CTCCL: Cost-Efficient Joint Device-Network Load Balancing for LLM Training in RoCE-based Intelligent Computing Network

Zhuotong Li
State Cloud
China Telecom
Shanghai, China
lizt20@chinatelecom.cn

Liang Xu
State Cloud
China Telecom
Shanghai, China
xul13@chinatelecom.cn

Ziqi Huang
State Cloud
China Telecom
Shanghai, China
huangzq31@chinatelecom.cn

Shuyun Qian
State Cloud
China Telecom
Shanghai, China
qiansy@chinatelecom.cn

Hongwei Bu
State Cloud
China Telecom
Shanghai, China
buhw@chinatelecom.cn

Ming Yang
State Cloud
China Telecom
Shanghai, China
yangm37@chinatelecom.cn

Mengyun Luan
State Cloud
China Telecom
Shanghai, China
luanmy@chinatelecom.cn

Weiguo Chen
State Cloud
China Telecom
Shanghai, China
chenwg6@chinatelecom.cn

Xu Wen
State Cloud
China Telecom
Shanghai, China
wenxu@chinatelecom.cn

Abstract

Pre-training large language models (LLMs) in data centers (DCs) is a complex yet essential task that requires vast computational resources and carefully designed infrastructures to enable efficient, large-scale distributed learning. However, without effective load balancing in RDMA over Converged Ethernet (RoCE) networks, network congestion and latency can create significant bottlenecks, disrupting data transmission, reducing resource utilization, and prolonging training times, ultimately compromising the scalability and performance of LLM training. To address these challenges, we propose and develop an innovative and cost-effective joint Device-Network Load Balancing (DNLB) approach. Built on our custom collective communication library, CTCCL, DNLB coordinates device-side flow optimization with network-side switch configuration to mitigate hash collisions and alleviate congestion. By managing UDP source port numbers and distributing traffic uniformly across multiple paths, DNLB achieves efficient load balancing without requiring hardware modifications. Moreover, DNLB demonstrates exceptional adaptability and fault tolerance, ensuring compatibility with diverse network architectures, task modes, and device deployments. Even in the event of uplink failures, it maintains robust network load balancing. Experiments conducted in real-world intelligent computing DCs demonstrate that DNLB improves communication efficiency by up to 40% and enhances overall pre-training efficiency of LLMs by 7%, offering a practical, scalable solution for modern LLM training.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICS '25, Salt Lake City, UT, USA

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1537-2/25/06
https://doi.org/10.1145/3721145.3725772

CCS Concepts

• **Computing methodologies** → **Distributed computing methodologies**; **Distributed artificial intelligence**; • **Networks** → **Network algorithms**.

Keywords

LLMs, RoCE, Collective Communication Library, Load Balancing

ACM Reference Format:
Zhuotong Li, Liang Xu, Ziqi Huang, Shuyun Qian, Hongwei Bu, Ming Yang, Mengyun Luan, Weiguo Chen, and Xu Wen. 2025.

CTCCL: Cost-Efficient Joint Device-Network Load Balancing for LLM Training in RoCE-based Intelligent Computing Network. In *2025 International Conference on Supercomputing (ICS '25), June 08–11, 2025, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3721145.3725772

1 Introduction

Since 2022, the demand for Large Language Models (LLMs) such as GPT [28], Gemini [30], and Llama [7] has reached unprecedented levels, driving transformative changes across various industries. These models, known for their exceptional ability to understand and generate human-like text, have become central to a wide range of applications, including chatbots, AI assistants, content creation tools, and beyond. As industries increasingly recognize their value, investments in LLM development and deployment have surged dramatically. Market projections highlight the remarkable trajectory of this technology, estimating that the global LLM market will grow from $1.59 billion in 2023 to a staggering $259.8 billion by 2030 [32]. This rapid growth is mirrored by a surge in demand for computational infrastructure capable of supporting LLM pre-training and deployment.

The exponential expansion of the LLM market places immense pressure on DC infrastructure, which must keep pace with this technological evolution. Advanced intelligent computing DCs equipped with ten of thousands of GPUs have become the backbone of LLM development, yet they face mounting challenges in meeting the resource-intensive requirements of these models. Unlike traditional distributed AI models, LLMs operate on an immense scale, comprising millions—and in some instances, billions—of parameters. This unparalleled complexity demands vast computational and network resources, particularly during the pre-training phase. LLM pre-training generates unique network traffic patterns characterized by low entropy, high burstiness, and the prevalence of elephant flows. These patterns exacerbate resource contention within DCs, where synchronization among GPUs relies heavily on collective communication. This communication process, although periodic and predictable, can drive data transmission to the line rate of Network Interface Cards (NICs) almost instantaneously, creating a highly dynamic and demanding network environment. As a result, bottlenecks in data transmission can critically hinder the training performance of LLMs, underscoring the need for innovative solutions in network optimization. Addressing these challenges is essential for maximizing efficiency and scalability in LLM training and ensuring that DCs can keep pace with the exponential growth of AI workloads.

Remote Direct Memory Access (RDMA) plays a pivotal role in accelerating data transfer rates for computationally intensive workloads like LLM pre-training. By bypassing the CPU and enabling direct memory access between nodes, RDMA significantly reduces latency and enhances overall efficiency. Historically, InfiniBand (IB) has been the gold standard for RDMA, offering unparalleled performance. However, its high cost has made it less feasible for widespread adoption in modern DCs. In recent years, RDMA over Converged Ethernet (RoCE) has emerged as a compelling alternative, especially in intelligent computing DCs focused on LLM pre-training [12, 34]. RoCE delivers comparable performance to IB while utilizing existing Ethernet infrastructure, making it a more cost-effective and scalable choice.

Despite its advantages, RoCE faces challenges when handling the demanding traffic patterns associated with LLM workloads. The prevalence of bursty, high-bandwidth elephant flows can lead to single-point congestion, undermining the overall effectiveness of the training process. Such bottlenecks emphasize the critical need for advanced load balancing mechanisms. One commonly employed strategy in general DCs is Equal-Cost Multi-Path (ECMP), a technique that effectively distributes network flows across multiple paths of equal cost. ECMP determines the optimal path for each flow by hashing its five-tuple attributes, including the source IP address, destination IP address, source port, destination port, and transport protocol. While this approach is computationally efficient and widely adopted, its static hash-based algorithm poses inherent limitations. In low-entropy flow scenarios, as commonly observed during LLM training, these limitations become particularly pronounced. Flows are frequently initiated in close temporal proximity and tend to have similar sizes, increasing the likelihood of hash collisions. When multiple flows are mapped to the same path, it leads to uneven load distribution and localized congestion, negating the benefits of ECMP [36].

To mitigate the challenges posed by low flow entropy, numerous load balancing strategies have been developed, particularly focusing on enabling RDMA multi-path transmission. Among these, techniques like fine-grained packet spraying [1] and MP-RDMA [3, 16] achieve load balancing by segmenting large elephant flows at the packet level and distributing them across multiple network paths. While effective in balancing the load, these methods frequently lead to packet out-of-order (OoO), necessitating the use of specialized hardware, such as NVIDIA's BlueField-3 DPU, to reorder packets [4, 11]. This hardware dependency introduces additional processing overhead and can degrade transmission efficiency, particularly in latency-sensitive environments. Meta initially attempted to deploy a path-pinning scheme, routing packets to specific paths based on destination slices [8]. However, this approach lacked adaptability and failed to ensure load balancing under differentiated parallel tasks or failure scenarios. Some alternative approaches [17, 31, 38] involve leveraging multi-Queue Pair (QP) transmission at the traffic source, a method commonly adopted in collective

communication libraries for distributed training. By ensuring that traffic originating from different QPs is transmitted along distinct network paths, this technique minimizes contention and congestion. However, implementing such a solution usually requires secondary development or adaptation and deployment of heterogeneous network devices, including modifications to RDMA NICs, switches, or the integration of dedicated traffic controllers to achieve precise distribution of flows in multi-QPs. While effective, these intricate designs substantially increase both the complexity and cost of equipment development and deployment, presenting a barrier to widespread adoption in cost-sensitive environments.

