RoCE SR-IOV Setup and Performance Study on vSphere 7.x

Using NVIDIA ConnectX adapter cards for HPC and machine learning workloads on vSphere—September 7, 2022

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

Table of Contents

1 Introduction ... 3

2 Configuration Workflow ... 4

2.1 BIOS configuration ... 6

2.2 ESXi configuration ... 7

2.2.1 Install Mellanox firmware tools and the first reboot ... 7

2.2.2 Install native Mellanox ESXi driver and then apply the second reboot ... 9

2.2.3 Configure RoCE SR-IOV on firmware and ESXi driver and the third reboot ... 10

2.3 The vSphere Client configuration ... 11

2.3.1 Create vSphere Distributed Switch (VDS) for SR-IOV communication on the cluster ... 13

2.3.2 Assign a VF as an SR-IOV passthrough adapter to a virtual machine ... 15

2.4 Guest OS configuration ... 16

3 Functionality Evaluation ... 19

3.1 ibverbs utility test ... 19

3.2 OSU Benchmark Test ... 22

4 Performance Study of HPC applications ... 24

4.1 OpenFOAM ... 24

4.2 WRF ... 25

4.3 LAMMPS ... 26

4.4 GROMACS ... 27

4.5 NAMD ... 28

5 Summary ... 30

6 References ... 30

1 Introduction

As three key Ethernet features have emerged—namely, RDMA over Converged Enhanced Ethernet (RoCE) v2, Priority-based Flow Control (PFC), and Enhanced Transmission Selection (ETS)—high speed Ethernet has become a more attractive option for the high-performance computing (HPC) network.

Currently, vendors like Mellanox, Intel, and HPE Cray have shifted heavily to Ethernet, and more than half of the TOP500 systems are using it [1][2]. Meanwhile, Single Root I/O Virtualization (SR-IOV) can efficiently share a physical device as multiple lightweight Virtual Functions (VFs), providing isolation for safety concerns and achieving near bare metal performance. The VMware Office of the CTO (OCTO) team recognized this trend—and the need of customers and HPC administrators to use RoCE and SR-IOV together to build intelligent infrastructure to run HPC/ML workload in their multi-cloud environments.

This is the second document in a series of technical guides. Here, we walk through the steps to enable RoCE SR-IOV on a dual-port Mellanox ConnectX-5 VPI adapter card in VMware vSphere 7.x. We cover the steps from the BIOS, ESXi, and the vSphere Client to the functionality test on the VM guest operating system. We also introduce how to use the vHPC toolkit [6], an open-source tool developed by VMware, to speed up the deployment of an HPC cluster in vSphere. Some of the steps are referenced from VMware documentation [7][8] on how to configure a VM to use SR-IOV devices and NVIDIA documentation [3][4][5][9] on how to set up and configure the firmware and driver of Mellanox ConnectX adapter cards in a vSphere environment. Finally, we present a performance study that uses five HPC applications across multiple vertical domains. We conclude that a virtual HPC cluster can perform nearly as well as a bare metal HPC cluster.

2 Configuration Workflow

Although Ethernet is broadly used in cloud computing, high-speed Ethernet was rarely used in the Top500 list before 2015 [12], when three key improvements emerged. The RoCE v2 (or Routable RoCE, RRoCE) protocol changes packet encapsulation by including IP and UDP headers to enable L2 network and L3 routing, thus overcoming the limitation of RoCE v1 being bound to a single broadcast domain (VLAN). Additionally, PFC and ETS provide the lossless computing fabric that high-performance RDMA communication requires. PFC allows the fabric to pause flows belonging to selected priority levels to avoid congestion. ETS is a quality-of-service (QoS) approach that uses a weighted round-robin algorithm to share bandwidth between priority levels, so it avoids the strict bandwidth allocation or prioritization of other QoS approaches. As these technologies mature, high-speed Ethernet becomes the favored platform for building HPC networks and existing software stacks in a cloud ecosystem.

Figure 1 illustrates the general idea of the RoCE SR-IOV configuration on two VMs. We first enable SR-IOV functionality on the physical adapter card, then attach the VFs to VMs in vSphere, and use the virtual distributed switch (VDS) for the communication between them. Different from the IB SR-IOV document, the only change here is the physical connection. We use a high-performance Ethernet switch—the Dell PowerSwitch S5232F 100GbE—to connect servers. We will use the example of 16 VMs on 16 servers for benchmarking and performance testing.

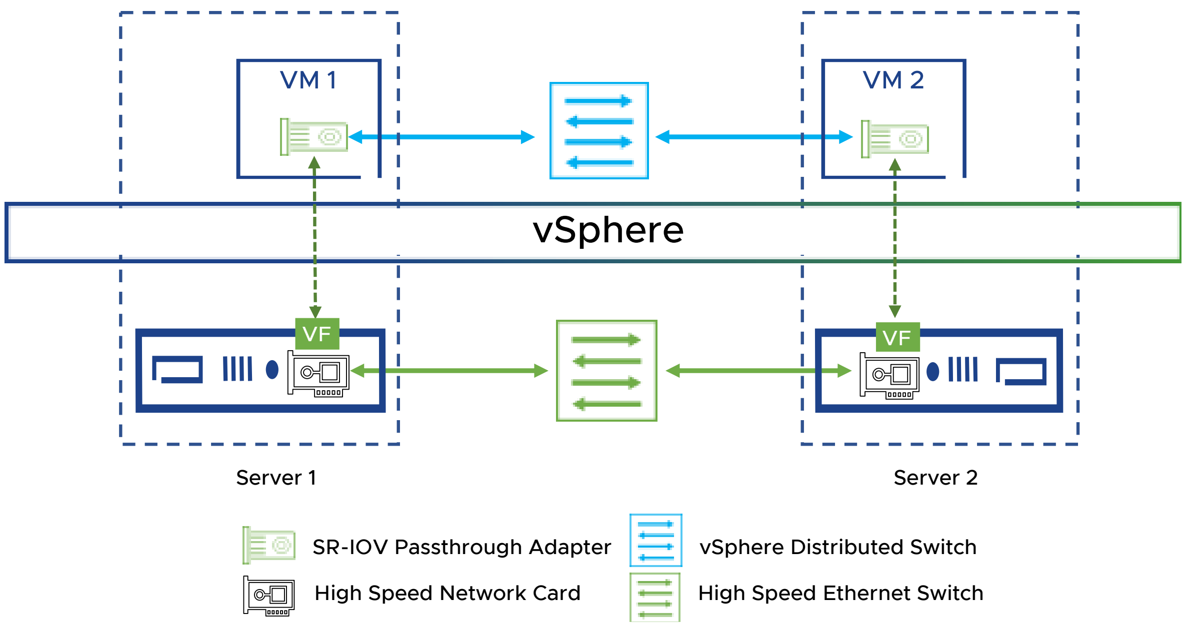


Figure 1: Illustration of the RoCE SR-IOV configuration.

Figure 2 presents the flow chart to enable RoCE SR-IOV. The configuration workflow is divided into four stages: the BIOS, ESXi, the vSphere Client and the VM guest.

