Media and Entertainment Workloads on vSphere 6.7

Best practices and recommendations for deployment and performance tuning

March 27, 2019

Mark Achtemichuk
Bob Goldsand
Shak Malik

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com

Table of Contents

Overview

This guide documents VMware best practices and recommendations for running low latency media and entertainment applications in vSphere 6.7 VMs in cloud or hybrid-cloud environments. Due to VMware technology advances, not only can you run streaming TV client applications in VMs, you can also host the core video streaming applications for many broadcast and media solutions.

Introduction

The IT landscape has evolved significantly over the past two decades. Some of these changes were gradual, while others—like virtualization and cloud computing—were revolutionary in nature. Many of these changes, which introduced new technologies, allowed once complex and expensive-to-implement solutions—like agility, resilience, and security—to finally be leveraged. Today, we can take for granted IT designs like high availability and disaster recovery, on-demand and instantaneous access, and robust security. These changes have been implemented across most industries while keeping up with the shifting changes in regulations and other market demands. Because of low latency needs, however, the media industry has traditionally been unable to take advantage of these solutions in virtual and cloud-based environments. VMware has advanced these technologies, which are now available to IT experts when they use vSphere 6.7.

The media industry is now looking to take advantage of standards from IP to virtualization, cloud, containers, and micro-services. This enables workloads to be more agile and highly available with disaster recovery available at a fraction of the cost of physical environments. The industry is moving toward new architectures and models, using virtualization, similar to Telco and their network function virtualization (NFV). These new designs will be based around media function virtualization (MFV).

Broadcast and media technologies supported in vSphere 6.7

vSphere supports high definition (HD), ultra-high definition (UHD or Ultra HD), and Video over IP.

vSphere can perform to the standards defined in the following:

- ST2022/ST2022-1 FEC for Real-Time Video/Audio Transport
- ST2022-2 Constant Bit Rate MPEG-2 Transport Streams
- ST2022-3Variable Bit Rate MPEG-2 Transport Streams
- ST2022-4 Non-Piecewise Constant Variable Bit Rate MPEG-2 Streams
- ST2022-6 High Bit Rate Media Signals over IP Networks
- ST2022-5 Forward Error Correction for Transport of High Bit Rate Media Signals
- ST2022-7 Seamless Protection Switching of SMPTE ST 2022 IP Datagrams

Tools we use in examples

In order to understand, optimize, and troubleshoot the vSphere network topology, we will be using the vSphere command-line interface (esxcli) and the VMkernel system information shell (vsish), as well as guest-operating-system network tools to gather current network configuration and modify these settings where necessary.

- Use **esxcli** to run system administration commands against vSphere systems from any machine with network access to those systems.

- Use **vsish** to extract system reliability information and check advanced performance counters of the ESXi host and virtual machines running on the host.

General vSphere network considerations

A note about network virtualization

The virtual environment provides networking elements similar to those in the physical environment. They are virtual network interface cards (virtual NICs), vSphere distributed switches (VDS), distributed port groups, vSphere standard switches (VSS), and port groups.

- A virtual switch works like a layer 2 physical switch.
- With vSphere standard switch, each server has its own virtual switches.
- With vSphere distributed switch, a single virtual switch spans many servers.

Find virtual network configurations and statistics

In order to fully leverage the vSphere esxcli and vsish utilities, you need to extract configuration information such as port IDs, world IDs, VM names, network adaptor type, form factor, driver, and so on, which are necessary components in many shell commands.

To determine the physical NIC (PNIC) layout, use the command **esxcli network nic list**, as shown in the following example. You can see the physical NIC name under **MTU Description**, and which virtual NIC each one is mapped to under **Name**.

```
[root@xxxx:~] esxcli network nic list
Name  PCI Device Driver Admin Status  Link Status  Speed  Duplex  MAC Address MTU  Description
------  ------------  ------  ------------  -----------  -----  ------  -----------------  ---
-  ------------------------------------------------
vmnic0  0000:1a:00.0  i40en   Up            Up           10000  Full    24:6e:96:b4:f5:9c  1500
Intel(R) Ethernet Controller X710 for 10GbE SFP+
vmnic1  0000:1a:00.1  i40en   Up            Up           10000  Full    24:6e:96:b4:f5:9e  1500
Intel(R) Ethernet Controller X710 for 10GbE SFP+
vmnic2  0000:1a:00.2  i40en   Up            Down             0  Half    24:6e:96:b4:f5:a0  1500
Intel(R) Ethernet Controller X710 for 10GbE SFP+
vmnic3  0000:1a:00.3  i40en   Up            Down             0  Half    24:6e:96:b4:f5:a2  1500
Intel(R) Ethernet Controller X710 for 10GbE SFP+
vmnic4  0000:5e:00.0  i40en   Up            Up           10000  Full    f8:f2:1e:0d:14:98  9000
Intel(R) Ethernet Controller X710 for 10GbE SFP+
vmnic5  0000:5e:00.1  i40en   Up            Down             0  Half    f8:f2:1e:0d:14:9a  1500
Intel(R) Ethernet Controller X710 for 10GbE SFP+
vmnic6  0000:d8:00.0  i40en   Up            Down             0  Half    f8:f2:1e:0b:95:88  1500
Intel(R) Ethernet Controller X710 for 10GbE SFP+
vmnic7  0000:d8:00.1  i40en   Up            Down             0  Half    f8:f2:1e:0b:95:8a  1500
Intel(R) Ethernet Controller X710 for 10GbE SFP+
vusb0   Pseudo
cdce      Up            Up               100  Full    58:8a:5a:f0:b8:23  1500  DellTM iDRAC
Virtual NIC USB Device
```

