Rajbhandari, R. Y. Aminabadi, O. Ruwase, S. Yang, M. Zhang, D. Li, and Y. He, "ZeRO-Offload: Democratizing Billion-Scale model training," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, Jul. 2021, pp. 551–564.

[45] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean, "Outrageously Large Neural Networks: The Sparsely-

Gated Mixture-of-Experts Layer," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[46] S. Shi, X. Pan, X. Chu, and B. Li, "Pipemoe: Accelerating mixture-of-experts through adaptive pipelining," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications, New York City, NY, USA, May 17-20, 2023*, 2023, pp. 1–10.

[47] S. Shi, X. Pan, Q. Wang, C. Liu, X. Ren, Z. Hu, Y. Yang, B. Li, and X. Chu, "Schemoe: An extensible mixture-of-experts distributed training system with tasks scheduling," in *Proceedings of the Nineteenth European Conference on Computer Systems*, ser. EuroSys '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 236–249.

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

[49] P. I. Team, S. Jaghouar, J. Mattern, J. M. Ong, J. Straube, M. Basra, A. Pazdera, K. Thaman, M. D. Ferrante, F. Gabriel, F. Obeid, K. Erdem, M. Keiblinger, and J. Hagemann, "INTELLECT-2: A reasoning model trained through globally decentralized reinforcement learning," *CoRR*, vol. abs/2505.07291, 2025.

[50] S. S. Vadhiyar, G. E. Fagg, and J. Dongarra, "Automatically tuned collective communications," in *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing (ICS)*, 2000, pp. 3–3.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS'17*, 2017, pp. 5998–6008.

[52] X. Wang, J. Li, Z. Ning, Q. Song, L. Guo, S. Guo, and M. S. Obaidat, "Wireless powered mobile edge computing networks: A survey," *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 263:1–263:37, 2023.

[53] X. Wang, Q. Li, Y. Xu, G. Lu, D. Li, L. Chen, H. Zhou, L. Zheng, S. Zhang, Y. Zhu, Y. Liu, P. Zhang, K. Qian, K. He, J. Gao, E. Zhai, D. Cai, and B. Fu, "Simai: Unifying architecture design and performance tuning for large-scale large language model training with scalability and precision," in *22nd USENIX Symposium on Networked Systems Design and Implementation, NSDI 2025, Philadelphia, PA, USA, April 28-30, 2025*, T. A. Benson and R. N. Mysore, Eds. USENIX Association, 2025, pp. 541–558.

[54] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, vol. 202, 2023, pp. 38 087–38 099.

[55] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, and Z. Qiu, "Qwen3 technical report," *CoRR*, vol. abs/2505.09388, 2025.

[56] M. Yue, "A survey of large language model agents for question answering," *CoRR*, vol. abs/2503.19213, 2025.

[57] Z. Zeng and D. Xiong, "Scomoe: Efficient mixtures of experts with structured communication," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

[58] M. Zhai, J. He, Z. Ma, Z. Zong, R. Zhang, and J. Zhai, "SmartMoE: Efficiently training Sparsely-Activated models through combining offline and online parallelization," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. Boston, MA: USENIX Association, Jul. 2023, pp. 961–975.

[59] Q. Zhou, P. Kousha, Q. Anthony, K. Shafie Khorassani, A. Shafi, H. Subramoni, and D. K. Panda, "Accelerating MPI All-to-All Communication with Online Compression on Modern GPU Clusters," in *High Performance Computing*, 2022, pp. 3–25.

[60] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Y. Zhao, A. Dai, Z. Chen, Q. Le, and J. Laudon, "Mixture-of-experts with expert choice routing," in *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[61] P. Zuo, H. Lin, J. Deng, N. Zou, X. Yang, Y. Diao, W. Gao, K. Xu, Z. Chen, S. Lu, Z. Qiu, P. Li, X. Chang, Z. Yu, F. Miao, J. Zheng, Y. Li, Y. Feng, B. Wang, Z. Zong, M. Zhou, W. Zhou, H. Chen, X. Liao, Y. Li, W. Zhang, P. Zhu, Y. Wang, C. Xiao, D. Liang, D. Cao, J. Liu, Y. Yang, X. Bai, Y. Li, H. Xie, H. Wu, Z. Yu, L. Chen, H. Liu, Y. Ding, H. Zhu, J. Xia, Y. Xiong, Z. Yu, and H. Liao, "Serving large language models on huawei cloudmatrix384," *CoRR*, vol. abs/2506.12708, 2025.