Therefore, we designed and implemented a cost-effective and efficient solution, joint Device-Network Load Balancing (DNLB). Efficient load balancing is realized through the coordination of UDP source port number control in CTCCL and policy-based routing (PBR) in network switches. Unlike existing approaches, DNLB seamlessly integrates into existing infrastructure with minimal additional cost. It offers strong compatibility with various networks and concurrent training tasks while maintaining effective load balancing even in the presence of failures. Our key contributions are summarized as follows:

- We developed DNLB, a cost-effective, highly adaptable and fault-tolerant load balancing solution implemented through our custom collective communication library, CTCCL. Combined with straightforward switch configurations, DNLB efficiently disperses low-entropy flows across multiple physical links, making it effective for RoCE-based intelligent computing networks.
- We designed a UDP port segmentation algorithm for LLM pre-training flows. This algorithm, characterized by its adaptability, matches UDP source ports with PBR rules in switches, achieving balanced flow distribution and ensuring effective load balancing across diverse scenarios and network environments.
- We constructed an experimental testbed with standard GPUs and RoCE-based networking to evaluate CTCCL and DNLB. Results demonstrated state-of-the-art improvements in collective communication efficiency and LLM pre-training performance even after failures, offering valuable insights for large-scale distributed AI training in the industry.

The remainder of this paper is structured as follows. Section 2 delves into the background and motivation that underpin our work, providing the necessary context and highlighting the challenges addressed. Section 3 offers a comprehensive overview of the proposed DNLB approach, which is built upon our custom collective communication library, CTCCL. This section also introduces the innovative UDP source port number calculation algorithm, detailing its implementation on both the device and network sides. Evaluation results are shown and discussed in Section 4, which is followed by a summary of related work in Section 5. Section 6 concludes this paper.

2 Background and Motivation

2.1 LLM, RoCE and Collective Communication

Pre-training LLMs is highly resource intensive, often involving models with parameter counts ranging from billions to trillions, which imposes significant challenges on memory and computational resources. In addition to the immense processing power required, it typically necessitates the integration of efficient parallel strategies across distributed computing environments. To address these demands, frameworks for **Data Parallelism (DP), Pipeline Parallelism (PP), and Tensor Parallelism (TP)** have been developed, each leveraging distinct aspects of hardware resources to facilitate the efficient training of LLMs.

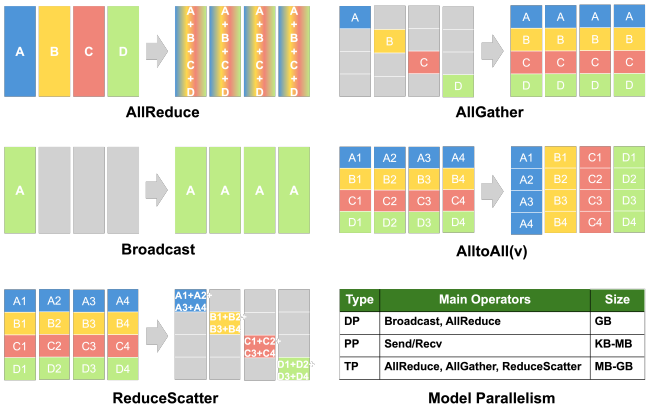


Figure 1: Distributed training collective communication operators and LLM parallelism.

In **Data Parallelism (DP)**, training data is distributed across multiple devices, each with a full replica of the model. Gradients computed on each device are averaged and synchronized using collective communication operations such as all-reduce, a process that ensures model consistency across devices while balancing the computational load. **Pipeline Parallelism (PP)**, on the other hand, partitions the model itself across multiple devices, enabling sequential processing through each model segment. While this approach helps distribute memory requirements, it may lead to suboptimal resource utilization, as devices can experience idle times while waiting for data from previous stages. Lastly, **Tensor**

Parallelism (TP) further divides the model's tensor computations across multiple devices, allowing each device to compute a part of the larger model tensor—a strategy critical for handling exceedingly large model architectures where single-device memory is insufficient. Some state-of-the-art frameworks, such as Megatron-LM [29], DeepSpeed [27], and Alpa [40], support the use of these parallelism strategies, as well as the 3D parallelism strategy that integrates all three types, enabling large-scale training of standard LLMs.

To synchronize the vast amounts of data required for 3D parallelism, collective communication plays a fundamental role. Collective operations such as **AllReduce (used for aggregating gradients), AlltoAll (facilitating data sharing across all devices), Broadcast (disseminating data from a central source to multiple devices) and ReduceScatter (aggregate data from all processes and distribute it to each device)** enable these distributed frameworks to operate cohesively. Figure 1 illustrates the principles of various collective communication operators and highlights the key operators commonly used in different parallel frameworks. Several high-performance libraries have been developed to implement these operations, each tailored to maximize efficiency under different conditions. Notable among these are NCCL (NVIDIA Collective Communication Library) [21], ACCL (Alibaba Collective Communication Library) [5], and Gloo (Meta's communication library) [18], which optimize communication patterns based on hardware architecture and load conditions.

RDMA significantly enhances the efficiency of collective communication operations by enabling direct access to remote memory through zero-copy and kernel bypass mechanisms. This reduces latency, improves throughput, and maximizes bandwidth utilization, making it ideal for large-scale distributed training environments. However, the Go-back-N mechanism inherent to RDMA makes it sensitive to packet loss and OoO delivery, which can substantially degrade performance [20]. Leading collective communication libraries like NCCL offer configurable environment variables to optimize RDMA features on IB or RoCE networks. For instance, Adaptive Routing (AR) in IB helps dynamically alleviate link congestion while maintaining packet order [35]. RoCE, in contrast, has gained popularity due to its cost efficiency and ability to operate on Ethernet infrastructure, although it faces greater challenges related to congestion, packet loss, and OoO delivery due to Ethernet's non-lossless nature. As a result, RoCE development in industries has diverged into two distinct paths: 1)lossless RoCE, which ensures zero packet loss through flow control mechanisms like Priority Flow Control (PFC) and Explicit Congestion Notification (ECN)-based congestion management, and 2)lossy RoCE, which employs techniques such as dynamic congestion control to tolerate packet loss while balancing performance [6].

2.2 Load Balancing Challenges

Both lossless and lossy RoCE rely heavily on congestion control and load balancing mechanisms to enhance network performance. Distributed training paradigms such as 3D parallelism involve frequent collective operations like AllReduce, AllGather and ReduceScatter. These operations, critical for synchronizing gradients and parameters across devices, often result in many-to-many and one-to-many data exchanges. Consequently, they lead to bandwidth spikes and highly variable, bursty traffic patterns, as shown in Fig. 2(a). During the backpropagation (BP) in each mini-batch of 3D parallelism, multiple DP, TP, and PP gradient synchronization processes are involved. This periodic low-entropy traffic poses substantial challenges to traditional load-balancing strategies, particularly ECMP.

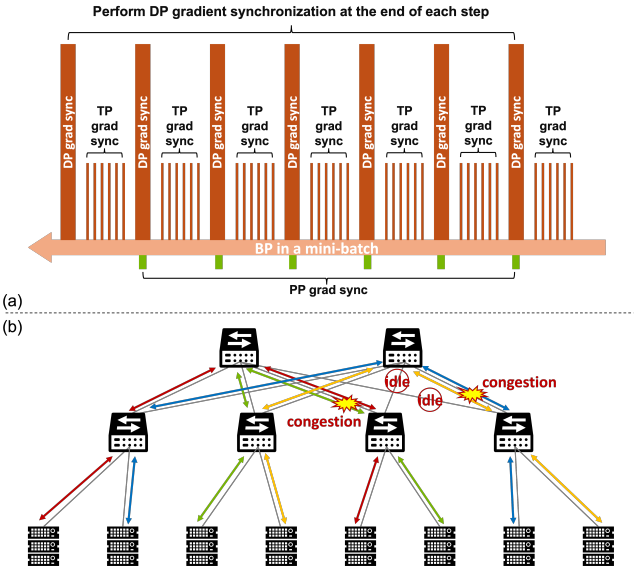


Figure 2: LLM pre-training: (a) Periodic traffic patterns in 3D parallel training (DP, TP, PP); (b) Challenges of ECMP.

