Troubleshoot C8000v Routers Performance Issues

Contents

Introduction

Components Used

General Troubleshooting

- Overruns
- Feature Drops
- Taildrops

Hypervisors

- VMware ESXi
- AWS
  - Multi-TX Queues
  - Exceeded Metrics
- Microsoft Azure
  - Accelerated Networking
  - Azure and Fragmentation
  - Supported Instance Types for Microsoft Azure

Additional Resources

Introduction

This document describes how to troubleshoot performance issues in C8000v enterprise routers across public clouds and ESXi scenarios.

Components Used

The information in this document is based on these hardware and software components:

- C8000v running 17.12 version
- ESXI Version 7.0 U3

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

General Troubleshooting

Although you can have your C8000v hosted in different environments, there are still a few troubleshooting steps that you can go through that are identical regardless where the C8000v is hosted. Let us start with the basics. The first thing you need to make sure is if the device is reaching its capacity limits or not. For that you can start by checking the these two outputs:

1. **show platform hardware qfp active datapath util summary** – This command gives you the complete information of the input/output that the C8000v is receiving and transmitting from every port. You must focus your attention on the processing load percentage. If you are in a scenario where you are reaching 100% it means that you are reaching the capacity limit

```
------------------ show platform hardware qfp active datapath utilization summary  ------------------

  CPP 0:                     5 secs        1 min        5 min       60 min
Input:     Total (pps)        93119        92938        65941        65131
                 (bps)    997875976   1000204000    708234904    699462016
Output:    Total (pps)        93119        92949        65944        65131
                 (bps)   1052264704   1054733128    746744264    737395744
Processing: Load (pct)           14           14           10           10
```

2. **show platform hardware qfp active datapath infrastructure sw-cio –** Think of this command as a more in-depth version of the one above. It provides better detail about the individual cores including the IO and crypto cores which are not part of the QFP utilization number. It is very useful in a scenario where you want to see if a specific data plane core is causing a bottleneck.



Tip: A very important detail when using this command, always run it twice. As it calculates the percentage core utilization between the time the command was executed.

```
------------------ show platform hardware qfp active datapath infrastructure sw-cio  ------------------

Credits Usage:

  ID      Port  Wght  Global WRKR0  WRKR1  WRKR2  WRKR3  WRKR4  WRKR5  WRKR6  WRKR7  WRKR8  WRKR9  WRKR
   1      rcl0    16:    492     0      0      0      0      0      0      0      0      0      0
   1      rcl0    32:    496     0      0      0      0      0      0      0      0      0      0
   2       ipc     1:    489     0      0      0      0      0      0      0      0      0      0
   3 vxe_punti     4:    490     0      0      0      0      0      0      0      0      0      0
   4       Gi1     4:   1999     0      0      0      0      0      0      0      0      0      0
   5       Gi2     4:   1991     0      0      0      0      0      0      0      0      0      0
   6       Gi3     4:   1991     0      0      0      0      0      0      0      0      0      0
   7       Gi4     4:   1993     0      0      0      0      0      0      0      0      0      0
   8       Gi5     4:   2009     0      0      0      0      0      0      0      0      0      0
   9       Gi6     4:   2015     0      0      0      0      0      0      0      0      0      0
  10       Gi7     4:   2002     0      0      0      0      0      0      0      0      0      0
  11      vpg0   400:    490     0      0      0      0      0      0      0      0      0      0

Core Utilization over preceding 107352.2729 seconds
-------------------------------------------------
      ID:       0       1       2       3       4       5       6       7       8       9      10
    % PP:    2.98    2.01    1.81    1.67    1.60    1.53    1.35    1.30    1.25    1.19    2.19
    % RX:    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

