Network Load Balancing with Parallel Flowlets for AI Training Clusters

Peirui Cao[1,2], Wenxue Cheng[1*], Shizhen Zhao[2], Yongqiang Xiong[1]
[1]Microsoft Research Asia, [2]Shanghai Jiao Tong university

ABSTRACT

Unlike traditional data center traffic, AI training traffic primarily consists of large-size flows that are fewer in number. This characteristic poses a challenge in balancing routing granularity with reorder overhead in existing routing strategies. Existing serial flowlet schemes aim to achieve a better trade-off in TCP scenarios than flow-level or packet spraying load balancing. However, they are not well-suited for AI training clusters with high-performance RDMA networks.

To tackle this issue, we propose a parallel-flowlet strategy, ParaLet, which effectively resolves the serial flowlet's problems of insufficient routing entropy in AI training traffic and the difficulty of identifying time gaps in RDMA networks. ParaLet requires only a small number of Queue Pairs, which are decoupled from the connections, thus circumventing scalability limits. The theoretical analysis and simulations indicate that ParaLet not only achieves near-optimal throughput but also diminishes flow completion time by 1.5-3.4 times compared to existing methods.

1 INTRODUCTION

The rapid development of AI necessitates high-performance RDMA networks to support the communication traffic generated by distributed training tasks. Due to the closed ecosystem and high cost of InfiniBand [10], Ethernet-based RDMA networks have gained favor among major vendors [11, 16, 19]. To support the expanding AI training models, it is necessary to interconnect more nodes, typically using an unblocking multi-level Clos topology, which inherently provides path diversity. In theory, multi-path routing can fully utilize the available bandwidth and capacity resources in the network. By simultaneously using multiple paths for data transmission, overall bandwidth utilization can be improved, reducing network congestion and bottlenecks. However, existing load balancing strategies in AI training clusters struggle to simultaneously meet the demands of the two dimensions.

Coarse-grained flow-level routing [12, 37], when faced with a small number of long flows generated by AI training jobs, is prone to flow conflicts resulting in tail latency and struggles to achieve uniform load balancing according to the law of large numbers. On the other hand, the finest-grained packet spraying [7, 9, 14] introduces significant overhead at the endpoint due to the disordering problem observed in AI training workloads. The intermediate-grained flowlet routing [2, 17, 23, 30] used in traditional TCP scenarios is also challenging to apply in RDMA scenarios as it requires large time gaps. Existing flowlets operate as serial flowlets, selecting a different path one by one. Meanwhile, the original AI training flows have a smaller number but larger size compared to traditional data center applications, resulting in the problem of limited routing path entropy [22].

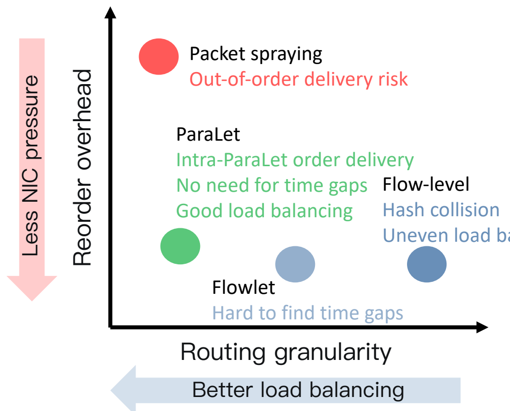


Figure 1: Design tradeoff for network load balancing.

As shown in Fig. 1, we propose the parallel flowlets scheme, ParaLet, to address the mentioned problems. ParaLet adds routing path entropy using parallel flowlets for better network load balancing. Unlike traditional serial flowlets that require finding optimal time gaps or pausing and waiting for time gaps, ParaLet directly splits every original flow to multiple flowlet blocks from the source. Meanwhile, ParaLet ensures that packets within each flowlet follow the same path for ordered delivery while allowing out-of-order arrival at the destination for parallel flowlets. This approach minimizes reorder overhead. Additionally, ParaLet achieves good performance by using a small number of parallel flowlets and ParaLet decouples Queue Pairs (QPs) and connections, which is crucial for enhancing scalability in RDMA scenarios.

The contributions of this paper are as follows:

*Wenxue Cheng is the corresponding author.

(1) We rethink the fundamental causes of existing load balancing issues (§2.2) during AI training workloads (§2.1), which can help guide the design of new load balancing routing strategies.
(2) We propose parallel flowlets scheme, ParaLet, to improve network load balancing for AI training clusters (§3) and provide theoretical evidence (§4). Specifically, ParaLet achieves wise routing granularity (§4.1) and minimizes reorder overhead (§4.2). Meanwhile, we discuss the implementation to support the deployment of ParaLet in real systems in the future (§5).
(3) Experimental results demonstrate that ParaLet achieves near-optimal throughput and significantly reduces flow completion time compared to existing strategies (§6).