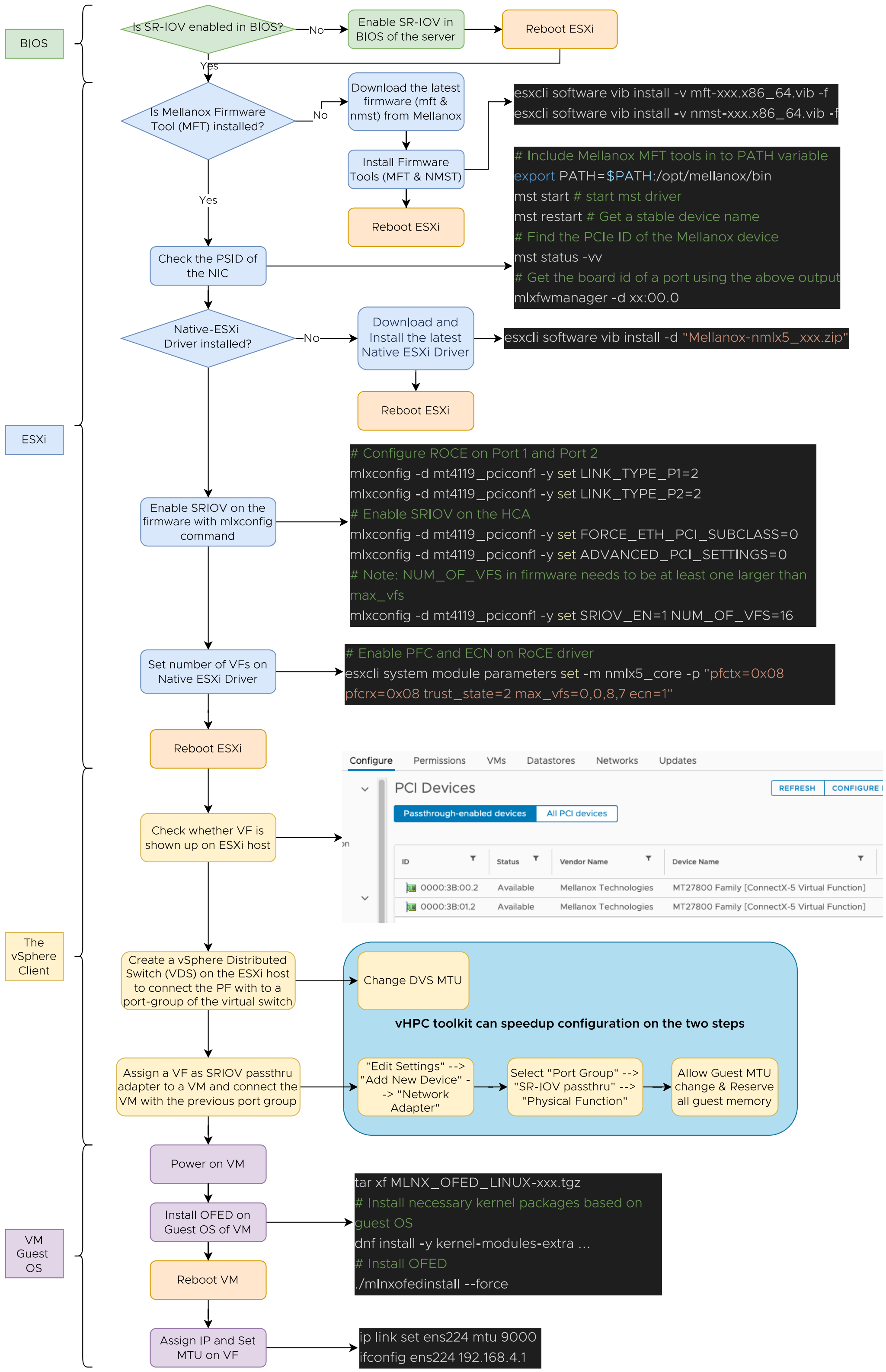


Figure 2: Flow chart to enable RoCE SR-IOV on NVIDIA Mellanox ConnectX-5 in ESXi 7.x.

2.1 BIOS configuration

For Dell servers, we enable the processor settings **Virtualization Technology** and **SR-IOV Global** on the BIOS in the iDRAC portal in **Figure 3**. If they are not set, changes will not take effect until after a reboot. You can take similar steps on other out-of-band management platforms, such as iLO on HPE servers, and so on. Refer to the specific documentation of your different server vendors.

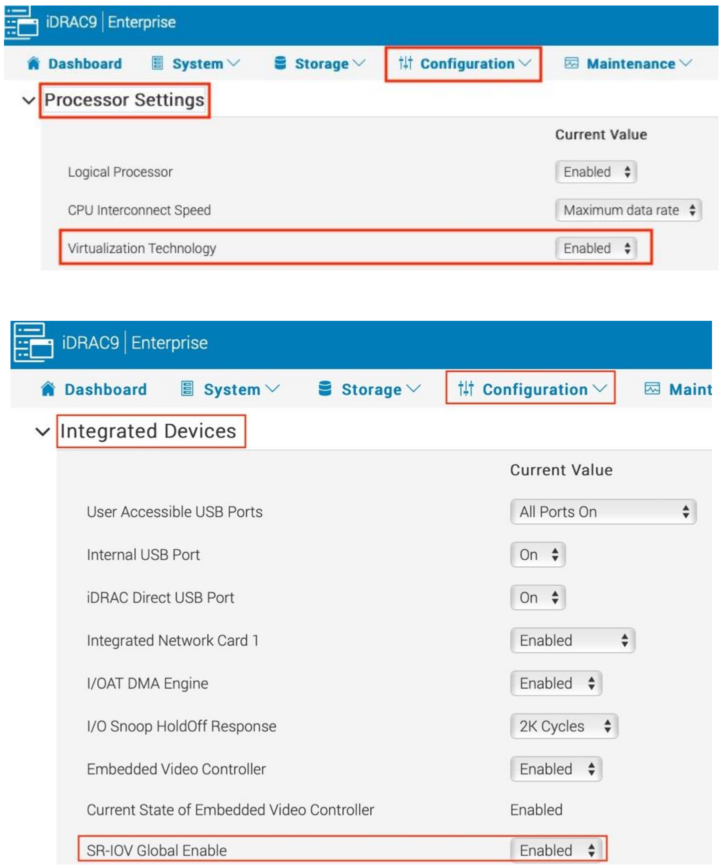


Figure 3: Enable virtualization and SR-IOV Global in BIOS of iDrac in Dell R740.

> **Best Practice:** For HPC workloads, **Performance Per Watt (OS)** is the recommended system power profile setting (**Figure 4**). Again, HPE servers have a similar profile setting. Then, when we get to the step where we can set the ESXi power management, we will choose **High Performance**.

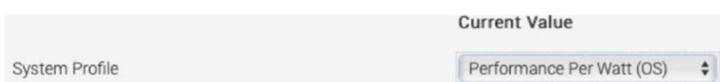


Figure 4: Power profile in BIOS.

2.2 ESXi configuration

After enabling SR-IOV in the BIOS, we need to configure the adapter card in ESXi by first configuring the firmware and then the native ESXi driver. We also need to download and install any missing software. At most, three reboots of the ESXi host are required in this section. (Most of the steps in this section refer to NVIDIA ConnectX-4 onwards NICs NATIVE ESXi Driver for VMware vSphere User Manual v4.19.71.1 [5].)

2.2.1 Install Mellanox firmware tools and the first reboot

First, we need to check whether the latest firmware tools [3] are installed on our ESXi host. **Figure 5** shows the two packages included: NMST and MFT. If they are installed, skip to the next section, 2.2.2.