```
   % TM:    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.
 % IDLE:   97.02   97.99   98.19   98.33   98.40   98.47   98.65   98.70   98.75   98.81   97.81   98.
```

Now, you have determined if you are reaching the platform limit or not. The next step would be to check for drops. These are inherently connected to performance issues. There are three types of drops you can consider depending on where they are occurring.

- Overruns: This type of packet drop occurs on the Rx end. They happen because one or more cores' processing capacity has been exceeded.
- Feature drops: This type of packet drop occurs in the PPE. They are related to features in the router such as an ACL or QoS.
- Taildrops: This type of packet drop occurs in the Tx end. They happen due to congestion in the Tx buffers.

To identify which drops you are experiencing, you can use the these outputs:

- **show platform hardware qfp active drop state clear**
- **show interface**
- **show policy map interface**

You verify how to identify which drops you are facing and how to mitigate them. Nevertheless, the biggest focus in this article is going to be the drops that known as Taildrops, as they are particularly tricky to troubleshoot in virtual routers.

Overruns

An overrun drop in Cisco IOS XE occurs when the network interface receives packets faster than it can process or store them in its buffer. Specifically, the internal buffers of the interfaces (FIFO queue) becomes full because the incoming data rate exceeds the ability of the hardware to handle it. As a result, new incoming packets cannot be stored and are dropped, which increments the overrun counter. This is essentially a packet loss caused by the interface being overwhelmed temporarily.

This type of packet drops occur on the Rx end. They happen because one or more cores' processing capacity has been exceeded, and Rx thread is unable to distribute incoming packets to the relevant PP thread and ingress buffers are already full. To make a simple analogy, you can think of it as a queue at a checkout counter that gets too full because packets are arriving faster than the cashier (interface hardware) can serve them. When the queue is full, new customers have to leave without being served - these are the overrun drops.

Even though hardware is mentioned in this section, the C8000v is a software-based router. In this case, overruns can be caused by:

- High data plane utilization: If the data plane utilization is high, the packets can not be polled fast enough, leading to overruns. For example, the presence of "elephant flows" (large, continuous data flows) can saturate processing resources and cause overruns on interfaces.

- Incorrect device template: Using an inappropriate device template can lead to inefficient buffer management and overruns. This can be checked with the command **show platform software cpu alloc** and it can be changed with the **platform resource <template>** command.

Each interface gets assigned a limited pool of credits, this mechanism prevents a busy interface and overloading the system resources. Each time a new packet arrives into the dataplane a credit is required. When packet processing is done, the credit is returned so Rx thread can use it again. If there is no available credit for the interface the packet needs to wait in the interface Rx ring. Generally, you expect the performance limit related drops to be Rx overruns because one or more cores' processing capacity has been exceeded.

To identify overruns, you typically check the interface statistics for input errors, specifically the overrun counter:

- Use the command **show platform hardware qfp active datapath infrastructure sw-cio** to identify the core utilization, and if the number of credits for a specific interface has been exceeded.
- Use the command **show interface <interface-name>** and look for the overrun count in the output.

Overruns are shown as part of input errors, for example:

```
Gig2 is up, line protocol is up
241302424557 packets input, 168997587698686 bytes, 0 no buffer
20150 input errors, 0 CRC, 0 frame, 20150 overrun, 0 ignored <<<<<<<<<<<
```

Let us suppose a case where Gig2 is observing performance issues caused by overruns. To determine the worker thread associated with this interface, you can use the this command:

```
show platform hardware qfp active datapath infra binding
Port Instance Bindings:

  ID       Port                 IOS Port WRKR 2
   1       rcl0                     rcl0  Rx+Tx
   2        ipc                      ipc  Rx+Tx
   3  vxe_punti              vxe_puntif  Rx+Tx
   4        Gi1         GigabitEthernet1  Rx+Tx
   5        Gi2         GigabitEthernet2  Rx+Tx >> in this case, WRKR2 is the thread responsible for bo
```

Then, you can analyze the utilization of the specific thread responsible for the Rx traffic of this interface and its number of credits.

In a scenario where Gig2 is observing performance issues due to overruns, you can expect the PP#2 constantly fully utilized (Idle = 0%), and low/zero credits for interface Gig2:

```
show platform hardware qfp active datapath infrastructure sw-cio
Credits Usage:

  ID      Port  Wght  Global WRKR0  WRKR1  WRKR2  Total
   1      rcl0    16:    487     0      0     25    512
   1      rcl0    32:    496     0      0     16    512
   2       ipc     1:    490     0      0     21    511
   3 vxe_punti     4:    459     0      0     53    512
   4       Gi1     4:    477     0      0     35    512
   5       Gi2     4:    474     0      0     38    512    <<< low/zero credits for interface Gig2:

Core Utilization over preceding 1.0047 seconds
-----------------------------------------------------
      ID:       0       1       2
    % PP:    0.77    0.00    0.00
    % RX:    0.00    0.00    0.44
    % TM:    0.00    0.00    5.63
  % IDLE:   99.23   99.72   93.93    <<< the core ID relevant in this case would be PP#2
```

Feature Drops

Packets are handled by any available data plane thread and are distributed strictly based on availability of QFP cores via software Rx function (x86) - Load Based Distribution (LBD). Packets that arrive in the PPE can be dropped with a specific QFP drop reason, which can be checked using the this output:

```
------------------ show platform hardware qfp active statistics drop detail ------------------

Last clearing of QFP drops statistics : never

-------------------------------------------------------------------------------
   ID  Global Drop Stats                          Packets                Octets
-------------------------------------------------------------------------------
  319  BFDoffload                                     403                 31434
  139  Disabled                                       105                  7487
   61  Icmp                                           135                  5994
   94  Ipv4NoAdj                                        1                   193
   33  Ipv6NoRoute                                   2426                135856
  215  UnconfiguredIpv4Fia                        1937573             353562196
  216  UnconfiguredIpv6Fia                        8046173            1057866418


------------------ show platform hardware qfp active interface all statistics drop_summary ------------

-----------------------------------------------------------------
Drop Stats Summary:
note: 1) these drop stats are only updated when PAL
         reads the interface stats.
      2) the interface stats include the subinterface

Interface                                   Rx Pkts            Tx Pkts
-------------------------------------------------------------------------
GigabitEthernet1                            9980371                  0
```

```
GigabitEthernet2                          4012                0
```

The reasons for the drops are diverse and are usually self-explanatory. To investigate further, a packet trace can be used.

Taildrops

As it was mentioned before, taildrops occur where the device is trying to transmit packets, but the transmission buffers are full.

In this sub-section you are going to look at which outputs you can examine when faced with this type of situation. Which values you can see in them mean and what you can do to mitigate the issue.

First, you need to know how to identify them. One such way is to simply look at the **show interface**. Keep an eye for any output drops increasing:

```
GigabitEthernet2 is up, line protocol is up
  Hardware is vNIC, address is 0050.56ad.c777 (bia 0050.56ad.c777)
  Description: Connected-To-ASR Cloud Gateway
  Internet address is 10.6.255.81/29
  MTU 1500 bytes, BW 10000000 Kbit/sec, DLY 10 usec,
     reliability 255/255, txload 2/255, rxload 3/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full Duplex, 10000Mbps, link type is force-up, media type is Virtual
  output flow-control is unsupported, input flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 03:16:21
  Input queue: 0/375/0/0 (size/max/drops/flushes); Total output drops: 7982350 >>> Output drops.
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 150449000 bits/sec, 20461 packets/sec
  5 minute output rate 89116000 bits/sec, 18976 packets/sec
```

This command is particularly useful to understand if you are experiencing congestion or not:

- **show platform hardware qfp active datapath infrastructure -** HQF stands for `Hierarchical Queueing Framework`. This is a feature that enables Quality of Service (QoS) management at different levels (physical, logical, and class) using the modular QoS command-line interface (MQC). It shows the current RX and TX costs. When the TX queue is full, as the output shows (full 1959)

```
pmd b1689fc0 device Gi1
RX: pkts 5663120 bytes 1621226335 return 0 badlen 0
```

```
Out-of-credits: Hi 0 Lo 0
pkts/burst 1 cycl/pkt 1565 ext_cycl/pkt 1173
Total ring read 12112962299, empty 12107695202
TX: pkts 8047873582 bytes 11241140363740
pri-0: pkts 8047873582 bytes 11241140363740
pkts/send 3
Total: pkts/send 3 cycl/pkt 452
send 2013612969 sendnow 1810842
forced 2013274797 poll 724781 thd_poll 0
blocked 2197451 retries 7401 mbuf alloc err 0
TX Queue 0: full 1959 current index 0 hiwater 224 >>>
```

The output suggests that the underlying hardware is not keeping up with the sending of packets. To debug the underlying interface, you need to potentially look outside of the C8000v and at the underlying environment where the C8000v is running on to see if there are any additional errors reported on the underlying physical interfaces.

In order to check the environment there is one step that you can do before checking in which hypervisor the C8000v router is running. This is to check the command **show controller** output. Nevertheless, you can find yourself lost on what each counter means or where to look at.

First, one important detail that you need to have in mind when looking at this output is that the information is mostly sourced from the vNICs themselves. Each NIC driver has a specific set of counters that they use, which can vary by driver, naturally. Different hypervisors have some sort of effect on what is presented as well. Some counters, such as mbuf counters, are stats from the DPDK driver. These can vary for different DPDK drivers. The actual counting is generally done by the hypervisor at the virtualization layer.

```
GigabitEthernet2 - Gi2 is mapped to UIO on VXE
rx_good_packets 1590
tx_good_packets 1402515
rx_good_bytes 202860
tx_good_bytes 1857203911
rx_missed_errors 0
rx_errors 0
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0_packets 1590
rx_q0_bytes 202860
rx_q0_errors 0
tx_q0_packets 1402515
tx_q0_bytes 1857203911
rx_q0_drop_total 0
rx_q0_drop_err 0
rx_q0_drop_fcs 0
rx_q0_rx_buf_alloc_failure 0
tx_q0_drop_total 976999540797
tx_q0_drop_too_many_segs 0
tx_q0_drop_tso 0
tx_q0_tx_ring_full 30901211518
```

Take a minute here to learn how to interpret and read these counters:

1. If you see subX, it means it is a sub-interface - a logical division of the main interface. The sub0 is generally the primary/default one. These are often used when multiple VLANs are involved.
2. Then, you have "rx = receiving" and "tx = transmitting".
3. Finally, **q0** refers to the first/default queue used by that interface

Although there is not a description for every counter, the article describes some of them, which can be relevant to your troubleshooting:

- “RX_MISSED_ERRORS:" is seen when the NIC buffer (Rx FIFO) becomes overfull. This condition leads to drops and an increase in latency. A possible solution to this is to increase the NIC buffer (which is impossible in our case) or change the NIC driver.
- "tx_q0_drop_total" and "tx_q0_tx_ring_full": These can tell you that the host is dropping packets, and the C8000v is experiencing Tail Drops in the C8000v because the host is back-pressuring the C8000v

In the output above, we do not see any “rx_missed_errors”. However, as we are focusing on taildrops we do see both "tx_q0_drop_total" and "tx_q0_tx_ring_full". With this, we can conclude that there is indeed congestion caused by the underlying hardware of the host.

As mentioned before, each hypervisor has some sort of effect on what is presented. The article focuses on this in the next section as it goes over the differences between the different hypervisors where the C8000v can be hosted. You can also find the different recommendations to try and mitigate this type of issue in each one of them.

Hypervisors

A hypervisor is a software layer that allows multiple operating systems (called virtual machines or VMs) to run on a single physical hardware host by managing and allocating the hardware resources such as CPU, memory, and storage to each VM. It ensures that these virtual machines operate independently without interfering with each other.

In the context of the Cisco Catalyst 8000V (C8000v), the hypervisor is the platform that hosts the C8000v virtual machine. How do you find out which hypervisor is hosting your C8000v? There is a rather useful output that gives us that information. Additionally, you can also check what kind of resources our virtual router has access to:

```
C8000v#show platform software system all
Processor Details
=================
Number of Processors : 8
Processor : 1 - 8
vendor_id : GenuineIntel
cpu MHz : 2593.906
cache size : 36608 KB
Crypto Supported : Yes
model name : Intel(R) Xeon(R) Platinum 8272CL CPU @ 2.60GHz

Memory Details
==============
```

```
Physical Memory : 32817356KB

VNIC Details
============
Name Mac Address Driver Name Status Platform MTU
GigabitEthernet1 0022.480d.7a05 net_netvsc UP 1500
GigabitEthernet2 6045.bd69.83a0 net_netvsc UP 1500
GigabitEthernet3 6045.bd69.8042 net_netvsc UP 1500

Hypervisor Details
==================
Hypervisor: AZURE
Manufacturer: Microsoft Corporation
Product Name: Virtual Machine
Serial Number: 0000-0002-0201-5310-5478-4052-71
UUID: 8b06091c-f1d3-974c-85a5-a78dfb551bf2
Image Variant: None
```

VMware ESXi

ESXi is a type-1 hypervisor developed by VMware that is installed directly on physical servers to enable virtualization. It allows multiple virtual machines (VMs) to run on a single physical server by abstracting the hardware resources and allocating them to each VM. The C8000v router is one of those VMs.

You can start by going over a common scenario where congestion is occurring. This can be confirmed by checking the **tx_q0_tx_ring_full counte**r:

Example:

```
------------------ show platform software vnic-if interface-mapping ------------------

------------------------------------------------------------
Interface Name Driver Name Mac Addr
------------------------------------------------------------
GigabitEthernet3 net_vmxnet3 <-- 0050.5606.2239
GigabitEthernet2 net_vmxnet3 0050.5606.2238
GigabitEthernet1 net_vmxnet3 0050.5606.2237
------------------------------------------------------------

GigabitEthernet3 - Gi3 is mapped to UIO on VXE
rx_good_packets 99850846
tx_good_packets 24276286
rx_good_bytes 78571263015
tx_good_bytes 14353154897
rx_missed_errors 0
rx_errors 0
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0packets 99850846
rx_q0bytes 78571263015
rx_q0errors 0
tx_q0packets 24276286
```

```
tx_q0bytes 14353154897
rx_q0_drop_total 0
rx_q0_drop_err 0
rx_q0_drop_fcs 0
rx_q0_rx_buf_alloc_failure 0
tx_q0_drop_total 160945155
tx_q0_drop_too_many_segs 0
tx_q0_drop_tso 0
tx_q0_tx_ring_full 5283588 <------
```

This congestion happens when the C8000V attempts to send packets through the VMXNET3 interface. However, the buffer ring is already full of packets, which end up in either delays or drops.

Under these conditions, those drops are happening on the hypervisor side as we have mentioned before. If all the recommendations are met, it is recommended to check with VMware support to understand what is happening on the NIC.

Here are some suggestions on how to improve the performance:

- Use a Dedicated vSwitch and Uplink for Optimal Performance
- By assigning the C8000V to a dedicated vSwitch backed by its own physical uplink, we can isolate its traffic from noisy neighbours and avoid shared resource bottlenecks.

There are a few commands that are worth having a look from the ESXi side. For example, to check for packet loss from the ESXi interface, we can do the this:

1. Enable SSH.
2. Connect to ESXi using SSH.
3. Run **esxtop**.
4. Type **n**.

The **esxtop** command can show packets dropped at the virtual switch if the network driver of the virtual machine runs out of Rx buffer memory. Even though **esxtop** shows the packets as dropped at the virtual switch, they are actually dropped between the virtual switch and the guest operating system driver.

Search for any packets being dropped under **%DRPTX** and **%DRPRX**:

```
12:34:43pm up 73 days 16:05, 907 worlds, 9 VMs, 53 vCPUs; CPU load average: 0.42, 0.42, 0.42

PORT-ID     USED-BY                   TEAM-PNIC   DNAME             PKTTX/s   MbTX/s   PSZTX   PKTRX/s
67108870    Management                 n/a         vSwitch-to-9200      0.00     0.00     0.00      0.00
67108872    Shadow of vmnic1          n/a         vSwitch-to-9200      0.00     0.00     0.00      0.00
67108876    vmk1                      vmnic1      vSwitch-to-9200      0.00     0.00     0.00      0.00
67108890    2101719:c8kv-gw-mgmt      vmnic1      vSwitch-to-9200  76724.83   792.35  1353.00  16180.39
100663305   Management                 n/a         vSwitch-to-Cisc      0.00     0.00     0.00      0.00
100663307   Shadow of vmnic0          n/a         vSwitch-to-Cisc      0.00     0.00     0.00      0.00
```

