2025 34th International Conference on Computer Communications and Networks (ICCCN) | 979-8-3315-0898-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICCCN65249.2025.11133974

TOPO-X: Co-optimize Flow Scheduling, Topology, and ML Training Parallelism

Yi-Xiang Hu*, Han Tian*‡, Yifang Zhao†, Feng Wu*, Xiang-Yang Li*‡
*School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
†School of Cyber Science and Technology, University of Science and Technology of China, Hefei, China
{yixianghu,zhaoyifang}@mail.ustc.edu.cn, {henrytian,wufeng02,xiangyangli}@ustc.edu.cn

***Abstract*—The rapid advancement of large-scale deep neural networks and large language models has intensified the demand for highly efficient GPU clusters. However, existing distributed training frameworks, like Fat-tree and TopoOpt, struggle with inefficient resource utilization and network bottlenecks. They often optimize communication, parallelism, and network topology independently, failing to leverage their interdependencies. To address this gap, we propose TOPO-X, a novel reconfigurable network framework that co-optimizes flow scheduling, training parallelism, and optical network topology. By formulating this integrated optimization challenge as a Resource-Constrained Project Scheduling Problem, TOPO-X dynamically adapts to changing workloads and network conditions using optical network reconfiguration capabilities. Our experimental results show that TOPO-X outperforms the state-of-the-art solution, TopoOpt, achieving a 2.22× speedup in training iteration times on average. These findings highlight TOPO-X as a promising approach for scalable, adaptive, and high-performance GPU clusters designed to meet the increasing demands of large-scale AI training workloads.**

***Index Terms*—reconfigurable network, network topology, optical networking, solver.**

I. INTRODUCTION

The recent developments of deep neural networks (DNNs) and large language models (LLMs) have dramatically reshaped artificial intelligence, leading to remarkable breakthroughs exemplified by models like ChatGPT [1]. The proliferation of such large-scale models, frequently consisting of billions of parameters, has significantly increased computational and memory demands, intensifying the need for efficient distributed training frameworks and robust GPU clusters [2]–[4].

However, existing distributed DNN training systems typically rely on conventional data center clusters employing multi-tier Fat-tree topologies [5]. Fat-tree networks, designed for legacy data center workloads characterized by short, bursty traffic patterns, offer uniform bandwidth and latency between server pairs [6]–[9]. However, they have become inadequate for contemporary DNN training workloads due to their inability to dynamically adapt to large, predictable communication demands [10]–[13]. For instance, training a 175-billion-parameter LLM across a 256-GPU cluster necessitates transferring approximately 1.4TB of gradients per iteration, resulting in communication overhead consuming more than 60% of the total training duration [14]. Similar bottlenecks have been observed in prior studies [11], [15]–[18], highlighting the limitations of existing data center networks in meeting the demands of large-scale AI training.

Current solutions addressing this communication bottleneck typically fall into three isolated optimization categories: topology, parallelism, and scheduling. First, topology optimization techniques, such as TopoOpt [14], dynamically adjust optical circuit switches (OCS) to minimize latency based on static traffic heatmaps. However, they overlook task dependencies and dynamic communication fluctuations occurring during different training stages (e.g., forward and backward passes). Second, parallelism optimization frameworks, including FlexFlow [19] and ColocRL [20], automate the search for optimal parallel configurations but are limited by fixed network topologies, diminishing their adaptability. Lastly, scheduling optimization methods, exemplified by Cassini [21] and Optimus [22], effectively allocate resources but lack the capability for real-time network reconfiguration. Consequently, despite individually optimizing specific dimensions, these isolated approaches fail to collectively resolve communication inefficiencies due to their lack of coordination.

To bridge this critical gap, we introduce TOPO-X, a novel reconfigurable network framework designed to holistically optimize flow scheduling, training parallelism, and optical network topology. By modeling this integrated optimization challenge as a Resource-Constrained Project Scheduling Problem (RCPSP), TOPO-X dynamically configures OCS connections, meticulously accommodating task dependencies and hardware constraints (e.g., port limits). This integrated strategy facilitates three core advantages: 1) **Adaptive Topology**: dynamically adjusting optical links in real-time to align with evolving traffic patterns; 2) **Task-Aware Scheduling**: strategically prioritizing tasks on critical execution paths to reduce iteration latency; and 3) **Parallelism Optimization**: employing FlexNet [19] to explore and implement optimal parallelization configurations within the dynamically configured network.