ECMP, a widely adopted routing mechanism in Ethernet-based DCs, balances traffic across multiple equal-cost paths by employing a hash-based flow allocation mechanism. This mechanism typically leverages a five-tuple (source IP, destination IP, source port, destination port, and protocol) to uniquely identify flows. By hashing these fields, ECMP distributes flows across paths, achieving effective load balancing under diverse, high-entropy traffic conditions. However, during LLM pretraining, the low-entropy traffic patterns generated by collective communication often result in disproportionate hash value clustering, leading to hash collisions [37]. These collisions cause severe congestion on specific paths while leaving others underutilized (shown in Fig. 2(b)),

thereby degrading network throughput and significantly increasing the latency of collective communication operations.

To address the challenge of low flow entropy, fine-grained packet spraying has emerged as a principal load balancing technique in DCs [4]. This method distributes data packets randomly across multiple equal-cost paths, thereby increasing flow entropy and mitigating the load imposed by low-entropy elephant flows. Similarly, MP-RDMA regulates transmission paths by specifying UDP source ports within data packets [31]. However, both approaches are subject to unavoidable processing overhead resulting from OoO packets. The path-pinning scheme initially implemented by Meta was discontinued due to its limited adaptability to a variety of concurrent tasks and network failures within a single rack, coupled with its failure to ensure uniform traffic distribution [8].

Collective communication libraries provide fine-grained control over network parameters to address congestion. For example, enabling multi-QP transmission enhances path diversity and mitigates single-flow bottlenecks. By splitting data across multiple QPs, RoCE-based systems achieve multi-path routing while maintaining flow order—a critical requirement for ensuring performance in congested environments. Leading companies such as Meta [8], Alibaba [5, 24] and Tencent [15] have implemented solutions that either optimize ECMP/Enhanced ECMP(E-ECMP) or develop advanced routing control algorithms to manage and distribute QP flows effectively.

However, cost considerations remain a critical constraint, particularly in industrial applications and cloud deployments. In real-world scenarios, achieving homogeneous network device configurations, especially for switches, is often impractical within a single AI cluster. Commercial and technical constraints frequently necessitate deploying heterogeneous infrastructures comprising switches, GPUs, and NICs from multiple vendors [10]. Modifying ECMP algorithms, switch routing policies, or NIC hardware for large-scale, heterogeneous deployments significantly increases R&D, operational, and commercial costs. **Consequently, general-purpose software-based solutions that require minimal hardware modifications and ensure broad compatibility and fault tolerance are often preferred.** Developing a cost-efficient load-balancing strategy tailored to the unique traffic patterns of LLM pretraining remains an urgent and essential research challenge.

Table 1: Our Notation

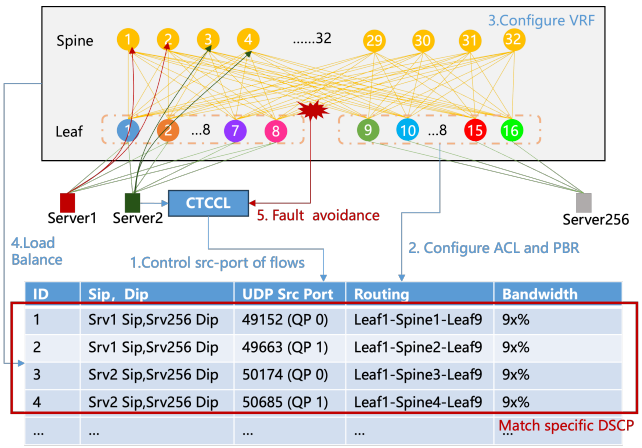
Variable	Description
$N_{i,j}$	i-th NIC on device j
$nqp^{\text{per\_conn}}$	Total number of QPs on each connection, obtained from the environment variable NCCL_IB_QPS_PER_CONNECTION
$q_{N_{i,j}}$	Index of QP in a single connection on the j-th NIC of the device i, $q_{N_{i,j}} \leq nqp^{\text{per\_conn}}$
$nlink^{\text{up}}$	Number of uplink ports on each Leaf switch
$nlink^{\text{down}}$	Number of downlink ports on each Leaf switch
$index_{N_{i,j}}$	Index of the j-th network card on device i
$nNIC^{\text{Leaf}}$	Number of NICs connected to the same Leaf
$step$	Step size for adjusting the UDP source port between different QPs
$sport^{\text{UDP}}_{N_{i,j},q_{N_{i,j}}}$	UDP source port number of the flow in the $q_{N_{i,j}}$-th QP of a single connection on the j-th network card of device i

3 Joint Device-Network Load Balancing (DNLB)

3.1 Overview of DNLB

In practice, the structure of intelligent computing networks is intentionally designed to remain stable and predictable during normal operations. These networks typically rely on fixed, well-defined topologies, such as Clos or Fat-Tree, which are specifically chosen to ensure consistent performance and simplify network management. The traffic patterns within these networks tend to be relatively uniform, with steady-state behavior prevailing under normal conditions. Adjustments to the network topology or alterations in traffic patterns are infrequent and generally occur only in response to unforeseen disruptions, such as hardware failures or link outages. Given this inherent stability, targeting load balancing through the static pre-configuration of collective communication traffic patterns—whether during normal steady-state operations or following recovery from failures—proves to be a more cost-effective strategy than relying on dynamic traffic adjustments. The latter would often necessitate extensive and costly network hardware upgrades or complex optimizations, making static configuration a more practical and scalable solution for most intelligent computing environments.

To address load balancing challenges, we propose a simplified and cost-effective DNLB scheme that utilizes multi-QP transmission within the collective communication library on the device side, coupled with minimal and simple network switch configurations. This solution effectively splits the transmission of elephant flows across multiple switch uplinks to evenly distribute the load. The streamlined pipeline of DNLB is illustrated in Fig. 3.



ID	Sip, Dip	UDP Src Port	Routing	Bandwidth
1	Srv1 Sip,Srv256 Dip	49152 (QP 0)	Leaf1-Spine1-Leaf9	9x%
2	Srv1 Sip,Srv256 Dip	49663 (QP 1)	Leaf1-Spine2-Leaf9	9x%
3	Srv2 Sip,Srv256 Dip	50174 (QP 0)	Leaf1-Spine3-Leaf9	9x%
4	Srv2 Sip,Srv256 Dip	50685 (QP 1)	Leaf1-Spine4-Leaf9	9x%
...	...	...	...	...

Figure 3: Cost-efficient Joint Device-Network Load Balancing (DNLB) approach.

On the device side, we have developed a custom collective communication library, CTCCL, building upon the foundation of NCCL. Within CTCCL, we introduced a specialized flow control module that manages the UDP source port of transmission flows when initializing QPs. This module ensures that each QP's flows on every NIC within the server is assigned continuous, unique UDP source ports. When enabled, this module also assigns a specific Differentiated Services Code Point (DSCP) to the flows, allowing the large model training traffic to be explicitly forwarded by DNLB. Flows with other DSCP values, typically those not associated with LLM training, are forwarded using standard ECMP routing. Additionally, we have implemented a fault avoidance mechanism in CTCCL that monitors and detects the status of QP communication. In the event of uplink failures on leaf switches, CTCCL redistributes the collective communication data to the functioning QPs, ensuring that load balancing is maintained across the operational uplinks. On the network side, Leaf switches are configured with access control lists (ACL) and PBR rules that match the DSCP values of the incoming traffic. These rules divide the full range of UDP source port numbers and map them to upstream forwarding links, directing flows with specific source ports to distinct upstream links leading to the Spine switches. Additionally, Spine switches can implement virtual routing and forwarding (VRF) as required, enabling the creation of virtual subnets to isolate and segregate different types of traffic. By carefully controlling the UDP source port numbers and applying ACL/PBR rules to the flows, the DNLB scheme ensures that all QP traffic from every NIC connected to a single Leaf switch is evenly distributed across the available uplinks. This approach not only achieves effective network load balancing but also prevents congestion that could arise from hash conflicts, thereby enhancing overall system performance and scalability.

