Introduction to Congestion Control in Juniper AI/ML Networks

Authors: Aninda Chatterjee, Vivek V

Documentation Feedback

We encourage you to provide feedback so that we can improve our documentation.
Send your comments to design-center-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

Table of Contents

Introduction

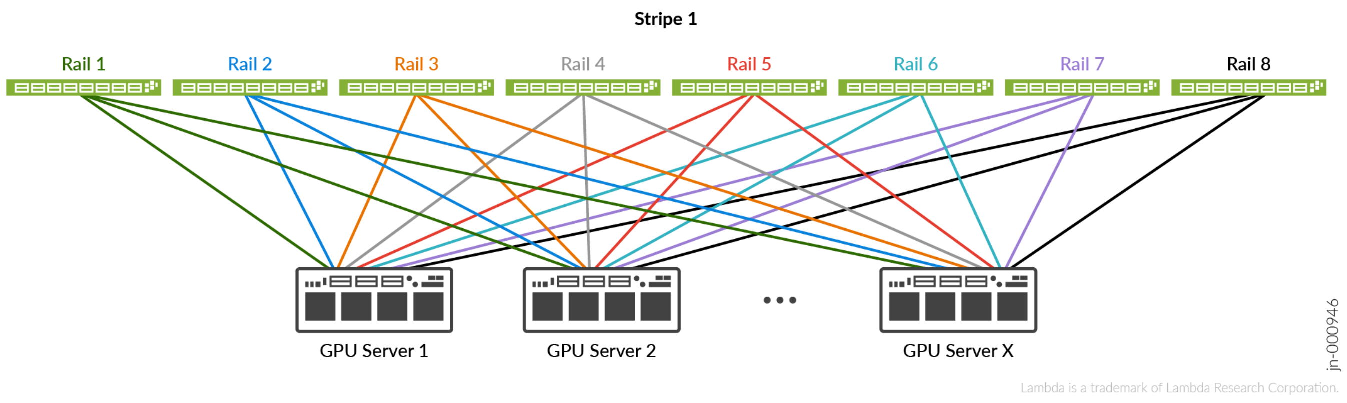
This document provides an introductory look at how to build a lossless fabric for Artificial Intelligence (AI)/ Machine Learning (ML) workloads using Dynamic Load-Balancing (DLB) and DCQCN (ECN and PFC) congestion control methods in Juniper AI/ML networks. This document uses the DLRM training model as a reference and demonstrates how different congestion parameters such as ECN and PFC counters, input drops and tail drops can be monitored to adjust configuration as needed to build a lossless fabric infrastructure for RoCEv2 traffic.

High-Level Design for Backend/Compute Network

In designing the network infrastructure for an AI/ML cluster, the key objectives are to provide maximum throughput, minimal latency, minimal network interference for AI/ML traffic flows, and a lossless fabric. Today, most AI clusters operate most efficiently over lossless networks for optimum training completion times. Packet loss can significantly impact training completion times.

A rail-optimized stripe, proposed by Nvidia, is a design that extends from the compute nodes to the leafs, which takes the network requirements that are necessary for AI/ML clusters into consideration, as shown in Figure 1.