Specifically, our contributions in this paper are:

- **Holistic Optimization**: Introducing TOPO-X, the first integrated framework to simultaneously optimize scheduling, parallelism, and network topology for distributed DNN training.
- **Comprehensive Problem Formulation**: Formulating the co-optimization of flow scheduling and topology configuration as an RCPSP, ensuring precise synchronization

‡Corresponding authors.

of computation and communication tasks within practical hardware constraints.

- **Empirical Validation and Performance Improvement**: Demonstrating through FlexNet simulations that TOPO-X achieves significant training acceleration, providing up to a 6.08× speedup compared to the state-of-the-art method, TopoOpt, when training the Deep Learning Recommendation Model (DLRM) [23].

II. MOTIVATION

Prior research highlights demand-aware network fabrics as promising solutions for building flexible and cost-efficient datacenter-scale networks [24]–[26]. However, accurately predicting future traffic distribution remains challenging in traditional datacenter environments due to their highly dynamic and unpredictable workloads.

In contrast, DNN training workloads offer distinct advantages in traffic predictability, making them particularly suitable for demand-aware network optimizations [14]. Traffic patterns in DNN training are inherently predictable, as these jobs typically have extended durations and exhibit repetitive, well-defined communication behaviors. This predictability arises because DNN training commonly utilizes stochastic gradient descent (SGD), an iterative process involving a selection of random data batches, computation of model errors, and gradient backpropagation for model updates. The iterative nature of SGD allows for precise precomputation of traffic distributions before job execution [27].

Moreover, 3D MEMS-based OCSs [28] can dynamically reconfigure optical links by adjusting tiny mirrors to alter the direction of light. A key advantage of OCS technology is its reconfiguration latency (approximately 10ms), which is four orders of magnitude faster compared to traditional patch panels that require several minutes [29].

Consequently, effectively optimizing flow scheduling and carefully adjusting optical network topology within a single training iteration is sufficient to significantly enhance overall performance.

This study first aims to validate whether incorporating explicit flow scheduling substantially improves the performance of existing topology optimization methods such as TopoOpt, which currently rely solely on static traffic heatmaps. Existing methods neglect the dynamic variations in communication patterns observed between different stages of a training iteration (e.g., forward and backward passes), thereby potentially overlooking opportunities for latency reduction and throughput maximization.

Secondly, this research aims to develop a comprehensive, integrated algorithm that simultaneously optimizes traffic flow scheduling, network topology configuration, and parallelization strategies tailored specifically to machine learning (ML) workloads. Our proposed approach addresses critical shortcomings in prior methodologies by holistically considering task dependencies, dynamic traffic fluctuations, and the rapid reconfiguration capabilities offered by OCS.

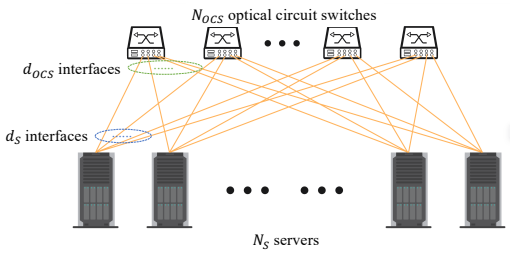

Fig. 1. Illustration of TOPO-X's interconnect.

Inspired by traffic intersection control systems [30], our co-optimization framework leverages the predictable traffic patterns characteristic of DNN workloads and exploits the swift (10ms) reconfiguration speed of OCSs [31]. Analogous to optimizing traffic signals at intersections to alleviate congestion and improve flow, our framework dynamically adjusts network topology configurations in response to anticipated network demand. Through this adaptive strategy, the proposed framework is anticipated to significantly improve resource utilization and reduce iteration times in distributed DNN training scenarios.

III. OPTIMIZATION FORMULATION

As shown in Figure 1, we illustrate a reconfigurable optical network architecture of TOPO-X, consisting of N_S servers, each equipped with d_S network interfaces. These servers are interconnected through N_{OCS} OCSs, where each OCS has d_{OCS} interfaces. The OCSs dynamically adjust their connections, allowing the network topology to adapt in real-time according to workload demands.

Given a DNN (or LLM) model and specific training hardware configurations, we design an ML training strategy that optimally utilizes the available resources. Specifically, it includes the following information: 1) Model characteristics, including type, architecture, size, and dataset scale used for training. 2) Hardware specifications, such as the number of servers, their hardware configurations, the number of optical switches and their settings, as well as the network cards and their configurations. Under these constraints, we determine the optimal training pattern that minimizes training iteration time.

