Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

PDLB: Path Diversity-aware Load Balancing with adaptive granularity in data center networks

Weimin Gao[1,2], Jiawei Huang[1*], Zhaoyi Li[1], Shaojun Zou[3] and Jianxin Wang[1]

Abstract

Modern data center topologies often take the form of a multi-rooted tree with rich parallel paths to provide high bandwidth. However, various path diversities caused by traffic dynamics, link failures and heterogeneous switching equipment widely exist in the production datacenter network. Therefore, the multi-path load balancer in data center should be robust to these diversities. Although prior fine-grained schemes such as RPS and Presto make the best use of available paths, However, they are prone to experiencing packet reordering problem under the asymmetric topology. The coarse-grained solutions such as ECMP and LetFlow effectively avoid packet reordering, but easily lead to under-utilization of multiple paths. To cope with these inefficiencies, we propose a load balancing mechanism called PDLB, which adaptively adjusts flowcell granularity according to path diversity. PDLB increases flowcell granularity to alleviate packet reordering under large degrees of topology asymmetry, while reducing flowcell granularity to obtain high link utilization under small degrees of topology asymmetry. PDLB is only deployed on the sender without any modification on switch. We evaluate PDLB through large-scale NS2 simulations. The experimental results show that PDLB reduces the average flow completion time by up to ~8-53% over the state-of-the-art load balancing schemes.

Keywords Path diversity-aware, Adaptive granularity, Load balancing, Data center networks

Introduction

The data center network connects a large number of hosts through switches to provide large-scale computing and storage. In order to improve network transmission performance, new multi-rooted tree topologies such as Fat-tree [1], Clos [2], and VL2 [3] have appeared in the architecture design of data center network. These new network topologies provide multiple available transmission paths between the source host and the destination host. Parallel multi-path transmission can greatly improve the network throughput and reliability of the data center.

The primary goal of load balancing is to evenly distribute traffic to each parallel path, improve network link utilization, and avoid network congestion caused by burst traffic. Equal Cost Multi-Path (ECMP) [4] is the most typical flow-level load balancing scheme in production datacenter, which uses flow hashing to transfer flows to available paths. LetFlow [5] and CONGA [6] reroute flowlets to avoid packet reordering. Random Packet Spraying (RPS) [7], Presto [8] and Distributed Randomized In-network Localized Load balancing (DRILL) [9] flexibly split flows at a finer granularity to make best use of all available paths.

*Correspondence:
Jiawei Huang
jiaweihuang@csu.edu.cn
[1] School of Computer Science and Engineering, Central South University, Changsha 410083, China
[2] Department of Computer Science and Engineering, Hunan Institute of Technology, Hengyang 421002, China
[3] School of Computer Science and Technology, Hainan University, Haikou 570228, China

However, as the data center runs over time, traffic dynamics, topology asymmetry, and switch failures will appear [8, 10]. Under such uncertainty, the multiple paths become diverse or asymmetric. When these load balancing schemes scatter data packets on congested paths, some flows will inevitably experience unpredicted congestion and packet disordering. The degradations of transmission delay and flow throughput result in a longer application response time and a worse user experience. For example, Presto uses the maximum TCP Segment Offload (TSO) size of 64 KB as the flow unit granularity, allowing fine-grained load balancing at the network bandwidth of more than 10 Gbps. However, since Presto uses fixed-size packet clusters to split the traffic without considering the path diversity, Presto has problems with out-of-order and packet loss in asymmetric topology.

In this paper, in order to solve the robustness and flexibility of the above methods under asymmetric topology, we propose a load balancing mechanism called Path Diversity-aware Load Balancing (PDLB), which adaptively adjusts flowcell sizes to obtain both low packet reordering and high link utilization. In particular, flowcell is defined as a packet cluster composed of *gran* packets, flowcell size is also *gran*, which is equal to 2^n KB (n=0, 1, 2... 6). Flowcell is also a balance loader with the packet cluster size of TCP Segmentation Offload (TSO) unit. To mitigate the impact of uncertainties under high path asymmetry, PDLB increases the size of flowcells to alleviate packet reordering. On the contrary, when the path asymmetry is low, PDLB reduces the size of the flowcells to achieve high link utilization. Moreover, PDLB is deployed on the sender with negligible overhead, while making no modifications on switch. From the perspective of performance, PDLB is sensitive to path latency and periodically adjusts flowcell granularity at the sending end. It exhibits high flexibility and resiliency under asymmetric topology and achieves good load balancing.

In summary, our major contributions are:

- We conduct an extensive simulation-based study to analyze the impact of path asymmetry on the load balancing performance. We demonstrate experimentally and theoretically why controlling the granularity of the flowcell is effective in avoiding large tailed flow completion time (FCT) and packet reordering under large path asymmetry.
- We propose a flowcell-level load balancing scheme, PDLB, to spread flowcell across the multiple paths, which are adaptively selected according to path diversity. PDLB rationally adjusts the size of the flowcell to improve link utilization under small path asymmetry and reduce tailed latency under large path asymmetry.
- By using large-scale NS2 simulations, we demonstrate that PDLB performs remarkably better than the state-of-the-art load balancing designs under different realistic traffic workloads. Especially, PDLB greatly reduces the mean FCT up to ~8-53% over RPS, ECMP, LetFlow, Presto and Queuing Delay Aware Packet Spraying (QDAPS).

The remainder of this paper is structured as follows. In Related works section, we introduce related works. In Background and motivation section, we demonstrate the design motivation. We present the design overview and details of PDLB in Adaptively adjusting flowcell granularity and The Algorithm of PDLB sections. In Simulation evaluation section, we evaluate the performance of PDLB using NS2 simulation. Finally, we conclude the conclusion and future work in Conclusion and future work section.

Related works

In recent years, researchers have proposed various load balancing mechanisms to facilitate parallel data transmission across multiple paths. The state-of-the-art algorithms mainly include RPS, LetFlow, ECMP, Presto, and QDAPS. The comparison is shown in the Table 1. RPS and QDAPS are classical packet-level load balancing algorithms, while LetFLow and Presto are classical flowlet-level load balancing algorithms, as well as ECMP, is typical of flow-level load balancing.

LetFlow [5] uses the natural attributes between data packets to automatically sense path congestion. When a flow encounters congestion on a certain path, the interval between data packets will increase, naturally forming a packet cluster with a time interval. LetFlow

Table 1 Comparisons with state-of-the-art load balancing algorithms

Schemes	Relevance
RPS	Randomly forwards packets of a flow to all paths at switches.
LetFlow	Picks a random path for each flowlet.
Presto	Splits a flow into many units with a fixed size (i.e.,64KB) and spreads the units to all paths.
ECMP	Selects a random path from *n* paths for forwarding.
QDAPS	Selects the output port for each packet based on the queueing delay of the last arriving packet in the same flow to avoid packet reordering.

Table 2 The comparison of flowlet-level load balancing schemes

Schemes	Key design	Advantage and Disadvantage (A &D)
LetFlow [5]	Selects the forwarding path for each packet cluster randomly at a fixed time interval.	Suitable for asymmetric networks, but routing randomly.
Presto [8]	Uses the central controller to collect topology information.	Suitable for asymmetric networks. It's not necessary to modify the protocol stack and hardware, but the deployment is complex.
Burst-Balancer [9]	Only manipulates a small amount of critical flowlets.	Suitable for symmetric and asymmetric topologies, but the overhead is not small.
CONGA [6]	The switch selects the lightest congestion path for each flowlet according to the congestion information table and the flowlet table.	Suitable for asymmetric networks, but the feedback delay is too large and the scalability is poor.
HULA [11]	The path of forwarding flowlet is the best next hop.	Solved the scalability issue of CONGA, and the overhead of forwarding tables is low, but has a herd effect.
Luopan [12]	Samples the congestion information of some paths, and forwards the fixed-size flowlet to the lightest congestion path.	Low overhead, suitable for asymmetric networks. However, sensing partial congestion cannot guarantee global optimization.

distinguishes the packet clusters according to the time interval threshold, and randomly sends the packet clusters to other paths. Since the interval threshold is generally greater than the maximal path difference, LetFlow can avoid disorder and effectively combat asymmetry problems. However, due to the randomness of flowlet scheduling, the optimal load balancing performance cannot be achieved.

Presto [8] is a typical flowcell-based mechanism that sprays the flowcells onto all available paths in a round-robin style. Presto uses a software switch to split flows into 64KB data slices. Presto uses the central controller to collect topology information. The software switch on the host sends each packet cluster of equal size of 64KB to all available paths in a poll. The receiving end modifies the Generic Receive Offload (GRO) mechanism to reduce disorder.