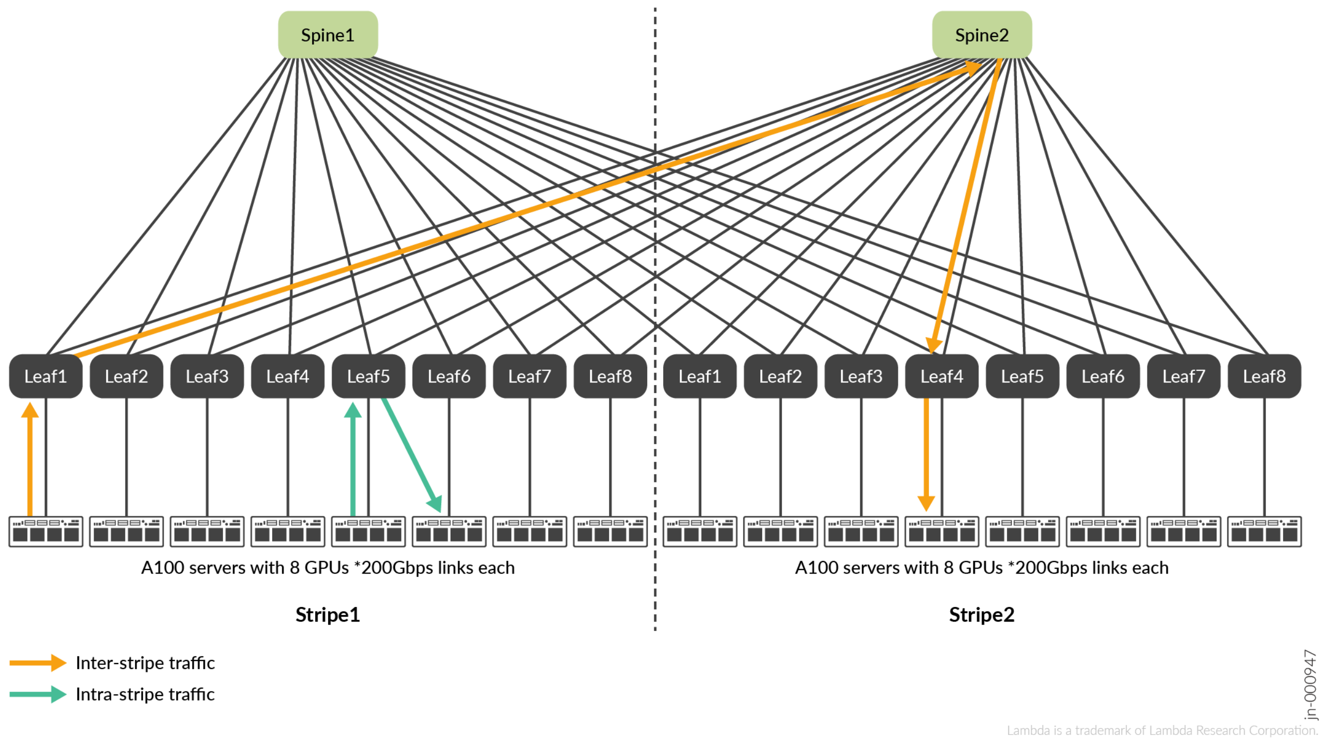
Figure 1 - Rail-Optimized Stripe



The GPUs are connected following a rail-optimized design. For example, for the Nvidia A100 severs (which have 8 GPUs and 8 NICs in the server), any-to-any GPU communication within the server is achieved via high throughput NVLink channels attached to an NVSwitch. In addition, each NIC in the server connects to a unique leaf (NIC1 to leaf1, NIC2 to leaf2, and so on), and the same methodology is followed on all servers, achieving a 'rail' design. With such a design, along with optimizations such as PXN (PCIe x NVLink), network interference is minimized by moving data to a GPU on the same rail as the destination, and thus sending data to the destination without crossing rails, which minimizes the number of network hops required. The spines are utilized when traffic crosses stripes (inter-stripe traffic pattern).

Sample traffic paths are shown in **Figure 2** for intra-stripe and inter-stripe traffic.

Figure 2 - Intra-Stripe and Inter-Stripe Traffic Pattern



IP Services for Congestion Avoidance

AI clusters pose unique demands on network infrastructure due to their high-density, low-entropy traffic patterns, characterized by frequent elephant flows with minimal flow variation. Ensuring uninterrupted packet flow, particularly for applications intolerant to packet loss or interruptions, presents significant challenges. The forthcoming section explores how DCQCN employs various techniques to strike a balance between reducing traffic rates and stopping traffic flow to alleviate congestion without resorting to packet drops.

Priority-Based Flow Control (PFC) is one technique to help relieve congestion by halting traffic flow in specific queues or ports. However, the resumption of traffic post-congestion often triggers a surge, potentially exacerbating or recreating the congestion scenario. While PFC mitigates data loss by halting flow, it does so at the expense of data transmission, impacting applications using the assigned queues during the congestion period.