2 BACKGROUND

2.1 New Features of AI Training Workloads

AI training workloads exhibit unique characteristics that set them apart from traditional datacenter workloads [1, 4, 5, 25]. These characteristics are primarily due to the use of multiple accelerators (e.g., GPU, NPU, TPU, etc.) interconnected through high-performance RDMA networks [21, 24, 31] in a multi-node software design, forming a distributed system. The Message Passing Interface (MPI) is commonly used for data communication in this setup. The distinctive features of AI training workloads include:
High-bandwidth long flows: AI training jobs typically involve higher bandwidth flows, ranging from several hundred MBs to GBs [26]. The flow size is often several times larger than the Bandwidth-Delay Product (BDP), necessitating network infrastructures capable of efficiently handling such high-bandwidth flows.
A small number of flows: AI training models can be categorized into various collective communication patterns [3, 34]. Each accelerator generates a small number of flows at any given time, as stated in the Nvidia network architecture white paper [22]. Despite the multi-stage optimizations applied to these communication patterns and the diverse implementations of different collective communication libraries (xCCL) [33], every accelerator triggers the NIC to generate a small number of active flows into the networks at any given moment, as shown in Table 1.
Synchronized communication phases: To ensure that the workers' model remains up-to-date, the model is synchronized in every iteration. At the start of each iteration, multiple flows are generated simultaneously, creating bursts of traffic for exchanging data among the workers [36]. In other words, during the training process, a group of flows begins simultaneously when the cluster enters a communication phase [27]. This synchronization contributes to bursts of network activity, which requires effective management of network resources and may lead to potential collision issues. The fact that some flows from the last phase are not finished will affect the communication in the subsequent phases.

xCCL	Collective communication primitives and the number of active flows (#AF)
NCCL [35]	Broadcast, Reduce, All-Gather, Reduce-Scatter, Ring / Double-binary-tree All-Reduce. #AF is usually 1, 2, up to 3.
MSCCL [6]	Ring for All-Reduce, Reduce-Scatter, and All-Gather. All-Pairs. Hierarchical. Microsoft generally set #AF to 8.
ACCL [8]	Hybrid All-Reduce: intra-node Reduce-Scatter + Halving-Doubling + intra-node All-Gather. #AF < 4
Other xCCL	Also include the above primitives, and #AF is also a small value.

Table 1: A small number of active flows in AI clusters.

2.2 Routing is Critical for AI Training

In large-scale AI clusters, multi-layer topology interconnections necessitate multi-path routing to fully utilize the diversity of paths and achieve high bandwidth and low latency. Optimizing routing is therefore crucial in these scenarios.

2.2.1 Limitations of Flow-level Routing in AI Training. Flow-level routing is coarse-grained load balancing and guarantees ordered delivery by allowing packets of a flow to select the same path through ECMP [12], using the 5-tuple (src-ip, dst-ip, src-port, dst-port, protocol number) as the hash factor, or WCMP [37], which allocates the proportion of different paths based on estimates of bandwidth and utilization. However, due to the small number of large-size flows and the multi-hop selection in large-scale AI networks, flow-level routing easily lead to uneven flow collision.

ConWeave [29] adds an additional opportunity for rerouting during the transmission of a flow based on ECMP and RTT awareness. ConWeave is suitable for traditional data centers where it finds a new path with low utilization through rerouting for reducing congestion. However, ConWeave requires the scenario including a large number of small flows combined with a small number of large flows, while AI training traffic includes mainly high-bandwidth large flows. This offers little room for ConWeave optimization.

2.2.2 Challenges of Packet Spraying in AI Training. Packet spraying [7, 9, 14, 15], which packets of every flow are randomly assigned to one of the available shortest paths, was proposed over a decade ago, but was not widely adopted in data centers. Compared with flow-level load balancing, packet spraying causes a significant out-of-order delivery problem. **RDMA is sensitive to out-of-order delivery.**

Routing granularity	Existing schemes	Dilemma in AI training clusters
Flow-level	ECMP [12], WCMP [37]	Uneven flow conflict and long tail delay.
Rerouting	ConWeave [29], Proteus [13]	Little room for rerouting.
Flowlet	Conga [2], LetFlow [30], HULA [18]	Hard to find time gap. Serial flowlet has the problem of limited routing path entropy.
Packet spraying	RPS [7], DRILL [9], SRD [28] QDAPS [14, 15]	Sorting overhead at the destination server or ToR switch.
ParaLet	Our schemes	Parallel flowlets solve these dilemmas.