BurstBalancer [9] is an efficient load balancing system, with the aim of manipulating only a small number of critical flowlets. BurstBalancer devises a sketch, namely BalanceSketch, and deploys it on each switch to detect and make forwarding decisions for each FlowBurst. BurstBalancer only needs small on-chip memory to keep critical flowlets (FlowBursts), and thus perfectly embraces the highly skewed flow distribution. Further, BurstBalancer only anipulates the critical flowlets, which are minimizing packet reordering.

CONGA [6] designs a distributed algorithm to obtain global congestion information in leaf-spine topologies, and assigns each flowlet to the least congested path at leaf switches. It can sense the congestion and failure of the path, and thus can be applied to asymmetric networks. However, CONGA needs to store a large amount of path information and use customized switches, which is difficult to deploy on a large scale.

Hop-by-hop Utilization-aware Load balancing Architecture (HULA) [11] uses programmable switches to achieve congestion sensing load balancing. HULA periodically sends detection packets to all available paths to collect global link utilization information. Based on the detection packet feedback information, each switch selects the next hop path with the lowest path utilization and notifies it to all neighbor nodes. At the same time, each switch maintains a congestion information table to store the best next hop path to the destination, and therefore it effectively eliminates the storage pressure of path explosion on the switch. However, because HULA only selects the best next-hop path, it will cause congestion on the best path.

Luopan [12] is a sample-based load balancing scheme. Luopan uses a fixed-size packet cluster as the scheduling unit, and the packet cluster size is set as the maximum TSO size of 64KB. For the multiple paths, Luopan periodically samples part of the paths, and then forwards the packet cluster directly to the path with the smallest queue length to achieve load balancing. The comparison of flowlet-level load balancing schemes is shown in Table 2.

Some packet-level load balancing research work are proposed, such as QDAPS and Adaptively Adjusting Concurrency (AAC). QDAPS [13] is a queueing delay-aware load balancing mechanism that significantly reduces the flow completion time. QDAPS selects a suitable output queue to ensure that the packets arrive at the receiver in order and avoid the low link utilization in a flexible manner. Moreover, QDAPS also designs a flow rerouting method to reduce the queueing delay of long flows. Compared with QDAPS, PDLB has its own features. Firstly, the granularity of load balancing is different. PDLB is a flowcell-based load balancing method,

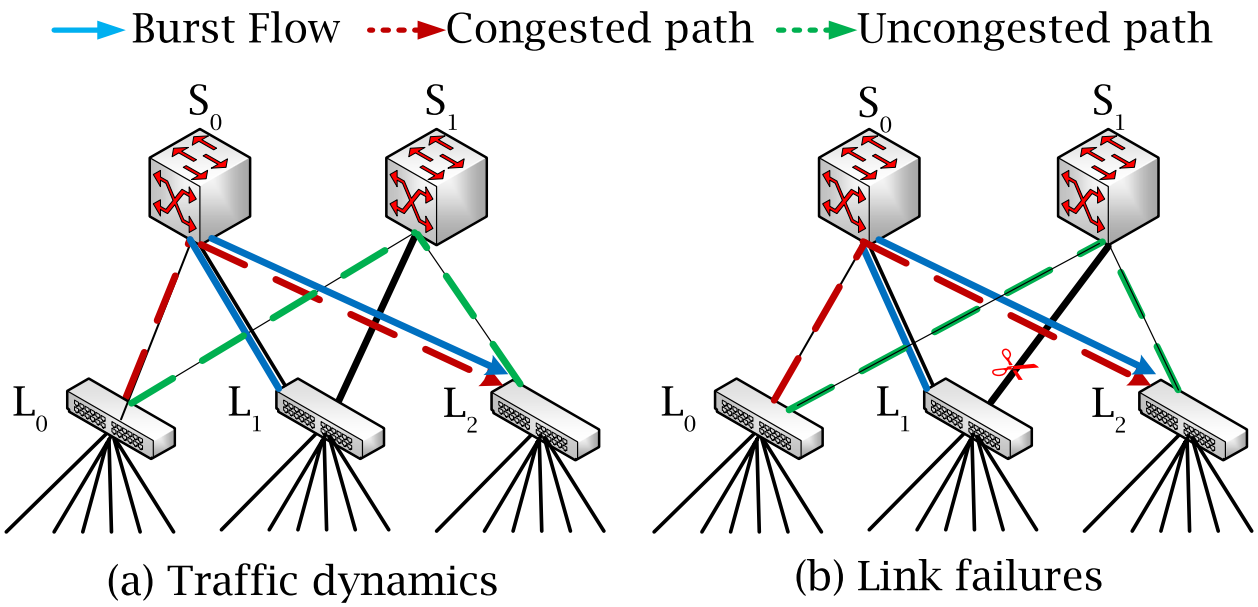


Fig. 1 Topology Asymmetry

while QDAPS is a packet-level load balancing method. Secondly, PDLB adaptively adjusts flowcell granularity according to path diversity. PDLB increases flowcell granularity to alleviate packet reordering under large degrees of topology asymmetry, while reducing flowcell granularity to obtain high link utilization under small degrees of topology asymmetry. But QDAPS only proposes a flow rerouting method to avoid packet reordering at the switch.

AAC [14] measures the leaf-to-leaf delay at a leaf switch to adjust the flow concurrency according to the degree of path asymmetry. Both AAC and PDLB are load balancing schemes designed for data center network. However, AAC and PDLB are essentially different works. Firstly, AAC is a load balancing scheme with packet-based granularity, while the switching granularity of PDLB is flowcell, which is adjusted according to the path asymmetry. Second, AAC performs routing selection on the switch by controlling the flow concurrency, while PDLB does not perform routing selection at switch and only adjusts the size of the flowcell at the sender. Finally, PDLB works at the sender side, without any deployment overhead at switch. However, AAC quickly senses network congestion at switches and adjusts the sending paths, incurring some deployment overhead.

Background and motivation

In this section, we provide empirical research to show that, the path asymmetry is very common in the modern data centers. Then, we analyze the impact of path asymmetry on load balancing performance and demonstrate that controlling the size of flowcells is effective in reducing latency under large path asymmetry.

Path asymmetry in production data center

Data center network traffic is known for its bursty traffic [15, 16]. When the traffic bursts instantaneously, it is easy to cause congestion, resulting in path asymmetry. In addition, due to link failures and the heterogeneity of network equipment, path asymmetry generally exists in data centers [5, 6, 17–20]. The main difference between symmetric topology and asymmetric topology is whether the delay and bandwidth of multiple paths between any pair of communication hosts are consistent. If the delay or bandwidth of paths between any pair of hosts is the same, it is a symmetric topology, otherwise it is an asymmetric topology. In the following, we use an example to show the wide existence of path asymmetry.

As shown in Fig. 1a, a bursty flow is transmitted by ECMP from leaf switch L_1 to leaf switch L_2. Since the link between spine switch S_0 and leaf switch L_2 is blocked by the burst flow, the two paths from L_0 to L_2 become asymmetric. Figure 1b shows the asymmetric topology caused by link failures. When the link between L_1 and S_1 breaks down, any traffic from L_1 to L_2 is forced to be transferred on the path L_1- S_1- L_2. Consequently, traffic from leaf switch L_1 to L_2 will go through two paths with different latencies. Besides, switch failures such as random packet dropping and heterogeneous switches with different link speeds can also induce topology asymmetry [5].

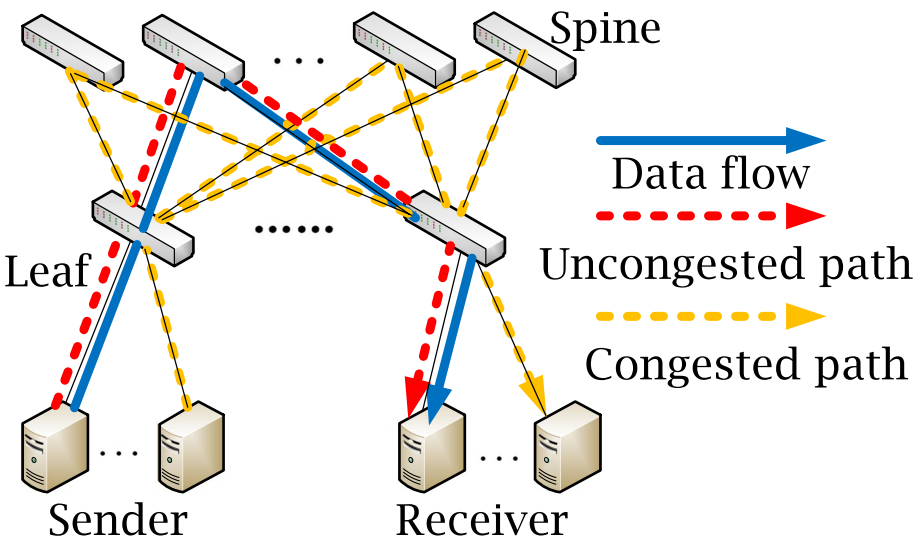
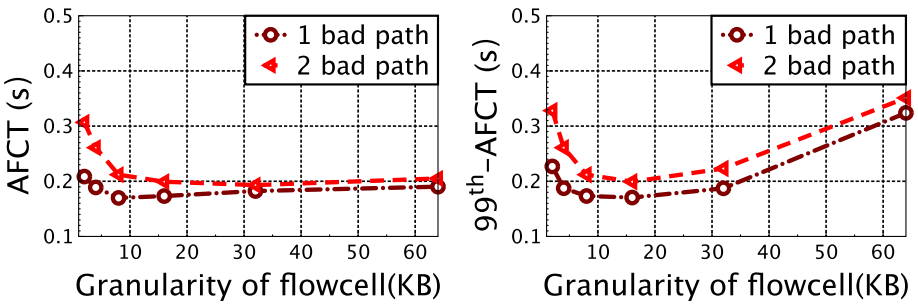


