2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA) | 979-8-3315-0971-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISPA63168.2024.00149

Improving Availability and Scalability for RDMA Load Balancing with In-network Reordering

Jinbin Hu[1], Ruiqian Li[1], Shuying Rao[1], Jin Wang[1,2]
[1]Changsha University of Science and Technology, Changsha, China
[2]Hunan University of Science and Technology, Xiangtan, China
jinbinhu@csust.edu.cn, jinwang@hnust.edu.cn
{leeerq, shuyingrao}@stu.csust.edu.cn

***Abstract*—Remote Direct Memory Access (RDMA) is widely deployed in datacenter networks (DCNs) due to its ultra-low latency, high throughput, and low CPU overhead. Since RDMA is sensitive to out-of-order packets, the previous load balancing schemes designed based on TCP do not work well in RDMA networks. Recently proposed load balancing schemes focus on solving packet reordering within the network. However, the existing solutions cannot be extended in practice because the required queues far exceed common switch capabilities. In this paper, we propose a scalable and efficient load balancing called SELB to improve availability and scalability. SELB employs a clustering algorithm to categorize equal-cost paths and then reroutes traffic to the same cluster parallel paths to reduce the degree of out-of-order and improve queue utilization. The NS-3 simulation results demonstrate that SELB reduces the average flow completion time (FCT) and the 99th percentile FCT by up to 33% and 21%, respectively, compared to the state-of-the-art load balancing schemes.**

***Index Terms*—RDMA, Datacenter networks, Load balancing, Packet reordering**

I. Introduction

Modern data centers are seeing a significant growth in the number of online applications and services, which has increased demand from businesses for ultra-low latency and high throughput transmission performance. Remote Direct Memory Access (RDMA) allows end hosts to bypass the CPU and directly access remote memory through RDMA-capable network interface cards (RNICs). Given its ability to achieve high-speed data transfer to meet the needs of ultra-low latency and high throughput, RDMA is well suited for modern datacenter networks (DCNs). As a result, DCNs have extensively embraced RDMA technology through RoCEv2 [1]–[3], [23], [27].

DCNs often adopt network topologies such as leaf-spine architecture to provide multiple equal-cost paths between hosts. In this multi-path transmission environment, load balancing technology is essential to optimize network performance. A widely used load balancing method in DCNs is equal-cost multi-path (ECMP) routing [4]. ECMP operates by distributing flows over multiple equal-cost paths based on a hashing mechanism, which helps to efficiently utilize network resources and alleviate congestion on links. Numerous studies have shown that ECMP is not ideal for load balancing due to factors such as hash collisions and traffic skew [5], [6]. Flowlet-based methods, such as LetFlow [7] and CONGA [8], divide flow into flowlets by detecting time intervals between packet clusters and then randomly forwarding flowlets to different paths. DRILL [9], as a packet-based method, makes routing strategies for each packet based on local queue occupancy and a random algorithm. These methods effectively avoid the shortcomings of ECMP, reduce path congestion, and improve link utilization.

However, the aforementioned load balancing schemes are designed for TCP rather than RDMA. Flowlet-based methods are not effective due to the specific flow characteristics of RDMA, and packet-based methods are challenging to adopt due to RDMA's sensitivity to out-of-order packets. Recently, a load balancing scheme called Conweave [10] has been proposed specifically for RDMA, addressing the challenge of out-of-order data packets within the network. Conweave performs fine-grained load balancing of RDMA flows by leveraging an in-network reordering mechanism to ensure that out-of-order packets can be reordered effectively. It ensures the deterministic arrival of flows at the destination Top of Rack (ToR) switch through cautious rerouting decisions, making it easy to reorder packets using a single queue. Therefore, it is crucial to have an adequate number of queues to support Conweave's reordering mechanism.