Table 2: The main issues of existing load balancing schemes in AI training clusters.

RDMA scenarios meet the assumption that there is generally no loss in the network and therefore packets are delivered in-order. Consequently, when an RNIC receives an out-of-order packet, it interprets it as a sign of packet loss, potentially due to network congestion. As a result, the RNIC promptly initiates loss recovery, leading to a decrease in the sending RNIC's transmission rate. In addition, RNICs are mostly fixed-function devices with limited resources, including packet buffering capabilities. Whether using the Go-Back-N or the enhanced Selective Repeat approach, RDMA experiences a significant performance decline when facing out-of-order delivery, even resulting in completion times for 1MB messages being slowed down by more than 3 times [29].

Adapting special HPC transport for AI training clusters is challenging. SRD [28] has designed a new full-stack transport that provides reliable but out-of-order delivery for HPC workloads, leaving the responsibility of order restoration to the layers above it. However, it can be difficult to adapt this transport for AI training applications as it requires modifications to the ordering semantics in the upper layer.

2.2.3 *Inefficiency of Flowlet Routing in AI Training.*

To achieve the accuracy and responsiveness of packet-based splitting while avoiding packet reordering, flowlet routing [2, 17, 18, 30] splits TCP flows into multiple flowlets based on the inactive time gap among packets. Simultaneously, it estimates real-time congestion on fabric paths and allocates flowlets to paths based on feedback from remote switches. However, when considering an inactive time gap, there are significantly fewer opportunities to find flowlets in RDMA scenarios compared to TCP scenarios, even when testing the gap threshold from $1us$ to $500us$ [20]. The reason for this is that RDMA utilizes hardware-based packet pacing per connection, also known as rate shaping, which leads to a continuous flow of packets with minimal time gaps. Consequently, due to the lack of sufficiently large flowlet gaps, flowlet switching-based approaches are not well-suited for RDMA. Therefore, **in AI training clusters under RDMA with a small number of large-size flows, typical flowlets tend to generate uneven packet groups, leading to low routing path entropy and packet disorder**.

In the online community, some speculate that subjecting flowlet to a stop-and-wait signal could artificially create a time gap. However, doing so undoubtedly forces the identification of a flowlet's segmentation threshold at the expense of the ultimate sending performance.

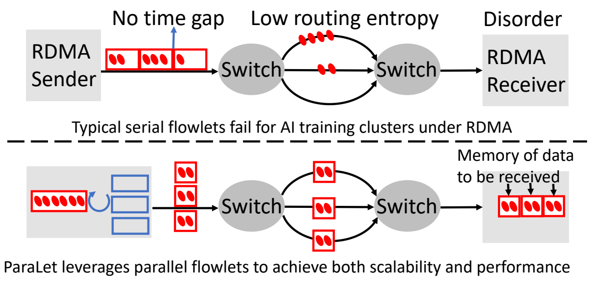

No time gap
Low routing entropy
Disorder
RDMA Sender
Switch
Switch
RDMA Receiver
Typical serial flowlets fail for AI training clusters under RDMA
Memory of data to be received
Switch
Switch
ParaLet leverages parallel flowlets to achieve both scalability and performance

Figure 2: ParaLet insight.

3 PARALET MOTIVATION

To address the issues of routing for AI training clusters (summarized in Table 2), we introduce ParaLet, a novel parallel-flowlet load balancing mechanism. The core concept of ParaLet is "parallel flowlet spraying." It divides the original flow into multiple segments and concurrently sends m flowlets into the network, effectively adding routing path entropy. As shown in Fig. 2, similar to traditional flowlets, each parallel flowlet can choose a new path. However, unlike traditional flowlets that are injected into the network with time gaps at the original rate, these m flowlets of every flow are delivered in parallel at a fraction of that rate.

Unlike MPRDMA [20], which utilizes multipath by letting every packet go to a different path with various windows based on ECN feedback, resulting in the reorder of all out-of-order packets and network scalability bottleneck, ParaLet does not require such reorder at the destination. Instead, ParaLet ensures that packets within a flowlet follow the same path, avoiding most of reorder, while still allowing parallel flowlets to arrive at the destination at different times.