Fig. 2 Leaf-spine topology



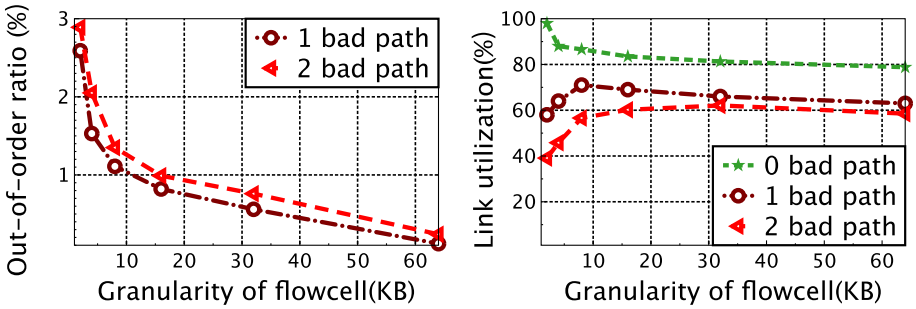
(a) AFCT (b) 99^{th} percentile AFCT

Fig. 3 FCT under different granularity of flowcell

The granularity of flowcell affects the performance of load balancing

In order to explore the impact of path asymmetry on load balancing performances, we use the NS2 simulation to test the performance of Presto, which is the typical datacenter load balancing design already implemented on the commodity switches [8, 21]. The test topology is a leaf-spine topology [11] with 8 spine switches and 2 leaf switches in Fig. 2. The bandwidth of each path and the buffer size of each switch are 1Gbps and 250 packets, respectively. Each sender sends a DCTCP flow to a single receiver via leaf switches with the RPS scheme, which randomly spreads the arriving packets to all 8 paths. To produce the path asymmetry, we change the round trip propagation delay of each path. We generate 10 flows with 10000 packets. The RTT of the uncongested path is set to 100μs, and the RTT of the congested path is set to 300μs and 600μs, respectively.

1) FCT under different granularity of flowcell: Figure 3 shows the average and 99^{th} percentile flow completion time with different granularity of flowcell. The RTTs of congested path is different. Figure 3a shows that average flow complete time(AFCT) firstly declines and then arises with the increasing size of flowcells. In Fig. 3b, the 99^{th} percentile FCT shows the similar trend. This result shows that, smaller flowcells provide higher link utilization to reduce flow completion time under smaller path asymmetry. However, under large path asymmetry, lager flowcells easily increase the tailed delay and reduce link utilization, resulting in large AFCT and 99^{th} percentile FCT.
2) Packet reordering and link utilization under different granularity of flowcell: We further investigate the reasons of above results. Figure 4a shows the ratio of 3-dupack events caused by out-of-order packets to all packets. As the flowcell is cut smaller, more data



(a) Packet reordering (b) Link utilization

Fig. 4 Performance under different granularity of flowcell

packets are scattered on the bad path, and the out-of-order rate of the data packet is higher. At the same time, increasing the number of bad paths will cause the same trend. Figure 4b shows the link utilization rate under symmetric and asymmetric paths. Under the symmetric path, the link utilization rate increases as the flowcell cutting granularity decreases; under the asymmetric path, the link utilization rate will show a convex curve as the flowcell cutting granularity increases.

Summary

Based on the above analysis, we draw the following conclusions that (i) smaller flowcells provide higher link utilization, but out-of-order is more likely to occur in path asymmetry, (ii) lager flowcells easily decrease out-of-order events are reduce the link utilizations. These conclusions motivate us to design a novel load balancing scheme that adjusts the size of flowcells to achieve a good tradeoff between the packet reordering and link utilization. In the following part, we design an adaptive granularity load balancing scheme, PDLB, to improve network performance under dynamic network conditions.

Adaptively adjusting flowcell granularity

In this section, we first describe the architecture of PDLB, presenting several challenges that need to be addressed. Then, we present the details how to estimate path asymmetry at the end-host. Moreover, we theoretically analyze how to obtain the granularity of the flowcell according to the network congestion state. Finally, we evaluate the accuracy of the model.

Design overview

Our goal is to design a load balancing mechanism that adjusts flowcell size based on the latency difference among multiple paths to achieve the tradeoff between packet reordering and link utilization. In Fig. 5, we plot the architecture of PDLB, which includes the path asymmetry estimation module and the flowcell size adjustment module. Firstly, PDLB implements the RTT measurement of the path at the sending end, which periodically sends the detection packet with the time stamp option to obtain the RTT sample information of each link in the network, and then the sampling information can be processed to obtain the real-time congestion states. The paths are divided into congested paths and uncongested ones. With the path states, PDLB makes flowcell-level forwarding decisions for each arrival packet. Specifically, PDLB calculates the

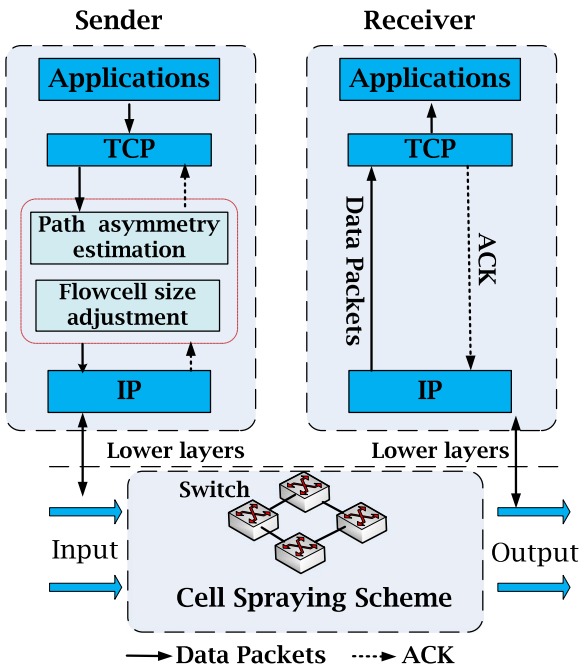


Fig. 5 The architecture of PDLB

optimal size of flowcells *gran* based on the path diversity. Then PDLB spreads flowcells on all paths to balance the out-of-order rate and link utilization.

PDLB design needs to solve several key challenges. Firstly, PDLB needs to periodically collect path delays with limited overhead to distinguish congested and non-congested paths. Secondly, the size of flowcell adjustment should quickly respond to the dynamic changes of the network. Finally, PDLB should be compatible with existing transport layer protocols and be easy to deploy in large-scale network.

Path asymmetry estimation

Load balancing schemes that require path congestion information, naturally, are much more complex [22]. Modern data center networks are usually organized in multi-rooted tree topologies, in which the load balancing schemes split traffic across multiple paths between the source leaf switch and the destination one. To obtain the path congestion state, PDLB should measure the end-to-end delay of the transmission path.

The path asymmetry can be estimated in advance at the sender based on the measurement of RTT [23]. Since PDLB mainly adjusts the granularity of flowcell adaptively according to the difference of the path, in order to realize the purpose of transferring the traffic from the congested path to the uncongested path. But the sender is unable to directly obtain the exact RTT for each path. Here, we utilize the TCP congestion control mechanism [15]. Specifically, when the sender receives the ACK packets, based on the corresponding RTT of each ACK, the bad path probability is calculated as the ratio of the number of ACK packets with large RTT to the total number of received ACK packets. For example, assuming that 10 flowcells are sent at the sender, the number of flowcells should generally be consistent with the number of paths. Since the switch uses polling scattering to route the flowcells, the sender calculates the probability of the bad path according to the path delay. If the transmission times of two flowcells are obviously longer than that of other 8 flowcells, it is possible that these two flowcells are taking the bad path. Then, the probability of a bad path P_b=0.2. To limit the computing and memory overhead, the sender measures the path delay every 100μs.