In a large-scale network environment, the limited queue resources are likely to be exhausted, resulting in the failure of the reordering mechanism [24], [26], [28]. To address this issue, this paper proposes a load balancing scheme called SELB, which improves reordering efficiency through improved rerouting strategy. First, to prevent out-of-order packets from arriving at the target ToR switch too early and causing the reordering queue to be occupied for extended periods, SELB classifies paths based on their queue lengths through a clustering algorithm. Second, SELB assesses path congestion by analyzing the growth rate of the queue length to inform rerouting decisions and efficiently reorder out-of-order packets at the destination ToR switch. SELB achieves better performance compared to existing methods.

The main contributions of this paper are as follows:

- We analyze the limitation of Conweave in larger-scale networks, including the inefficient utilization of reorder-

*Jin Wang is the corresponding author.

2158-9208/24/$31.00 ©2024 IEEE
DOI 10.1109/ISPA63168.2024.00149

ing queues caused by Conweave's rerouting mechanism and the impact of insufficient queues on transmission performance.

- We design a load balancing scheme called SELB that classifies paths according to queue length differences to improve rerouting strategies. Our design reduces the time that out-of-order packets occupy queue resources, meets the reordering requirements of more flows, and has better availability and scalability.
- We evaluate SELB by NS-3 simulation under two realistic workloads. The result shows that SELB effectively improves the transmission performance of RDMA network and significantly reduces tail latency and average flow completion time (AFCT).

The paper is organized as follows: The motivation is given in Section II. The design is described in depth in Section III. The evaluation results are presented in Section IV. Related works are discussed in Section V. The paper is finally concluded in Section VI.

II. Motivation

In this section, we investigate the influence of insufficient queues on Conweave packet reordering in typical DCNs scenarios to motivate our design.

A. Limited Queue Resource

Conweave periodically decides whether rerouting is beneficial based on network measurements. Packets are rerouted only when out-of-order packets can be effectively reordered within the network before being delivered to the end host. To minimize the cost of reordering at the destination ToR switch, it is necessary to produce predictable packet arrival patterns. Specifically, only flows that have completed the last rerouting operation are rerouted again, ensuring that any flow can have packets in transit on at most two paths at any time. Therefore, at the destination ToR switch, Conweave requires only one queue per flow to cache packets that arrive out-of-order due to rerouting (through the queue pausing/resuming feature of the programmable switch [11]). When the last packet on the old path reaches the destination ToR switch, the switch releases the out-of-order packets to ensure the reordering of packets within the network.

However, effective Conweave reordering relies on sufficient available queue resources. When available queue resources are insufficient, Conweave applies ECMP to the remaining connections as a backup. In large-scale networks, this issue becomes significant, which will seriously affect the transmission performance of the RDMA network. Minimizing the queue resource occupation of Conweave reordering is thus an urgent problem. We also observe that the earlier the out-of-order packets of the flow arrive at the destination ToR switch, the greater the degree of disorder, causing the out-of-order packets to occupy the reordering queue for a longer time. Therefore, the path for rerouting should not just be faster but should consider the delay differences of the paths. Conweave simply forwards the flow to a non-congested path without considering path delay differences, reducing the efficiency of queue resources utilization and exacerbating the problem of queue resources shortage.