Optimization Input. The optimization inputs primarily consist of task-related parameters and hardware constraints, as detailed in Table I. Specifically, J^T and J^C represent the sets of transfer and compute tasks, respectively. Each transfer task j_i^T has an associated pair (x_i, y_i) that defines its source and destination servers. The running time of any given task j is denoted by p_j. Task dependencies are captured within the set P.

Additionally, hardware-specific parameters such as the number of servers (N_S), number of OCSs (N_{OCS}), and respective numbers of ports per server (d_S) and per OCS (d_{OCS}) are included to ensure realistic modeling constraints. Notably, TOPO-X can easily generalize to accommodate variations in

TABLE I
OPTIMIZATION MODEL SYMBOLS AND DEFINITIONS

Symbol	Definition
τ	the time slot size
m	the number of time slots
$J^T = \{j_i^T\}$	the set of transfer tasks
(x_i, y_i)	transfer direction of task j_i^T
$J^C = \{j_i^C\}$	the set of compute tasks
p_j	running time of task j
N_S	the number of the servers
N_{OCS}	the number of the OCSs
d_S	the number of ports on the server
d_{OCS}	the number of ports on the OCS
$P = \{(j_i, j_k)\}$	the set of task dependencies
j_v	a virtual task
$x[j_i^T, t']$	transfer task j_i^T starts at t' th time slot or not
$s[j]$	the start time of task j
$A = \{a[r, x, y, t]\}$	allocation of OCS

these hardware configurations, demonstrating the flexibility and scalability of the proposed model.

Auxiliary variables. Auxiliary variables facilitate the formulation of constraints and simplify the optimization problem. As detailed in Table I, a virtual task j_v with zero processing time is introduced, which depends on all compute and transfer tasks. The binary variable $x[j_i^T, t]$ indicates whether a transfer task j_i^T starts at time slot t ($x[j_i^T, t] = 1$) or not, ($x[j_i^T, t] = 0$).

Optimization Output. The model outputs include task start times and the OCS reconfiguration schedule, presented in Table I. The OCS schedule is represented as a four-dimensional binary array $a[r, x, y, t]$, defined as:

$$a[r, x, y, t] = \begin{cases} 1, \text{ if OCS } r \text{ connects server } x \text{ and } y \text{ at } t \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

For example, if $a[r_2, x_1, y_3, t_2] = 1$, OCS r_2 establishes connectivity from server x_1 to y_3 at time slot t_2. Here, we present the optimization model:

Objective:

$$\min \ s[j_v] \tag{2}$$

Subject to:

$$\sum_{t'=0}^{m-1} x[j_i^T, t'] = 1, \ \forall j_i^T \in J^T \tag{3}$$

$$\sum_{t'=0}^{m-1} \tau \times t \times x[j_i^T, t'] = s[j_i^T], \ \forall j_i^T \in J^T \tag{4}$$

$$s[j_i] + p_{j_i} \leq s[j_k], \ \forall (j_i, j_k) \in P \tag{5}$$

$$\sum_{r=1}^{N_{OCS}} a[r, x_i, y_i, t'] \geq \sum_{t=\max\left(t' - \left\lceil \frac{p_{j_i^T}}{\tau} \right\rceil + 1, 0\right)}^{t'} x[j_i^T, t], \quad \forall j_i^T \in J^T, \ \forall t' \in [0, m-1] \tag{6}$$

$$\sum_{x=0}^{N_S-1} \sum_{y=0}^{N_S-1} a[r, x, y, t'] \leq d_{OCS}, \quad \forall t' \in [0, m-1] \ \forall r \in [1, N_{OCS}] \tag{7}$$

$$\sum_{r=1}^{N_{OCS}} \sum_{x=0}^{N_S-1} a[r, x, y, t'] + a[r, y, x, t'] \leq d_S, \quad \forall t' \in [0, m-1] \ \forall y \in [0, N_S - 1] \tag{8}$$

$$\sum_{x=0}^{N_S-1} a[r, x, y, t'] \leq 1, \quad \forall t' \in [0, m-1] \ \forall r \in [1, N_{OCS}] \ \forall y \in [0, N_S - 1] \tag{9}$$

$$\sum_{y=0}^{N_S-1} a[r, x, y, t'] \leq 1, \quad \forall t' \in [0, m-1] \ \forall r \in [1, N_{OCS}] \ \forall x \in [0, N_S - 1] \tag{10}$$

Optimization Objective. The optimization goal (equation 2) aims to minimize the start time of the virtual task j_v, as it is the last task, thereby effectively reducing the overall training iteration time for the DNN or LLM.