PDLB brings about limited overhead, since it only measures the one-way-delay by using source and destination hosts. Moreover, to reduce overhead and enhance scalability, PDLB periodically uses a few data and ACK packets to carry the delay information in the option field of TCP header. Thus, the path delay is collected with very small traffic overhead and deployment overhead.

Flowcell size adjustment

The flowcell size affects both the TCP reordering probability and network utilization under different asymmetric degrees. We give the analysis on how to get the optimal value of flowcell size as follows.

Let F_{size} and *gran* denote the size of a TCP flow and flowcell granularity, respectively. Because the smallest granularity is a packet, and the largest granularity is 64KB (44 packets) [8, 18], the value of *gran* ranges from 1 to 44. Then the flow is cut into $\frac{F_{size}}{gran}$ flowcells according to *gran* granularity.

When the flowcell are transferred on multiple paths, a flowcell is out-of-order only when at least one flowcell sent later arrives at the destination earlier. We assume that the n flowcells may select m parallel paths, which consist of N_b congested paths and N_g uncongested paths with the propagation delay D_b and D_g, respectively. Assuming that the ratio of the number of bad paths to the number of good paths is R, then R is equal to $\frac{N_b}{N_g}$. Let X denote the ratio of bad paths to good paths, then X is equal to $\frac{D_b}{D_g}$.

Here, we classify the path types according to the path delay with the following considerations. Firstly, since RTT asymmetry is caused by dynamic traffic and hardware failures such as frame checksum errors and high CPU utilization, the delay suddenly jumps up when an incident occurs [24]. Since the uncongested paths between host pair have the same numbers of hops, they have similar RTT [25]. Secondly, in the model analysis, references [18, 23, 26] divide the paths into congested paths and non-congested paths. For the convenience of modeling, we also divide the paths into two types. Thirdly, the transmission of the flowcell should be considered, because the flowcell is composed of multiple packets which are transmitted on each path. Since the size of the flowcell is not large, the variance of experienced delay is also not large. Moreover, to track the real-time delay information, the sender will update the path delay once receiving the ACK packets.

Therefore, for simplicity, we classify the paths into congested and uncongested ones according to their delay. The congested paths are defined as the ones with a latency larger than 2x the average RTT of all paths. If P_g and P_b are the probabilities that the flowcell selects the uncongested and congested paths, respectively. Specifically, P_g and P_b may be various under different load balancing schemes. In our design, in order to avoid synchronization effect, each flowcell is randomly assigned to one of the available paths. Thus, the probability P_b that a congested path is selected is calculated as $\frac{N_b}{m}$.

Then, we get the probability P_g that an uncongested path is selected as

$$P_g = 1 - \frac{N_b}{m}. \tag{1}$$

At the receiving host, we can use the monotonically increasing sequence number method to determine whether a out of order occurs. That is, the sequence number of the received data packet is monotonically increasing, then the data packet arrives in order, otherwise, a out of order occurs.

The out-of-order event occurs when one packet train is transmitted on congested path and at least one flowcell sent later is transmitted on uncongested path. Therefore, the reordering probability P of n flowcells is calculated as

$$\begin{aligned} P &= 1 - (P_b^n + P_b^{n-1} \times P_g + P_b^{n-2} \times P_g^2 + ... \\ &\quad + P_b^{n-k} \times P_g^k + P_b \times P_g^{n-1} + P_g^n) \\ &= 1 - \sum_{k=0}^{n} P_b^{n-k} \times P_g^k. \end{aligned} \tag{2}$$

Substitute P_b and P_g into Eq. (2), we get the reordering probability P as

$$P = 1 - \sum_{k=0}^{n} (1 - \frac{N_b}{m})^k \times (\frac{N_b}{m})^{n-k}. \tag{3}$$

Assume that the largest window in each round of transmission on a path is $MaxW$, and the initial value of the network congestion window is W_0. When detecting the out-of-order packet, the TCP sender reduces its congestion window by half. Thus, in each round of data transmission, if n_i represents the number of flowcells in i-th round, the i-th round window W_i and the out-of-order ratio of i-th round P_i are

$$\begin{cases} n_1 = f(W_0) = min(m, \frac{W_0}{gran}) \\ P_1 = g(n_1) = 1 - \sum_{k=0}^{n_1} P_b^{n_1-k} \times P_g^k \\ W_1 = \delta(P_1) \\ = min(n_1 \times MaxW, (W_0 + 1) \times (1 - P_1) + \frac{W_0}{2} \times P_1) \end{cases}$$

...

$$\begin{cases} n_i = f(W_i) = min(m, \frac{W_{i-1}}{gran}) \\ P_i = g(n_i) = 1 - \sum_{i=0}^{n_i} P_b^{n_i-i} \times P_g^i \\ W_i = \delta(P_i) \\ = min(n_i \times MaxW, (W_{i-1} + 1) \times (1 - P_i) + \frac{W_{i-1}}{2} \times P_i) \end{cases}$$

In the slow start phase of the protocol, the congestion window increases exponentially. After the sending rate reaching the link capacity, the slow start phase is switched to the congestion avoidance phase. Then we assume that at the switch point (i.e., i=0), the total maximum window is $m \times MaxW$ for m paths. Therefore, the congestion window for each round W_i is given by:

$$W_i = \begin{cases} m \times MaxW & i = 0; \\ min(n_i \times MaxW, (W_{i-1} + 1) \times (1 - P_i) \\ + \frac{W_{i-1}}{2} \times P_i) & i\> 0. \end{cases} \tag{4}$$

Let r and W_s represent the number of rounds required to transmit n flowcells and the sum of congested windows respectively, then W_s can be expressed as

$$W_s = \sum_{i=0}^{r} W_i. \tag{5}$$

When W_s is firstly greater than or equal to F_{size}, the subscript of W_i corresponds to the number of transmission rounds r, then we get the average congestion window $\overline{W}$ as

$$\overline{W} = \frac{W_s}{r}. \tag{6}$$

Though the small size of flowcell leads to large packet reordering probability, the small flowcell could utilize more paths, increasing the total utilized bandwidth. Given the link bandwidth C for each path, since each flowcell randomly picks its transmission path, the total bandwidth for n flowcell is $n \times C$.

Typically, the end-to-end latency mainly consists of the queueing and propagation delay. Given the average congestion window $\overline{W}$, we obtain the average end-to-end round-trip time $\overline{RTT}$ as

$$\overline{RTT} = (1 - (\frac{N_b}{m})^{\overline{W}}) \times D_g + (\frac{N_b}{m})^{\overline{W}} \times D_b + \frac{\overline{W}}{n \times C}. \tag{7}$$

Substitute D_b and $\overline{W}$ into Eq. (7), we can get the $\overline{RTT}$ as

$$\overline{RTT} = ((X - 1) \times (\frac{N_b}{m})^{\overline{W}} + 1) \times D_g + \frac{W_s}{r \times n \times C}. \tag{8}$$

After introducing the end-to-end delay of each round of transmission, we only need to calculate the number of rounds of transmission, and use the product of both the end-to-end delay of each round and the number of rounds to deduce the flow completion time. So FCT can be expressed as

$$\begin{aligned} FCT &= \frac{F_{size}}{\overline{W}} \times \overline{RTT} \\ &= \frac{F_{size}}{\overline{W}} \times ((X - 1) \times (\frac{N_b}{m})^{\overline{W}} + 1) \times D_g + \frac{F_{size}}{n \times C} \\ &= \frac{F_{size}}{\overline{W}} \times ((X - 1) \times (\frac{N_b}{m})^{\overline{W}} + 1) \times D_g + \frac{gran}{C} \end{aligned} \tag{9}$$

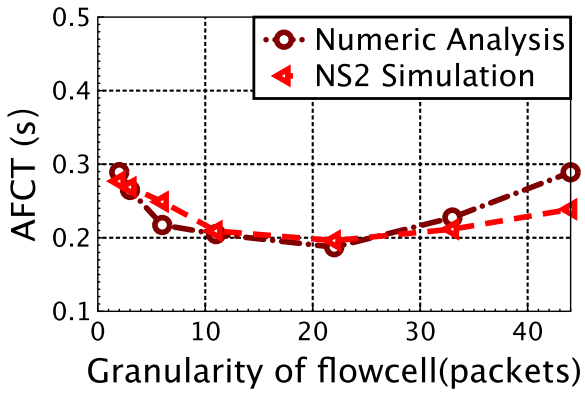


Fig. 6 Model verification

wherein F_{size}, C, N_b, m, D_g, X are constant coefficients. There is a quadratic function relationship between the variable $\overline{W}$ and the independent variable *gran*. Finally, according to Eq. 9 we can get the optimal granularity *gran* to obtain the minimum value of average flow completion time FCT as

$$gran = \arg\min_{gran_{i\in}[1,44]} \left\|\mathrm{FCT}(\mathrm{gran_i})\right\|. \tag{10}$$

Model verification

We evaluate the correctness of the model by NS2 simulations. We use DCTCP [27] as the underlying transport protocol, and the experimental topology shown is Fig. 2. Besides, the flow size, the delay of two bad paths(D_b), and the delay of good path (D_g) are 10000 packets, 300μs, 600μs and 100μs, respectively. Other parameters are the same as the experimental scenario in Sect. III-B.