> **Best Practice:** We recommend downloading them [16] to the vSAN datastore or Network File System (NFS) so that all ESXi hosts in the cluster can conveniently access these files for large-scale deployment.

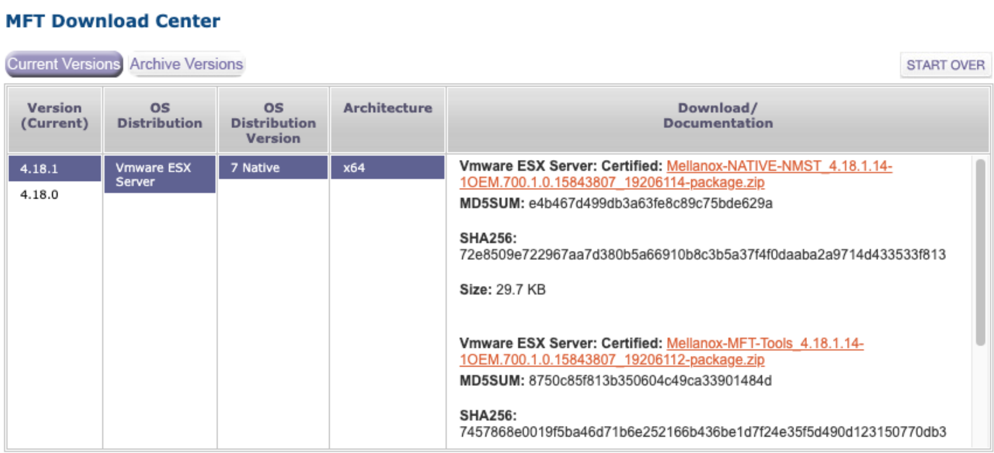


Figure 5: Download firmware tools from the NVIDIA Mellanox website.

After extracting the two zip files, we use the following commands to install them on the ESXi in # Reboot *the host for the first time*

Figure 6. We also add the installation directory to the `$PATH` variable for convenience in the remaining steps. When the installation completes, we must reboot the host for the first time.

```
# Install MFT and NMST
[esxi]$ esxcli software vib install -v mft-xxx.x86_64.vib -f
[esxi]$ esxcli software vib install -v nmst-xxx.x86_64.vib -f
# Best Practice: Add installation directory to PATH variable
[esxi]$ echo 'export PATH=$PATH:/opt/mellanox/bin' >> etc/profile.local
# Reboot the host for the first time
```

Figure 6: Commands to install firmware tools.

After the reboot, we can use the firmware tools to check whether they function well—for example, by querying the status, firmware version, and board id, and then updating the firmware online.

```
# start mst driver
[esxi]$ mst start

# Restart mst to get a stable device name
[esxi]$ mst restart

# Find the PCIe ID of the Mellanox device
[esxi]$ mst status -vv
PCI devices:
------------
DEVICE_TYPE             MST                           PCI
ConnectX4LX(rev:0)      mt4117_pciconf0               1a:00.0
ConnectX4LX(rev:0)      mt4117_pciconf0.1             1a:00.1
ConnectX5(rev:0)        mt4119_pciconf1               3b:00.0
ConnectX5(rev:0)        mt4119_pciconf1.1             3b:00.1

# Get the board id of a port using the output of the above command
[esxi]$ mlxfwmanager -d 3b:00.0
Querying Mellanox devices firmware ...
Device #1:
----------
  Device Type:      ConnectX5
  Part Number:      MCX556A-ECA_Ax
  Description:      ConnectX-5 VPI adapter card; EDR IB (100Gb/s) and 100GbE;
dual-port QSFP28; PCIe3.0 x16; tall bracket; ROHS R6
  PSID:             MT_0000000008
  PCI Device Name:  3b:00.0
...
  Versions:         Current        Available
     FW             16.32.1010     N/A
     PXE            3.6.0502       N/A
     UEFI           14.25.0017     N/A
  Status:           No matching image found

# If the current firmware version is lower than the online version,
# an update can be done by this command.
[esxi]$ mlxfwmanager --online -u -d 3b:00.0 -f
```

Figure 7: Commands to query the HPC NIC with firmware tools.

> **Note**: The `mst status` command in **Figure 7** discovers four devices because we have two adapter cards on our ESXi host, each with two ports. The `ConnectX4LX` card is used as a service network interface card (NIC) for connecting to vCenter and vSAN, while the `ConnectX5`, on which we intend to enable RoCE SR-IOV, is used for the HPC/ML workload. Setting up two NICs is typical for an HPC workload using vSphere [10].

Note: To query the firmware version and board ID (PSID) of our ConnectX-5 card, we use the `mlxfwmanager` command with the peripheral component interconnect express (PCIe) ID generated by `mst status`, which is `3b:00.0` in this case. We are currently using the 16.32.1010 firmware version. The PSID of the Host Channel Adapter (HCA) is `MT_0000000008`, which we compare with the latest online firmware version [17] on the NVIDIA website in Figure 8. If an online update of the firmware is not available, we can choose to manually burn the firmware with the flint command [5].

ConnectX-5 VPI/InfiniBand Firmware Download Center

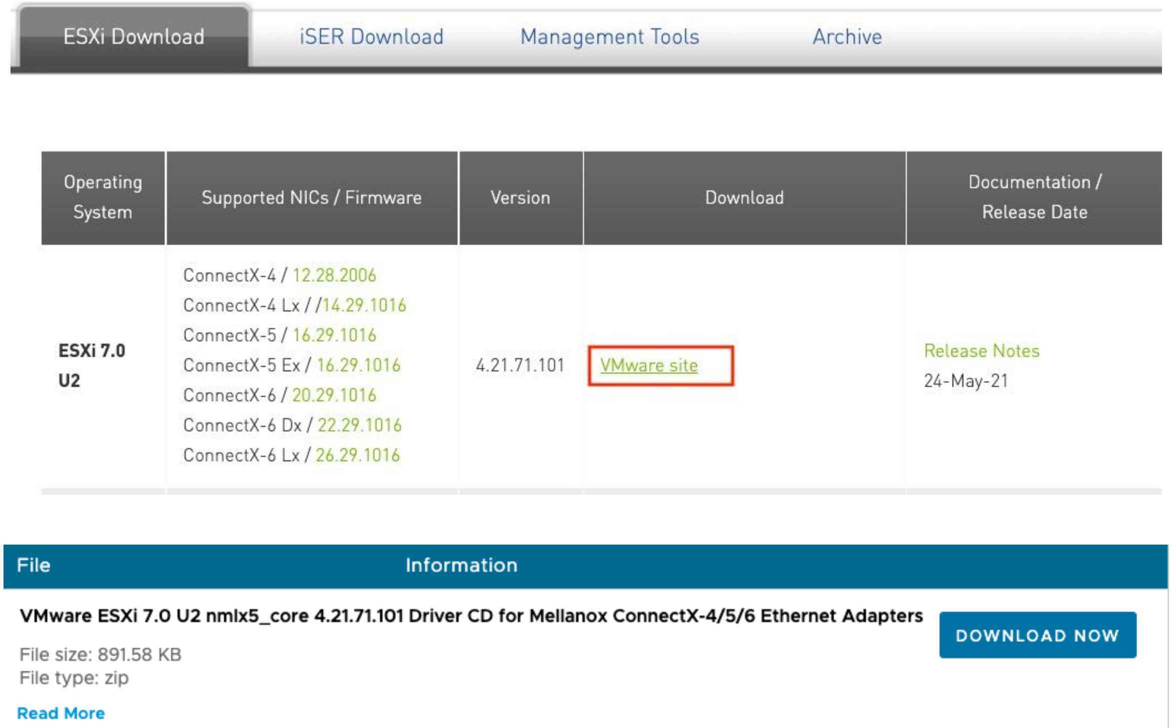
Current Versions | Archive Versions | START OVER