Optimization Constraints. The following optimization constraints guarantee feasibility and ensure adherence to system resource limitations: 1) Task Assignment and Scheduling Constraints: Each transfer task is initiated exactly once (Equation 3), with start times explicitly coordinated via auxiliary scheduling variables (Equation 4). 2) Precedence Constraints: Task dependencies are strictly enforced by Equation 5, ensuring sequential execution of interdependent tasks. 3) Link Availability Constraints: Equation 6 verifies sufficient connectivity between servers during data transfer tasks. 4) Hardware Port Constraints: Port usage is meticulously managed through Equations 7-10. Collectively, these constraints guarantee valid scheduling decisions consistent with system capacities and operational requirements.

As an NP-complete Mixed Integer Programming (MIP) problem, the initial RCPSP-based model formulation involves continuous running-time parameters, potentially increasing computational complexity. To alleviate this, we convert task execution and transfer durations into discrete integer units (e.g., time slots). This integer-based representation substantially accelerates the solver performance, enabling quicker and more scalable optimization for large-scale GPU clusters.

IV. TOPO-X SYSTEM DESIGN

In this section, we detail the system design of our proposed TOPO-X framework, addressing the critical research question: *Can we construct a fine-grained reconfigurable optical network that effectively supports distributed DNN training workloads?*

TOPO-X extends TopoOpt to provide fine-grained optimization for distributed DNN workloads. TOPO-X comprises three core modules: (1) the *Strategy Optimization Module*, (2) the *Topology Optimization Module*, and (3) the *Task Optimization Module*. These modules collaboratively optimize parallelization strategies, network topology configurations, and

precise flow scheduling, resulting in substantial performance improvements.

Strategy Optimization. Given a specific DNN model and training configuration, the Strategy Optimization Module utilizes FlexNet [19] to identify the optimal parallelization approach. FlexNet comprehensively evaluates potential parallelization strategies [12]—including model parallelism [32], [33], data parallelism [11], [34], and hybrid parallelism [4]—to minimize computational overhead and communication latency, ultimately selecting the most efficient configuration.

Topology Optimization. Based on the parallelization strategy identified by FlexNet, the Topology Optimization Module extracts detailed traffic demands corresponding to each communication task. The TOPOLOGYFINDER algorithm [14] dynamically computes an optimal optical network topology tailored to the predicted traffic demands, swiftly adapting to traffic fluctuations across different training stages.

Task Optimization. The Task Optimization Module differentiates TOPO-X from previous solutions such as TopoOpt. Utilizing the optimized network topology, we construct a comprehensive task dependency-directed acyclic graph (DAG) representing all computation and communication tasks explicitly. The Task Optimization Module then develops a fine-grained scheduling plan, carefully coordinating task start times and OCS reconfiguration actions. This synchronized scheduling minimizes iteration latency by aligning task execution precisely with network topology changes.

Task inputs consist of the following categories: 1) Forward tasks, 2) Backward tasks, 3) Communication tasks, 4) Update tasks, 5) Barrier tasks, 6) Nominal communication tasks, and 7) All-reduce tasks.

Each task is described by 1) Execution time: Communication tasks (j_i^T) have transfer time p_i^T, while computation tasks (j_i^C) have computation time p_i^C; 2) Transfer size: Size of data transferred across the network; 3) Transfer direction: Communication tasks (j_i^T) specify source server x_i and destination server y_i; 4) Device: Computing resource on which the task executes; 5) Dependency: Task dependencies based on output requirements from other tasks.

We formulate an optimization model to minimize total training time, introducing a virtual task j_v (with zero execution time), dependent on all other tasks. Minimizing the start time of j_v equates to minimizing iteration duration.

The resulting optimization model is solved using an MIP solver. The solver execution times increase with network scale; thus, we employ solver time limits or accept optimality gaps to expedite solutions for large-scale problems. The optimization model yields detailed task start times and OCS reconfiguration schedules. Using these schedules, OCS devices can proactively adjust mirrors to establish necessary optical links in advance.