To determine the individual capabilities of each PNIC, use the command **esxcli network nic get -n vmnic0**, where **vmnic0** is the name of the NIC found in the previous example.

```
[root@xxxx:~] esxcli network nic get -n vmnic0
   Advertised Auto Negotiation: false
   Advertised Link Modes: 10000BaseT/Full
   Auto Negotiation: false
   Cable Type: DA
   Current Message Level: -1
   Driver Info:
         Bus Info: 0000:1a:00:0
         Driver: i40en
         Firmware Version: 6.00 0x800034eb 18.3.6
         Version: 1.4.3
   Link Detected: true
   Link Status: Up
   Name: vmnic0
   PHYAddress: 0
   Pause Autonegotiate: false
   Pause RX: true
   Pause TX: true
   Supported Ports: DA
   Supports Auto Negotiation: false
   Supports Pause: true
   Supports Wakeon: true
   Transceiver:
   Virtual Address: 00:50:56:53:a2:4c
   Wakeon: MagicPacket(tm)
```

From the output of these commands, you can derive the physical characteristics of the PNICs, including the driver and version information covered in the next section.

Use native network device drivers

vSphere fully supports native network device drivers, and we recommend using these, instead of the older ESXi vmklinux drivers, for the best performance. The benefits of native drivers are:

- Efficient and flexible device driver
- Standardized information for debugging/troubleshooting
- Improved performance
- Support for PCIe hot-plug

Native network drivers end with an "n", while the vmklinux drivers do not. For example, the Intel Ethernet 700 series network adapter native driver for ESXi is named "i40en".

> **Note:** When looking for compatible network device drivers from the VMware Compatibility Guide, be sure to select a native device driver—the version number might be lower than that of the vmklinux driver.

To list the parameters of a driver, using the Intel example, you would enter **esxcli system module parameters list -m i40en**:

```
[root@xxxx:~] esxcli system module parameters list -m i40en
Name     Type          Value  Description
-------  ------------  -----  -----------------------------------------------------------------
RxITR    int                  Default RX interrupt interval (0..0xFFF), in microseconds
(default = 50)
TxITR    int                  Default TX interrupt interval (0..0xFFF), in microseconds,
(default = 100)
VMDQ     array of int         Number of Virtual Machine Device Queues: 0/1 = disable, 2-16 enable
(default = 8)
max_vfs  array of int         Maximum number of VFs to be enabled (0..128)
```

Offload work to NICs

In this section, we discuss and recommend offloading techniques that maximize performance for media and entertainment environments. The goal is to offload the work to network interface cards (NICs) to preserve vCPU resources while maximizing network performance. In high performance systems, the preservation of overall vCPU utilization ultimately leads to increased application efficiency.

Enable TPC/IP offload (TSO) engine

For media and entertainment workloads, use TCP segmentation offload (TSO) in VMkernel network adapters and virtual machines to improve the network performance in workloads that are latency sensitive.

TSO on the transmission path of physical network adapters, and VMkernel and virtual machine network adapters improve the performance of ESXi hosts by reducing the overhead of the CPU for TCP/IP network operations. When TSO is enabled, the network adapter divides larger data frames (up to 64 KB), sent from the host, into multiple frames which are sent out to the network. This results in less compute utilization on the ESXi host.

The NIC must support TSO; if it does, TSO is typically enabled by default. Also, in order to leverage TSO, it must be activated in vSphere.

> **Note**: TSO is referred to as LSO (*large segment offload* or *large send offload*) in the latest VMXNET3 driver attributes.