Version (Current)	OPN	PSID	Download/ Documentation
16.32.1010	MCX556M-ECAT-S25 MCX556A-EDAT MCX556A-ECUT MCX556A-ECAT MCX555A-ECAT MCX553Q-ECAS MCX546A-EDAN MCX545M-ECAN	MT_0000000008	**ConnectX5IB:** fw-ConnectX5-rel-16_32_1010-MCX556A-ECA_Ax-UEFI-14.25.17-FlexBoot-3.6.502 **MD5SUM:** d29fe16fffdd82eeba8ed3168b320127 **SHA256:** f7e14a93e7b111e4443a22eb0f9b5025c58857fdf244267754cf352384065bd6 **Release Date:** 5-Dec-21 **Documentation:** Release Notes EULA

Figure 8: The latest firmware version of our HPC NIC.

2.2.2 Install the native Mellanox ESXi driver and then apply the second reboot

After the firmware tools function well, we can configure the native Mellanox ESXi (nmlx) driver. If it is not installed, refer to Native ConnectX Driver for VMware ESXi Server [4]. At the time of this writing, the Mellanox website shows that the driver is defined for Ethernet only, not for InfiniBand. But, as we confirmed with the Mellanox support team, the 4.21.71.101 version can be used to support IB SR-IOV. We just need to treat the IB device as an Ethernet device so that vSphere can detect it. This webpage then directs to a VMware site to download the nmlx_core driver.



ESXi Download | iSER Download | Management Tools | Archive

Operating System	Supported NICs / Firmware	Version	Download	Documentation / Release Date
ESXi 7.0 U2	ConnectX-4 / 12.28.2006 ConnectX-4 Lx / /14.29.1016 ConnectX-5 / 16.29.1016 ConnectX-5 Ex / 16.29.1016 ConnectX-6 / 20.29.1016 ConnectX-6 Dx / 22.29.1016 ConnectX-6 Lx / 26.29.1016	4.21.71.101	VMware site	Release Notes 24-May-21

File	Information
VMware ESXi 7.0 U2 nmlx5_core 4.21.71.101 Driver CD for Mellanox ConnectX-4/5/6 Ethernet Adapters File size: 891.58 KB File type: zip Read More	DOWNLOAD NOW

Figure 9: Download the native ESXi driver.

> Best Practice: We also recommend downloading the nmlx driver to a location in vSAN or NFS for the same reason as before.

Now we use the following commands to install the driver and reboot the ESXi host the second time.

```
# Install Native Mellanox ESXi Driver (nmlx)
[esxi]$ esxcli software vib install -d "Mellanox-nmlx5_xxx.zip"
# Reboot the host for the second time
```

Figure 10: Commands to install the nmlx ESXi driver and reboot the host.

2.2.3 Configure RoCE SR-IOV on firmware and ESXi driver, and the third reboot

After the second reboot, we can enable SR-IO IB on the firmware and the native ESXi driver using the commands in Figure 11.

```
1  # Configure RoCE on the firmware of Port 1 and Port 2 of ConnectX-5
2  [esxi]$ mlxconfig -d mt4119_pciconf1 -y set LINK_TYPE_P1=2
3  [esxi]$ mlxconfig -d mt4119_pciconf1 -y set LINK_TYPE_P2=2
4
5  # Enable SRIOV on the firmware of ConnectX-5
6  # Clear Advanced PCI setting
7  [esxi]$ mlxconfig -d mt4119_pciconf1 -y set ADVANCED_PCI_SETTINGS=0
8  # Note: "NUM_OF_VFS" in firmware needs to be at least one larger than "max_vfs"
9  [esxi]$ mlxconfig -d mt4119_pciconf1 -y set SRIOV_EN=1 NUM_OF_VFS=16
10
11 # Set number of VFs on Native ESXi Driver
12 [esxi]$ esxcli system module parameters set -m nmlx5_core -p "pfctx=0x08 pfcrx=0x08
   trust_state=2 max_vfs=0,0,8,7 ecn=1"
13
14 # Reboot the host for the third time
```

Figure 11: Commands to enable RoCE SR-IOV the firmware and nmlx driver.

In lines 2 and 3, we set both ports to be Ethernet by setting `LINK_TYPE_P1=2` and `LINK_TYPE_P2=2`.

In line 7, we set `ADVANCED_PCI_SETTINGS=0` since RoCE doesn't need to enable it.

In line 9, we set `NUM_OF_VFS=16` to create 16 VFs. Note that this number should be at least one larger than the sum of VFs created by `max_vfs` in the following command since at least one VF should be reserved for the PF.

In line 12, we set `max_vfs="0,0,8,7"`, where each number is the number of VFs enabled on each port of this host. Since we have two ConnectX cards on the ESXi host and each NIC has two ports, we need to set four numbers here. We don't intend to enable SR-IOV on the ConnecX-4 card, so we set the first two numbers to zero. But we would like to create 8 VFs on the first port and 7 VFs on the second port on our ConnectX-5, so we set the last two numbers to 8 and 7.

We also follow the Mellanox user manual [5] to enable PFC and ECN on the RoCE ports. The parameters `pfctx` and `pfcrx` are specified per host to enable PFC so that Global Pause will be disabled. The bitmap value "0x08" enables lossless applications only on priority 3. The `ecn=1` is set by default, but we explicitly set it to clarify that the feature is turned on.

2.3 The vSphere Client configuration

After configuring SR-IOV on the firmware and driver on the ESXi, you can see the VFs by logging into vSphere, going to the **Hosts and Clusters** view, and selecting the relevant ESXi server, followed by **Configure** → **Networking** → **Physical Adapters** → the vmnic showing 100Gbps → **Edit**. (Since vSphere can detect Ethernet adapters, the RoCE adapter vmnic3 shows 100Gbps in **Figure 12**. We can check that the SR-IOV status shows as **Enabled**. We can also change the number of VFs in this step to whatever we need. Here, we set it to one VF for simplicity. (If changing the number of VFs is not responding in the vSphere Client, you can also log into the ESXi host UI followed by

Host → **Manage** → **Hardware** → **PCI Devices** → **Configure SR-IOV** → Select the physical adapters in the list → Change the **Number of virtual functions**.)