ParaLet balances routing granularity and reorder overhead by transmitting one flow across multiple, but not infinite, paths. In contrast, flow-level load balancing restricts all packets of a flow to a single path, while packet spraying can distribute a flow across as many paths as there are in-flight packets. ParaLet, however, selects m paths concurrently for every flow. As a result, ParaLet maintains in-order packets delivery within each flowlet and only needs to ensure that all flowlets from one flow successfully reach their destination. ParaLet can be seen as a hybrid approach: it will degrade to flow-level load balancing when $m = 1$ (treating the entire flow as one flowlet) and to packet spraying when $m = \infty$ (treating each packet as a separate flowlet).

4 THEORETICAL VERIFICATION

4.1 Routing Granularity

Considering one communication step of AI training over an unblocking cluster, where each GPU starts to communicate with its peer under other ToRs simultaneously. Focusing on one ToR switch first. Assume the ToR has n ingress ports and n egress ports, thus it should forward n flows from n ingress ports towards n egress ports at line rate. We can calculate the routing collision effect for different routing strategies.

1) Flow-level Routing: We expected one flow towards one egress ports without collision. Under the random strategy of ECMP, the collision probability would be $1 - \frac{n!}{n^n}$. As the increase of switch radix n to support large scaled AI cluster, the collision ratio could be almost 1. That is, at least two flows would select the same port and then the flow completed time has to be at least doubled than optimal. Even when we take the rerouting strategies like ConWeave [29], the second choice still suffers the almost 1 collision ratio since the number of congested flows are still comparable to the number free ports. This is quite different from traditional data center where you can always find enough free ports.

2) Packet Spraying: The success of Packet spraying for load balancing is based on *the Law of Large Numbers.* With infinite tries (one packet select once), the workload ratio distributed to one egress port equals to probability to select this port for each packet, i.e., $\frac{1}{n}$. Thus each egress port just need to handle as the same as the workload of each ingress port, even through the packets has been totally disordered.

3) ParaLet: With m flowlets arrive at each ingress port in parallel, the switch need to distribute the total of $m * n$ flowlets to n egress ports. Support there are X flowlets distributed to one egress port, thus X can be regarded as the success number of $n * m$ times of Bernoulli tests with probability $\frac{1}{n}$, i.e., $X \sim B(mn, \frac{1}{n})$, then $E(X) = m$, $Var(X) = m$. Consequently we can calculate the slowdown of flow completion time as $\max(1, \frac{X}{m})$. Let $S = \frac{X}{m}$, then S approximates to a Normal distribution $N(\mu, \sigma^2)$ with $\mu = 1$, $\sigma^2 = \frac{1}{m}$. We can approximates the CDF (cumulative density function) of S as

$n' = \Phi^{-1}(1 - \frac{1}{n})$	$n = \frac{1}{1-\Phi(n')}$	$n' = \Phi^{-1}(1 - \frac{1}{n})$	$n = \frac{1}{1-\Phi(n')}$
1	6	5	3488555
2	43	6	1.013×10^9
3	740	7	7.813×10^{11}
4	31574	8	1.501×10^{15}

Table 3: Impact of the switch radix n. Let $n' = \Phi^{-1}(1-\frac{1}{n})$, then $n = \frac{1}{1-\Phi(n')}$. When n increases to 1e+15, n' is still no more than 8.

$\Phi(\frac{S-\mu}{\sigma})$, where Φ is the CDF of the Standard Normal Distribution. Considering the n egress ports, the maximal slowdown $\max\{S_0, S_1, ..., S_{n-1}\}$ can be estimated by the $(1 - \frac{1}{N})$ percentile of S, i.e.

$$E(S_{\max n}) = \mu + \sigma * \Phi^{-1}(1 - \frac{1}{n}) = 1 + \frac{\Phi^{-1}(1 - \frac{1}{n})}{\sqrt{m}}$$

When we considering the end-to-end flow transmission through the whole AI cluster by multiple hops, we can find this equation are still correct but the switch radix n would be changed to the whole number of GPU nodes N. That is, the final slowdown for ParaLet can be estimated as

$$SlowDown = 1 + \frac{\Phi^{-1}(1 - \frac{1}{N})}{\sqrt{m}} \quad (1)$$

Compared to ECMP, ParaLet is expected to speedup by $\sqrt{m}$ times, and only $\frac{1}{\sqrt{m}}$ slower than optimal. As shown in Table 3, the switch radix n, which indicates the AI network scale, has really few impact on the maximal slowdown.