To check if TSO is supported and enabled on the physical NIC, run the following command:

```
[root@xxxx:~] esxcli network nic tso get
NIC     Value
------  -----
vmnic0  on
vmnic1  on
vmnic2  on
vmnic3  on
vmnic4  on
vmnic5  on
vmnic6  on
vmnic7  on
vusb0   off
```

To check if TSO is enabled within the ESXi kernel, run the following command:

```
[root@xxxx:~] esxcli system settings advanced list -o /Net/UseHwTSO
   Path: /Net/UseHwTSO
   Type: integer
   Int Value: 1
   Default Int Value: 1
   Min Value: 0
   Max Value: 1
   String Value:
   Default String Value:
   Valid Characters:
   Description: When non-zero, use pNIC HW TSO offload if available
```

A **Default Int Value** of 1 indicates TSO is enabled within the ESXi kernel advanced settings. If the value is 0, use the following command to enable TSO in vSphere:

```
esxcli system settings advanced set -o /Net/UseHwTSO -i 1
```

Disable large receive offload (LRO)

Large receive offload (LRO) is used to reduce the CPU overhead for processing packets that arrive from the network at a high rate. LRO reassembles incoming network packets into larger buffers and transfers the resulting larger but fewer packets to the network stack of the host or virtual machine. When LRO is enabled, the CPU processes fewer packets than when it is disabled, which reduces its utilization for networking, especially in the case of connections that have high bandwidth. By default, LRO is enabled in the VMkernel and in the VMXNET3 virtual machine adapters.

This however has a negative impact on latency-sensitive workloads due to the time necessary to checksum and reassemble packets that are sent to the network stack. For this reason, we recommend **disabling** LRO for media and entertainment workloads.

To disable LRO:

1. In the vSphere Web Client, navigate to the host.
2. On the Configure tab, expand System.
3. Click Advanced System Settings.
4. Change the value of the **Net.Vmxnet3SwLRO** parameter for VMXNET3 adapters; set it to **0** to disable LRO.
5. Click OK to apply the changes.

To verify LRO has been disabled, run the following command. The description section should say **LRO enabled for TCP/IP**.

```
[root@xxxx:~] esxcli system settings advanced list -o /Net/TcpipDefLROEnabled
   Path: /Net/TcpipDefLROEnabled
   Type: integer
   Int Value: 0
   Default Int Value: 1
   Min Value: 0
   Max Value: 1
   String Value:
   Default String Value:
   Valid Characters:
   Description: LRO enabled for TCP/IP
```

Enable checksum offload (CSO)

The TCP header contains a 16-bit checksum field which is used to verify the integrity of the header and data. For performance reasons, the checksum calculation on the transmit side and verification on the receive side may be offloaded from the operating system to the network adapter. By enabling CSO, this eliminates the vCPU load on the host, which preserves more compute power for the virtual machines without negatively affecting latency. For this reason, we recommend enabling CSO.

To determine if CSO is enabled and supported for each NIC, run the following command:

```
[root@xxxx:~] esxcli network nic cso get
NIC     RX Checksum Offload  TX Checksum Offload
------  -------------------  -------------------
vmnic0  on                   on
vmnic1  on                   on
vmnic2  on                   on
vmnic3  on                   on
vmnic4  on                   on
vmnic5  on                   on
vmnic6  on                   on
vmnic7  on                   on
vusb0   off                  off
```

Physical server and vSphere host configuration

Change these BIOS settings at the hardware level

We recommend the following BIOS settings in the bare metal machine for the best performance:

- **Power management:** OS controlled
- **Hyperthreading:** enabled
- **Turbo boost:** enabled
- **C states:** disabled
- **C1E:** disabled
- **For Intel machines, Intel VT technology:** enabled
- **QPI power management:** disabled
- **Execute disable bit:** enabled
- **Node interleaving:** disabled

Change these host settings at the ESXi host operating system level

We recommend the following ESXi host settings for the best performance:

- **Power management:** high performance
- **TxSplit mode:** active

```
vsish -e set /net/pNics/vmnicX/sched/txMode 1
```

Virtual machine configurations

Set latency sensitivity to high

You can turn on latency sensitivity for each VM at the VM level or at the host level. This feature:

- Gives exclusive access to physical resources to avoid resource contention due to sharing
  - With exclusive physical CPU (pCPU) access given, each vCPU entirely owns a specific pCPU; no other vCPUs and threads (including VMkernel I/O threads) are allowed to run on it. This achieves nearly zero ready time and no interruption from other VMkernel threads, improving response time and jitter under CPU contention.
- Bypasses virtualization layers to eliminate the overhead of extra processing
  - Once exclusive access to pCPUs is obtained, the feature allows the vCPUs to bypass the VMkernel's CPU scheduling layer and directly halt in the virtual machine monitor (VMM), since there are no other contexts that need to be scheduled. That way, the cost of running the CPU scheduler code and the cost of switching between the VMkernel and VMM are avoided, leading to much faster vCPU halt/wake-up operations.
- Tunes virtualization layers to reduce the overhead
  - When the VMXNET3 paravirtualized device is used for VNICs in the VM, VNIC interrupt coalescing and LRO support for the VNICs are automatically disabled to reduce response time and jitter.
  - Although these and other features are disabled automatically when setting latency sensitivity to high, we recommend disabling each of these features independently to avoid any cascading effects when a single parameter is altered when tuning your virtual machines.