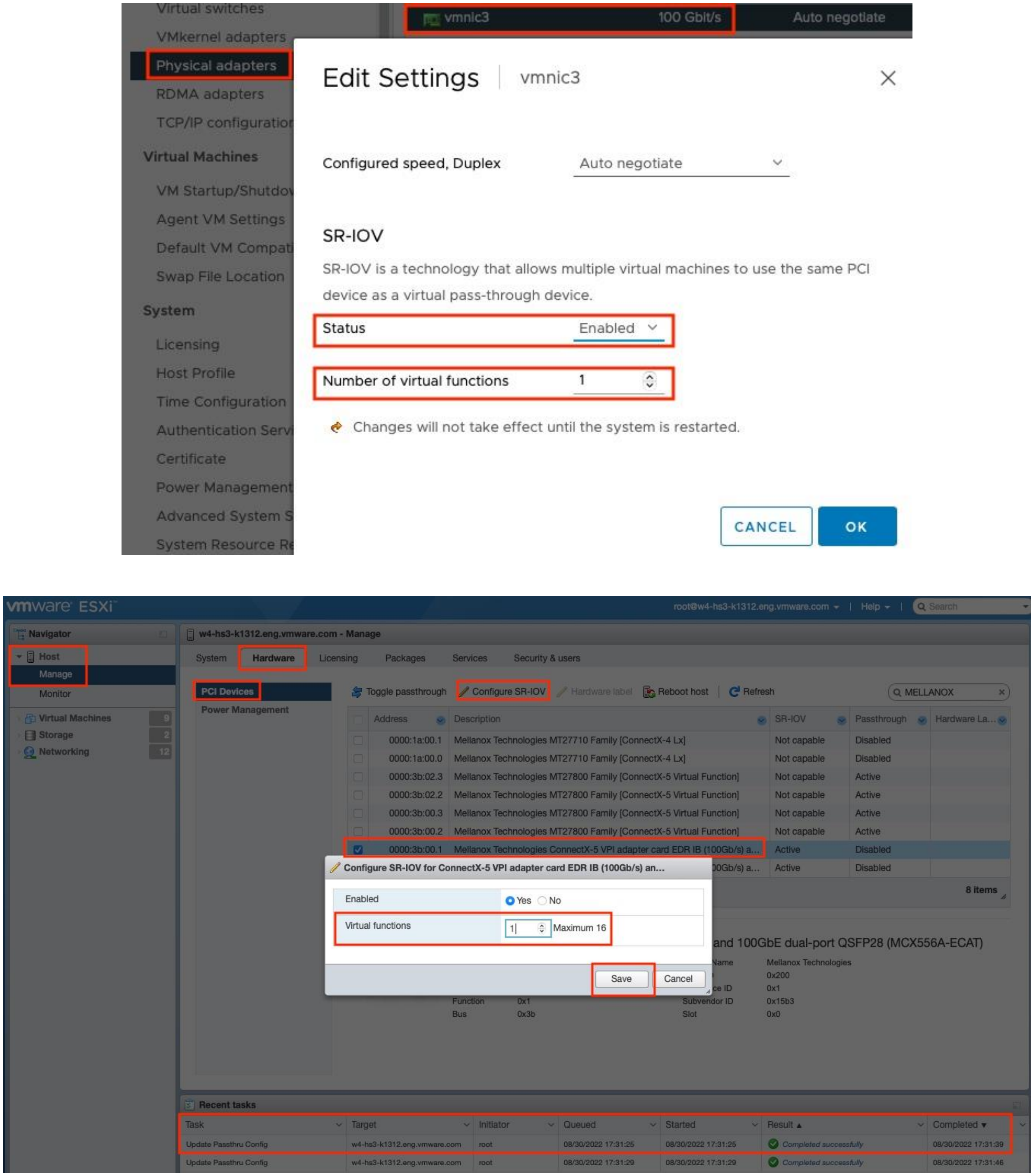


Figure 12: Edit the number of VFs in the vSphere Client or in the ESXi Host.

Next, we can view the VFs shown as **Passthrough-enabled devices** by clicking the **Configure** → **Hardware** → **PCI Devices** tab. **Figure 13** shows that we enable one VF on each of the two ports of the ConnectX-5.

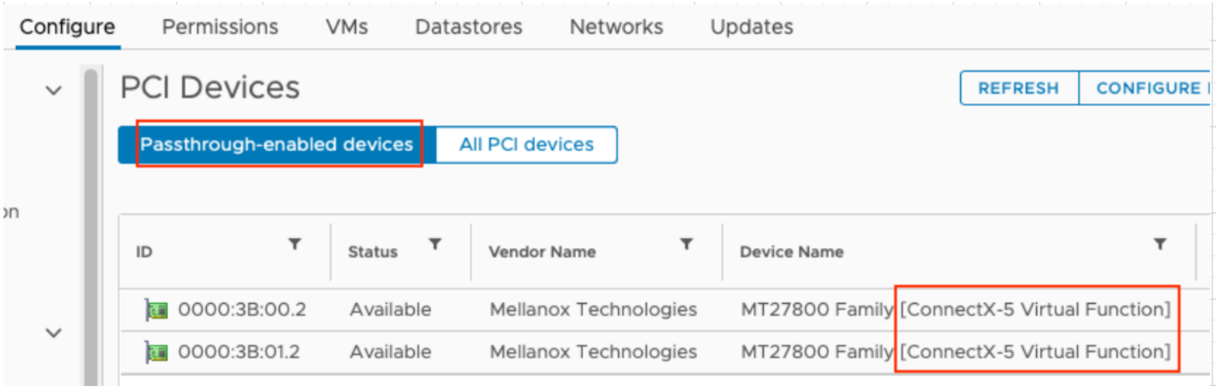


Figure 13: VFs are shown as PCI Passthrough-enabled devices in the vSphere Client.

> Best Practice: In Figure 14, we choose to use High Performance in the Power Policy by clicking the relevant ESXi server → Configure → Hardware → Overview, scrolling down to Power Management, clicking Edit Power Policy, and selecting High Performance.

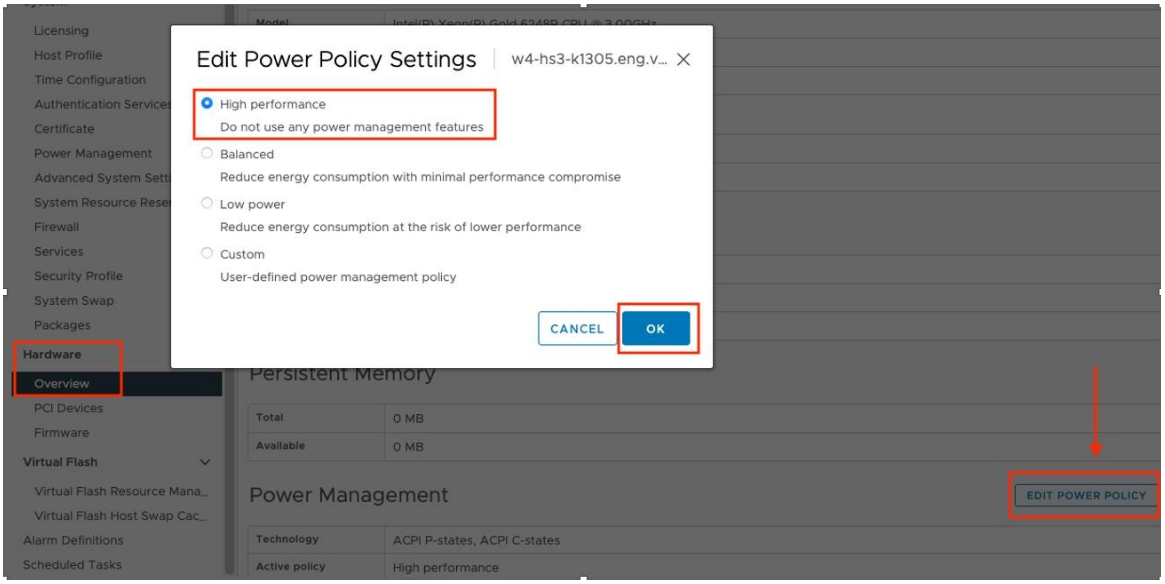


Figure 14: Choose High Performance as the power policy for the ESXi server.