When the number of flowcells increases from 1 to 44, we calculate the theoretical completion time FCT for ten flows with a size of 100000 packets. The value of numeric analysis is consistent with the varying trend in the NS2 simulation test (Fig. 6).

The Algorithm of PDLB

Based on the above theory and verification experiment analysis, we can get the optimal granularity of flowcell. However, in the production data center network, with thousands of servers, complex topology and frequent burst traffic, it is obviously unreasonable to use a fixed flowcell granularity for transmission. Therefore, we have to adopt certain adjustment strategies as shown in Algorithm 1 to quickly respond to network changes. PDLB algorithm consists of path asymmetry estimation module and flowcell size adjustment module.

Input: m /*The number of paths in DCN*/
Output: $gran$ /*The granularity of flowcell will be selected*/

Initialization;
$p_i.mode \leftarrow \emptyset$; $path[] \leftarrow allpaths$;
$gran \leftarrow 0$; $fct_{min} \leftarrow +\infty$;
/* path asymmetry estimation module */ ;
if *The timer is timeout* **then**
  $D_b = +\infty; D_g = 0; Sum = 0;$
  for *each path* $p_i \epsilon path[]$ **do**
    $D_b = MIN(D_b, p_i.RTT)$;
    $D_g = MAX(D_g, p_i.RTT)$;
    $Sum+ = p_i.RTT$;
  $D_{avg} = \frac{Sum}{m}$;
  $X = \frac{D_b}{D_g}$;
  for *each path* $p_i \epsilon path[]$ **do**
    if $p_i.RTT > 2 \times D_{avg}$ **then**
      $p_i.mode = congested$;
    else
      $p_i.mode = uncongested$;
/* Flowcell size adjustment module */ ;
if *the path state changes* **then**
  while $i \geq 1$ *and* $i \leq 44$ **do**
    $fct = f(gran_i)$;
    if $fct < fct_{min}$ **then**
      $fct_{min} = fct$;
      $gran = i$;
    $i = i + 1$;
return $gran$;

Algorithm 1 Pseudo-code of PDLB*Path asymmetry estimation module:* The path congestion estimation of PDLB at the sender side periodically (e.g.100μs) sends probe flowcells to measure the congestion state. When the timer expires after the timeout *T*, the PDLB first estimates the delay of all paths between the sender and the destination host. Then the PDLB updates the number of congested paths with a delay greater than 2*X* the average RTT [15, 28]. According to the path delay information, PDLB calculates the difference of path asymmetry.

Flowcell size adjustment module: When the path state changes, PDLB meausres the difference of paths and calculates the optimal flowcell granularity.

To find the optimal flowcell granularity corresponding to the minimum flow completion time, we check all possible value of flowcell granularity one by one. The time complexity is O(*gran*), where *gran* is the maximum value of flowcell granularity. Since the flowcell granularity is

Table 3 Comparing the time complexity of different algorithms

Schemes	Time complexity
RPS	$O(n)$
LetFlow	$O(n^2)$
Presto	$O(n)$
ECMP	O(1)
QDAPS	$O(n)$

limited by the maximum receiving window, the computational overhead is still small. We enforce the optimal granularity to all active flows in network. When the number of arriving packets from a given flow is less than the optimal granularity, the packets follow the path of the same cell by using the same five-tuple of packet header.

Simulation evaluation

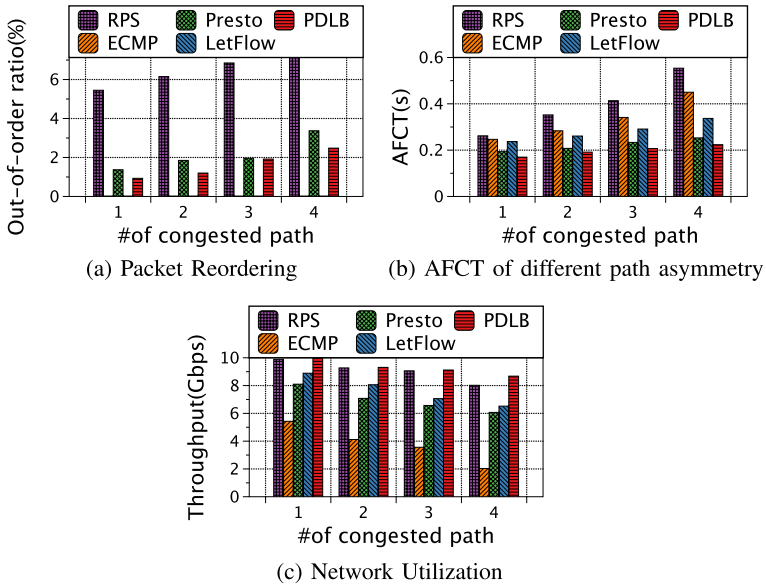
In this section, we conduct the NS2 simulation tests to evaluate the performance of PDLB. We firstly test the basic performance of PDLB by adjusting the number of congested paths and the asymmetry of the path respectively, and then compare PDLB with the state-of-the-art schemes under datacenter workloads in a large-scale test. We use FCT, link utilization and out-of-order rate as the primary performance metrics.

Basic performance

In this section, we test the basic performance of PDLB. We compare the packet reordering, link utilization and flow completion time of RPS, ECMP, Letflow and Presto[1]. The comparison of time complexity is shown in Table 3, where n represents the number of paths from the sending end to the receiving end.

1) Adjusting the number of congested paths The experimental topology is shown in Fig. 2. There are 8 available paths from the one leaf switch to another leaf switch. The downlink and uplink bandwidth of leaf switch are 10 Gbps. At the edge layer of the data center network, the arrival process of packets is represented by an ON/OFF model, the arrival time interval of packets between the OFF period obeys a normal distribution [1, 29]. In addition, to obtain fine-grained characteristics (such as packet size distribution, arrival time, etc.), the length of data flow in the data center network conforms to a heavy-tailed distribution [30]. That is 99% of flows are less than 100 MB in size, while 1% of long flows exceed 90% of data traffic. We generate 200 flows from 50KB to 200KB in heavy-tailed distribution and the start time of these flows follows the Poisson distribution. The RTT of good paths are 100μs. The hotspot path with a latency of 300μs, then we increase the number of hot paths from 1 to 4, we compare the performances of PDLB and others methods. In this section, the parameter values are taken from reference work [1, 10, 18, 19]. We only randomly generated some small-scale data for testing, trying to compare the performance of PDLB with other load balancing schemes. From a macro perspective, the heterogeneous traffic exhibits the stable heavy-tailed distribution in data centers [27, 29]. From a micro viewpoint, however, datacenter traffic is very bursty and unpredictable at short timescales (e.g., 10-100s of microseconds) [1]. PDLB detects traffic changes by periodically updating the number of flows. But for the sake of simplicity, we used only one traffic mode in the test, and did not test the performances of PDLB under varying traffic patterns. Figure 7 shows the basic performance with different load balancing mechanisms. Figure 7a shows that out-of-order ratio of RPS, ECMP, Letflow, Presto and PDLB under path asymmetry. Since RPS forwards packets of each flow to all paths in a packet-level spraying manner, with the increasing number of bad paths, the packets scattered on the bad paths will experience a long transmission delay, so there will be a large number of out-of-order packets. ECMP and LetFlow are coarse-grained scheduling schemes, which can completely avoid packet reordering, while fail to fully utilize all parallel paths. Figure 7b and c show the similar trend. Smaller flowcells provide higher link utilization to reduce flow completion time under small path asymmetry. However, smaller flowcells easily increase the tailed delay under large path asymmetry. As RPS experiences more disorder, its average flow completion time is the worst. ECMP's performance is also relatively poor, because ECMP has a higher probability of hashing flows to the same hot path. LetFlow can effectively avoid packet reordering, while fail to fully utilize all parallel paths. PDLB still outperforms Presto. The reason is that PDLB uses path asymmetry-aware switch granularity adjustment. Therefore, PDLB always maintains the highest throughput and the smallest FCT compared to the other schemes.
2) Adjusting the asymmetry degree of the path The experimental topology used in the same as above. We randomly choose 4 parallel paths as the bad paths, while the remaining 4 paths are the good paths. Because there are eight paths in the test topology, the number of good and bad paths is set to be equal for the convenience of calculation and testing. We