Reserve sufficient memory and CPU

Media and entertainment workloads require many of the same recommendations as mission-critical enterprise workloads that require setting memory and CPU reservations to exhibit consistent performance and maintain enterprise service-level agreements. In addition, the latency-sensitivity feature requires a full CPU and memory reservation. Also, we recommend not over-provisioning vCPUs to reduce contention; the number of vCPUs in the host should be less than the number of pCPUs to leave one or more pCPUs for VMkernel threads for I/O processing and system management.

Disable vCPU hot add

A necessary performance optimization includes exposing the NUMA topology to the virtual machine. This allows in-guest NUMA-aware applications to better control the allocation of memory in relation to which CPU the consuming thread is scheduled on. Without vNUMA, application latencies might increase. By enabling vCPU hot add, you disable vNUMA and the guest OS will only see a UMA topology which may not be as performant.

Make sure VMs meet minimum requirements

These are the minimum requirements for the virtual machine and the VMware driver versions within it:

- Virtual Hardware 14
- VMware Tools 10.3.5
- vmxnet3 1.8.3.1

Network tuning considerations

Set up receive-side scaling (RSS)

RSS is on by default in Linux VMs. For other guest operating systems, you should set it up.

About RSS

RSS, which was introduced in vSphere 5.1, can be confused with NetQueue because the basic functionality is the same: both technologies distribute packets among different queues. However, there is a difference in the implementation. NetQueue operates on MAC address filters, which means that packets belonging to the same MAC address will be processed by the same queue. Thus, a virtual NIC (VNIC) can receive packets from only one queue and, as a result, can limit performance. RSS allows distribution based on flows and as a result a single VNIC with multiple flows can leverage parallelism in the physical NIC (PNIC) and the ESXi stack using multiple queues. The distribution can vary among various NICs because some NICs support 5-tuple hashing techniques, while some others don't.

RSS is a mechanism that allows the network driver to spread incoming TCP traffic across multiple CPUs, resulting in increased multi-core efficiency and processor cache utilization. If the driver or the operating system is not capable of using RSS, or if RSS is disabled, all incoming network traffic is handled by only one CPU. In this situation, a single CPU can be the bottleneck for the network while other CPUs might remain idle.

> **Note:** To make use of RSS, the hardware version of the virtual machine must be 7 or higher, the virtual network card must be set to VMXNET3, and the guest operating system must be capable and configured properly. On some systems it has to be enabled manually.

Configure in-guest and kernel driver

num_tqs: Number of Tx queues for each adapter. Comma separated list of integers, one for each adapter. When set to 0, the number of Tx queues is made equal to the number of vCPUs. The default is 0.

If a virtual machine has three VMXNET3 adapters, this command configures the first VMXNET3 VNIC with four Tx queues, the second VMXNET3 VNIC with one Tx queue, and the third with two Tx queues:

```
modprobe vmxnet3 num_tqs=4,1,2
```

Disable interrupt coalescing

Interrupt coalescing is a feature on high-performance NICs; it triggers a single hardware interrupt that notifies the guest operating system kernel when a group of network frames is received. Without interrupt coalescing, multiple interrupts would occur for each network frame, which would decrease vCPU performance.

When latency sensitivity is set to high on the vSphere host (which is recommended), interrupt coalescing is automatically disabled.

To make sure interrupt coalescing is off, run the following command:

```
[root@xxxx:~] esxcli network nic coalesce get
NIC     RX microseconds  RX maximum frames  TX microseconds  TX Maximum frames  Adaptive RX  Adaptive TX  Sample interval seconds
------  ---------------  -----------------  ---------------  -----------------  -----------  -----------  -----------------------
vmnic0  N/A              N/A                N/A              N/A                N/A          N/A          N/A
vmnic1  N/A              N/A                N/A              N/A                N/A          N/A          N/A
```

If interrupt coalescing were enabled, the above output would contain statistical behavior of the physical NICs; however, since it has been disabled, no statistics are available; all appear as N/A.

If you have latency sensitivity set to something else, we still recommend disabling interrupt coalescing.

To disable interrupt coalescing, use the Advanced Settings of the virtual machine Edit, or add the following setting for each PNIC. Set ethernetX.coalescingScheme to 0.