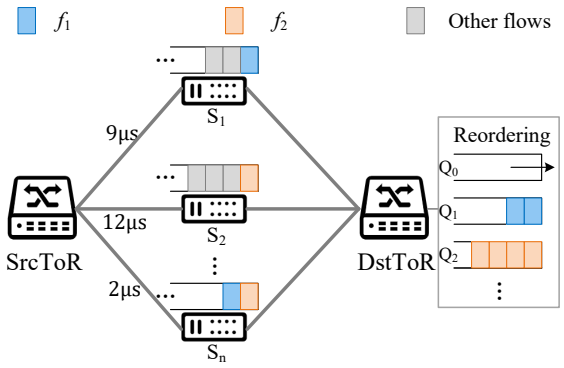


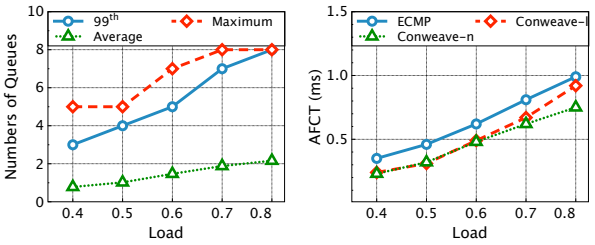
Fig. 1: Study case

We illustrate the problem Conweave faces when the queue for reordering is insufficient through an example. As shown in Fig. 1, we assume that three queues are available on the switch, with the default queue Q_0 having the lowest scheduling priority and being used for flows that arrive in order (according to Conweave). If congestion is detected on the paths where spine switches S_1 and S_2 are located, the source ToR switch will reroute flows on congested paths. The packets of f_1 and f_2 arrive at the destination ToR switch out of order through the path where spine switch S_n is located. The destination ToR switch uses Q_1 and Q_2 to cache the out-of-order packets of f_1 and f_2, respectively. At this point, the queue resources are exhausted, and other flows on congested paths have no chance to be rerouted. This not only prevents timely alleviation of path congestion but also reduces the transmission performance of other flows.

B. Experimental Observation

We examine the effect of Conweave on flow completion time in the case of insufficient queues using NS-3 [12] simulation experiments. We employ a leaf-spine architecture, which consists of 8 leaf switches and 8 spine switches, that is frequently found in data centers. With a 2:1 over-subscription ratio, each rack contains 16 servers. The link bandwidth is 100 Gbps, with a latency of 1 μs. We use AliCloud storage [13] as the workload in the simulation. In this example experiment, we limit the number of queues available for reordering on each port of Conweave-l to 8, while Conweave-n is unlimited, so as to better show the influence of insufficient queues.

In order to simulate a moderately and heavily loaded network, respectively, we conduct the simulations at 0.4 and 0.8 of the typical traffic load. Fig. 2. (a) displays how many queues are in use for every switch port when simulating Conweave-l. Under high load, some ports reach the limit of 8 queues. Since there are more active flows than available queues, Conweave has to use ECMP as a fallback. Therefore,



(a) Numbers of Queues (b) AFCT

Fig. 2: Influence of insufficient queues

as shown in Fig. 2. (b), the AFCT increases significantly under high load compared to the case where there is no limit on queue resources. On a commercial switch, there may be 32 or more queues per port, but there are also far more active flows than in the simulation situation. Therefore, the number of queues actually required is likely to reach the upper limit, thus limiting the opportunity for traffic rerouting and causing Conweave's actual transmission performance to degrade, making it closer to the performance of ECMP.

III. Design

In this section, we present the SELB design. We combine an improved rerouting approach with Conweave's reordering method to increase queue usage effectiveness and mitigate the queue scarcity issue.

A. Design Overview

The key to SELB is to classify paths using clustering algorithms and reroute flows between similar paths to avoid large delay differences that cause out-of-order packets to arrive too early. Specifically, timely rerouting decisions can steer flows away from congested paths. Additionally, rerouting to a path with a similar queue length, but relatively better, can minimize the occupancy time of the reordering queue and release queue resources faster. There are three components making up the SELB framework, which is shown in Fig. 3.

Path Classification: SELB performs path classification on the source ToR switch by categorizing paths based on the length of the egress port queue. The switch periodically measures the queue length for each path and uses a clustering algorithm to classify them into different clusters. Unlike Conweave, SELB does not continuously monitor Round Trip Time (RTT), which reduces resources consumption and enables faster path updates.

Rerouting Strategy: SELB determines whether a flow needs rerouting based on changes in queue length. It does not always reroute to the path with the shortest queue. Instead, if a path's queue length increases rapidly, it is marked as congested, and traffic is rerouted only after the last packet is sent. SELB then randomly selects a non-congested path of the same type for rerouting. If all paths of the same cluster are congested, traffic is rerouted to the path with the shortest queue to prevent further congestion.