4.2 Reorder Overhead

Packet Spraying will make the packets arrive at receiver in unpredictable pattern, it is really hard to judge whether the data between two disorder packets are lost or still in transmission in the path. Current RDMA takes the Go-Back-N loss recovery thus any disorder event will call re-transmission, resulting in serious under-utilization. Technologies like SACK could be much better than Go-Back-N to avoid unnecessary re-transmission, but SACK requires more fine-grained transport support. What is worse, it still have the risk to transmits the packet in path, even through the risk maybe mitigated by a magic timeout. But with ParaLet, packets insider one flowlet are sequential and through the same path, thus the disorder events inside one flowlet indicates loss event and retransmission is necessary, and the disorder events between different flowlets can be ignored directly.

5 DESIGN

5.1 ParaLet Steps

Theory (§3 and §4) has proved that ParaLet can achieve better performance for AI training clusters. Next, we will show the concrete steps of ParaLet, as shown in Fig. 3.

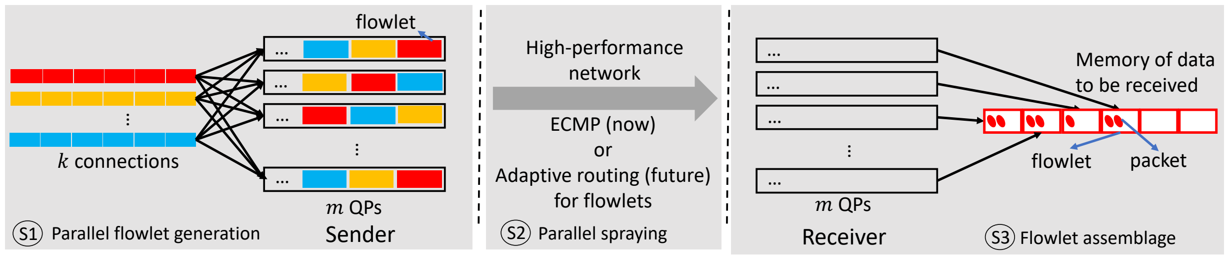


Figure 3: ParaLet design.

Ⓢ1 **Parallel flowlet generation**. ParaLet needs m QPs with different UDP source port to output m parallel flowlets using decoupling (§5.3). Each QP will pull the waiting flowlets and transmit data at $\frac{1}{m}$ of the line rate. Note that, ParaLet only modifies RNICs at hosts.

Ⓢ2 **Parallel spraying**. Since we create different QP with different UDP source ports, the m flowlets can directly be sprayed to different paths based on ECMP hash function. In the future, adaptive routing can also be incorporated into ParaLet.

Ⓢ3 **Flowlet assemblage**. A parallel flowlet is a message consisting of several packets. Splitting at the MPI layer allows for the utilization of the RDMA feature, which enables out-of-order delivery between messages. The receiver only needs to ensure that all flowlets of a flow arrive successfully and does not need to reorder packets, because packets within a message follow the same path.

5.2 Parameter Tuning

The number of QPs m and the message size S are important.

m: Based on Equation(1), we should select m based on expected slowdown and the AI cluster scales. For example, for a cluster contain tens-of-thousands accelerators, and we expected the slowdown no more than 2, there would be $m \geq \left(\frac{\Phi^{-1}(1-10^{-5})}{2-1}\right)^2 \approx 25$. And when cluster extends more than 10^9 nodes, to achieve the same slowdown, we only need $m \geq \left(\frac{\Phi^{-1}(1-10^{-9})}{2-1}\right)^2 \approx 36$.

S: The selection of S does not impact the correctness of ParaLet, but the QP should support a deeper tx-depth with small S to achieve full bandwidth utilization. And a larger S may loss flexibility. Experiential, we suggested to select the message (flowlet) size S as $4BDP/m$.

5.3 Decoupling QPs and Connections

It is crucial to consider the decoupling of QPs and connections to support parallel-flowlet traffic. This aspect is currently our ongoing work, and we provide the following guide.

First, ParaLet utilizes a QP scheduler to schedule m QPs from host memory, similar to SRNIC [32]. This scheduler determines which QP to send data next and retrieves work queue elements (WQEs) and data from the corresponding send queue (SQ) using a data mover. Each SQ can support multiple flowlets from different flows. Additionally, incorporating MPI layer management in ParaLet involves handling various tasks such as managing connection status, coordinating receive buffers, and efficiently processing batch acknowledgments for packet groups. One method for managing the MPI layer is by utilizing a networking plugin, such as the *NCCL_NET_PLUGIN*.