3.2 Segmentation of UDP Source Port

RoCE specifies that the valid range of UDP source port numbers for flows lies between [0xC000, 0xFFFF]. The fundamental mechanism of DNLB revolves around achieving effective load balancing by selecting UDP source port numbers within this range in an even and discrete manner. These carefully assigned port numbers ensure that flows are distributed across multiple uplinks, thereby mitigating congestion and improving network efficiency. The notation used in this work is summarized in Table 1.

In our approach, each QP on a NIC connected to the same Leaf switch is treated as the minimum discrete unit of granularity for flow segmentation. Consequently, the incremental step size for UDP source port numbers assigned to flows transmitted by adjacent QPs is defined as:

$$step = \frac{0\text{xFFFF} - 0\text{xC000} + 1}{nlink^{\text{up}}} \tag{1}$$

On the network side, the ACLs configured on the Leaf switches should evenly partition the total range of UDP source port numbers into $nlink^{\text{up}}$ segments. Each ACL division interval is defined as:

$$[0\text{xC000} + n * step, 0\text{xC000} + (n+1) * step] \tag{2}$$

where $n \in [0, nlink^{\text{up}} - 1]$.

Correspondingly, PBR rules are applied to ensure that flows with UDP source port numbers within a specific range are forwarded to their designated uplink ports.

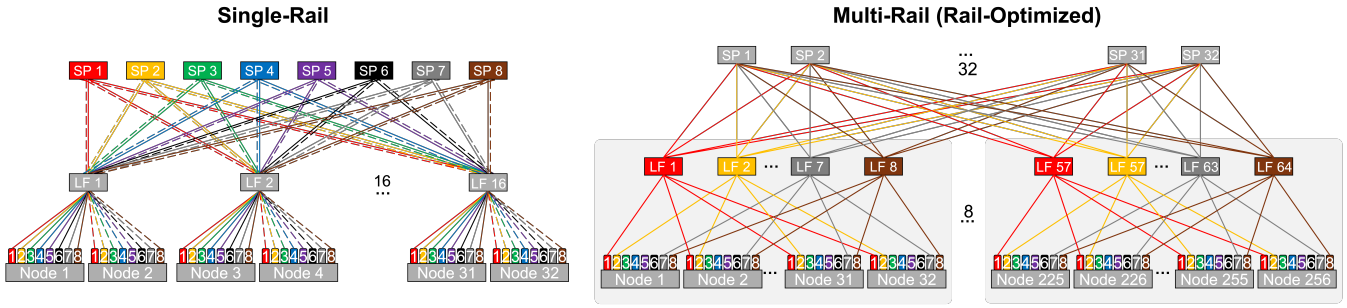
On the device side, the UDP source port numbers for flows transmitted by each QP are calculated within CTCCL based on the NIC index and the QP number, which ensures a systematic and distributed assignment of source port numbers, enabling balanced traffic distribution across network links:

$$sport^{\text{UDP}} = 0\text{xC000} + \left(index_{N_{i,j}} * nqp^{\text{per\_conn}} + q_{N_{i,j}}\right) * step\%\,(0\text{xFFFF} - 0\text{xC000} + 1) \tag{3}$$

where $index_{N_{i,j}}$ could be the host number of the IP address of the j-th NIC on device i.

3.3 Adaptability and Fault Tolerance

DNLB exhibits excellent compatibility, enabling seamless integration with various intelligent computing network architectures and deployment scenarios. To achieve optimal load balancing, it is essential that the product of the number of NICs connected to the same Leaf switch on the selected training host/task and the number of QPs configured in CTCCL



	Single-Rail			Multi-Rail (Rail-Optimized)		
NIC	Node 1 NIC 1	Node 1 NIC 2	Node 1 NIC 3	Node 1 NIC 1	Node 2 NIC 1	Node 3 NIC 1
Index (IP)	xxx.xxx.xxx.1	xxx.xxx.xxx.2	xxx.xxx.xxx.3	xxx.xxx.xxx.1	xxx.xxx.xxx.2	xxx.xxx.xxx.3

Figure 4: Adaptability of DNLB under single/multi-rail GPU clusters.

meets or exceeds the number of uplink ports on the Leaf. This requirement is typically straightforward to satisfy in most large-scale LLM training tasks, given the resource-intensive nature of these workloads and their associated complex hardware configurations:

$$nNIC^{\text{Leaf}} * nqp^{\text{per\_conn}} \geq nlink^{\text{up}} \tag{4}$$

Given this constraint, DNLB ensures consistent uplink traffic even with multiple concurrent training tasks, where some tasks require uplink resources while others do not. Unlike Meta's path-pinning solution, DNLB effectively avoids uneven uplink traffic resulting from differing task requirements on Spine.

Another significant advantage of DNLB lies in its flexibility with respect to NIC indexing. It only requires that the indices of the multiple NICs connected to the same Leaf switch form a continuously increasing sequence. This holds true regardless of whether the network utilizes a single-rail or multi-rail (rail-optimized) structure, as illustrated in Fig. 4. While many contemporary GPU clusters implement rail-optimized networking to enhance performance, certain deployments still rely on single-rail configurations due to the unique requirements of specific GPU models. A practical method for fulfilling this indexing requirement is by utilizing the host portion of the NIC's IP address. In the initial setup of intelligent computing networks, NICs connected to the same Leaf switch are typically assigned sequential IP addresses, making this approach both efficient and easy to implement.

Also, DNLB is designed as a minimalist and cost-effective solution, ensuring its broad applicability and resilience across diverse server deployments. It remains unaffected by variables such as GPU types (e.g., A100, H100, or L40S), the presence of NvLink, or the number of NICs per server. Furthermore, DNLB eliminates the need to account for the number of ranks/tasks concurrently engaged in computation and data transmission within the actual network deployment. This inherent versatility guarantees that DNLB can be deployed in a wide array of intelligent computing environments, making it a reliable choice for optimizing load balancing and enhancing the overall efficiency of large-scale computational systems.

Moreover, DNLB demonstrates robust fault tolerance. In the event of uplink failures, CTCCL detects the status of the affected QPs and reroutes the failed transmission data to other available, functional QPs, while preserving the UDP source port of flows in the operational QPs. Consequently, load balancing across the operational uplinks is maintained.

4 Evaluation

4.1 Experimental Testbed

In this work, we present a comprehensive performance comparison between the proposed CTCCL-based DNLB and the widely used NCCL (version: 2.19.4) combined ECMP, evaluating performance across multiple dimensions. These include collective communication benchmarks conducted using the NCCL-Tests tool, as well as LLM pre-training. As illustrated in Fig. 5(a), our experimental setup consists of a cluster of eight GPU servers, each configured with two Non-Uniform Memory Access (NUMA) structures. Each NUMA structure is equipped with one CPU, four L40S GPUs, and one Mellanox NIC (CX-7/BF3 operating at 400 Gbps).