The TOPO-X framework leverages FlexNet [19] for parallelism analysis and task graph extraction, integrating Gurobi [35] to efficiently resolve the formulated integer RCPSP. The FlexNet–Gurobi pipeline ensures accurate modeling of distributed training workloads and efficient solution generation, making TOPO-X practically deployable.

Overall, through its integrated modules and advanced optimization techniques, TOPO-X significantly enhances distributed training performance. By simultaneously optimizing flow scheduling, training parallelism strategies, and optical network reconfiguration, TOPO-X delivers robust, adaptive, and high-performance GPU cluster solutions.

V. EXPERIMENTS

We compare TOPO-X against the state-of-the-art co-optimization approach TopoOpt. Our evaluation is conducted using FlexFlow's widely adopted simulator [12], which given a specific DNN model and batch size, explores numerous parallelization strategies and GPU placements to minimize iteration training time. The output from FlexFlow is a comprehensive task graph outlining the computation and communication tasks assigned to each GPU, including detailed dependencies. TOPO-X then utilizes this output (i.e., optimized parallelization strategies, task execution times, transfer sizes, directions, and dependencies) to construct and solve the optimization model using Gurobi, one of the most efficient and widely-used integer optimization solvers available [35].

Our simulation environment represents realistic distributed GPU clusters consisting of N_S servers, each equipped with four NVIDIA A100 GPUs. We systematically vary N_S across different experimental setups and simulate the following network architectures:

- **TOPO-X**: Each server contains d_S Network Interface Cards (NICs) with bandwidth B, interconnected through N_{OCS} OCSs, each having d_{OCS} ports. Thus, the maximum number of single-directional connections at any given time slot is $N_{OCS} \times d_{OCS}/2$. TOPO-X dynamically reconfigures OCS connections in each time slot based on the outputs of our integrated co-optimization model. In our simulations, we set the reconfiguration latency $\tau = 1$ ms, highlighting real-time adaptability.
- **TopoOpt**: Similar to TOPO-X, servers in TopoOpt configurations also have d_S NICs with bandwidth B but are connected through a static flat layer of optical devices. At the start of each job, TopoOpt optimally configures the network topology based on initial traffic heatmaps. However, unlike TOPO-X, TopoOpt maintains its static configuration throughout the entire training process.

We evaluate and compare the iteration training time achieved by TOPO-X and the alternative network architectures in a scenario where the entire GPU cluster is dedicated exclusively to a single DNN training task. Specifically, we use the DLRM [23], a widely adopted, real-world benchmark for evaluating distributed training frameworks. Table II summarizes the DLRM model configurations and batch sizes used in our experiments.

As shown in Table III, the experimental results demonstrate that TOPO-X significantly outperforms the state-of-the-art TopoOpt framework across various network configurations. The performance gains observed in our evaluation stem from TOPO-X's ability to holistically optimize flow scheduling,

TABLE II
MODEL CONFIGURATIONS OF DLRM

No.	Batch/GPU	Dense Layer Size	Feature Layer Size	Embedding Table Size	#Embedding Tables
1	128	2048	4096	128×10^7	64
2	256	1024	2048	256×10^7	16

TABLE III
COMPARISON OF TRAINING ITERATION TIME (MS) BETWEEN TOPOOPT AND TOPO-X ACROSS VARIOUS NETWORK CONFIGURATIONS

No.	N_S	N_{OCS}	d_S	d_{OCS}	B (Gbps)	Batch Size	TopoOpt	Topo-X	Speed Up
1	128	4	4	128	40	128	303.020	150.864	2.01
2	128	6	6	128	40	128	177.487	108.617	1.63
3	128	4	4	128	100	128	126.908	66.093	1.92
4	128	6	6	128	100	128	75.131	52.533	1.43
5	16	4	4	16	10	256	195.958	32.226	6.08
6	16	4	4	16	25	256	89.299	23.278	3.84
7	16	4	4	16	40	256	61.517	21.236	2.90
8	16	4	4	16	100	256	27.974	20.227	1.38
9	16	4	4	16	200	256	16.623	16.623	1.00
10	16	8	8	16	10	256	83.417	23.226	3.60
11	16	8	8	16	25	256	37.610	18.796	2.00
12	16	8	8	16	40	256	24.504	18.364	1.33
13	16	8	8	16	100	256	15.832	15.832	1.00
14	16	8	8	16	200	256	14.667	14.667	1.00

topology reconfiguration, and parallelism strategies. We provide a detailed analysis of the key contributing factors to these improvements.