Note that before decoupling QPs and connections, we attempted other methods. The first method involved creating multiple QPs for each source-destination pair to support parallel flowlets. However, in large-scale scenarios, this method resulted in performance degradation due to the excessive number of QPs. The second method was to create a single QP for each source-destination pair but required modifying the port for each flowlet. This introduced port-switching overhead and also posed a scalability bottleneck.

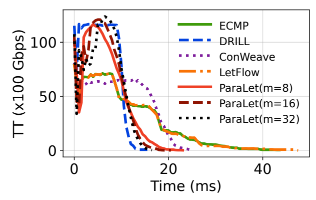
5.4 Benefits

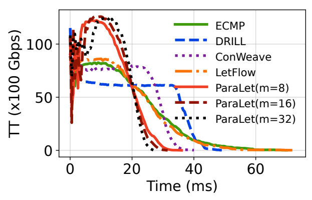
Adaptive for various collective options and communication patterns: The xCCL will translate the collective options into any kinds of communication patterns, thus the number of concurrent active connections are dynamic. This design can naturally adapt for any number of concurrent active connections. Flowlets from different connections has the same probability to be distribute to all m QPs, and then the routing granularity can be also guaranteed.

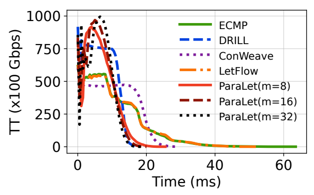
High scalability: Typically in AI training, each node needs to setup connections with all other nodes at the beginning of the job. To catch up with the increase rate of AI scales, how to handle the tens of thousands connections using limited resources becomes a big challenge. Aimed at this, our design decouples the connection management and QP transmission. Besides, ParaLet limits the number of completed flows in network, which make congestion control more flexible.

6 SIMULATION

Topology: We use NS3 simulator to simulate Clos topologies with an over-subscription ratio of 1:1, including small-scale


(a) Ring in small-scale 128 topology.


(b) DBT in small-scale 128 topology.


(c) Ring in large-scale 1024 topology.

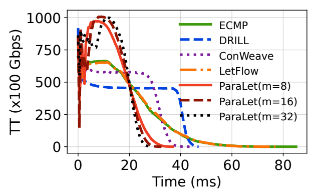
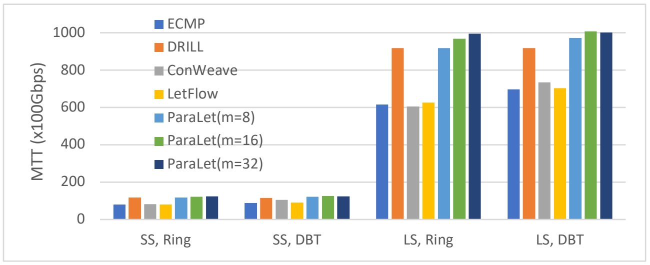

(d) DBT in large-scale 1024 topology.

Figure 4: Total throughput (TT) performance.


(a) MTT

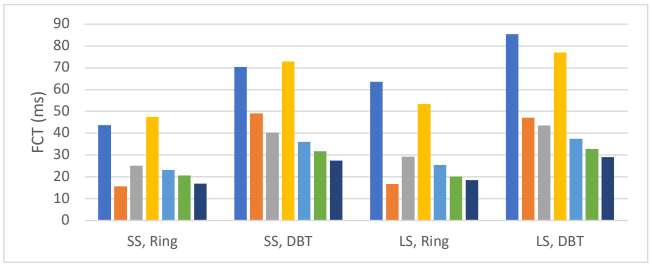

(b) FCT

Figure 5: Maximum Total Throughput (MTT) and Flow Completion Time (FCT) under AI training workloads of Ring and double binary tree (DBT) in small-scale (SS) and large-scale (LS) typologies, respectively.

128 accelerators and large-scale 1024 accelerators, respectively. All links are 100Gbps with 1*us* latency. Each switch has a buffer size of 9MB.

Workload: As shown in Table 1, evaluating All-Reduce encompasses the most combinations of other collective communication primitives. Specifically, we use traffic snapshots of the communication pair generated by the Ring-based All-Reduce and the Double Binary Tree (DBT)-based All-Reduce.

Comparison objects: We mainly compare ParaLet using different *m* parallel flowlets for every flow with ECMP, DRILL [9], LetFlow [30] and ConWeave [29]. DRILL uses packet spraying settings, which choose a new output port with the smallest queue among 2 random samples and the current port. LetFlow is an advanced resilient asymmetric flowlet scheme. We select the time gap threshold of 50 us, 100 us, and 200 us, and choose to present the optimal result. ConWeave is the state-of-the-art rerouting solution.