Within each L40S server, internal communication between components within the same NUMA is facilitated via PCIe Gen 4, ensuring high-speed, low-latency data transfer. The servers are connected to the network through two NICs, each linked to a pair of Leaf switches via multi-rail, with switch ports operating at 400 Gbps. These Leaf switches aggregate the server traffic, with each Leaf switch providing eight uplink ports that are divided into two groups of

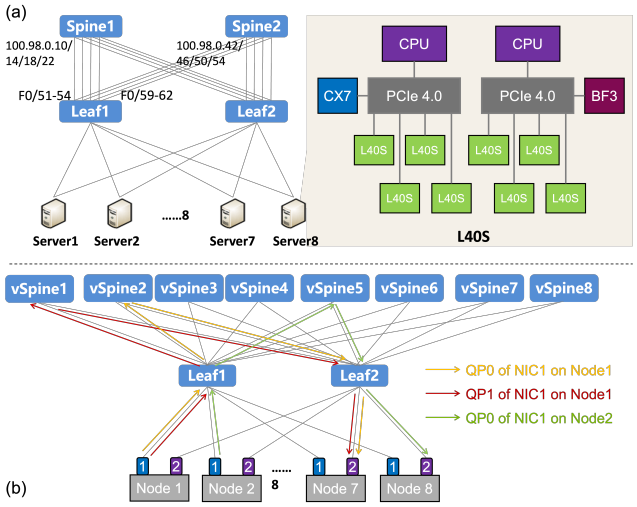


Figure 5: Experimental testbed: (a) physical network structure; (b) logical network structure.

four. The uplink ports are connected to two Spine switches (FHGigabitEthernet0/51-54, FHGigabitEthernet0/59-62), establishing a scalable, high-throughput network topology. When DNLB is enabled, the ACL rules and PBR configured on each Leaf switch divide the total range of UDP port numbers [0xC000, 0xFFFF] into eight equal segments. These rules ensure that traffic flows with DSCP values matching the DNLB configuration, and UDP port numbers falling within a specific segment, are selectively allowed to pass through. The matching flow is then forwarded to one of the eight corresponding uplink ports on the Leaf switch. By mapping DSCP-tagged flows to specific port ranges and uplinks, the system creates a predictable and manageable routing pattern, making the configuration suitable for large-scale distributed training environments. The following example shows partial ACL and PBR on a Leaf switch:

```
<ACL>
ip access-list extended udf1
  10 permit udp any range 49152 51199 any eq 4791
ip access-list extended udf2
  10 permit udp any range 51200 53247 any eq 4791
...
ip access-list extended udf8
  10 permit udp any range 63488 65535 any eq 4791

<PBR>
route-map udf permit 10
  match ip address udf 1
  set ip next-hop 100.98.0.10
!
route-map udf permit 20
  match ip address udf 2
  set ip next-hop 100.98.0.14
!
...
route-map udf permit 80
  match ip address udf 8
  set ip next-hop 100.98.0.54
!
```

At the Spine switch layer, VRF is configured to isolate subnets, effectively dividing the two physical Spine switches into eight logically distinct virtual Spine switches, as shown in Fig. 5(b).

4.2 Evaluation of collective operations

We evaluated the collective communication performance of NCCL with ECMP and CTCCL integrated with DNLB using the NCCL-Tests, as illustrated in Fig. 6. NCCL-Tests provides critical performance metrics for collective operations, making it an essential tool for optimizing and understanding multi-GPU communication patterns. In our testbed, a mismatch exists between the PCIe 4.0 bandwidth within a node and the NIC/switch bandwidth between nodes. The maximum theoretical bandwidth of PCIe 4.0 is 32 GB/s, which is lower than the bandwidth of the NIC and switch ports (400 Gbps). As a result, a single NCCL-test task cannot generate significant network load or congestion. To address this, we concurrently ran two NCCL-Tests tasks to saturate the NIC traffic. Each task uses 4 GPUs per node, with the GPUs evenly distributed across two NUMA, that is, two GPUs in each Numa used by one task, which will be equivalent to the original node structure. We adopt the inherent bus bandwidth (busbw) metric introduced by NCCL-Tests, which reflects the communication speed between GPUs and identifies hardware bottlenecks (e.g., NVLink, PCIe, QPI, or network). The computation formula varies depending on the specific collective operation [22].

Figure 6(a) presents busbw and collective communication time for the AllReduce operation between 8 nodes with varying message sizes (QP=8), both averaged over two concurrent tasks. For large message transmissions, a notable performance gap emerges between NCCL (with ECMP) and CTCCL (with DNLB). Specifically, in our testbed, ECMP hash conflicts lead to network congestion, limiting NCCL's maximum busbw to only 16.1 GB/s. In contrast, CTCCL with DNLB effectively distributes the network load evenly, significantly reducing the likelihood of congestion and achieving a peak bandwidth of 22.6 GB/s. The impact of these differences can be observed in Fig. 6(b), which display the load distribution across the eight Leaf switch ports connected to the Spine during the execution of AllReduce in transmiting 8 GB messages. With ECMP, some ports experience

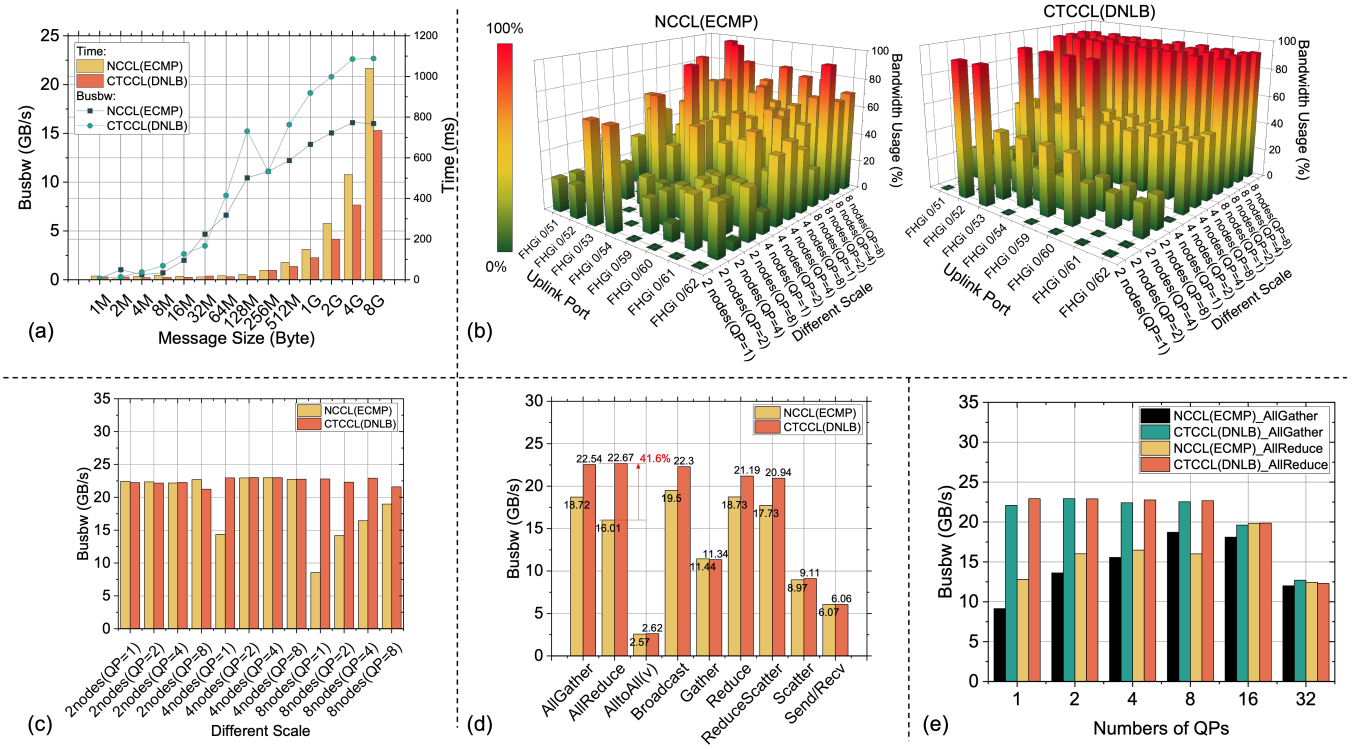


Figure 6: 1.Collective communication performance with (a) different message sizes; (c) different scale; (d) different operators; (e) different numbers of QPs; 2.(b) bandwidth usages of each port of Leaf.

severe congestion, slowing down the overall AllReduce performance. Conversely, DNLB ensures balanced utilization of all ports and links, maximizing resource usage and enhancing communication efficiency. Furthermore, Fig. 6(a) also presents CTCCL integrated with DNLB demonstrates clear advantages in the average execution time of collective communication, particularly when handling a higher volume of messages. Compared to NCCL combined with ECMP, DNLB significantly shortens the AllReduce execution time, reducing it from approximately 1 s to 700 ms.