[1] Code and trace here: https://github.com/gwm2022/PDLB-test



(a) Packet Reordering (b) AFCT of different path asymmetry

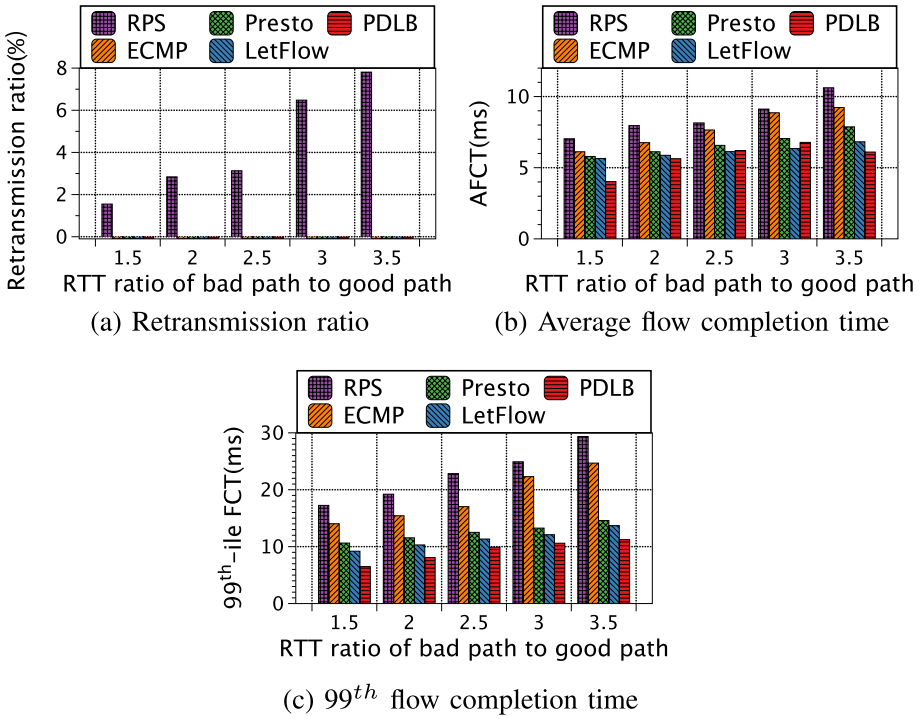
(c) Network Utilization

Fig. 7 The basic performance with different load balancing mechanisms

refer to BurstBalancer [9] and QDAPS [13] for this part of experimental parameter setting. According to typical link bandwidth in data center, we set the link bandwidth to 10Gbps, and the round trip propagation delay of good paths is 200μs. For the bad paths, we gradually increase the propagation link delay to enlarge the degree of topology asymmetry. Therefore, the ratio of bad paths' RTT to good paths' RTT varies from 1.5 to 3.5. In our tests, the senders generate 100 DCTCP flows based on the data mining workload according to the Poisson process. The threshold of flowlet used in LetFlow is set as 500μs [5, 9]. We evaluate the performances of various schemes in terms of the average flow completion times (AFCTs) of short flows (100KB), the total throughput of long flows ($\geq$ 100KB) [22, 31], the ratio of retransmission packets caused by packet reordering. Figure 8a compares the retransmission ratio of short flows under different schemes. The rise of asymmetric degree causes RPS to experience increasingly serious packet reordering, generating a much higher retransmission ratio compared to the other schemes. On the contrary, PDLB can effectively control packet reordering, and obtain high link utilization, thus achieving the shortest average and 99th percentile flow completion times across all cases, as shown in Fig. 8b and c. The performance of long flows is shown in Fig. 9. RPS can distribute load in the most balanced way, but possess a high retransmission ratio. Presto and PDLB perform load balancing based on flowcell granularity. In asymmetric topology, a small part of long flows may be out of order and cause retransmission as shown in Fig. 9a. In Fig. 9b, ECMP and LetFlow can completely avoid packet reordering, while failing to fully utilize all parallel paths. Fortunately, PDLB can adjust the granularity adaptively according to path conditions, and effectively avoid packet reordering. Therefore, PDLB is not affected by the rise of asymmetric degree, and always maintains the highest total throughput for long flows compared to the other schemes.

3) Adaptive Granularity In this section, we compare the fixed granularity scheme and the adaptive granularity scheme in three aspects: the out-of-order rate of the data packet, the link utilization rate and whether the granularity can be adjusted adaptively. We use the Leaf-Spine topology with 4 paths. The round trip propagation delay is 200μs and the link bandwidth is 10Gbps. The buffer size of each switch is 100 packets. We generate 2 TCP flows with a size of 30MB. There is a 500Mbps UDP background flow on each path to induce congestion. Meanwhile, to produce topology asymmetry, we increase the sending rate of UDP flow to 800Mbps on one randomly selected path from 100*ms* to 200*ms*. We set the sample interval as 500μs.

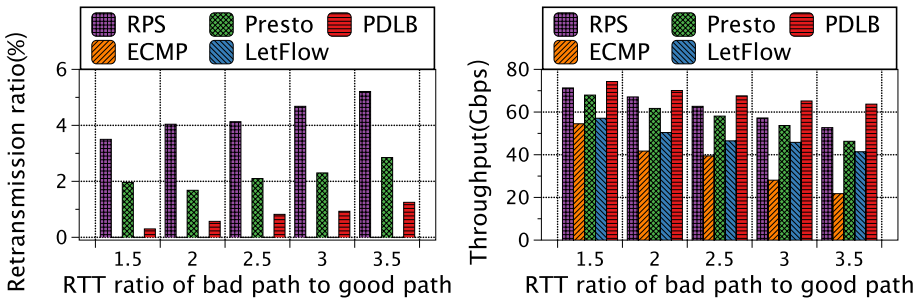
Figure 10a shows the flowcell granularity of PDLB. When the multiple paths are symmetric before 100*ms* or after 200*ms*, PDLB sends flowcells with smaller granularity to improve link utilization. Although the asymmetric degree of multiple paths becomes large, PDLB increases the granularity of flowcell to alleviate packet reordering.



(a) Retransmission ratio (b) Average flow completion time

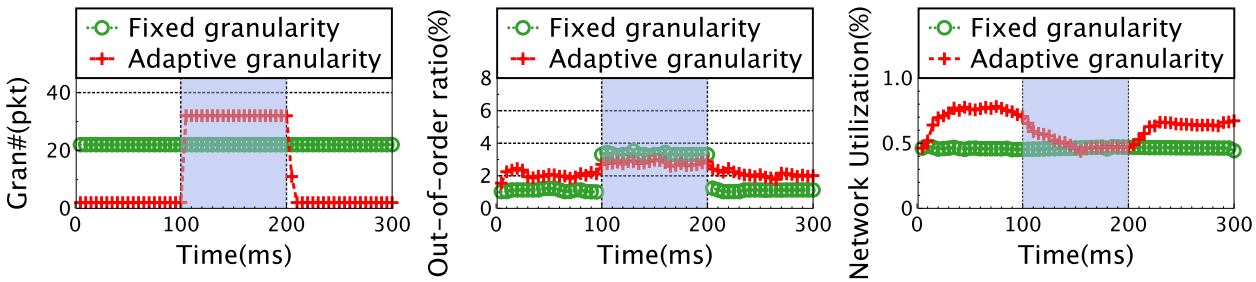
(c) 99^{th} flow completion time

Fig. 8 The performances of short flows



(a) Retransmission ratio (b) Total throughput

Fig. 9 The performances of long flows



(a) Adaptively Granularity (b) Packet Reordering (c) Network Utilization

Fig. 10 Comparison of fixed granularity and adaptive granularity schemes

Table 4 Parameter settings

Parameter	Setting values
D_g: The delay of uncongested path	100 μs
D_b: The delay of congested path	300 μs
m: The number of paths	8
C: Link bandwidth	10 Gbps
RTO: Retransmission Timeout	10 ms
Sample Interval	500 μs
Buffer Size	200 packets

Figure 10b shows the ratio of 3-dupack events caused by out-of-order packets in all packets. Since the asymmetry of the path occurs in [100, 200]ms, PDLB experiences a little more packet reordering due to its random scheduling pattern. Fortunately, it can estimate the path asymmetric, and adjust the granularity of flowcell adaptively. Therefore, the packet reordering ratio of PDLB is still acceptable.

Figure 10c shows the overall network utilization. The network utilization of fixed granularity schemes is less than 50%, because 2 flows respectively almost fully utilize 2 paths at a time, while the other 2 paths are unused. PDLB achieves high network utilization, because it adaptively adjusts the granularity of flowcell based on path asymmetry to make a trade-off between the packet reordering and link utilization.

Large-scale simulation test

In this part, we select topology parameter values that are taken from the references AG [25, 32] and TR [27], and the detailed settings are shown in Table 4 [33, 34]. We construct a large-scale leaf-spine network in which 256 hosts are connected via 8 leaf switches, 8 spine switches, and 10Gbps links. The switch buffer size is set to 250KB. There are 8 equal cost paths with the propagation delay of 100μs between any pair of hosts. In order to generate path asymmetry, we randomly select one of the paths and set its round-trip propagation delay to 300μs.