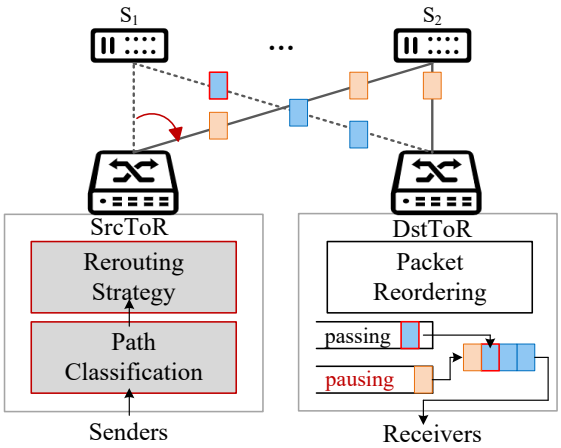


Fig. 3: SELB overview

Packet Reordering: SELB uses the same method as Conweave to reorder out-of-order packets on the destination ToR switch. The rerouting strategies of both schemes allow a flow to be rerouted at most once at the same time, leading to a predictable flow arrival pattern. Packets that arrive in order pass through the highest priority queue Q_0. Out-of-order packets are paused in the reorder queue until the tail packet on the old path arrives. Since SELB's out-of-order packets do not arrive at the destination ToR switch too early, each rerouted flow does not occupy the queue for too long, allowing queue resources to be released more quickly.

B. Design Details

To better illustrate the design of SELB, we explain the difference between SELB and Conweave in rerouting through the following example. As shown in Fig. 4, let's first look at how Conweave operates. When continuous RTT monitoring detects an RTT timeout, the switch randomly selects some sample paths and checks whether they are marked as unavailable. If there is an available path, the flow is rerouted. If not, rerouting is not done because the network is deemed to be extremely congested. The rerouting path is completely unpredictable, leading to extreme situations. For example, when tail packet 3 on the old path has not yet arrived, packets 4, 5, 6, and 7 may have already arrived at the destination ToR switch early from a better path. This means that this flow will occupy the reordering queue resources for a long time, which is inefficient and unnecessary. SELB's approach is to select a rerouting path within the same cluster of paths with similar queue lengths. We can see in Fig. 4 that out-of-order packets will not arrive too early. When data packet 4 arrives, tail packet 3 on the old path will also arrive soon. Queue Q_1 only needs to maintain the out-of-order packets for a short time and can be quickly emptied for other flows to reorder.

Path Classification: The switch periodically obtains local queue length information and then classifies the paths using a clustering algorithm. Algorithm 1 outlines a clustering method for organizing paths based on queue lengths. The algorithm

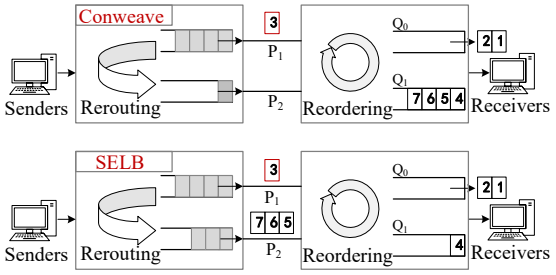


Fig. 4: Differences between Conweave and SELB

takes as input a set of paths P, a corresponding set of queue lengths Q, and the desired number of clusters k. Initially, k centroids are randomly selected from Q. The algorithm iteratively assigns each path to the cluster with the nearest centroid and subsequently updates each cluster's centroid to the mean queue length of the paths within that cluster. This process repeats until the centroids stabilize, resulting in k clusters of similar paths based on queue lengths. After classification, the path set with the shortest average queue length will be marked as a backup option for rerouting flows on highly congested paths, while other flows are only allowed to be rerouted within the same path cluster. In fact, highly congested path clusters occur rarely because we can predict path congestion based on the growth rate of queue length and make timely rerouting decisions to alleviate it.

Rerouting Strategy: For each egress queue, SELB continuously monitors the change in queue length to detect congestion. When the queue length growth rate exceeds θ, the corresponding path is marked as congested. Similar to θ_{reply} in Conweave, which affects the switching granularity, a smaller value of θ provides more opportunities for rerouting but also increases the overhead of reordering. We set a compromise value for θ in the actual configuration to balance these trade-offs. The switch then searches the cluster where the path is located to see if there are non-congested queues. If so, these paths are selected as rerouting paths. After the flow on the congested path sends the tail packet on the old path, it is rerouted to the new path in the same cluster. If there is no available queue, it indicates that all paths in the cluster are highly congested, making rerouting within the same cluster ineffective. Therefore, flows on these congested paths are directly rerouted to the path with the shortest queue length to balance the load between paths.