Figure 6(b) and (c) provide additional insights into how network load (determined by the number of nodes and QPs) affects collective communication performance. We analyzed the network load and the corresponding performance of AllReduce in transmitting 8GB messages under various configurations, including tests involving 2 nodes, 4 nodes, and 8 nodes participating in collective communication with different numbers of QPs. The results indicate a significant imbalance in network load when NCCL is paired with ECMP, a problem that becomes increasingly pronounced as the number of nodes and QPs grows. In the case of 8 nodes, hash conflicts result in single-port congestion, leading to a noticeable degradation in overall busbw performance. For example, with 8 nodes and QP=8, the variance in load across the 8 switch ports is 92.22. In contrast, CTCCL integrated with DNLB ensures balanced load distribution across switch ports, reducing the variance in port bandwidth utilization to 0.06 at 8 nodes with QP=8—substantially lower than that of NCCL(ECMP)—thereby enhancing network resource utilization. This balance enables the AllReduce operation to consistently achieve performance levels of approximately 22 GB/s, even under high-demand conditions. It is worth noting that in scenarios with limited node or QP configurations, such as 2 nodes with QP=1 or 2, and 4 nodes with QP=1, Formula (4) is not satisfied, which prevents optimal utilization of network resources. However, this limitation has minimal practical impact, as the pre-training of LLMs typically involves configurations with a higher number of GPU nodes and QPs for parallel transmission.

Figure 6(d) provides a comprehensive comparison of the performance of NCCL (using ECMP) and CTCCL (using DNLB) when transmitting 8GB messages across various collective operations between 8 nodes with QP=8. The results clearly highlight significant performance improvements

when CTCCL is integrated with DNLB. Notably, operations such as AllGather, AllReduce, Broadcast, Reduce, and ReduceScatter benefit substantially from this approach, with the busbw for AllReduce increasing by approximately 40%. However, for other operators, the performance differences are less significant, as the network load and congestion remain relatively low, even with parallel execution of two tasks, thereby reducing the impact of load-balancing mechanisms.

Additionally, Fig. 6(e) illustrates the performance trends of NCCL and CTCCL under varying QP numbers when transmiting 8 GB messages between 8 nodes. For both AllGather and AllReduce operations, the performance exhibits a slight peak as the QP count increases but begins to decline once the number of QPs exceeds 16. The observed performance degradation is attributed to the growing overhead associated with managing a larger number of QPs. As the QP configuration scales, the system incurs additional costs related to the coordination of parallel transmissions and the management of multiple communication paths. These factors collectively impact the overall efficiency of collective operations, underscoring the trade-offs involved in configuring higher numbers of QPs.

The fault avoidance mechanism of CTCCL (DNLB) was also evaluated. During the execution of the collective communication, we triggered an uplink (FHGigabitEthernet0/51) failure on one Leaf switch (FHGigabitEthernet0/1-8 are downlink port of the leaf switch). As shown in Fig. 7, after the failure, traffic was effectively redistributed to other functional ports, maintaining excellent load balancing and overcoming the limitations of the path-pinning scheme, thus highlighting DNLB's robust fault tolerance.

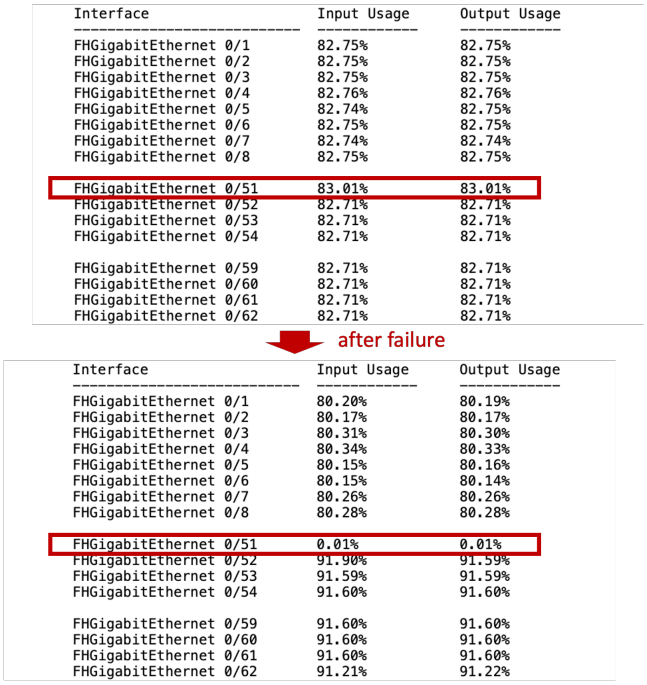


Figure 7: Traffic variation of switch ports before and after failures under DNLB.

4.3 Evaluation of LLM pre-training performance

To further evaluate the performance advantages of the collective communication library integrated with DNLB, we conducted pre-training experiments using Llama2-7B [19] with 3D parallelism on the same test platform. In the collective communication library, the number of QPs is configured to 8. The Megatron [29] pre-training framework is utilized to design parallelization strategies, while DeepSpeed [27] is employed to accelerate model training. The global batch size is set to 32, with a micro-batch size of 1. Figure 8(a) illustrates the iteration time progression over the first 250 iterations in an 8-node, 8-GPU per node configuration, where DP, TP and PP were set to 32, 2, and 1 respectivly. The product of the three values should equal the total number of GPUs. Excluding the first iteration, which involves model cold start processes such as data preprocessing, resource initialization and allocation, and delayed initialization, the iteration time stabilized at approximately 1.67 seconds when using NCCL (ECMP) for computational graph construction and operator execution. In contrast, enabling CTCCL (DNLB) reduced the iteration time to approximately 1.54 seconds, representing an significant 7% improvement in training efficiency. This result aligns with the collective communication performance assessments presented earlier, given that collective communication typically accounts for 15-30% of the total iteration time in large-scale model training [26, 29]. The performance gain underscores the effectiveness of DNLB in alleviating network congestion and optimizing resource utilization during distributed training, demonstrating its potential to accelerate large-scale model training workflows.

Further results under different 3D parallel configurations are presented in Fig. 8(b), which reports the average iteration time for each setting. For CTCCL (DNLB), the most significant improvements over NCCL (ECMP) were observed when the parallelism strategy (DP, TP, PP) was set to (32,2,1) or(16,4,1), where performance improved by approximately 7%. However, as DP decreases further, the performance gains diminish. A reduction in DP scale, which involves fewer devices in data parallelism, leads to lower inter-node communication traffic, thereby limiting the effectiveness of network load optimization techniques like DNLB. While TP/PP primarily involves intra-node communication, and PP typically relies on Send/Recv operations, which involves smaller data

volumes. In such scenarios, addressing congestion at the device level, NIC level, or within nodes may be a more effective strategy for improving performance.