Besides, to make a comprehensive comparison, we use two representative data center workloads such as web search and data mining [35, 36]. Specifically, in the web search workload, over 95% of the bytes are from 30% of flows larger than 1MB. In the data mining workload, 95% of all bytes are from 3.6% flows that are larger than 35MB, while more than 80% of flows are less than 10KB. The average flow sizes are 1.6MB and 7.4MB in the web search and data mining applications, respectively. Overall, across two workloads, the ratios of short flows are always higher than those of long flows, following the heavy-tail distribution of data center traffic [17, 23, 27]. Flows are generated between random pairs of hosts following a Poisson process with load varying from 0.1 to 0.8, We evaluate the performance of PDLB by comparing with the state-of-the-art load balancing schemes, such as RPS [7], LetFlow [5], ECMP [4], Presto [8] and QDAPS [13].

The performance test results of web search and data mining applications are shown in Figs. 11 and 12, respectively. Figures 11a and 12a show the AFCT of short flows, which increases with a larger traffic load. We observe that RPS performs poorly for small flows because of it scatters the data flow to all paths at packet granularity. In the case of bad paths, the long flows will block the short flows, thus affecting the completion time of the short flows. For ECMP, because it forwards packets to paths in a flow hashing method, data flows are prone to collisions, so the AFCT of short flows is only better than RPS. Presto splits each flow into a fixed granularity (i.e. 64KB). Though mitigating the issues of low link utilization and packet disordering to a certain extent, the fixed granularity of Presto is not adaptable to all asymmetric degrees. Under LetFlow, the switch automatically senses path congestion by using the elasticity of the flowlet size. LetFlow sets a time interval threshold (i.e. 500μs) for packet clusters, resulting in no out-of-order and smaller AFCT for short flows. QDAPS chooses the path for the packet with the least queuing delay based on the output buffer. Due to the adaptive switching granularity, PDLB reduces the average FCT of short flows by up to 43%.

Figures 11b and 12b show the 99^{th} percentile FCT of short flows under different traffic loads. Compared with the other four schemes, PDLB significantly reduces the tail FCT of around 8%-59% and 20%-73% under web search and data mining workload, respectively. Note that under data mining workload, since the flow sizes of 80% flows are less than 100KB [3], the average FCT of short flows is lower than that of web search workload. Meanwhile, as the long flow size in data mining is larger, the short flows' 99^{th} FCT are higher than that of web search workload.

Figures 11c and 12c show the AFCT of long flows under different traffic loads. RPS experiences packet reordering under highly asymmetric topology. Therefore, RPS displays the poor performance because of the packet reordering problem. ECMP may cause hash collisions due to random routing, so that the AFCT of long flows is only better than the RPS in asymmetric topology. Since QDAPS selects a suitable output queue to ensure that the packets arrive at the receiver in order and avoid the low link utilization in a flexible manner, it has obtained sub-optimal performance. Meanwhile, Presto sends flowcells at a fixed granularity, under asymmetric topology, thereby obtaining worse

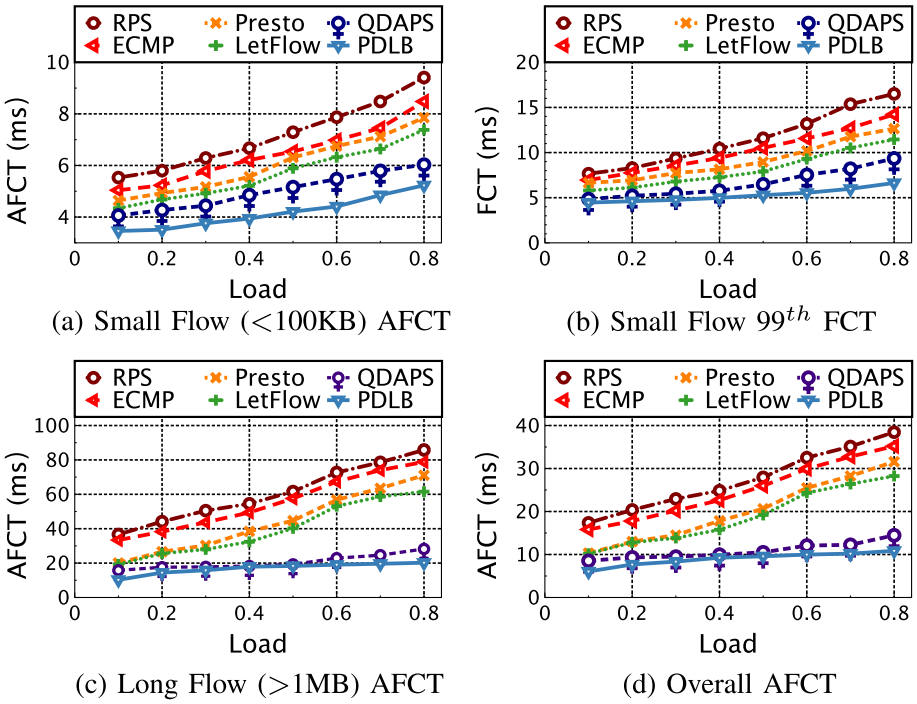


Fig. 11 FCT for the web search workload in the asymmetric topology

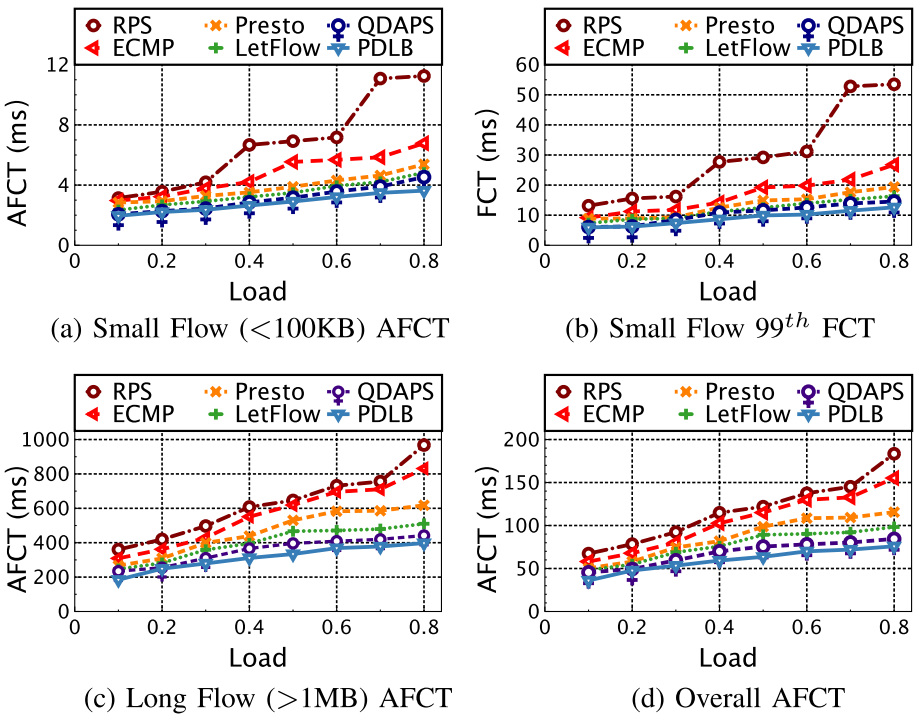


Fig. 12 FCT for the data mining workload in the asymmetric topology

performance than LetFlow. Compared with the other schemes, PDLB alleviates the impacts of large queueing delay and out-of-order problem by adaptively adjusting the flowcell granularity of long flows according to path congestion states. Therefore, PDLB reduces the AFCT of long flows by up to 8%-53% over other schemes.

Figure 11d and 12d show the AFCT of overall flows in web search and data mining workloads, respectively. Since RPS and Presto spread packets to all and some available paths, respectively, they experience many reordered packets and large FCT. ECMP and LetFlow easily degrade the link utilization due to their inflexibility. QDAPS flexibly selects the port with the shortest buffer queue for forwarding packets to avoid congestion, so its FCT is close to PDLB [37–39].

Conclusion and future work

In this paper, we proposed PDLB, a novel load balancing scheme that reduces FCT and simultaneously improves link utilization. Specifically, PDLB performs path asymmetry estimation by periodically sending detection flowcells at the sender, and then adaptively adjusts the granularity according to the path diversity. By using large-scale NS2 simulations, we demonstrate that PDLB performs remarkably better than the state-of-the-art load balancing designs under different realistic traffic workloads.

In the future, we will implement PDLB on the hardware programmable switches in a real testbed environment and conduct more experiments using NetBench to evaluate PDLB performance [24, 40].

Acknowledgements
Not applicable.

Authors' contributions
All authors have participated in conception and design, or analysis and interpretation of this paper.