The improved rerouting strategy based on path classification makes packet reordering more efficient. Specifically, under ideal conditions, most out-of-order packets arrive only slightly earlier than the old path, and occupy the queue for a very short time. Only a small number of flows that cannot be rerouted on the same cluster path need to occupy the reordering queue for a long time. But overall, the efficiency of reordering has increased, so SELB has better availability and scalability compared with Conweave.

Algorithm 1: Path Clustering Algorithm

Input: Set of paths P, set of queue lengths Q, number of clusters k
Output: k clusters of similar paths C
Initialize k centroids $centroids \leftarrow$ randomly select k different values from Q;
repeat
  for *each path* $p_i \in P$ **do**
    $q_i \leftarrow$ queue length of path p_i;
    $c_{closest} \leftarrow$ centroid closest to q_i;
    Assign p_i to cluster corresponding to $c_{closest}$;
  end
  for *each cluster* C_j **do**
    if C_j *is not empty* **then**
      Update centroid of C_j to be the mean queue length of all paths in C_j;
    end
  end
until *centroids do not change*;
return k *clusters of similar paths* C

IV. Evaluation

A. Simulation Setup

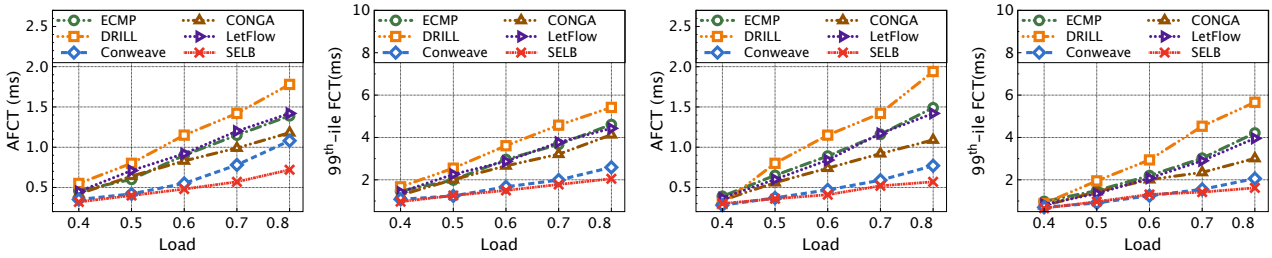
Workloads: Our simulation involves two classic workloads: AliCloud Storage [13] and Meta Hadoop [14]. Both workloads exhibit a heavy-tailed distribution typical of production data centers. Traffic loads range from 0.4 to 0.8, and they are generated randomly between host pairs using a Poisson distribution. For IRN RDMA [15] networks, we use end-to-end flow control that limits the number of in-flight packets to one BDP (BDP-FC); for Lossless RDMA networks, we use priority-based flow control (PFC) [16].

Topology: We use the leaf-spine topology, a typical topology of data centers, which has 8 leaf switches and 8 spine switches. There are 256 servers in the network, which means that each leaf switch is connected to 32 servers. Each link has a latency of 1 μs and a bandwidth of 100 Gbps. For congestion control, we employ the standard DCQCN [17] scheme for commercial RNICs.

Baseline: We contrast SELB with LetFlow, CONGA, DRILL, ConWeave, and ECMP. LetFlow and CONGA both have a flowlet timeout of 10 μs. We utilize the suggested DRILL(2,1) configuration for DRILL, in which the new output port with the smallest backlog is chosen from two random samples and one current port. We use the suggested parameter values for a two-tier topology for ConWeave, setting θ_{reply}, $\theta_{\text{path\_busy}}$, and θ_{inactive} to 8 μs, 8 μs and 300 μs, respectively.