Disable queue pairing

Some PNICs support a feature called *queue pairing*, which indicates to the ESXi uplink layer that the receive thread will also process the completion of transmitted packets on a paired transmit queue. For transmit-heavy media and entertainment workloads, this can cause delays in processing transmit completions and therefore can cause the transmit ring of the vNIC to run out of room for transmitting additional packets, forcing the vNIC driver in the guest OS to drop packets. This feature can be disabled on an ESXi host for all pNICs, thereby creating a separate thread for processing transmit completions for the pNICs so that they are processed in a timely manner to make room in the vNIC's transmit ring for additional packets.

The ESXi command to disable queue pairing is:

```
esxcli system settings advanced set -o /Net/NetNetqRxQueueFeatPairEnable -i 0
```

For this to take effect, the ESXi host needs to be rebooted. Like some of the other configuration options, this too should be carefully considered as it increases the CPU usage with the additional thread to handle transmit completions, impacting the amount of CPU resources available for running other workloads on the same host.

Disable interrupt throttling

Much like interrupt coalescing works at the virtual machine layer, interrupt throttling will coalesce interrupts from the physical NIC to the hosts with the same goal of preserving vCPU cycles. When enabled, this too will add latency to the system, so we recommend you disable interrupt throttling as well.

To disable interrupt throttling, run the following command:

```
esxcli system module parameters set -m ixgbe -p "InterruptThrottleRate=0"
```

Make sure it is 0 for off. The other settings are 1 for for dynamic, and 3 for dynamic conservative (default).

Right-size ring buffers

Ring buffers are used to handle bursty network traffic and can be configured both on the physical NIC and the guest operating system vmxnet3 NIC to minimize packet loss. If these buffers are filled before they are emptied, the virtual NIC will drop any additional incoming frames. This condition is known as buffer or ring exhaustion.

In order to combat ring exhaustion and optimize these setting for latency sensitive workloads, it's necessary to determine the current settings and behavior of the physical and virtual NICs. It's important to note that the

ring buffers sizes depend on the physical NICs used. Check with your vendor to determine the maximum and minimum sizes.

Determine the host settings

To get the current Rx and Tx buffers sizes of the physical NIC, run the following command on the host:

```
[root@xxxx:~] esxcli network nic ring current get -n vmnic0
   RX: 255
   RX Mini: 0
   RX Jumbo: 0
   TX: 511
```

As an example, if you want to increase the Rx buffer to 1024 and Tx buffer to 511, run the following command on the host:

```
esxcli network nic ring current set -n vmnic0 -r 1024 -t 511
```

Now when checking the ring size buffers with the **esxcli network nic ring current get -n vmnic0** command, the new values should be reflected. As previously mentioned, the physical NIC ring buffer sizes depend on the hardware vendor; if there is a mismatch when you try to set the desired value, you will see **invalid argument**.

Determine the in-guest settings

In the previous section, you changed the physical NIC settings through the ESXi host. To have end-to-end ring buffer alignment, you need to change the guest settings as well.

> Note: To change the in-guest ring buffer settings, you must have the VMXNET3 virtual network adapter installed.

To determine the current settings for a Linux guest, run the following command in the virtual machine:

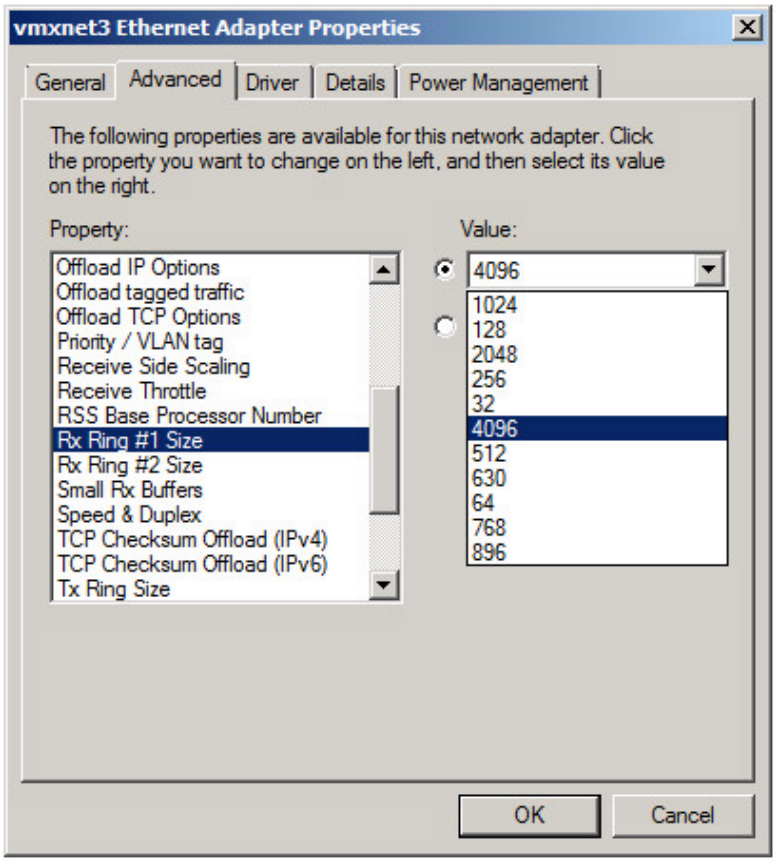
```
xxxx:~ # ethtool -g eth0
Ring parameters for eth0:
Pre-set maximums:
RX:             4096
RX Mini:        0
RX Jumbo:       0
TX:             4096
Current hardware settings:
RX:             4096
RX Mini:        0
RX Jumbo:       0
TX:             4096
```