Authors' information
Weimin Gao is currently a professor with the School of Computer Science and Engineering, Hunan Institute of Technology, China. His current research interest is data center networks.
Jiawei Huang(M'07) is currently a professor with the School of Computer Science and Engineering, Central South University. His research interests include performance modeling, analysis, and optimization for data center networks.
Zhaoyi Li is currently pursuing the Ph.D. degree in the School of Computer Science and Engineering at Central South University, China. His research interests are in the area of data center networks.
Shaojun Zou(Member, IEEE) is now an associate professor in the School of Computer Science and Technology, Hainan University, China. His current research interests include congestion control, load balancing and data center networks.
Jianxin Wang(SM'12) is currently a professor with the School of Computer Science and Engineering, Central South University. His current research interests include algorithm analysis and optimization, parameterized algorithm, bio-informatics, and computer network.

Funding
This work is supported by the National Natural Science Foundation of China (62132022, 61872387, 62102047), Key Research and Development Program of Hunan(2022WK2005), Natural Science Foundation of Hunan Province, China(2021JJ30867), Philosophy and Social Science Foundation of Hunan Province (21YBA224), Hengyang Science and Technology Innovation Project, China(202250045133).

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 18 August 2021 Accepted: 10 November 2023
Published online: 07 December 2023

References

1. Huang Q, Jin X, Lee PPC et al (2015) Sketchvisor: Robust network measurement for software packet processing. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 113–126
2. Bredel M, Bozakov Z, Barczyk A et al (2014) Flow-based load balancing in multipathed layer-2 networks using OpenFlow and multipath-TCP. In: Proc. HotSDN, New York, NY, USA, pp. 213–214
3. Greenberg A, Hamilton JR, Jain N, Kandula S, Kim C, Lahiri P, Maltz DA, Patel P, Sengupta S (2009) VL2: A scalable and flexible data center network. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 51–62
4. Hopps C (2000) Analysis of an Equal-Cost Multi-Path Algorithm. In: RFC 2992, IESG, ISOC
5. Vanini E, Pan R, Alizadeh M et al (2017) Let it flow: Resilient asymmetric load balancing with flowlet switching. In: Proc. USENIX Symposium on Networked Systems Design and Implementation, Berkeley, CA 94710 USA. pp. 407–420
6. Alizadeh M, Edsall T, Dharmapurikar S et al (2014) CONGA: Distributed congestion-aware load balancing for datacenters. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 503–514
7. Dixit A, Prakash P, Hu YC, Kompella RR (2013) On the Impact of Packet Spraying in Data Center Networks. In: Proc. IEEE INFOCOM, Turin, Italy. pp. 2130–2138
8. He K, Rozner E, Agarwal K, Felter W, Carter J, Akellay A (2015) Presto: Edge-based Load Balancing for Fast Datacenter Networks. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 466–478
9. Ghorbani S, Yang Z, Godfrey PB, Ganjali Y, Firoozshahian A (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 (SIGCOMM '17). Association for Computing Machinery, New York, NY, USA. pp. 225–238
10. Zhang H, Zhang J, Bai W et al (2017) Resilient datacenter load balancing in the wild. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 253–266
11. Katta N, Hira M, Kim C et al (2016) HULA: Scalable load balancing using programmable data planes. In: Proc. ACM SOSR, New York, NY, USA. pp. 1–12
12. Wang P, Trimponias G, Xu H, Geng YH (2019) Luopan: Sampling based load balancing in data center networks. IEEE Trans Parallel Distrib Syst 30(1):133–145
13. Huang J, Lyu W, Li W, Wang J, He T (2021) Mitigating Packet Reordering for Random Packet Spraying in Data Center Networks. IEEE/ACM Trans Netw 29(3):1183–1196
14. Gao W, Huang J, Zou S et al (2021) AAC: Adaptively Adjusting Concurrency by Exploiting Path Diversity in Datacenter Networks. J Netw Syst Manag 29(3):111–135

15. Mittal R, Lam VT, Dukkipati N et al (2015) TIMELY: RTT-based Congestion Control for the Datacenter. ACM SIGCOMM Comput Commun Rev 45(4):537–550
16. Kabbani A, Sharif M (2017) Flier: Flow-level congestion-aware routing for direct-connect data centers. In: Proc. IEEE INFOCOM, Atlanta, GA. pp. 1–9
17. Zou S, Huang J, Wang J et al (2019) Improving TCP Robustness over Asymmetry with Reordering Marking and Coding in Data Centers. In: Proc. IEEE ICDCS, Dallas, TX, USA. pp. 57–67
18. Liu J, Huang J, Li W et al (2019) AG: Adaptive Switching Granularity for Load Balancing with Asymmetric Topology in Data Center Network. In: Proc. IEEE ICNP, Chicago, IL, USA. pp. 1–11
19. Zhang T, Lei Y, Zhang Q et al (2021) Fine-grained Load Balancing with Traffic-aware Rerouting in DataCenter Networks. J Cloud Comput 10(37):1–20
20. Olmedilla et al (2020) Optimizing Packet Dropping by Efficient Congesting-Flow Isolation in Lossy Data-Center Networks, IEEE Symposium on High-Performance Interconnects (HOTI), Piscataway, NJ, USA, 2020, pp. 47–54. https://doi.org/10.1109/HOTI51249.2020.00022
21. Abbasloo S, Xu Y, Chao HJ (2020) To schedule or not to schedule: When no-scheduling can beat the best-known flow scheduling algorithm in datacenter networks. Comput Netw 172:107–177
22. Floyd S, Jacobson V (1993) Random early detection gateways for congestion avoidance. IEEE/ACM Trans Netw 1(4):397–413
23. Hu J, Huang J, Lv W et al (2019) CAPS: Coding-based adaptive packet spraying to reduce flow completion time in data center. IEEE/ACM Trans Netw 27(6):2338–2353
24. Guo C et al (2015) Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 139–152
25. Liu J, Huang J, Li W et al (2022) Asymmetry-aware Load Balancing with Adaptive Switching Granularity in Data Center. IEEE/ACM Trans Netw 30(5):2374–2387
26. Duan Y, Li C, Chao G et al (2015) Finding the shortest path in huge data traffic networks: A hybrid speed model. In: Proc. IEEE International Conference on Communications, London, UK. pp. 6906–6911
27. Alizadeh M, Greenberg A, Maltz D A et al (2010) Data center tcp (DCTCP). In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 63–74
28. Noormohammadpour M, Raghavendra CS (2017) Datacenter traffic control: Understanding techniques and tradeoffs. IEEE Commun Surv Tutor 20(2):1492–1525
29. Benson T, Akella A, Maltz DA (2010) Network traffic characteristics of data centers in the wild. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 267–280
30. Theophilus B, Ashok A, Aditya A et al (2010) Understanding data center traffic characteristics. ACM SIGCOMM Comput Commun Rev 40(1):92–99
31. Zou S, Huang J, Jiang W et al (2020) Achieving high utilization of flowlet-based load balancing in data center networks. Futur Gener Comput Syst 108:546–559
32. Huang J, Li W, Li Q et al (2020) Tuning high flow concurrency for MPTCP in data center networks. J Cloud Comput 9(1):1–15
33. Poutievski L, Singh A, Vahdat A (2014) Wcmp:weighted cost multipathing for improved fairness in data centers. In: Proceedings of the Ninth European Conference on Computer Systems, New York, NY, USA. pp. 1–14
34. Lee C, Park C, Jang K et al (2017) DX: Latency-Based Congestion Control for Datacenters. IEEE/ACM Trans Netw 25(1):335–348
35. Zhangy W, Lingy D, Zhangy Y et al (2020) Achieving optimal edge-based congestion-aware load balancing in data center networks. In: Proc. IEEE Networking Conference. pp. 109–117
36. Kumar G, Dukkipati N, Jang K, Wassel HMG, Wu X, Montazeri B, Wang Y, Springborn K, Alfeld C, Ryan M, Wetherall D, Vahdat A (2020) Swift: Delay is Simple and Effective for Congestion Control in the Datacenter. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 514–528
37. Mittal, Radhika et al (2015) TIMELY: RTT-based Congestion Control for the Datacenter. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 537–550
38. Diao X, Gu H, Yu X et al (2022) Flex: A flowlet-level load balancing based on load-adaptive timeout in DCN. Futur Gener Comput Syst 130:219–230
39. Hu J, Huang J, Lyu W et al (2021) Adjusting Switching Granularity of Load Balancing for Heterogeneous Datacenter Traffic. IEEE/ACM Trans Netw 29(5):2367–2384
40. Chen L, Chen K, Bai W, Alizadeh M (2016) Scheduling mix-flows in commodity datacenters with karuna. In: Proc. ACM SIGCOMM, New York, NY, USA. pp. 174–187

Publisher's Note