Explicit Congestion Notification (ECN), on the other hand, curtails transmit rates during congestion, enabling traffic to persist albeit at reduced rates until congestion subsides. Combining PFC and ECN offers congestion relief while safeguarding against packet loss for protocols incapable of retransmission.

Despite equitably distributed paths from leafs to spines, traffic characteristics may impede optimal link utilization. For example, default hash algorithms often favor certain links due to limited flow variation. Consequently, highly utilized links may impede the transmission of smaller, low-bandwidth flows, leading to potential collisions.

To address this issue, Dynamic Load Balancing (DLB) should be implemented on leafs within the IP Fabric supporting AI clusters. Additionally, congestion control mechanisms are vital given the lossless network requirements for AI cluster-generated traffic. DCQCN (Data Center Quantized Congestion Notification) is employed, particularly for RDMA over Converged Ethernet (RoCEv2) traffic.

In a lossless IP fabric where RoCEv2 is in use, the objective is to prioritize ECN over PFC for congestion management. This entails marking packets with ECN bits at congestion points, prompting receivers to generate Congestion Notification Packets (CNPs) and signal the source to throttle traffic rates accordingly.

Configuring a Lossless IP Fabric for RoCEv2 Traffic

A lossless IP fabric for RoCEv2 traffic is configured by leveraging the IP services described in the previous section, which is a combination of DLB (dynamic load-balancing) and DCQCN (PFC+ECN). This ensures that traffic is marked with ECN bits to indicate congestion to the recipient, enabling backoff mechanisms to kick in to slow down traffic prior to drops.

In general, traffic loss in such fabrics may present itself as either input drops (receive errors) or tail drops on a Juniper device. Input drops indicate that ingress buffers are being overwhelmed for the corresponding priority group, while tail drops indicate that egress buffers are being overwhelmed for the corresponding queue.

The reason that the drop profile fill-level thresholds are configured to trigger ECN before PFC is because PFC can cause excess ingress buffering, leading to input drops. These statistics act as markers to determine how to slide the drop profile fill-level thresholds that control when ECN marking occurs as follows:

- If input drops (with PFC pause frames) or tail drops are seen, the fill-level minimum threshold should be **lowered** so that ECN kicks in first and reduces the traffic rate from the source.

- If ECN marking is seen, the fill-level minimum threshold should be **increased in small increments**. The ECN increase should not cross threshold levels where tail drops or PFC pause frames are seen. If PFC is seen, decrease the value until the ECN marking occurs without PFC or tail drops.

The following section gives guidance on how to configure a Juniper QFX device for lossless traffic queueing. The settings covered below should be considered a starting point utilizing the drop profile fill-level. The configuration shown here is deployed across all leafs and spines in the fabric.

DLB configuration with the inactivity-interval set to a value of 16 micro-seconds is as follows:

```
admin@leaf1# show forwarding-options enhanced-hash-key
ecmp-dlb {
    flowlet {
        inactivity-interval 16;
    }
    ether-type {
        ipv4;
    }
}
```

The Class of Service (CoS) configuration for a lossless IP fabric for RoCEv2 traffic is explored below:

- Classify the RoCEv2 traffic based on the DSCP markings that the servers are using. This example uses the decimal value 26, which is a binary value of 011010. This assumes the servers are sending RoCEv2 traffic marked with a DSCP value of 26. By default, this feature is not enabled on the Nvidia servers and must be configured separately. In addition, classify the Congestion Notification Packets (CNP) using a DSCP value of 48 (110000), which Nvidia servers use as a standard for CNP packets.

  ```
  classifiers {
      dscp mydscp {
          forwarding-class CNP {
              loss-priority low code-points 110000;
          }
          forwarding-class NO-LOSS {
              loss-priority low code-points 011010;
          }
      }
  }
  ```

- Set the drop profile and fill-level threshold which controls the point at which the switch starts marking packets with ECN to indicate congestion and avert tail-dropping. This will be discussed in detail in the sections below. For now, the below configuration means that at 55% fill level, the switch starts marking the traffic with ECN bit 11 using a probability number between 0 and 100. At 90% fill level, it marks 100% of the packets with ECN bits set to 11, indicating congestion.