To set the in-guest ring buffers to the same values as the example shown for physical NICs, execute the following command on the virtual machine:

```
ethtool -G eth0 rx 4096 tx 4096
```

The Rx and Tx values can be changed as shown above, or individually by excluding the desired arguments.

For Windows, you can modify the ring buffer values by changing the VMXNET3 Ethernet Adapter Properties under advanced settings, as shown in the following screenshot.



Determine ring buffer exhaustion

To determine the correct ring buffer values and if ring buffer exhaustion is occurring, use vsish.

But first, you need to obtain the portid and virtual machine of interest. There are several ways to get this data; here, we use **esxtop -n**. Here is a portion of the output. Look at the **port-id** and **used-by** sections on the left.

```
  PORT-ID USED-BY                                TEAM-PNIC DNAME
 33554433 Management                                   n/a vSwitch0
 33554434 vmnic4                                         - vSwitch0
 33554435 Shadow of vmnic4                             n/a vSwitch0
 33554440 353496890:img4-vmxnet-fixed (2          vmnic4 vSwitch0
 50331649 Management                                   n/a DvsPortset-0

 50331657 vmk4                                      vmnic0 DvsPortset-0
 50331662 353496890:img4-vmxnet-fixed (2          vmnic0 DvsPortset-0
 67108865 Management                                   n/a vSwitchiDRACvus
 67108866 vusb0                                          - vSwitchiDRACvus
 67108867 Shadow of vusb0                              n/a vSwitchiDRACvus
```

In this example, notice virtual machine **img4-vmxnet-fixed** with the port-id of **33554440**. You'll use the port-id in the following vsish command.

Note: Here, don't use esxtop because it may report zero packet loss even though packet loss is occurring.

The below vsish command lets you see if packets are being lost at the vNIC level.

```
[root@xxxx:~] vsish -e get /net/portsets/vSwitch0/ports/33554440/stats
packet stats {
   pktsTx:106521
   pktsTxMulticast:97130
   pktsTxBroadcast:9370
   pktsRx:975737540318
   pktsRxMulticast:975737491875
   pktsRxBroadcast:48413
   droppedTx:0
   droppedRx:1297
```

As you can see, the vNIC is experiencing packet loss. To determine if buffer ring exhaustion is the culprit for packet loss, we can take a closer look at the transmit statistics, as well as other vNIC characteristics, using the below vsish command:

```
[root@imgrail-4:~] vsish -e get /net/portsets/vSwitch0/ports/33554440/vmxnet3/rxSummary
stats of a vmxnet3 vNIC rx queue {
   LRO pkts rx ok:0
   LRO bytes rx ok:0
   pkts rx ok:502829850903
   bytes rx ok:725080550789161
   unicast pkts rx ok:8
   unicast bytes rx ok:612
   multicast pkts rx ok:502829821748
   multicast bytes rx ok:725080547307291
   broadcast pkts rx ok:29147
   broadcast bytes rx ok:3481258
   running out of buffers:199
   pkts receive error:0
   1st ring size:4096
   2nd ring size:4096
   # of times the 1st ring is full:199
   # of times the 2nd ring is full:0
   fail to map a rx buffer:0
   request to page in a buffer:0
   # of times rx queue is stopped:0
   failed when copying into the guest buffer:0
   # of pkts dropped due to large hdrs:0
   # of pkts dropped due to max number of SG limits:0
   pkts rx via data ring ok:0
   bytes rx via data ring ok:0
   Whether rx burst queuing is enabled:1
   current backend burst queue length:0
   maximum backend burst queue length so far:1477
   aggregate number of times packets are requeued:183
   aggregate number of times packets are dropped by PktAgingList:0
   # of pkts dropped due to large inner (encap) hdrs:0
   number of times packets are dropped by burst queue:0
}
```

This command yields a lot of information, you can see that Large Receive Offload (LRO) is disabled since zero packets have been transferred, the ring buffer sizes can be verified as well. To determine if the buffers are large enough the number of times the first ring fills is a key statistic. This number should be kept below 100. IF this number exceeds 100 then the ring buffer values should be increased. Normally the second ring represents jumbo frame traffic.