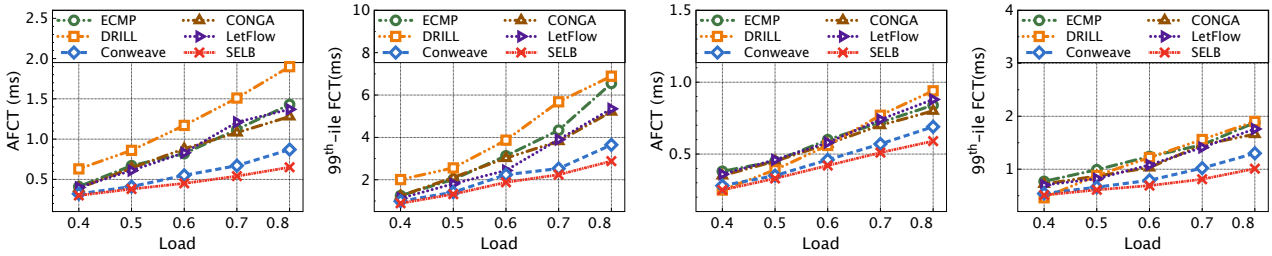
B. Performance under AliCloud Storage Workload

First, we assess how well SELB and five different load balancing strategies perform when used under AliCloud Storage. The performance enhancement of SELB in AliCloud Storage workload is displayed in Fig. 5. The findings show that, particularly in scenarios with high workloads, SELB



(a) AFCT in Lossless RDMA (b) 99^{th}-ile FCT in Lossless RDMA (c) AFCT in IRN RDMA (d) 99^{th}-ile FCT in IRN RDMA

Fig. 5: Performance under Alicloud Storage Workload



(a) AFCT in Lossless RDMA (b) 99^{th}-ile FCT in Lossless RDMA (c) AFCT in IRN RDMA (d) 99^{th}-ile FCT in IRN RDMA

Fig. 6: Performance under Meta Hadoop Workload

dramatically lowers the AFCT and the 99^{th} percentile FCT of short flows. Specifically, ECMP cannot perform fine-grained rerouting in the case of path congestion, resulting in long queuing delays. LetFlow and CONGA are difficult to work due to the characteristics of RDMA flows so that there is a lack of sufficiently large flowlet gaps. Even if the flowlet timeout is set to 10μs, flows that encounter congestion have difficulty rerouting. DRILL performs particularly poorly in RDMA networks because it reroutes at the packet granularity level, causing severe packet reordering. ConWeave can effectively reroute traffic and prevent packet disorder, performing significantly better than the aforementioned methods at low and medium loads. However, under high load, its reordering function is affected due to limited queue resources.

SELB reorders packets more efficiently through an improved rerouting strategy, enabling frequent rerouting and ensuring in-order delivery of packets even under high loads. SELB effectively balances the load and reduces the AFCT and tail latency. For example, under 0.8 workload, SELB reduces the AFCT in Lossless RDMA by 48%, 49%, 39%, 60%, and 33%, and reduces the 99^{th} percentile FCT by 56%, 54%, 50%, 62%, and 21% compared to ECMP, LetFlow, CONGA, DRILL, and ConWeave, respectively. Besides, SELB also achieves better performance in IRN RDMA, reducing the AFCT by 62%, 60%, 48%, 71%, and 26%, and the 99^{th} percentile FCT by 62%, 59%, 46%, 71%, and 21%.

C. Performance under Meta Hadoop Workload

We continue to simulate SELB and five baseline load balancing schemes under Meta Hadoop workloads to determine its performance in a wider range of situations. As shown in Fig. 6, we still analyze the results both for IRN and Lossless RDMA networks. In general, SELB demonstrates better transmission performance than existing schemes. CONGA, LetFlow, and ECMP all perform similarly, with poor performance. It is worth noting that DRILL unexpectedly performs well in IRN RDMA networks under low load. This is because the IRN RDMA with the Selective-Repeat (SR) loss recovery mechanism has a slightly higher tolerance for out-of-order packets than the Lossless RDMA network with GO-BACK-N (GBN). Besides, the Meta Hadoop workload has a higher proportion of short flows than AliCloud Storage, resulting in fewer out-of-order packets. However, DRILL still performs poorly under high loads or in Lossless RDMA networks. Obviously, due to RDMA's low tolerance to out-of-order transmission, it is not suitable in most cases.