Next, we will create a virtual distributed switch to connect the SR-IOV devices on hosts, then assign the VF to a VM. The logical view is shown in the top part of Figure 1.

2.3.1 Create vSphere Distributed Switch (VDS) for SR-IOV communication on the cluster

In this step, we need to create a VDS and its port group on the cluster, which connects ESXi hosts to the port group using the physical adapter. To do this manually with the vSphere Client, refer to Create a vSphere Distributed Switch [7].

Best Practice: We can use the vHPC toolkit [6] to automate the operations in Figure 15. In this case, we need to specify the following information: datacenter, the name of the created VDS and its port group, the PCIe ID of the Physical Function (PF) of our ConnectX-5 card, the name of the physical adapter, and the ESXi host list.

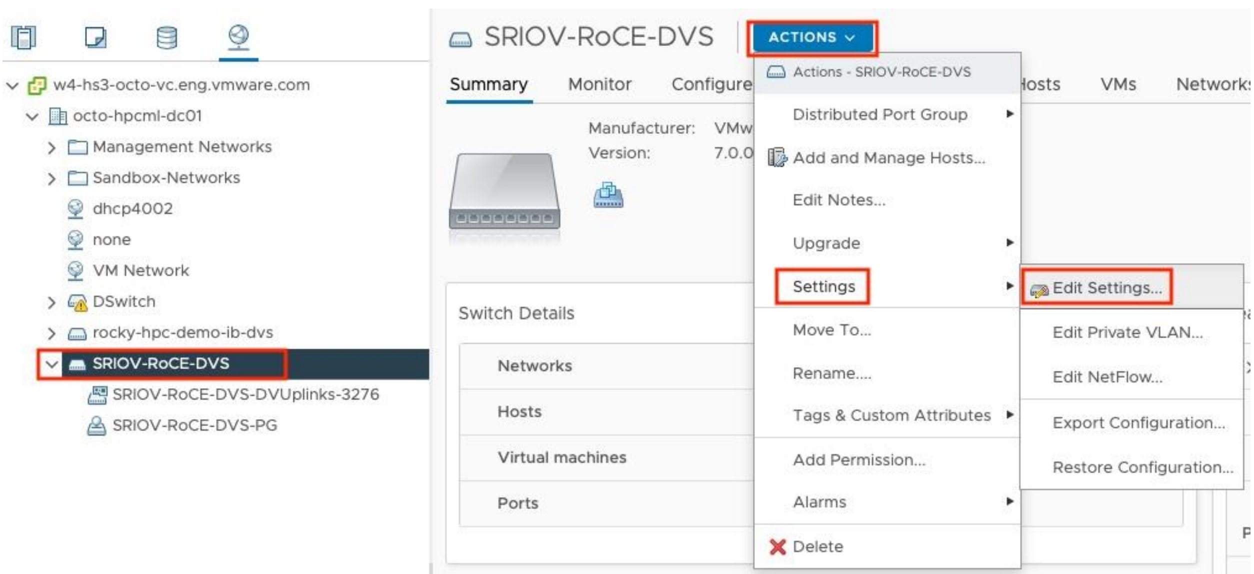
```
[vhpc]$ vHPC_BIN_DIR=/user_dir/vhpc-toolkit/bin
[vhpc]$ source /user_dir/vhpc-toolkit/venv/bin/activate
[vhpc]$ cd $vHPC_BIN_DIR

[vhpc]$ dc="octo-hpcml-dc01"
[vhpc]$ sriov_dvs="SRIOV-ROCE-DVS"
[vhpc]$ sriov_dvs_pg="SRIOV-ROCE-DVS-PG"
[vhpc]$ PF_ID="0000:3b:00.1"
[vhpc]$ physical_adapter="vmnic3"
[vhpc]$ host_list="${esxi_host_name0} ${esxi_host_name1} ..."

# Use vhpc_toolkit to create a VDS and its port group,
# then connect ESXi hosts to the port group with the physical adapter
[vhpc]$ ./vhpc_toolkit dvs --create --name $sriov_dvs --datacenter $dc --host $host_list
--pnic $physical_adapter --port_group $sriov_dvs_pg
```

Figure 15: Create a VDS using the vHPC toolkit.

Next, we change the Maximum Transition Unit (MTU) of the VDS from its default of 1500 to 9000 to meet the network performance requirement for HPC workloads (**Figure 16**).



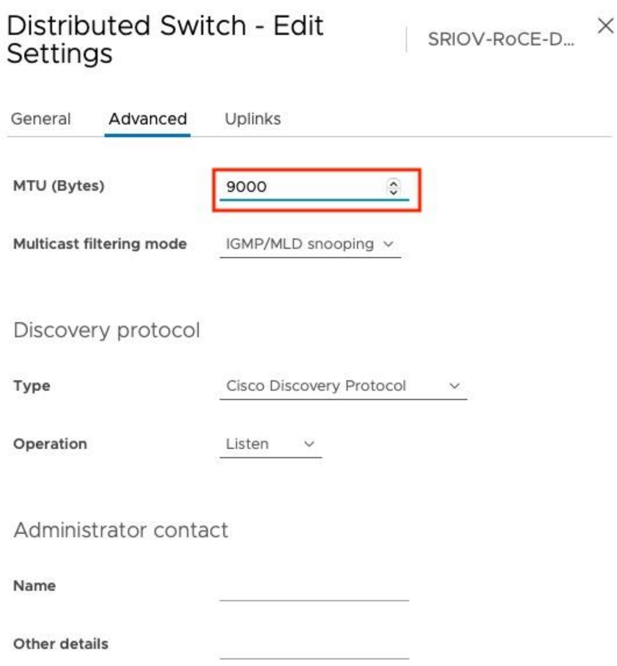


Figure 16: Change MTU=9000 on the VDS.

Using a Virtual Standard Switch (VSS) on each host is another option to achieve the same goal in this step. But we prefer VDS since it provides a single management point and prevents configuration drift.

2.3.2 Assign a VF as an SR-IOV passthrough adapter to a virtual machine

In this step, we assign a VF as an SR-IOV passthrough adapter to a VM. Figure 17 shows this operation by following the steps in Assign a Virtual Function as SR-IOV Passthrough Adapter to a Virtual Machine [8] by using the vSphere Client. Note that the VM requires reserved memory, and **Allow the Guest MTU** is changed to **Allow** SR-IOV.