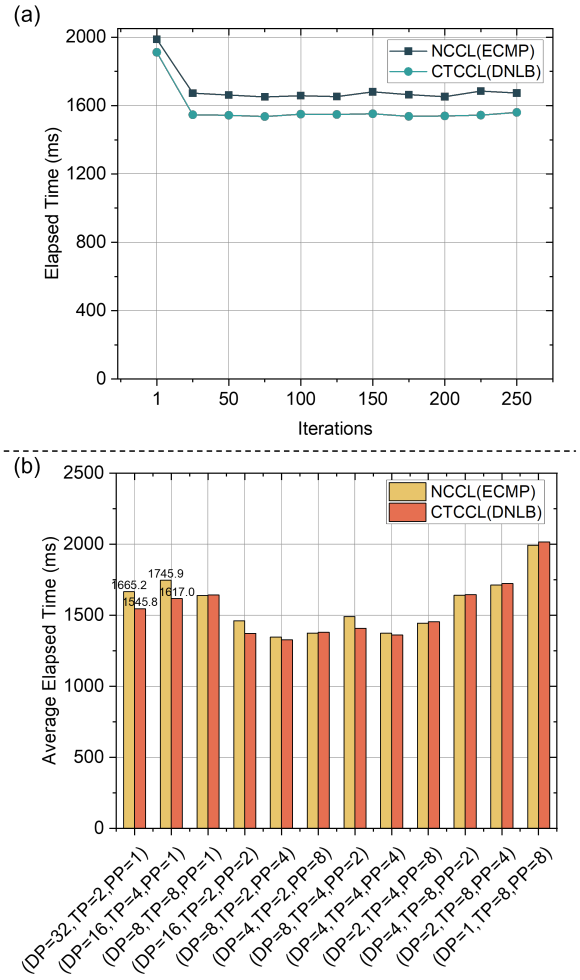


Figure 8: Performance of LLM pretraining: (a) elapsed time of the first 250 iterations; (b) average elapsed time under different 3D parallelism setup.

5 Related Work

Increasing flow entropy is the key for addressing the challenges posed by ECMP in mitigating the hash collisions low-entropy flows during LLM pre-training. Various approaches have been developed to tackle load-balancing challenges in large-scale GPU clusters, which can be broadly categorized into three classes based on the granularity of load distribution control.

The first category, as mentioned earlier, involves increasing flow entropy by controlling and forwarding data at the packet level, although this approach incurs additional overhead for managing OoO packets. Packet spraying splits traffic into packets that are randomly sprayed across all available parallel links [4]. DRILL employs a per-packet decision and randomization algorithm at each switch, guided by local queue occupancy, to allocate load [9]. Microsoft's MP-RDMA represents the first RDMA transport method to support multi-path routing [31]. MP-RDMA not only controls paths by specifying UDP source ports but also selectively drops slower paths to minimize reordering overhead. Furthermore, MP-RDMA synchronizes receiver-side delays to optimize memory usage for OoO handling. Nvidia's Spectrum technology leverages tight integration between network switches and DPUs to enable real-time dynamic monitoring of ECMP link bandwidth and port congestion [23]. This ensures precise traffic distribution and efficient data transfer.

The second category encompasses flowlet-based routing schemes, such as Conga [2], which decompose flows into smaller flowlets based on inactivity thresholds to mitigate OoO packet challenges [14, 25, 33]. This approach leverages the intermittent nature of TCP traffic, where natural transmission gaps provide opportunities for effective flowlet segmentation. However, RDMA traffic, particularly RoCE, is characterized by continuous, high-throughput communication with minimal idle periods, rendering flowlet-based methods less effective in this context.

The third category diverges from logically splitting flows at the protocol level by fully utilizing multi-QP transmission at the traffic source. This approach divides flows into multiple streams while maintaining packet order. For instance, Ethereal [1] demonstrated that greedily assigning paths to flows in LLM training traffic can achieve uniform load distribution across all network paths, effectively resolving ECMP hash collisions. However, this method requires modifications to both NICs and switches, as well as adjustments to how path IDs are identified within packet headers.

Alternatively, control over QPs in collective communication libraries has drawn increasing attention and also represent a key area of focus in this work. Beside the path-pinning scheme, Meta also tried to integrate QP extension with E-ECMP by using user-defined fields (UDF) to incorporate destination QP numbers into the hashing process [8]. Meta reported a 40% improvement in AllReduce collective performance through two extension methods—splitting and cycling. Alibaba's high-performance network (HPN) deploys host switches that enable its collective communication library to control the UDP source port of each flow, thereby balancing traffic across all available paths [5, 24]. Additionally, Alibaba introduced a Relative Path Control (RePaC) scheme [39], which uses deterministic mappings between header changes and path alterations to facilitate on-demand flow migration for load balancing. However, both approaches

necessitate switch modifications, which can increase costs. Similarly, Tencent explored a related approach by optimizing its TCCL library and injecting traffic routing through a network controller to achieve load balancing [15]. Compared to these, our solution stands out for its ease of implementation and deployment and fault tolerance, combined with a notably lower cost.

Beyond these methods, MegaScale has shown that topology optimization at the orbit level can also alleviate ECMP hash collision issues [13].

6 Conclusion

This paper presents an innovative Joint Device-Network Load Balancing (DNLB) solution that addresses key challenges in LLM pre-training within RoCE networks. DNLB controls UDP source ports through our custom collective communication library CTCCL and coordinates switch configuration with PBR to enhance network load distribution by mitigating hash collisions and flow congestion. The proposed approach offers a cost-effective, scalable load-balancing solution for distributed training without requiring significant hardware modifications. It is adaptable to various network architectures and task scenarios while providing fault avoidance capabilities. Our evaluation demonstrates that DNLB can improve communication efficiency by up to 40% and boost overall LLM training performance by 7% in real-world intelligent computing DC environments, all while maintaining excellent performance under network failures. Future research could explore DNLB's performance on test platforms equipped with NVLink and investigate its scalability for larger dynamic and heterogeneous network conditions, further enhancing its cost-effective load-balancing efficiency across diverse distributed training environments.

References

[1] Vamsi Addanki, Prateesh Goyal, and Ilias Marinos. 2024. Challenging the Need for Packet Spraying in Large-Scale Distributed Training. arXiv:2407.00550 [cs.NI] https://arxiv.org/abs/2407.00550

[2] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. 2014. CONGA: distributed congestion-aware load balancing for datacenters. 44, 4 (Aug. 2014), 503–514. doi:10.1145/2740070.2626316

[3] Guo Chen, Yuanwei Lu, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, and Thomas Moscibroda. 2019. Mp-rdma: enabling rdma with multi-path transport in datacenters. *IEEE/ACM Transactions on Networking* 27, 6 (2019), 2308–2323.

[4] Advait Dixit, Pawan Prakash, Y. Charlie Hu, and Ramana Rao Kompella. 2013. On the impact of packet spraying in data center networks. In *2013 Proceedings IEEE INFOCOM*. 2130–2138. doi:10.1109/INFCOM.2013.6567015

[5] Jianbo Dong, Shaochuang Wang, Fei Feng, Zheng Cao, Heng Pan, Lingbo Tang, Pengcheng Li, Hao Li, Qianyuan Ran, Yiqun Guo, Shanyuan Gao, Xin Long, Jie Zhang, Yong Li, Zhisheng Xia, Liuyihan Song, Yingya Zhang, Pan Pan, Guohui Wang, and Xiaowei Jiang. 2021. ACCL: Architecting Highly Scalable Distributed Training Systems With Highly Efficient Collective Communication Library. *IEEE Micro* 41, 5 (2021), 85–92. doi:10.1109/MM.2021.3091475

[6] Jiangfei Duan, Shuo Zhang, Zerui Wang, Lijuan Jiang, Wenwen Qu, Qinghao Hu, Guoteng Wang, Qizhen Weng, Hang Yan, Xingcheng Zhang, Xipeng Qiu, Dahua Lin, Yonggang Wen, Xin Jin, Tianwei Zhang, and Peng Sun. 2024. Efficient Training of Large Language Models on Distributed Infrastructures: A Survey. arXiv:2407.20018 [cs.DC] https://arxiv.org/abs/2407.20018

[7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[8] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, Shuqiang Zhang, Mikel Jimenez Fernandez, Shashidhar Gandham, and Hongyi Zeng. 2024. RDMA over Ethernet for Distributed Training at Meta Scale *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 57–70. doi:10.1145/3651890.3672233

[9] Soudeh Ghorbani, Zibin Yang, P. Brighten Godfrey, Yashar Ganjali, and Amin Firoozshahian. 2017. DRILL: Micro Load Balancing for Low-latency Data Center Networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 225–238. doi:10.1145/3098822.3098839

[10] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. 2009. The cost of a cloud: research problems in data center networks. *SIGCOMM Comput. Commun. Rev.* 39, 1 (Dec. 2009), 68–73. doi:10.1145/1496091.1496103

[11] Taylor Groves, Damian Hazen, Glenn Lockwood, and Nicholas J Wright. 2021. Use It or Lose It: Cheap Compute Everywhere. In *Smoky Mountains Computational Sciences and Engineering Conference*. Springer, 280–298.