Metrics: Besides **Flow Completion Time (FCT)**, we collected statistics on the throughput of switch ports connected to all source. By summing these values, we obtained the **Total Throughput (TT)**, which reflects the effective throughput of the entire network as it changes over time.

Fig. 4 illustrates that ParaLet achieves near-optimal throughput. A coarse granularity of spraying is sufficient, and there is no need for per-packet spraying. In the DBT workload, DRILL's TT experiences two sharp degradations, because of out-of-order delivery, which not only adds overhead to the receiver but also affects the source's congestion control. ECMP performs the worst across different scenarios because, under the characteristics of AI training workloads, its uneven flow hash conflicts become more pronounced. ConWeave outperforms DRILL in the DBT workload, but in the Ring workload, where rerouting opportunities are fewer, its performance falls between DRILL and ECMP. The state-of-the-art flowlet scheme (LetFlow) achieves poor performance in AI training clusters. This is because LetFlow struggles to identify the time gaps under RDMA, and the size of flows is large but the number of flows in the AI training traffic is fewer than in traditional data centers. Consequently, the available number of flowlets is also limited, resulting in constrained routing entropy.

As shown in Fig. 5, ParaLet (*m*=32) increases Max Total Throughput (MTT) by 1.4-1.6 times and reduces FCT by 2.6-3.4 times compared to ECMP. Compared to DRILL, ParaLet increases MTT by about 1.1 times and reduces FCT by about 1.8 times. Compared to ConWeave, ParaLet increases MTT by 1.2-1.6 times and reduces FCT by 1.5-1.6 times. Compared to LetFlow, ParaLet increases MTT by 1.4-1.6 times and reduces FCT by 2.6-2.8 times. The performance of the ParaLet improves with an increasing value of *m*, although the rate of improvement gradually diminishes. ParaLet requires only 32 QPs to achieve good simulation performance without the need for an excessive number.

7 CONCLUSION

To tackle the routing issues posed by AI training workloads, we propose ParaLet, a parallel flowlets load balancing strategy, which achieves the simultaneous reduction of flow collisions and reorder overhead. Experimental results demonstrate that ParaLet achieves near-optimal throughput and significantly reduces flow completion time compared to existing strategies.

8 ACKNOWLEDGMENT

We sincerely thank anonymous reviewers for their helpful comments. This work was supported by the NSF China (No. 62272292).

REFERENCES

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review* 38, 4 (2008), 63–74.

[2] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, et al. 2014. CONGA: Distributed congestion-aware load balancing for datacenters. In *Proceedings of the 2014 ACM conference on SIGCOMM*. 503–514.

[3] Zixian Cai, Zhengyang Liu, Saeed Maleki, Madanlal Musuvathi, Todd Mytkowicz, Jacob Nelson, and Olli Saarikivi. 2021. Synthesizing optimal collective algorithms. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 62–75.

[4] Peirui Cao, Shizhen Zhao, Min Yee The, Yunzhuo Liu, and Xinbing Wang. 2021. TROD: Evolving From Electrical Data Center to Optical Data Center. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 1–11.

[5] Peirui Cao, Shizhen Zhao, Dai Zhang, Zhuotao Liu, Mingwei Xu, Min Yee Teh, Yunzhuo Liu, Xinbing Wang, and Chenghu Zhou. 2023. Threshold-Based Routing-Topology Co-Design for Optical Data Center. *IEEE/ACM Transactions on Networking* (2023).

[6] Meghan Cowan, Saeed Maleki, Madanlal Musuvathi, Olli Saarikivi, and Yifan Xiong. 2022. MSCCL: Microsoft collective communication library. *arXiv preprint arXiv:2201.11840* (2022).

[7] 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*. IEEE, 2130–2138.

[8] Jianbo Dong, Shaochuang Wang, Fei Feng, Zheng Cao, Heng Pan, Lingbo Tang, Pengcheng Li, Hao Li, Qianyuan Ran, Yiqun Guo, et al. 2021. Accl: Architecting highly scalable distributed training systems with highly efficient collective communication library. *IEEE Micro* 41, 5 (2021), 85–92.

[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*. 225–238.

[10] Paul Grun. 2010. Introduction to infiniband for end users. *White paper, InfiniBand Trade Association* 55 (2010).

[11] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. 2016. RDMA over commodity ethernet at scale. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 202–215.

[12] Christian Hopps. 2000. *Analysis of an equal-cost multi-path algorithm.* Technical Report.