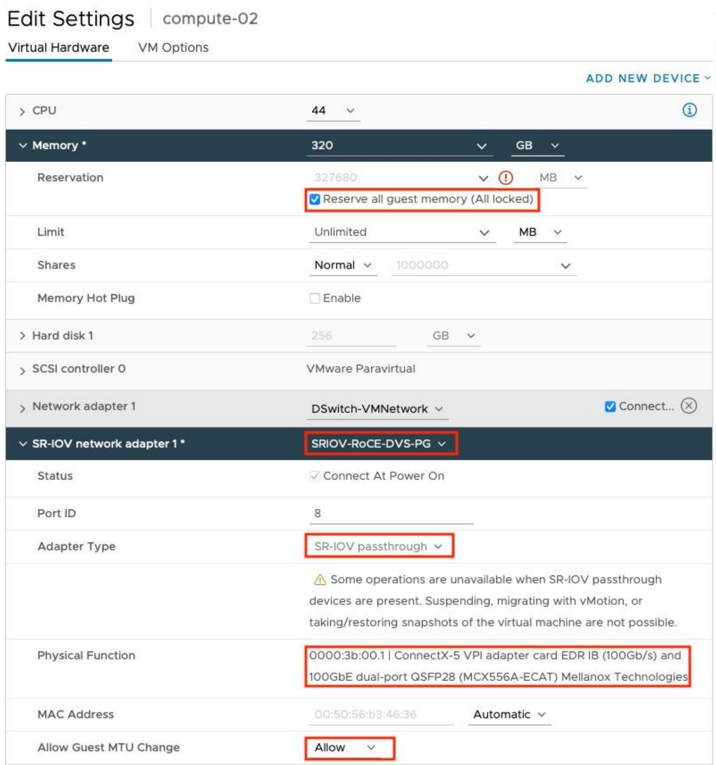


Figure 17: Use the vSphere Client to assign a VF as an SR-IOV passthrough adapter to a VM.

> **Best Practice:** We can use the vHPC toolkit to speed up the operation as shown in Figure 18.

```
[vhpc]$ sriov_dvs="SRIOV-ROCE-DVS"
[vhpc]$ sriov_dvs_pg="SRIOV-ROCE-DVS-PG"
[vhpc]$ PF_ID="0000:3b:00.1"
[vhpc]$ vm_name="compute-02"

# Assign a VF as a SR-IOV Passthrough Adapter to a VM
[vhpc]$ ./vhpc_toolkit sriov --add --vm $vm_name --sriov_port_group $sriov_dvs_pg --dvs_name
$sriov_dvs --pf $PF_ID
```

Figure 18: Use the vHPC toolkit to assign a VF as an SR-IOV passthrough adapter to a VM.

2.4 Guest configuration

Now, we can power on VM. If Mellanox's version of OpenFabrics Enterprise Distribution (OFED) is not installed on the VM, download it from Guest OFED Download link [9]. **Figure 19** shows the command to install OFED. A reboot of the VM is required after the installation.

```
[guest OS]$ tar xf MLNX_OFED_LINUX-xxx.tgz
# For RHEL, install necessary dependent packages
[guest OS]$ yum install -y kernel-modules-extra
# For CentOS, install necessary dependent packages
[guest OS]$ yum install -y tk
# Install the latest driver and firmware
[guest OS]$ ./mlnxofedinstall –force --add-kernel-support
# Reboot VM
```

Figure 19: Install OFED on the guest operating system.

RoCE doesn't have a subnet manager like IB, but we still need to enable PFC and set MTU=9000 on the Ethernet switch to achieve the performance requirement for HPC workloads. For the ethernet switch commands to do so, please refer to the documentation of your ethernet switch.

After OFED is installed on the guest OS, we first need to force restart the OFED driver, and then we can check its version with `ofed_info -s`. Use `ip a` to list the network interface, and we see the RoCE interface, `ens256`. Next, we set the MTU and assign an IP to it. Note that the IP should be different from existing subnets on the VM. Otherwise, an IP conflict will appear [15]. Then we can use `ibv_devinfo` or `ibstatus` to check the status of the RoCE port. **Figure 20** shows that the port `mlx5_1` is in the active state with `active_MTU: 4096` and uses Ethernet as the link layer.

```
# Load the updated OFED driver
[guest OS]$ /etc/init.d/openibd force-restart

# Check OFED version
[guest OS]$ ofed_info -s
MLNX_OFED_LINUX-5.4-3.0.3.0:

# Check interface, ens256 is the interface of RoCE
[guest OS]$ # ip a
. . .
ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
        link/ether 00:50:56:b3:49:dc brd ff:ff:ff:ff:ff:ff

# Set MTU on the RoCE interface
[guest OS] ip link set ens256 mtu 9000
# Assign an IP on the RoCE interface. Note: this IP should be different from existing subnets
on the host. Otherwise, IP conflict will appear.
[guest OS] ip addr add 192.168.xx.xx/24 dev ens256

# Check device information
[guest OS]$ ibv_devinfo
hca_id: mlx5_1
    transport:          InfiniBand (0)
    fw_ver:             16.32.1010
    node_guid:          0050:56ff:feb3:9926
    sys_image_guid:         0c42:a103:00d3:9846
    vendor_id:          0x02c9
    vendor_part_id:         4120
    hw_ver:             0x0
    board_id:           MT_0000000008
    phys_port_cnt:          1
        port:   1
            state:          PORT_ACTIVE (4)
            max_mtu:        4096 (5)
            active_mtu:     4096 (5)
            sm_lid:         0
            port_lid:       0
            port_lmc:       0x00
            link_layer:     Ethernet
```

Figure 20: Load OFED and check the OFED version and device information on the guest operating system.

3 Functionality Evaluation

In this section, we will evaluate the functionality of the RoCE SR-IOV that we configured using two tests: ibverbs utility test and the OSU microbenchmark suite.

Table 1 describes our testbed hardware, BIOS settings, and the firmware and driver versions used in the above SR-IOV configuration. These versions were the latest available when we conducted these experiments. We recommend that you consult your product vendor and use the appropriate versions.

Table 1: Testbed Details of the virtual clusters.

<table>
<tr><th colspan="2">Environment</th><th>Bare Metal</th><th>Virtual Machine</th></tr>
<tr><td rowspan="7">Hardware</td><td>Server</td><td colspan="2">PowerEdge R740 vSAN ReadyNode</td></tr>
<tr><td>Processor</td><td colspan="2">2 x Intel Xeon Gold 6248R @ 3.00GHz</td></tr>
<tr><td>HPC InfiniBand Network NIC</td><td colspan="2">100 GbE NVIDIA Mellanox ConnectX-5 VPI Dual Ports</td></tr>
<tr><td>Service Network NIC</td><td colspan="2">10/25 GbE NVIDIA Mellanox ConnectX-4 Dual Ports</td></tr>
<tr><td>HPC Ethernet Network Switch</td><td colspan="2">Dell PowerSwitch S5232F 100GbE</td></tr>
<tr><td>Service Network Switch</td><td colspan="2">Dell PowerSwitch S5248F-ON</td></tr>
<tr><td>ConnectX-5 firmware</td><td colspan="2">16.32.1010</td></tr>
<tr><td rowspan="3">BIOS</td><td>Power Profile</td><td colspan="2">Performance Per Watt (OS controlled)</td></tr>
<tr><td>Hyperthreading</td><td colspan="2">Enabled</td></tr>
<tr><td>Virtualization</td><td colspan="2">Intel VT-d Enabled</td></tr>
<tr><td colspan="2">Cores</td><td>All 48 cores used</td><td>44 vCPU reserved, High Latency sensitivity</td></tr>
<tr><td colspan="2">Memory</td><td>24 * 16GB RDIMM,
All 384GB used</td><td>144GB reserved for the VM</td></tr>
<tr><td rowspan="5">Operating system</td><td>Host</td><td>RHEL 8.1</td><td>VMware vSphere 7.0 U2,
Guest: RHEL 8.1</td></tr>
<tr><td>Power Policy</td><td>Default</td><td>High Performance</td></tr>
<tr><td>Mellanox Firmware Tools</td><td colspan="2">MFT & NMST 4.18.1</td></tr>
<tr><td>Native Mellanox (NMLX) Driver</td><td>N/A</td><td>4.21.71.101</td></tr>
<tr><td>OFED</td><td colspan="2">5.4-3.0.3.0</td></tr>
<tr><td rowspan="6">Build Libraries</td><td>Compiler</td><td colspan="2">GCC 9.3.0</td></tr>
<tr><td>MPI</td><td colspan="2">OpenMPI 4.1.2</td></tr>
<tr><td>UCX</td><td colspan="2">1.12.0</td></tr>
<tr><td>Intel One API / Cluster Checker</td><td colspan="2">2022.2 / 2021 Update 6 (build 20220318)</td></tr>
<tr><td>Spack</td><td colspan="2">0.17.1</td></tr>
<tr><td>OSU Microbenchmark</td><td colspan="2">5.7.1</td></tr>
</table>