We focus on the improvement of SELB compared to Conweave. ConWeave performs better than other TCP-based solutions, but still not as well as SELB. This verifies the problem raised in this paper: if queue resources are insufficient, Conweave's reordering mechanism is limited, and performance will naturally be affected. Therefore, once the number of active flows is large, the superiority of SELB is reflected. For example, in a Lossless RDMA network under 0.8 load, SELB reduces the AFCT and 99^{th} percentile FCT by 25% and 21%, respectively compared to ConWeave, while in an IRN RDMA network, it reduces them by 16% and 23%, respectively. SELB demonstrates superior performance in both scenarios, proving its scalability and effectiveness.

V. Related Work

Load balancing schemes: To fully exploit multi-path, various schemes implement load balancing at fine granularities. [9], [18] employs packets as the scheduling granularity to maximize network bandwidth utilization, but it often leads to significant packet reordering. In contrast, [7], [8], [19], [20] operate at the flowlet granularity, identifying new flowlets for rerouting based on the inter-packet time intervals, which allows for dynamic adaptation to network congestion. On the other hand, [4], [21] prevent packet out-of-order delivery with flow granularity, despite the cost of low link usage.

Multipath transports for RDMA: MP-RDMA [22] uses a novel multipath ACK clock and out-of-order path selection method to select the best network path and distribute packets among them in a congestion-aware manner. MP-RDMA is implemented through custom-designed RNICs or is purely software-based, but may not be compatible with traditional RNICs. Maestro [23] is a purely software-based modular multipath RDMA solution. It proposes a virtual network card-based user space path control mechanism and a middleware transport layer to achieve efficient multipath transmission in RDMA without introducing additional memory copies.

VI. Conclusion

This paper introduces a scalable and efficient load balancing scheme called SELB, which classifies paths through a clustering algorithm and reroutes traffic to parallel paths in the same cluster to reduce the degree of disorder. SELB uses an improved rerouting strategy to better balance traffic and more efficiently utilize queues to ensure that traffic arrives at the host in order. Simulation evaluation show that SELB outperforms existing designs, reducing AFCT and 99^{th} percentile FCT by 33% and 21%, respectively.

Acknowledgment

This work was funded by the National Natural Science Foundation of China (62472050, 62473146, 62102046, 62072056), and by the Key Project of Natural Science Foundation of Hunan Province (2024JJ3017), and by the New Generation Information Technology Innovation Project 2023 (2023IT271), and by the Science and Technology Project of Hunan Provincial Department of Water Resources (XSKJ2024064-36).

References

[1] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn. RDMA over Commodity Ethernet at Scale. In Proc. ACM SIGCOMM, 2016, 202-215.

[2] G. Yao, Q. Li, L. Tang, Y. Xi, P. Zhang, W. Peng, B. Li, Y. Wu, S. Liu, and L. Yan. When Cloud Storage Meets RDMA. In Proc. USENIX NSDI, 2021, 519-533.

[3] J. Hu, H. Shen, X. Liu, J. Wang. RDMA Transports in Datacenter Networks: Survey. IEEE Network, 2024, DOI: 10.1109/MNET.2024.3397781.

[4] C. Hopps, Analysis of an Equal-Cost Multi-Path Algorithm. Technical Report, 2000.

[5] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In Proc. USENIX NSDI, 2010, 89-92.

[6] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella. Presto: Edge-Based Load Balancing for Fast Datacenter Networks. ACM SIGCOMM Computer Communication Review, 2015, 45(4), 465-478.

[7] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, and T. Edsall. Let It Flow: Resilient Asymmetric Load Balancing with Flowlet Switching. In Proc. USENIX NSDI, 2017, 407-420.

[8] M. Alizadeh, T. Edsall, S. Dharmapurikar, et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters. In Proc. ACM SIGCOMM, 2014, 503-514.