Adaptive Topology Reconfiguration. Unlike TopoOpt, which relies on static network topology at the start of training, TOPO-X continuously adapts to dynamic communication patterns across different training stages. This adaptability ensures that OCS resources are efficiently allocated to high-priority data transfers, reducing network contention and improving bandwidth utilization.

Fine-Grained Flow Scheduling. TOPO-X introduces an explicit flow scheduling mechanism that strategically prioritizes critical data transfers within each training iteration. By aligning data movement with computation dependencies, TOPO-X minimizes idle times and maximizes GPU utilization. This is evident in the cases with limited OCS and network interface resources (e.g., setups with small d_S and d_{OCS} values), where TOPO-X achieves up to a 6.08× speedup over TopoOpt by avoiding bottlenecks in gradient communication.

Scalability Considerations. The experimental findings suggest that TOPO-X scales efficiently across different GPU cluster sizes and network conditions. The performance improvements remain consistent even as N_S and N_{OCS} increase. Notably, TOPO-X exhibits diminishing returns in extremely high-bandwidth configurations (e.g., cases with $B \geq 100$ Gbps), where network congestion is inherently lower. However, even in these scenarios, TOPO-X matches or slightly outperforms TopoOpt by optimizing task scheduling and minimizing redundant data transfers.

VI. RELATED WORK

We review recent efforts to optimize distributed ML workloads, focusing on three key areas: 1) network topology, 2) training parallelism, and 3) scheduling strategies.

Network Topology Optimization. Reducing communication overhead through better topology design is central to cluster efficiency. TopoOpt [14] co-optimizes topology and parallelization strategies, reducing DNN training time. TPU v4 [36] adopts optical switches and flexible interconnects like twisted 3D torus to improve performance and scalability. Taurus [37] employs adaptive, topology-aware techniques to reduce latency and increase throughput.

Training Parallelism Optimization. Deep learning frameworks such as MXNet [38] support data, model, and hybrid parallelism. However, selecting optimal strategies remains nontrivial. FlexFlow [12], ColocRL [20], and MERLIN automate parallelization using search or RL-based methods.

Scheduling Optimization. Effective scheduling improves utilization and minimizes training time. Cassini [21] incorporates network-aware placement; Optimus [22] uses performance models for adaptive scheduling. Themis [39] promotes fair GPU sharing, while TE-CCL [40] enhances communication efficiency via traffic engineering.

VII. CONCLUSION

This paper introduces TOPO-X, a solver-based framework that co-optimizes flow scheduling, training parallelism, and optical topology to accelerate distributed DNN training. Modeled as an RCPSP, TOPO-X efficiently manages task execution and network reconfiguration, yielding significant speedups over state-of-the-art methods. Future work includes validating on real GPU clusters, refining communication modeling beyond the current simplified ring all-reduce model, and developing scalable heuristics to tackle the NP-hard nature of RCPSP for larger deployments. These steps are crucial for enabling broader adoption in large-scale AI training systems.

ACKNOWLEDGMENT

The research is partially supported by National Key R&D Program of China under Grant 2021ZD0110400, Anhui Provincial Natural Science Foundation under Grant 2208085MF172, Innovation Program for Quantum Science and Technology 2021ZD0302900 and China National Natural Science Foundation with No. 62132018, 62231015, "Pioneer" and "Leading Goose" R&D Program of Zhejiang, 2023C01029, and 2023C01143. We are also thankful to Yijun Sun for his helpful suggestions and assistance.

REFERENCES

[1] OpenAI, "Chatgpt: Transforming text generation with deep learning," *OpenAI Blog*, 2022. [Online]. Available: https://openai.com/blog/chatgpt

[2] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM computer communication review*, vol. 38, no. 4, pp. 63–74, 2008.

[6] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 63–74.

[7] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 435–446, 2013.

[8] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 267–280.

[9] P. X. Gao, A. Narayan, S. Karandikar *et al.*, "Network requirements for resource disaggregation," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 249–264.

[10] M. Cho, U. Finkler, D. Kung, and H. Hunter, "Blueconnect: Decomposing all-reduce for deep learning on heterogeneous network hierarchy," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 241–251, 2019.

[11] Y. Huang, Y. Cheng, A. Bapna *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *Advances in neural information processing systems*, vol. 32, 2019.