[13] Jinbin Hu, Chaoliang Zeng, Zilong Wang, Junxue Zhang, Kun Guo, Hong Xu, Jiawei Huang, and Kai Chen. 2023. Enabling load balancing for lossless datacenters. In *2023 IEEE 31st International Conference on Network Protocols (ICNP)*. IEEE, 1–11.

[14] Jiawei Huang, Wenjun Lv, Weihe Li, Jianxin Wang, and Tian He. 2018. QDAPS: Queueing delay aware packet spraying for load balancing in data center. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 66–76.

[15] Jiawei Huang, Wenjun Lyu, Weihe Li, Jianxin Wang, and Tian He. 2021. Mitigating packet reordering for random packet spraying in data center networks. *IEEE/ACM Transactions on Networking* 29, 3 (2021), 1183–1196.

[16] Yimin Jiang, Yibo Zhu, Chang Lan, Bairen Yi, Yong Cui, and Chuanxiong Guo. 2020. A unified architecture for accelerating distributed {DNN} training in heterogeneous {GPU/CPU} clusters. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 463–479.

[17] Srikanth Kandula, Dina Katabi, Shantanu Sinha, and Arthur Berger. 2007. Dynamic load balancing without packet reordering. *ACM SIGCOMM Computer Communication Review* 37, 2 (2007), 51–62.

[18] 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.

[19] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, et al. 2019. HPCC: High precision congestion control. In *Proceedings of the ACM Special Interest Group on Data Communication*. 44–58.

[20] 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)*. 357–371.

[21] Qingkai Meng and Fengyuan Ren. 2021. Lightning: A practical building block for rdma transport control. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 1–10.

[22] NVIDIA. 2023. *NVIDIA Spectrum-X Network Platform Architecture - The First Ethernet Network Designed to Accelerate AI Workloads.* Technical Report. NVIDIA.

[23] 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.

[24] Yufei Ren, Xingbo Wu, Li Zhang, Yandong Wang, Wei Zhang, Zijun Wang, Michel Hack, and Song Jiang. 2017. irdma: Efficient use of rdma in distributed deep learning systems. In *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 231–238.

[25] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. 2015. Inside the social network's (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 123–137.

[26] James Salamy. 2022. *Network Requirements for Distributed Machine Learning Training in the Cloud.* Ph.D. Dissertation. Massachusetts Institute of Technology.

[27] Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan Ports, and Peter Richtárik. 2021. Scaling distributed machine learning with

{In-Network} aggregation. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 785–808.

[28] Leah Shalev, Hani Ayoub, Nafea Bshara, and Erez Sabbag. 2020. A cloud-optimized transport protocol for elastic and scalable hpc. *IEEE Micro* 40, 6 (2020), 67–73.

[29] Cha Hwan Song, Xin Zhe Khooi, Raj Joshi, Inho Choi, Jialin Li, and Mun Choon Chan. 2023. Network Load Balancing with In-network Reordering Support for RDMA. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 816–831.

[30] 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.

[31] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. 2023. {TopoOpt}: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 739–767.

[32] Zilong Wang, Layong Luo, Qingsong Ning, Chaoliang Zeng, Wenxue Li, Xinchen Wan, Peng Xie, Tao Feng, Ke Cheng, Xiongfei Geng, et al. 2023. {SRNIC}: A Scalable Architecture for {RDMA}{NICs}. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1–14.

[33] Adam Weingram, Yuke Li, Hao Qi, Darren Ng, Liuyao Dai, and Xiaoyi Lu. 2023. xCCL: A Survey of Industry-Led Collective Communication Libraries for Deep Learning. *Journal of Computer Science and Technology* 38, 1 (2023), 166–195.

[34] William Won, Midhilesh Elavazhagan, Sudarshan Srinivasan, Ajaya Durg, Swati Gupta, and Tushar Krishna. 2023. TACOS: Topology-Aware Collective Algorithm Synthesizer for Distributed Training. *arXiv preprint arXiv:2304.05301* (2023).

[35] Cliff Woolley. 2015. Nccl: Accelerated multi-gpu collective communications. (2015).

[36] Jiacheng Xia, Gaoxiong Zeng, Junxue Zhang, Weiyan Wang, Wei Bai, Junchen Jiang, and Kai Chen. 2019. Rethinking transport layer design for distributed machine learning. In *Proceedings of the 3rd Asia-Pacific Workshop on Networking 2019*. 22–28.

[37] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. 2014. WCMP: Weighted cost multipathing for improved fairness in data centers. In *Proceedings of the Ninth European Conference on Computer Systems*. 1–14.