  During congestion testing, the best balance between ECN marking and PFC was found to be a fill-level of [55 90] and drop-probability of [0 100] in our test environment. Depending on customer requirements, these values will likely differ, but the tuning method will be the same.

  If during a congestion event, ECN marking is experienced, but PFC is not, you can increase the first fill-level number by five until PFC is triggered, for example, “fill-level [60 90]”. Once PFC is seen, reduce the first fill-level value by five and monitor the validation commands below to ensure PFC or tail drops do not occur. This new value is the best balance during congestion to ensure network packets can transmit at a reduced rate versus pausing network packets (see **Validating Congestion Parameters for Tuning** in this document).

```
drop-profiles {
    dp1 {
        interpolate {
            fill-level [ 55 90 ];
            drop-probability [ 0 100 ];
        }
    }
}
```

- Allocate buffer partitions to the lossless, lossless-headroom, and lossy traffic types in the ingress and egress direction. The only restriction is that the ingress lossless buffer has to be equal to the egress lossless buffer.

```
shared-buffer {
    ingress {
        buffer-partition lossless {
            percent 80;
        }
        buffer-partition lossless-headroom {
            percent 10;
        }
        buffer-partition lossy {
            percent 10;
        }
    }
    egress {
        buffer-partition lossless {
            percent 80;
        }
        buffer-partition lossy {
            percent 10;
        }
    }
}
```

- Create forwarding classes for CNP and RoCEv2 traffic and assign the RoCEv2 traffic to a lossless queue (CNP in queue 3 and RoCEv2 in queue 4, in this example). Lossless forwarding-classes (queues) can use the lossless shared buffer that was partitioned previously.
- A PFC priority of 3 indicates that the priority of 3 is set/enabled in the PFC pause frame generated by the device.

```
forwarding-classes {
    class CNP queue-num 3;
    class NO-LOSS queue-num 4 no-loss pfc-priority 3;
}
```

- Create input and output congestion notification profiles, which is responsible for PFC generation and processing. Please note that both input and output profiles are needed, as shown in the output below, because input deals with generating PFC pause frames when the switch is congested, and output deals with receiving and processing PFC pause frames if a connected device is congested.

```
congestion-notification-profile {
    cnp {
        input {
            dscp {
                code-point 011010 {
                    pfc;
                }
            }
        }
        output {
            ieee-802.1 {
                code-point 011 {
                    flow-control-queue 4;
                }
            }
        }
    }
}
```

- Add the drop profiles and enable ECN for them within a scheduler. Connect the scheduler to a forwarding class using scheduler maps. Assign the congestion notification profile (responsible for PFC) and the scheduler maps(responsible for ECN/CNP) to the necessary interfaces.

```
interfaces {
    et-* {
        congestion-notification-profile cnp;
        scheduler-map sm1;
        unit * {
            classifiers {
                dscp mydscp;
            }
        }
    }
}
scheduler-maps {
    sm1 {
        forwarding-class CNP scheduler s2-cnp;
        forwarding-class NO-LOSS scheduler s1;
    }
}
schedulers {
    s1 {
        drop-profile-map loss-priority any protocol any drop-profile dp1;
        explicit-congestion-notification;
    }
    s2-cnp {
        transmit-rate percent 5;
        priority strict-high;
    }
}
```

Validating Congestion Parameters for Tuning

As highlighted in the above sections, when using Juniper infrastructure, the following congestion parameters can be used to indicate the need for tuning ECN thresholds:

Table 1: Congestion Parameters and ECN Threshold Tuning

EVENT	MONITOR CLI COMMAND	INDICATOR	ACTION	EXPECTED OUTCOME
ECN marked packets	*show interfaces <int> extensive*	Review "*Output Errors*" and "*ECN Marked Packets*" This indicates that the buffer is utilized to the fill level value, and the switch requests senders to reduce the transmit rate. If this prematurely impacts application performance, tune as indicated in the Action field.	Increase the drop profile fill level value until PFC occurs. Then, reduce by decrements of 5 until PFC stops.	ECN marking occurs later in the buffer utilization, reducing the frequency of traffic throttling.
PFC Pause Frames	*show interfaces <int> extensive*	Review "*Priority Flow Control Statistics*" Indicates traffic is coming in at a rate greater than the shared input buffer.	Reduce drop profile fill level value until PFC no longer occurs.	ECN occurs earlier in buffer utilization to mitigate the Pause Frame and interruption of traffic transmission.
Tail Drops & Egress queue peak buffer occupancy	*show interfaces <int> extensive* *show interfaces queue <int>* *show interfaces queue buffer-occupancy <int>*	Review "*Egress Queues*" and "*Dropped Packets*" Indicates packets are being dropped based on WRED profile.	Reduce the drop profile fill level values until ECN occurs before drops/PFC.	Traffic reduction should happen earlier, allowing queues to clear without drops.
Input drops & Ingress priority-group buffer occupancy	*show interfaces <int> extensive* *show interfaces priority-group <int> buffer-occupancy*	Review "*Drops*" and "*Resource Errors*" The ingress shared buffer is being exceeded and unable to store the incoming packets. This event is common when PFCs are generated.	Increase shared buffer > ingress > buffer-partition <%> For "Resource Errors", Increase the "ingress buffer-partition lossless-headroom" percentage. If the ingress shared buffer partition is at the expected value, Reduce the fill level values until ECN occurs before drops/PFC.	More memory is allocated to the specified buffer if needed. No or minimal input drops were experienced.

This section describes how to validate these different parameters on the QFX Series. This specific example uses a QFX5230. Buffer monitoring must be enabled to view buffer occupancy per priority-group or per egress queue. This can be enabled as follows:

```
jnpr@gpu-backend-stripe-001-leaf4# show chassis
fpc 0 {
    traffic-manager {
        buffer-monitor-enable;
    }
}
```

The peak buffer utilization for an ingress port can be viewed using show interfaces priority-group buffer-occupancy [*interface-name*], as follows:

```
jnpr@gpu-backend-stripe-001-leaf4> show interfaces priority-group et-0/0/1
buffer-occupancy
Physical interface: et-0/0/1, Enabled, Physical link is Up
  Interface index: 1061, SNMP ifIndex: 615
            PG: 0
                PG-Utilization bytes  :
                Peak                  : 7993380
            PG: 1
                PG-Utilization bytes  :
                Peak                  : 0
            PG: 2
                PG-Utilization bytes  :
                Peak                  : 0
            PG: 3
                PG-Utilization bytes  :
                Peak                  : 0
            PG: 4
                PG-Utilization bytes  :
                Peak                  : 0
            PG: 5
                PG-Utilization bytes  :
                Peak                  : 0
            PG: 6
                PG-Utilization bytes  :
                Peak                  : 0
            PG: 7
                PG-Utilization bytes  :
                Peak                  : 0
```

The number of PFC pause frames generated for a specific priority group can be viewed using show interfaces [*interface-name*] extensive. This also shows any input drops due to overflowing ingress buffer (these are also counted as resource errors). Since the configuration generates PFC pause frames with a priority of 3 and processes PFC pause frames with a priority of 3, this is the priority-level that should be analyzed. Following is an example of this output using one ingress port et-0/0/1.

```
jnpr@gpu-backend-stripe-001-leaf4> show interfaces et-0/0/1 extensive
```

```
Physical interface: et-0/0/1, Enabled, Physical link is Up
  Interface index: 1061, SNMP ifIndex: 615, Generation: 652835030207
  Description: facing_spine1:et-0/0/7
  Link-level type: Ethernet, MTU: 9216, LAN-PHY mode, Speed: 400Gbps, BPDU
  Error: None, Loop Detect PDU

*snip*

  Input errors:
    Errors: 0, Drops: 0, Framing errors: 0, Runts: 0, Policed discards: 0, L3
    incompletes: 0, L2 channel errors: 0, L2 mismatch timeouts: 0, FIFO errors:
    0, Resource errors: 0
  Output errors:
    Carrier transitions: 0, Errors: 0, Drops: 0, Collisions: 0, Aged packets: 0,
    FIFO errors: 0, HS link CRC errors: 0, MTU errors: 0, Resource errors: 0,
    ECN Marked packets: 0

*snip*

  MAC Priority Flow Control Statistics:
    Priority :  0                              0                 0
    Priority :  1                              0                 0
    Priority :  2                              0                 0
    Priority :  3                              0                 0
    Priority :  4                              0                 0
    Priority :  5                              0                 0
    Priority :  6                              0                 0
    Priority :  7                              0                 0
```