```
100663309   vmk0                      vmnic0      vSwitch-to-Cisc      3.64     0.01   280.00     3.29
100663310   2100707:gsoaresc-On_Prem  vmnic0      vSwitch-to-Cisc      0.00     0.00     0.00     2.43
100663311   2100993:cats-vmanage      void        vSwitch-to-Cisc      0.00     0.00     0.00     0.00
100663312   2100993:cats-vmanage      void        vSwitch-to-Cisc      0.00     0.00     0.00     0.00
100663313   2100993:cats-vmanage      vmnic0      vSwitch-to-Cisc      5.38     0.01   212.00     9.71
100663314   2101341:cats-vsmart       void        vSwitch-to-Cisc      0.00     0.00     0.00     0.00
100663315   2101341:cats-vsmart       vmnic0      vSwitch-to-Cisc      2.60     0.00   164.00     6.94
100663316   2101522:cats-vbond        vmnic0      vSwitch-to-Cisc      0.00     0.00     0.00     0.00
100663317   2101522:cats-vbond        vmnic0      vSwitch-to-Cisc      0.00     0.00     0.00     0.00
100663318   2101522:cats-vbond        vmnic0      vSwitch-to-Cisc      4.33     0.01   174.00     7.80
100663319   2101522:cats-vbond        vmnic0      vSwitch-to-Cisc      0.00     0.00     0.00     4.16
100663320   2101547:gdk-backup        vmnic0      vSwitch-to-Cisc      0.00     0.00     0.00     3.12
100663321   2101703:sevvy             vmnic0      vSwitch-to-Cisc      0.00     0.00     0.00     3.12
100663323   2101719:c8kv-gw-mgmt      vmnic0      vSwitch-to-Cisc  16180.91     9.09    73.00  76755.87
100663324   2137274:telemetry-server  vmnic0      vSwitch-to-Cisc      0.00     0.00     0.00     3.12
100663335   2396721:netlab            vmnic0      vSwitch-to-Cisc      0.00     0.00     0.00     3.12
2214592519  vmnic1                    -           vSwitch-to-9200  76727.26   792.38  1353.00  16182.64
2248146954  vmnic0                    -           vSwitch-to-Cisc  16189.05     9.32    75.00  76736.97
```

This command lists all NICs configured on a host:

```
esxcli network nic list
Name PCI Device Driver Admin Status Link Status Speed Duplex MAC Address MTU Description
------ ------------ ------ ------------ ----------- ----- ------ ----------------- ---- -----------
vmnic0 0000:01:00.0 igbn Up Up 1000 Full fc:99:47:49:c5:0a 1500 Intel(R) I350 Gigabit Network Connectio
vmnic1 0000:01:00.1 igbn Up Up 1000 Full fc:99:47:49:c5:0b 1500 Intel(R) I350 Gigabit Network Connectio
vmnic2 0000:03:00.0 ixgben Up Up 1000 Full a0:36:9f:1c:1f:cc 1500 Intel(R) Ethernet Controller 10 Gigab
vmnic3 0000:03:00.1 ixgben Up Up 1000 Full a0:36:9f:1c:1f:ce 1500 Intel(R) Ethernet Controller 10 Gigab
```

There is also a useful command to check the status of the vNIC assigned to a particular VM.

```
esxcli network vm list
World ID  Name                Num Ports  Networks
--------  ------------------  ---------  ----------------------------------------
2137274   telemetry-server    1          Cisco Backbone 10.50.25.0/24
2101703   sevvy               1          Cisco Backbone 10.50.25.0/24
2396721   netlab              1          Cisco Backbone 10.50.25.0/24
2101547   gdk-backup          1          Cisco Backbone 10.50.25.0/24
2101522   cats-vbond          4          VPN0, VPN0, VPN0, VPN0
2101719   c8kv-gw-mgmt        2          c8kv-to-9200l, c8kv-to-cisco
2100707   gsoaresc-On_Prem    1          Cisco Backbone 10.50.25.0/24
2100993   cats-vmanage        3          VPN0, VPN0, VPN0
2101341   cats-vsmart         2          VPN0, VPN0
```

Looking at the c8kv-gw-mgmt , which is a C8000v VM, there are 2 Networks assigned:

- c8kv-to-9200l
- c8kv-to-cisco

You can use the world ID to look for more information about this VM:

```
[root@localhost:~] esxcli network vm port list -w 2101719
Port ID:           67108890
vSwitch:           vSwitch-to-9200L
Portgroup:         c8kv-to-9200l
DVPort ID:
MAC Address:       00:0c:29:31:a6:b6
IP Address:        0.0.0.0
Team Uplink:       vmnic1
Uplink Port ID:    2214592519
Active Filters:

Port ID:           100663323
vSwitch:           vSwitch-to-Cisco
Portgroup:         c8kv-to-cisco
DVPort ID:
MAC Address:       00:0c:29:31:a6:ac
IP Address:        0.0.0.0
Team Uplink:       vmnic0    <----
Uplink Port ID:    2248146954
Active Filters:
[root@localhost:~]
```

Once you have this information, you can identify which Network the vSwitch is assigned to.

To check some traffic statistics of the physical NIC assigned to the vSwitch we have the this command:

```
# esxcli network nic stats get -n <vmnic>
```

This command displays information such as packets received, bytes received, packets dropped and received errors. This can help to identify if there are any drops happening at the NIC.