Leave SplitRx mode enabled

SplitRx mode uses multiple physical CPUs to process network packets received in a single network queue. This feature can significantly improve network performance for certain workloads through improved CPU

efficiency. Where multiple virtual machines on one ESXi host all receiving multicast traffic from the same source. Use of multiple cores for a single NIC, increased CPU but faster processing of incoming packets

This feature is enabled by default in vSphere 5.1 and greater and can be set at the host level or at the virtual NIC level.

We recommend you leave this feature enabled, although it will only benefit the ESXi host when multiple VMs are used on the same host.

Enable SplitTx mode

In vSphere 6.5, a new parameter is used to configure SplitTx mode. SplitTx mode allows two separate threads to be created for a single flow. SplitTx is configured for a pNIC on a host. When active, it enables the hypervisor to use two transmission threads for the pipeline: one for the vNIC backend, another for the pNIC.

If the workload is transmit heavy, consider enabling SplitTx mode.

SplitTx mode must be enabled per host with the command:

```
vsish –e set /net/pNics/vmnicX/sched/txMode 1
```

Right-size virtual machines

The size of virtual machines, in relation to the physical server on which they are hosted, is critical to ensuing the best performance to the virtual machine and reducing hypervisor contention for work it does on behalf of the virtual machines.

Set vNUMA appropriately

For details on setting vNUMA, see the blog post, "Virtual Machine vCPU and vNUMA Rightsizing – Rules of Thumb at https://blogs.vmware.com/performance/2017/03/virtual-machine-vcpu-and-vnuma-rightsizing-rules-of-thumb.html

Set exclusive affinity (system contexts)

Similar to setting Latency Sensitivity to High, a new vSphere 6.0 VMX feature can grant cores exclusive affinity to System Contexts that are associated with low latency virtual machine workloads. For these workloads the set of contexts are identified as those which have high communication with the latency-sensitive VMs. In vSphere 6.0, this only applies to worldlets. In vSphere 6.5, this applies to all worlds. For this reason in vSphere 6.0 only Tx threads can be scaled, however both Tx and Rx threads can be scaled in 6.5 or greater via the **sched.cpu.latencySensitivity.sysContexts** VMX parameter.

The vSphere scheduler functions independently of the **sched.cpu.latencySensitivity.sysContexts** VMX setting. As long as there is an active heavy communication established between the VM and the system context, the system context is a potential candidate for exclusive affinity. Exclusive Affinity on system contexts are best-effort. The scheduler will try its best to assign the exclusive core if deemed necessary. However, in the case of conflict of insufficiently CPU reservation, no exclusive core will be granted.

The **sched.cpu.latencySensitivity.sysContexts** is set to the number of cores you wish to reserve and grant worlds/worldlets exclusive access to, so if you wanted to reserve 3 cores for Tx threads and 3 cores for Rx threads then it would be set to 6.

```
sched.cpu.latencySensitivity.sysContexts=6
```

Earlier we suggested to using full CPU reservations for M&E virtual machines to guarantee performance levels and maintain service level agreements, this will affect the behavior of system contexts exclusive affinity if resource contention occurs. If virtual machines with full reservations are deployed on the same NUMA node, then in order to bring up that VM the **sched.cpu.latencySensitivity.sysContexts** setting will be disregarded and aborted allowing the cores associated with this setting will be made available for other virtual machines.

NUMA considerations must be taken into account, if the goal is to contain the virtual machine to a single NUMA while maintaining exclusive affinity. Since the **sched.cpu.latencySensitivity.sysContexts** can be unset by the scheduler, a VI admin may mistakenly believe a low-latency VM is running with exclusive cores assigned to Rx and Tx threads when it actually is not.

Right-size or scale transmit threads

By default, vSphere uses a single transmit and a single receive thread regardless of the number of virtual network interface devices configured. This single thread corresponds to a single core, if that core cannot process the network packets from the physical NIC then performance can be degraded, and packet loss may occur. If a particular vNIC is overloaded additional worlds can be configured to parallelize the threads. This is accomplished on a per vNIC basis with the **ethernetX.ctxPerDev** advanced setting in the VMX file.

The default value of the **ethernetX.ctxPerDev** is set to **2**. Change this setting to **1**; this allows additional system contexts to be configured, which will accommodate additional threads to be configured. As mentioned, this is done on a per vNIC basis. The **ethernetX** corresponds to the vNIC being configured.

Using the exclusive affinity example of 3 Tx threads and 3 Rx threads above:

```
sched.cpu.latencySensitivity.sysContexts=6
ethernet1.ctxPerDev=1
```

> **Note:** The **ethernetX.ctxPerDev** parameter is not a count of worlds; a setting of **1** enables you to configure additional worlds.