[9] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian. Drill: Micro Load Balancing for Low-Latency Data Center Networks. In Proc. ACM SIGCOMM, 2017, 225-238.

[10] C. H. Song, X. Z. Khooi, R. Joshi, I. Choi, J. Li, and M. C. Chan. Network Load Balancing with In-Network Reordering Support for RDMA. In Proc. ACM SIGCOMM, 2023, 816-831.

[11] J. Hu, C. Zeng, Z. Wang, et al. Load Balancing with Multi-level Signals for Lossless Datacenter Networks. IEEE/ACM Transactions on Networking, 2024, 32(3), 2736-2748.

[12] G. F. Riley and T. R. Henderson, The ns-3 network simulator. Modeling and Tools for Network Simulation, Springer, 2010, 15-34.

[13] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, and M. Alizadeh. HPCC: High Precision Congestion Control. In Proc. ACM SIGCOMM, 2019, 44-58.

[14] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the Social Network's (Datacenter) Network. In Proc. ACM SIGCOMM, 2015, 123-137.

[15] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker. Revisiting Network Support for RDMA. In Proc. ACM SIGCOMM, 2018, 313-326.

[16] IEEE 802.1 Qbb - Priority-based Flow Control. https://1.ieee802.org/-dcb/8021qbb/.

[17] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang. Congestion Control for Large-Scale RDMA Deployments. ACM SIGCOMM Computer Communication Review, 2015, 45(4), 523-536.

[18] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella. On the Impact of Packet Spraying in Data Center Networks. In Proc. IEEE INFOCOM, 2013, 2130-2138.

[19] N. Katta, A. Ghag, M. Hira, I. Keslassy, A. Bergman, C. Kim, and J. Rexford. Clove: Congestion-Aware Load Balancing at the Virtual Edges. In Proc. ACM CoNEXT, 2017, 323-335.

[20] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford. Hula: Scalable Load Balancing Using Programmable Data Planes. In Proc. ACM SIGCOMM Symposium on SDN Research (SOSR), 2016, 1-12.

[21] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In Proc. USENIX NSDI, 2010, 89-92.

[22] Y. Lu, G. Chen, B. Li, K. Tan, Y. Xiong, P. Cheng, J. Zhang, E. Chen, and T. Moscibroda. Multi-Path Transport for RDMA in Datacenters. In Proc. USENIX NSDI, 2018, 357-371.

[23] W. Dou, B. Liu, C. Lin, X. Wang, X. Jiang, and L. Qi. Architecture of Virtual Edge Data Center with Intelligent Metadata Service of a Geo-Distributed File System. Journal of Systems Architecture, 2022, 128, 102545.

[24] J. Hu, J. Huang, J. Wang, and J. Wang. A Transmission Control Mechanism for Lossless Datacenter Network Based on Direct Congestion Notification. Acta Electronica Sinica, 2023, 51(9), 2355-2365.

[25] F. Tian, Y. Zhang, W. Ye, C. Jin, Z. Wu, and Z.-L. Zhang. Accelerating Distributed Deep Learning Using Multi-Path RDMA in Data Center Networks. In Proc. ACM SIGCOMM Symposium on SDN Research (SOSR), 2021, 88-100.

[26] J. Hu, Y. He, W. Luo, J. Huang, and J. Wang. Enhancing Load Balancing With In-Network Recirculation to Prevent Packet Reordering in Lossless Data Centers. IEEE/ACM Transactions on Networking, 2024, 32(5), 4114-4127.

[27] X. Wang, L. T. Yang, H. Liu, and M. J. Deen. A Big Data-as-a-Service Framework: State-of-the-Art and Perspectives. IEEE Transactions on Big Data, 2017, 4(3), 325-340.

[28] J. Wang, Y. He, W. Luo, S. Rao and J. Hu. A Novel Load Balancing Scheme based on PFC Prediction in Lossless Datacenter Networks. Human-centric Computing and Information Sciences (HCIS), 2024, DOI: https://doi.org/10.22967/HCIS.2024.14.059