For egress queue buffer utilization, the peak buffer occupancy can be viewed using show interfaces queue buffer-occupancy [*interface-name*], as follows:

```
jnpr@gpu-backend-stripe-001-leaf4> show interfaces queue buffer-occupancy
et-0/0/28
Physical interface: et-0/0/28, Enabled, Physical link is Up
  Interface index: 1069, SNMP ifIndex: 543
Forwarding classes: 12 supported, 5 in use
Egress queues: 12 supported, 5 in use
            Queue: 0, Forwarding classes: best-effort
                Queue-depth bytes  :
                Peak               : 0
            Queue: 3, Forwarding classes: CNP
                Queue-depth bytes  :
                Peak               : 0
            Queue: 4, Forwarding classes: NO-LOSS
                Queue-depth bytes  :
                Peak               : 9086082
            Queue: 7, Forwarding classes: network-control
                Queue-depth bytes  :
                Peak               : 0
            Queue: 8, Forwarding classes: mcast
```

```
                    Queue-depth bytes  :
                    Peak               : 0
```

The number of ECN marked packets can be viewed using show interfaces [*interface-name*] extensive, where the interface is the egress interface for the traffic. This also shows the traffic queuing statistics, along with any tail-dropped packets for a specific queue. In this case, packets are hitting queue 4 (as expected) and there are no tail-drops.

In addition to this, granular queuing statistics can be viewed using the show interfaces queue [*interface-name*] command.

```
jnpr@gpu-backend-stripe-001-leaf4> show interfaces et-0/0/28 extensive
Physical interface: et-0/0/28, Enabled, Physical link is Up
  Interface index: 1069, SNMP ifIndex: 543, Generation: 652835030263
  Description: to.ixia-aresone
  Link-level type: Ethernet, MTU: 9216, LAN-PHY mode, Speed: 200Gbps, BPDU
  Error: None, Loop Detect PDU Error: None, MAC-REWRITE Error: None, Loopback:
  Disabled, Source filtering: Disabled, Flow control: Disabled,
  Auto-negotiation: Disabled, Media type: Fiber
  Device flags   : Present Running
  Interface flags: SNMP-Traps
  CoS queues     : 0 supported, 0 maximum usable queues
  Hold-times     : Up 0 ms, Down 0 ms
  Damping        : half-life: 0 sec, max-suppress: 0 sec, reuse: 0, suppress:
  0, state: unsuppressed
  Current address: d0:81:c5:ec:cc:77, Hardware address: d0:81:c5:ec:cc:77
  Last flapped   : 2024-03-11 11:11:33 PDT (1d 11:30 ago)
  Statistics last cleared: 2024-03-12 22:27:44 PDT (00:14:24 ago)
  Traffic statistics:
   Input  bytes  :          2012730642             18872496 bps
   Output bytes  :      21322851537681         197378071848 bps
   Input  packets:            28803217                33714 pps
   Output packets:         14230153733             16465436 pps
  Input errors:
    Errors: 0, Drops: 0, Framing errors: 0, Runts: 0, Policed discards: 0, L3
    incompletes: 0, L2 channel errors: 0, L2 mismatch timeouts: 0, FIFO errors:
    0, Resource errors: 0
  Output errors:
    Carrier transitions: 0, Errors: 0, Drops: 0, Collisions: 0, Aged packets:
    0, FIFO errors: 0, HS link CRC errors: 0, MTU errors: 0, Resource errors:
    0, ECN Marked packets: 0
  Egress queues: 12 supported, 5 in use
  Queue counters:       Queued packets  Transmitted packets     Dropped packets
    0                                0                    0                   0
    3                                0                    0                   0
    4                      14230148345          14230148345                   0
    7                                0                    0                   0
    8                                7                    7                   0

*snip*
```

```
jnpr@gpu-backend-stripe-001-leaf4> show interfaces queue et-0/0/28
Physical interface: et-0/0/28, up, Physical link is Up
  Interface index: 1084, SNMP ifIndex: 543
Forwarding classes: 12 supported, 5 in use
Egress queues: 12 supported, 5 in use
Queue: 0, Forwarding classes: best-effort
  Queued:
    Packets              :                     0                     0 pps
    Bytes                :                     0                     0 bps
  Transmitted:
    Packets              :                     0                     0 pps
    Bytes                :                     0                     0 bps
    Tail-dropped packets :                     0                     0 pps
    Tail-dropped bytes   :                     0                     0 bps
    RED-dropped packets  :                     0                     0 pps
    RED-dropped bytes    :                     0                     0 bps
Queue: 3, Forwarding classes: CNP
  Queued:
    Packets              :                     0                     0 pps
    Bytes                :                     0                     0 bps
  Transmitted:
    Packets              :                     0                     0 pps
    Bytes                :                     0                     0 bps
    Tail-dropped packets :                     0                     0 pps
    Tail-dropped bytes   :                     0                     0 bps
    RED-dropped packets  :                     0                     0 pps
    RED-dropped bytes    :                     0                     0 bps
Queue: 4, Forwarding classes: NO-LOSS
  Queued:
    Packets              :            1355306301              16468511 pps
    Bytes                :         2030294255848          197363366344 bps
  Transmitted:
    Packets              :            1355306301              16468511 pps
    Bytes                :         2030294255848          197363366344 bps
    Tail-dropped packets :                     0                     0 pps
    Tail-dropped bytes   :                     0                     0 bps
    RED-dropped packets  :                     0                     0 pps
    RED-dropped bytes    :                     0                     0 bps

*snip*
```