```
[root@localhost:~] esxcli network nic stats get -n vmnic0
NIC statistics for vmnic0
Packets received: 266984237
Packets sent: 123640666
Bytes received: 166544114308
Bytes sent: 30940114661
Receive packets dropped: 0
Transmit packets dropped: 0
Multicast packets received: 16773454
Broadcast packets received: 36251726
Multicast packets sent: 221108
Broadcast packets sent: 1947649
Total receive errors: 0
Receive length errors: 0
Receive over errors: 0
Receive CRC errors: 0
```

```
Receive frame errors: 0
Receive FIFO errors: 0
Receive missed errors: 0
Total transmit errors: 0
Transmit aborted errors: 0
Transmit carrier errors: 0
Transmit FIFO errors: 0
Transmit heartbeat errors: 0
Transmit window errors: 0
```

There are a few configurations to be verified that can improve the performance of the Cisco Catalyst 8000V running on an ESXi environment by modifying the settings on the host and the virtual machine:

- Set the Virtual Hardware: CPU reservation setting to Maximum.
- Reserve all the guest memory in Virtual Hardware: Memory.
- Select VMware Paravirtual from Virtual Hardware: SCSI Controller.
- From the Virtual Hardware: Network Adapter: Adapter Type option, select SR-IOV for the supported NICs
- Set the General Guest OS Version > VM Options option to Other 3.x or later Linux (64-bit).
- Set the VM Options option under Advanced Latency Sensitivity to High.
- Under VM Options > Advanced Edit Configuration, add "numa.nodeAffinity" to the same NUMA node as the SRIOV NIC
- Enable the hypervisor performance settings.
- Limit the overhead of vSwitch by enabling SR-IOV on the supported Physical NICs.
- Configure the vCPUs of the VM to run on the same NUMA node as Physical NICs.
- Set the VM Latency Sensitivity to High.

AWS

The C8000v supports deployment on AWS by launching as an Amazon Machine Image (AMI) within an Amazon Virtual Private Cloud (VPC), allowing users to provision a logically isolated section of the AWS cloud for their network resources.

Multi-TX Queues

In a C8000v running on AWS, a key feature is the use of Multi-TX Queues (Multi-TXQs). These queues help reduce internal processing overhead and improve scalability. Having multiple queues makes it faster and simpler to assign incoming and outgoing packets to the correct virtual CPU (vCPU).

Unlike some systems where RX/TX queues are assigned per vCPU, in the C8000v, these queues are assigned per interface. The RX (receive) and TX (transmit) queues serve as the connection points between the Catalyst 8000V application and the AWS infrastructure or hardware, managing how network traffic is sent and received. AWS controls the number and speed of RX/TX queues available for each interface, depending on the instance type.

To create multiple TX queues, the Catalyst 8000V needs to have multiple interfaces. When multiple TX queues are enabled, the device keeps the order of packet flows by using a hashing method based on the 5-tuple of the flow (source IP, destination IP, source port, destination port, and protocol). This hashing decides which TX queue to use for each flow.

Users can create multiple interfaces on the Catalyst 8000V by using the same physical network interface card (NIC) attached to the AWS instance. This is done by configuring loopback interfaces or adding secondary IP addresses.

With Multi-TXQs, there are multiple transmit queues to handle outgoing traffic. In the example, there are twelve TX queues (numbered 0 to 11). This setup allows you to monitor each queue individually to see if any are becoming full.

Looking at the output, you can see that **TX Queue 8** has a very high "full" counter (56,406,998), which means its buffer is filling up frequently. The other TX queues show zero for the "full" counter, indicating they are not congested.

```
Router#show platform hardware qfp active datapath infrastructure sw-cio
pmd b17a2f00 device Gi2
RX: pkts 9525 bytes 1229599 return 0 badlen 0
Out-of-credits: Hi 0 Lo 0
pkts/burst 1 cycl/pkt 560 ext_cycl/pkt 360
Total ring read 117322273, empty 117312792
TX: pkts 175116324 bytes 246208197526
pri-0: pkts 157 bytes 10238
pkts/send 1
pri-1: pkts 75 bytes 4117
pkts/send 1
pri-2: pkts 91 bytes 6955
pkts/send 1
pri-3: pkts 95 bytes 8021
pkts/send 1
pri-4: pkts 54 bytes 2902
pkts/send 1
pri-5: pkts 75 bytes 4082
pkts/send 1
pri-6: pkts 104 bytes 8571
pkts/send 1
pri-7: pkts 74 bytes 4341
pkts/send 1
pri-8: pkts 175115328 bytes 246208130411
pkts/send 2
pri-9: pkts 85 bytes 7649
pkts/send 1
pri-10: pkts 106 bytes 5784
pkts/send 1
pri-11: pkts 82 bytes 7267
pkts/send 1
Total: pkts/send 2 cycl/pkt 203
send 68548581 sendnow 175024880
forced 1039215617 poll 1155226129 thd_poll 0
blocked 2300918060 retries 68534370 mbuf alloc err 0
TX Queue 0: full 0 current index 0 hiwater 0
```