[12] Z. Jia, M. A. Zaharia, and A. Aiken, "Beyond data and model parallelism for deep neural networks," *ArXiv*, vol. abs/1807.05358, 2018.

[13] D. Mudigere, Y. Hao, J. Huang *et al.*, "Software-hardware co-design for fast and scalable training of deep learning recommendation models," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 993–1011.

[14] W. Wang, M. Khazraee, Z. Zhong *et al.*, "TopoOpt: Co-optimizing network topology and parallelization strategy for distributed training jobs," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, 2023, pp. 739–767.

[15] E. Chung, J. Fowers, K. Ovtcharov *et al.*, "Accelerating persistent neural networks at datacenter scale," in *Hot Chips*, vol. 29, 2017.

[16] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[17] H. Pan, Z. Li, J. Dong, Z. Cao, T. Lan, D. Zhang, G. Tyson, and G. Xie, "Dissecting the communication latency in distributed deep sparse learning," in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 528–534.

[18] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "Pipedream: Generalized pipeline parallelism for dnn training," in *Proceedings of the 27th ACM symposium on operating systems principles*, 2019, pp. 1–15.

[19] X. Miao, G. Oliaro, Z. Zhang *et al.*, "Specinfer: Accelerating large language model serving with tree-based speculative inference and verification," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. Association for Computing Machinery, 2024, p. 932–949.

[20] A. Mirhoseini, H. Pham, Q. V. Le *et al.*, "Device placement optimization with reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR.org, 2017, p. 2430–2439.

[21] S. Rajasekaran, M. Ghobadi, and A. Akella, "CASSINI: Network-Aware job scheduling in machine learning clusters," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Apr. 2024, pp. 1403–1420.

[22] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: an efficient dynamic resource scheduler for deep learning clusters," in *Proceedings of the Thirteenth EuroSys Conference*, 2018.

[23] Meta Research, "Deep learning recommendation model for personalization and recommendation systems," 2022.

[24] N. Farrington, G. Porter, S. Radhakrishnan *et al.*, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 339–350.

[25] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "Projector: Agile reconfigurable data center interconnect," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 216–229.

[26] S. Rajbhandari, O. Ruwase, J. Rasley, S. Smith, and Y. He, "Zero-infinity: breaking the gpu memory wall for extreme scale deep learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Association for Computing Machinery, 2021.

[27] J. Brownlee, *Better deep learning: train faster, reduce overfitting, and make better predictions*. Machine learning mastery, 2018.

[28] X. Li, "On scheduling optical packet switches with reconfiguration delay," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 7, pp. 1156–1164, 2003.

[29] M. Zhang, R. N. Mysore, S. Supittayapornpong, and R. Govindan, "Understanding lifecycle management complexity of datacenter topologies," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019, pp. 235–254.

[30] J. Ma and F. Wu, "Learning to coordinate traffic signals with adaptive network partition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 1, pp. 263–274, 2023.

[31] "Polatis optical circuit switch," 2022. [Online]. Available: http://www.polatis.com/series-7000-384x384-port-software-controlled-optical-circuit-switch-sdn-enabled.asp

[32] Z. Jia, S. Lin, C. R. Qi, and A. Aiken, "Exploring hidden dimensions in accelerating convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2274–2283.

[33] S. Lee, J. K. Kim, X. Zheng *et al.*, "On model parallelization and scheduling strategies for distributed machine learning," *Advances in neural information processing systems*, vol. 27, 2014.

[34] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[35] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[36] N. Jouppi, G. Kurian, S. Li *et al.*, "Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*. Association for Computing Machinery, 2023.

[37] T. Swamy, A. Rucker, M. Shahbaz, I. Gaur, and K. Olukotun, "Taurus: a data plane architecture for per-packet ml," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 1099–1114.

[38] T. Chen, M. Li, Y. Li *et al.*, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015.

[39] K. Mahajan, A. Balasubramanian, A. Singhvi *et al.*, "Themis: Fair and efficient GPU cluster scheduling," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, 2020, pp. 289–304.

[40] X. Liu, B. Arzani, S. K. R. Kakarla *et al.*, "Rethinking machine learning collective communication as a multi-commodity flow problem," in *Proceedings of the ACM SIGCOMM 2024 Conference*. Association for Computing Machinery, 2024, p. 16–37.