Save 4-6 physical cores for ESXi I/O completion

In addition to hypervisor services, each ESXi host has a number of worlds doing work on behalf of the virtual machine. These include worlds doing network receive and transmit prior to traffic being passed in/out of the virtual machine. In order to ensure we reduce scheduling contention on these worlds, and ensure low latency packet processing, you must save a number of physical cores (pCores) for these worlds and not allocate them to virtual machines.

It is recommended that 4-6 pCores be excluded from the total host count.

Example: If your ESXi host has two sockets with 18 pCores per socket (36 pCores, 72 logical thread with hyper-threading), then only 30-32 vCPUs can be allocated to virtual machines.

Use a third-party tool for precision timekeeping

Precise timekeeping is a key attribute for many types of applications. At a high level, it allows these applications to accurately construct the precise sequence of events that have occurred or are occurring in real time. For example, the media and entertainment industry often deploys multiple audio/video equipment with high sampling frequencies, from which the results are re-assembled prior to editing or playback. In all these

applications, the events can happen in any order, and the results of these events can arrive for processing in any order, but there must be sufficient timing information to correctly recreate the actual order.

We recommend you use an in-guest precision timing client because the ESXi host with NTP provides only sub-millisecond timing. Microsecond level timing requires a 3rd party client.

For more information, see "Timekeeping within ESXi" at https://blogs.vmware.com/vsphere/2018/07/timekeeping-within-esxi.html.

Carefully consider using passthrough with DirectPath I/O

VMware DirectPath I/O is a technology, available from vSphere 4.0 and higher, that leverages hardware support (Intel VT-d and AMD-Vi) to allow guests to directly access hardware devices. In the case of networking, a VM with DirectPath I/O can directly access the physical NIC instead of using a paravirtualized (vmxnet3) device. While both paravirtualized devices and DirectPath I/O can sustain high throughput (beyond 10Gbps), DirectPath I/O can additionally save CPU cycles in workloads with very high packet count per second (say > 50k/sec). However, **DirectPath I/O does not support many features such as physical NIC sharing, memory overcommitment, vMotion, and Network I/O Control**. Therefore, we recommend using DirectPath I/O only for workloads with very high packet rates, where the CPU savings from DirectPath I/O may be needed to achieve the desired performance.

Conclusion

This paper provides the benefits of media function virtualization (MFV) and shares best practices that VMware and its partners have adopted to successfully enable providers to leverage MFV solutions while reducing costs and delivering efficient and effective scale. VMware's industry-leading portfolio allows for a true hybrid-cloud solution to emerge for carriers and providers.

Appendix

esxcli - network

network nic coalesce get	Get coalesce parameters
network nic coalesce set	Set coalesce parameters on a nic
network nic cso get	Get checksum offload settings
network nic cso set	Set checksum offload settings on a nic
network nic ring current get	Get current RX/TX ring buffer parameters of a NIC
network nic ring current set	Set current RX/TX ring buffer parameters of a NIC
network nic ring preset get	Get preset RX/TX ring buffer parameters of a NIC
network nic stats get	Get NIC statistics for a given interface
network nic tso get	Get TCP segmentation offload settings
network nic tso set	Set TCP segmentation offload settings on a nic
network nic vlan stats get	List VLAN statistics for active VLAN's on the NIC
network nic vlan stats set	Enable/disable VLAN statistics collection on the NIC
network port filter stats get	Filter statistics for a given port
network port stats get	Packet statistics for a given port
network sriovnic list	This command will list the SRIOV Enabled NICs (PFs) currently installed and loaded on the system
network vswitch standard portgroup list	List all of the port groups currently on the system
network nic list	List the Physical NICs currently installed and loaded on the system

VMkernel system information shell (vsish) commands

vsish -e get /net/portsets/vSwitch0/ports/portid/vmxnet3/rxSummary

vsish -e get /net/portsets/vSwitch0/ports/portid/stats

vsish -e get /net/pNics/vmnic4/stats

Related knowledgebase articles and docs

- Enabling and disabling native drivers in ESXi 6.5 (2147565)
- VMXNET3 resource considerations on a Linux virtual machine that has vSphere DirectPath I/O with vMotion enabled (2058349)
- RSS and multiqueue support in Linux driver for VMXNET3 (2020567)
- Troubleshooting network performance issues in a vSphere environment (1004087)
- Large packet loss at the guest operating system level on the VMXNET3 vNIC in ESXi (2039495)
- Poor network performance or high network latency on Windows virtual machines (2008925)
- Enable or disable LRO for all VMkernel adapters on an ESXi host (docs.vmware.com)
- The output of esxtop shows dropped receive packets at the virtual switch (1010071)
- Understanding TCP Segmentation Offload (TSO) and Large Receive Offload (LRO) in a VMware environment (2055140)
- SR-IOV support status FAQ (2038739)