```
TX Queue 1: full 0 current index 0 hiwater 0
TX Queue 2: full 0 current index 0 hiwater 0
TX Queue 3: full 0 current index 0 hiwater 0
TX Queue 4: full 0 current index 0 hiwater 0
TX Queue 5: full 0 current index 0 hiwater 0
TX Queue 6: full 0 current index 0 hiwater 0
TX Queue 7: full 0 current index 0 hiwater 0
TX Queue 8: full 56406998 current index 224 hiwater 224 <<<<<<<<<<
TX Queue 9: full 0 current index 0 hiwater 0
TX Queue 10: full 0 current index 0 hiwater 0
TX Queue 11: full 0 current index 0 hiwater 0
```

Monitoring the "full" counters of TX queues helps identify if any transmit queue is overloaded. A consistently increasing "full" count on a particular TX queue points to a traffic flow that is stressing the device. Addressing this can involve traffic balancing, adjusting configurations, or scaling resources to improve performance.

Exceeded Metrics

AWS sets certain network limits at the instance level to ensure consistent and high-quality network performance across different instance sizes. These limits help maintain stable networking for all users.

You can check these limits and related statistics using the **show controllers** command on your device. The output includes many counters, but here we focus only on the most important ones for monitoring network performance:

```
c8kv-2#sh control | inc exceed
<snipped>
bw_in_allowance_exceeded 0
bw_out_allowance_exceeded 0
pps_allowance_exceeded 0
conntrack_allowance_exceeded 0
linklocal_allowance_exceeded 0
<snipped>
```

You can now dive in and see what those counters refer to exactly:

- **bw_in_allowance_exceeded:** Number of packets queued or dropped because incoming bandwidth went over the limit of the instance.
- **bw_out_allowance_exceeded:** Number of packets queued or dropped because outgoing bandwidth went over the limit of the instance.
- **pps_allowance_exceeded:** Number of packets queued or dropped because the total packets per second (PPS) exceeded the limit of the instance.
- **conntrack_allowance_exceeded:** Number of connections tracked that reached the maximum allowed for the instance type.
- **linklocal_allowance_exceeded:** Number of packets dropped because traffic to local proxy services (like Amazon DNS, Instance Metadata Service, and Time Sync Service) exceeded the PPS limit for

the network interface. This does not affect custom DNS resolvers.

What this means for your C8000v performance:

- If you notice these counters increasing and experience performance issues, it does not always mean the C8000v router is the problem. Instead, it often indicates that the AWS instance you are using has reached its capacity limits. You can check the specifications of your AWS instance to ensure it can handle your traffic needs.

Microsoft Azure

In this section, explore how Microsoft Azure and the Cisco C8000v virtual router combine to deliver scalable, secure, and high-performance virtual networking solutions in the cloud.

Go through how Accelerated Networking (AN) and packet fragmentation can impact performance. As well as reviewing the importance of using a supported instance type for Microsoft Azure.

Accelerated Networking

In performance issues cases where the C8000v is hosted in the Microsoft Azure Cloud. One aspect that you cannot overlook is whether Accelerated Network is enabled or not. As it greatly increases the performance of the router. In a nutshell, accelerated networking enables single root I/O virtualization (SR-IOV) on VMs such as a Cisco Catalyst 8000V VM. The accelerated networking path bypasses the virtual switch, increases the speed of network traffic, improves the networking performance, and reduces the network latency and jitter.

There is a very simple way to check if Accelerated Network is enabled. That is to check the **show controllers** output and verify if a certain counters is present or not:

```
------------------ show controllers ------------------

GigabitEthernet1 - Gi1 is mapped to UIO on VXE
  rx_good_packets 6497723453
  tx_good_packets 14690462024
  rx_good_bytes 2271904425498
  tx_good_bytes 6276731371987
<snip>
  rx_q0_good_packets 58576251
  rx_q0_good_bytes 44254667162
<snip>
  vf_rx_good_packets 6439147188
  vf_tx_good_packets 14690462024
  vf_rx_good_bytes 2227649747816
  vf_tx_good_bytes 6276731371987
```

The counters that you are looking for are the ones that start with **vf** such as **vf_rx_good_packets**. If you

verify that these counters are present, you can be absolutely sure that accelerated networking is enabled.

Azure and Fragmentation

Fragmentation can have negative performance implications. One of the main reasons for the effect on performance is the CPU/memory effect of the fragmentation and reassembly of packets. When a network device needs to fragment a packet, it has to allocate CPU/memory resources to perform fragmentation.

The same thing happens when the packet is reassembled. The network device must store all the fragments until they are received so it can reassemble them into the original packet.

Azure does not process fragmented packets with Accelerated Networking. When a VM receives a fragmented packet, the nonaccelerated path processes it. As a result, fragmented packets miss the benefits of Accelerated Networking, such as lower latency, reduced jitter, and higher packets per second. For this reason, the recommendation is to avoid fragmentation if possible.

Azure, by default, drops fragmented packets that arrive at the VM out of order, meaning the packets do not match the transmission sequence from the source endpoint. This issue can occur when packets travel over the internet or other large WANs.

Supported Instance Types for Microsoft Azure

It is important that the C8000v is using a supported instance type as per Cisco standards. They can be found in the Cisco Catalyst 8000V Edge Software Installation And Configuration Guide.

The reason for this is because the instance types in that list are the ones where the C8KV was properly tested. Now, there is the valid question if the C8000v works on an instance type that is not listed? The answer is most probably yes. However, when you are troubleshooting something as complex as performance issues you do not want to add another unknown factor into the issue. For that reason alone, Cisco TAC always recommends to you to stay in a supported instance type.

Additional Resources

A performance issue can only be truly troubleshooted when it is happening in the moment. However, this can be hard to catch as it can happen at any given moment. For that reason, we provide this EEM script. It helps to capture important outputs the moment packets start getting dropped and performance issues ensue:

```
ip access-list extended TAC
permit ip host <source> host <destination>
permit ip host <destination> host <source>

conf t
event manager applet CONNECTIONLOST1 authorization bypass
event track 100 state down maxrun 500
```

```
action 0010 syslog msg "Logging information to file bootflash:SLA-DROPS.txt and bootflash:FIASLA_Decode
action 0020 cli command "enable"
action 0021 cli command "term length 0"
action 0022 cli command "term exec prompt timestamp"
action 0023 cli command "term exec prompt expand"
action 0095 cli command "show clock | append bootflash:SLA-DROPS.txt"
action 0096 cli command "show platform hardware qfp active statistics drop detail | append bootflash:SL
action 0097 cli command "show logging | append bootflash:SLA-DROPS.txt"
action 0099 cli command "show interfaces summary | append bootflash:SLA-DROPS.txt"
action 0100 cli command "show interfaces | append bootflash:SLA-DROPS.txt"
action 0101 cli command "show platform hardware qfp active statistics drop clear"
action 0102 cli command "debug platform packet-trace packet 2048 fia-trace"
action 0103 cli command "debug platform packet-trace copy packet both"
action 0104 cli command "debug platform condition ipv4 access-list TAC both"
action 0105 cli command "debug platform condition start"
action 0106 cli command "show platform hardware qfp active data infrastructure sw-cio | append bootflas
action 0110 wait 60
action 0111 cli command "debug platform condition stop"
action 0112 cli command "show platform packet-trace packet all decode | append bootflash:FIASLA_Decode.
action 0120 cli command "show platform packet-trace statistics | append bootflash:FIASLA_Decode.txt"
action 0121 cli command "show platform packet-trace summary | append bootflash:FIASLA_Decode.txt"
action 0122 cli command "show platform hardware qfp active datapath utilization summary | append bootfl
action 0123 cli command "show platform hardware qfp active statistics drop detail | append bootflash:SL
action 0124 cli command "show platform hardware qfp active infrastructure bqs queue output default all
action 0125 cli command "show platform software status control-processor brief | append bootflash:SLA-D
action 0126 cli command "show platform hardware qfp active datapath infrastructure sw-pktmem | append b
action 0127 cli command "show platform hardware qfp active infrastructure punt statistics type per-caus
action 0128 cli command "show platform hardware qfp active statistics drop | append bootflash:SLA-DROPS
action 0129 cli command "show platform hardware qfp active infrastructure bqs queue output default all
action 0130 cli command "show platform hardware qfp active data infrastructure sw-hqf config 0 0 | appe
action 0131 cli command "show platform hardware qfp active feature lic-bw oversubscription | append boo
action 0132 cli command "show platform hardware qfp active data infrastructure sw-hqf config 0 0 | appe
action 0133 cli command "show platform hardware qfp active data infrastructure sw-cio | append bootflas
action 0134 cli command "show platform hardware qfp active data infrastructure sw-hqf sched | append bo
action 0135 cli command "show platform hardware qfp active data infrastructure sw-dist | append bootfla
action 0136 cli command "show platform hardware qfp active data infrastructure sw-nic | append bootflas
action 0137 cli command "show platform hardware qfp active data infrastructure sw-pktmem | append bootf
action 0138 cli command "show controllers | append bootflash:SLA-DROPS.txt"
action 0139 cli command "show platform hardware qfp active datapath pmd controllers | append bootflash:
action 0140 cli command "show platform hardware qfp active datapath pmd system | append bootflash:SLA-D
action 0141 cli command "show platform hardware qfp active datapath pmd static-if-config | append bootf
action 0150 cli command "clear platform condition all"
action 0151 cli command "clear platform packet-trace statistics"
action 0152 cli command "clear platform packet-trace configuration"
action 0153 cli command "show log | append bootflash:througput_levelinfoSLA.txt"
action 0154 cli command "show version | append bootflash:througput_levelinfoSLA.txt"
action 0155 cli command "show platform software system all | append bootflash:througput_levelinfoSLA.tx
action 0156 syslog msg "EEM script and FIA trace completed."
action 0180 cli command "conf t"
action 0181 cli command "no event manager applet CONNECTIONLOST1"
end
```