Running the DLRM Training Model with Optimized Class of Service

Using the network infrastructure and the IP services defined and configured in the earlier sections, the DLRM training model was run. The following output confirms the training time (in minutes) over three iterations of the model with 64 Nvidia A100 GPUs.

```
+------------------------------------------------------------------------------------------+---------------+---------------------+------+

|                                            File                                          | time to train |    Avg epoch time   | epoch|

+------------------------------------------------------------------------------------------+---------------+---------------------+------+

|   /mnt/nfsshare/logs/dlrm/A100-RAILS-ALL/03182024_22_59_10/240318225911319397655_1.log   |    2.76225    | 0.15345740740740743 |  18  |

|   /mnt/nfsshare/logs/dlrm/A100-RAILS-ALL/03182024_22_59_10/240318225911319397655_2.log   |    3.08295    | 0.15414666666666668 |  20  |

|   /mnt/nfsshare/logs/dlrm/A100-RAILS-ALL/03182024_22_59_10/240318225911319397655_3.log   |    2.73265    |  0.1518138888888889 |  18  |

+------------------------------------------------------------------------------------------+---------------+---------------------+------+
```

These training times fall within the optimum MLPerf training benchmarks defined by MLCommons found here - https://mlcommons.org/benchmarks/training/.

Corporate and Sales Headquarters
Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, CA 94089 USA
Phone: 888.JUNIPER (888.586.4737)
or +1.408.745.2000
Fax: +1.408.745.2100
www.juniper.net

APAC and EMEA Headquarters
Juniper Networks International B.V.
Boeing Avenue 240
1119 PZ Schiphol-Rijk
Amsterdam, The Netherlands
Phone: +31.207.125.700
Fax: +31.207.125.701

Copyright 2023 Juniper Networks, Inc. All rights reserved. Juniper Networks, the Juniper Networks logo, Juniper, Junos, and other trademarks are registered trademarks of Juniper Networks, Inc. and/or its affiliates in the United States and other countries. Other names may be trademarks of their respective owners. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.