[12] Jinbin Hu, Houqiang Shen, Xuchong Liu, and Jin Wang. 2024. RDMA Transports in Datacenter Networks: Survey. *IEEE Network* (2024).

[13] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, et al. 2024. {MegaScale}: Scaling large language model training to more than 10,000 {GPUs}. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 745–760.

[14] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. 2016. Hula: Scalable load balancing using programmable data planes. In *Proceedings of the Symposium on SDN Research*. 1–12.

[15] Baojia Li, Xiaoliang Wang, Jingzhu Wang, Yifan Liu, Yuanyuan Gong, Hao Lu, Weizhen Dang, Weifeng Zhang, Xiaojie Huang, Mingzhuo Chen, Jie Chen, Chunzhi He, Yadong Liu, Xiaoyuan Hu, Chen Liu, Xuefeng Ji, Yinben Xia, Xiang Li, Zekun He, Yachen Wang, and Xianneng Zou. 2024. TCCL: Co-optimizing Collective Communication and Traffic Routing for GPU-centric Clusters. In *Proceedings of the 2024 SIGCOMM Workshop on Networks for AI Computing* (Sydney, NSW, Australia) *(NAIC '24)*. Association for Computing Machinery, New York, NY, USA, 48–53. doi:10.1145/3672198.3673799

[16] Yuanwei Lu, Guo Chen, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, Enhong Chen, and Thomas Moscibroda. 2018. Multi-Path Transport for RDMA in Datacenters. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 357–371. https://www.usenix.org/conference/nsdi18/presentation/lu

[17] Huimin Luo, Jiao Zhang, Mingxuan Yu, Yongchen Pan, Tian Pan, and Tao Huang. 2024. SeqBalance: Congestion-Aware Load Balancing with

no Reordering for RoCE. arXiv:2407.09808 [cs.NI] https://arxiv.org/abs/2407.09808

[18] Meta. [n. d.]. Gloo: Collective communications library with various primitives for multi-machine training. https://github.com/facebookincubator/gloo?tab=readme-ov-file

[19] Meta. 2024. *Llama 2*. Retrieved January 15, 2025 from https://huggingface.co/meta-llama/Llama-2-7b "

[20] Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, and Scott Shenker. 2018. Revisiting network support for RDMA. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 313–326. doi:10.1145/3230543.3230557

[21] Nvidia. 2016. *NVIDIA Collective Communications Library (NCCL)*. Retrieved January 15, 2025 from https://developer.nvidia.com/nccl

[22] Nvidia. 2017. *NCCL-Tests*. Retrieved January 15, 2025 from https://github.com/NVIDIA/nccl-tests/blob/master/doc/PERFORMANCE.md

[23] Nvidia. 2023. *NVIDIA Spectrum-X Networking Platform*. Retrieved January 15, 2025 from https://www.nvidia.com/en-us/networking/spectrumx/

[24] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, Chao Wang, Peng Wang, Pengcheng Zhang, Xianlong Zeng, Eddie Ruan, Zhiping Yao, Ennan Zhai, and Dennis Cai. 2024. Alibaba HPN: A Data Center Network for Large Language Model Training. In *Proceedings of the ACM SIGCOMM 2024 Conference* (Sydney, NSW, Australia) *(ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 691–706. doi:10.1145/3651890.3672265

[25] Mubashir Adnan Qureshi, Yuchung Cheng, Qianwen Yin, Qiaobin Fu, Gautam Kumar, Masoud Moshref, Junhua Yan, Van Jacobson, David Wetherall, and Abdul Kabbani. 2022. PLB: congestion signals are simple and effective for network load balancing. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 207–218.

[26] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*. PMLR, 18332–18346.

[27] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3505–3506. doi:10.1145/3394486.3406703

[28] Katharine Sanderson. 2023. GPT-4 is here: what scientists think. *Nature* 615, 7954 (2023), 773.

[29] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. arXiv:1909.08053 [cs.CL] https://arxiv.org/abs/1909.08053

[30] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2024. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805 [cs.CL] https://arxiv.org/abs/2312.11805

[31] Feng Tian, Yang Zhang, Wei Ye, Cheng Jin, Ziyan Wu, and Zhi-Li Zhang. 2021. Accelerating Distributed Deep Learning using Multi-Path RDMA in Data Center Networks. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)* (Virtual Event, USA) *(SOSR '21)*. Association for Computing Machinery, New York, NY, USA, 88–100. doi:10.1145/3482898.3483363

[32] Serhii Uspenskyi. 2024. *Large Language Model Statistics And Numbers (2024)*. https://springsapps.com/knowledge/large-language-model-statistics-and-numbers-2024#

[33] Erico Vanini, Rong Pan, Mohammad Alizadeh, Parvin Taheri, and Tom Edsall. 2017. Let it flow: Resilient asymmetric load balancing with flowlet switching. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 407–420.

[34] Manoj Wadekar. 2013. InfiniBand, iWARP, and RoCE. In *Handbook of Fiber Optic Data Communication*. Elsevier, 267–287.

[35] Ruolan Wu, Wei Wan, and Junhong Li. 2024. Research and simulation of adaptive routing mechanism in InfiniBand network. In *Third International Conference on Advanced Manufacturing Technology and Manufacturing Systems (ICAMTMS 2024)*, Vol. 13226. SPIE, 1315–1327.

[36] Yunhong Xu, Keqiang He, Rui Wang, Minlan Yu, Nick Duffield, Hassan Wassel, Shidong Zhang, Leon Poutievski, Junlan Zhou, and Amin Vahdat. 2022. Hashing Design in Modern Networks: Challenges and Mitigation Techniques. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. 805–818.

[37] Yunhong Xu, Keqiang He, Rui Wang, Minlan Yu, Nick Duffield, Hassan Wassel, Shidong Zhang, Leon Poutievski, Junlan Zhou, and Amin Vahdat. 2022. Hashing Design in Modern Networks: Challenges and Mitigation Techniques. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. USENIX Association, Carlsbad, CA, 805–818. https://www.usenix.org/conference/atc22/presentation/xu

[38] Ling Zhang, Xuefei Yang, Zhenlong Wan, Hang Liu, Wei Gu, Pingjing Liu, Qilin Dai, Shanwei Ye, and Yingcheng Lin. 2024. A High-Performance RDMA NIC With Ultra-Highly Scalable Connections. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2024), 1–1. doi:10.1109/TCAD.2024.3514782

[39] Zhehui Zhang, Haiyang Zheng, Jiayao Hu, Xiangning Yu, Chenchen Qi, Xuemei Shi, and Guohui Wang. 2021. Hashing linearity enables relative path control in data centers. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 855–862.

[40] Lianmin Zheng, Zhuohan Li, Hao Zhang, Yonghao Zhuang, Zhifeng Chen, Yanping Huang, Yida Wang, Yuanzhong Xu, Danyang Zhuo, Eric P. Xing, Joseph E. Gonzalez, and Ion Stoica. 2022. Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. USENIX Association, Carlsbad, CA, 559–578. https://www.usenix.org/conference/osdi22/presentation/zheng-lianmin