3.1 ibverbs utility test

We first use the ibverbs bandwidth and latency utility test to evaluate RoCE performance between two VMs in **Figure 21** and **Figure 22**.

```
[root@compute-02 ~]$ ib_send_bw -a –report_gbits -d mlx5_1 compute-03
---------------------------------------------------------------------------------------
                    Send BW Test
 Dual-port       : OFF      Device         : mlx5_1
 Number of qps   : 1        Transport type : IB
 Connection type : RC       Using SRQ      : OFF
 PCIe relax order: ON
 ibv_wr* API     : ON
 TX depth        : 128
 CQ Moderation   : 100
 Mtu             : 4096[B]
 Link type       : Ethernet
 GID index       : 3
 Max inline data : 0[B]
 rdma_cm QPs     : OFF
 Data ex. Method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0000 QPN 0x00e9 PSN 0xba8f4c
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:04:02
 remote address: LID 0000 QPN 0x00e9 PSN 0x296312
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:04:03

 #bytes   #iterations  BW peak[Gb/sec]  BW average[Gb/sec]  MsgRate[Mpps]
 2        1000         0.085666         0.082800            5.175012
 4        1000         0.17             0.17                5.228020
 8        1000         0.35             0.35                5.443561
 16       1000         0.70             0.70                5.437884
 32       1000         1.40             1.27                4.955672
 64       1000         2.79             2.78                5.436817
 128      1000         5.58             5.57                5.443363
 256      1000         11.17            10.25               5.003909
 512      1000         22.21            22.16               5.410438
 1024     1000         43.78            43.69               5.333671
 2048     1000         69.26            64.76               3.952334
 4096     1000         91.23            88.17               2.690763
 8192     1000         93.85            93.84               1.431817
 16384    1000         95.31            95.30               0.727082
 32768    1000         96.56            96.56               0.368339
 65536    1000         96.64            96.63               0.184312
 131072   1000         96.73            96.73               0.092252
 262144   1000         96.40            96.39               0.045963
 524288   1000         94.98            94.98               0.022644
 1048576  1000         95.49            95.20               0.011349
 2097152  1000         95.78            95.51               0.005693
 4194304  1000         96.06            95.85               0.002857
 8388608  1000         96.33            96.15               0.001433
---------------------------------------------------------------------------------------
```

Figure 21: ibverbs bandwidth test.

Figure 21 shows that the `ib_send_bw` bandwidth of 95Gbps on larger packet sizes is close to the line rate of 100Gbps of the ConnectX-5 adapter card, which indicates that RoCE SR-IOV is configured correctly.

```
[root@compute-02 ~]$ ib_send_lat -a --report_gbits -d mlx5_1 compute-03
---------------------------------------------------------------------------------------
                    Send Latency Test
 Dual-port       : OFF      Device         : mlx5_1
 Number of qps   : 1        Transport type : IB
 Connection type : RC       Using SRQ      : OFF
 PCIe relax order: ON
 ibv_wr* API     : ON
 TX depth        : 1
 Mtu             : 4096[B]
 Link type       : Ethernet
 GID index       : 3
 Max inline data : 236[B]
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0000 QPN 0x00ea PSN 0x8b939c
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:04:02
 remote address: LID 0000 QPN 0x00ea PSN 0x6214c1
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:04:03
---------------------------------------------------------------------------------------
 #bytes  #iterations t_min[usec] t_max[usec] t_typical[usec] t_avg[usec] t_stdev[usec] 99%[usec]   99.9%[usec]
 2       1000        1.98        5.08        2.04            2.06        0.11          2.28        5.08
 4       1000        1.99        4.41        2.03            2.04        0.10          2.20        4.41
 8       1000        1.98        4.39        2.03            2.04        0.12          2.17        4.39
 16      1000        1.99        3.79        2.04            2.05        0.07          2.18        3.79
 32      1000        2.02        6.10        2.07            2.08        0.10          2.23        6.10
 64      1000        2.09        4.55        2.14            2.15        0.12          2.26        4.55
 128     1000        2.12        4.14        2.18            2.19        0.12          2.42        4.14
 256     1000        2.56        4.89        2.62            2.63        0.12          2.82        4.89
 512     1000        2.63        4.91        2.68            2.71        0.13          3.33        4.91
 1024    1000        2.76        6.15        2.83            2.86        0.15          3.10        6.15
 2048    1000        3.01        5.15        3.11            3.14        0.15          3.63        5.15
 4096    1000        3.55        5.83        3.67            3.69        0.11          3.91        5.83
 8192    1000        4.23        5.98        4.41            4.41        0.10          4.63        5.98
 16384   1000        5.58        8.74        5.86            5.88        0.18          6.65        8.74
 32768   1000        7.73        9.22        7.98            7.98        0.11          8.23        9.22
 65536   1000        10.42       12.25       10.66           10.67       0.11          10.93       12.25
 131072  1000        15.78       18.00       16.04           16.06       0.14          16.38       18.00
 262144  1000        27.24       31.29       27.89           27.92       0.26          28.68       31.29
 524288  1000        49.03       51.14       49.92           49.94       0.33          50.87       51.14
 1048576 1000        93.67       97.21       95.21           95.25       0.60          96.73       97.21
 2097152 1000        180.64      184.85      182.59          182.56      0.71          184.18      184.85
 4194304 1000        354.99      361.64      356.99          357.02      0.76          359.06      361.64
 8388608 1000        703.53      712.74      706.95          706.86      1.06          709.32      712.74
---------------------------------------------------------------------------------------
```

Figure 22: ibverbs latency test.

Figure 22 shows that the latency of `ib_send_lat` is averaging 2 microseconds for small messages. Switch latency is documented to be 877 nanoseconds for the Dell PowerSwitch S5232F 100GbE **Error! Reference source not found.** and accounts for most of the latency for small message t ransfers. Thus, two hops between two VMs at around 2 microseconds is an acceptable value.

3.2 OSU Benchmark Test

Since our server has been configured as dual-boot (bare metal and ESXi), we use the OSU benchmark to compare the communication performance first on the 16 bare metal nodes. Then on the 16 VMs, we run the OSU multiple bandwidth/message rate benchmark (mbw_mr) (the results are in **Figure 23**) and collective benchmark (all_to_all) (the results are in **Figure 24**) with the first 2, then 4, 8, 12, and 16 VMs. Each datapoint uses an average of five runs. Since the VMs are using 44 vCPUs, for a fair comparison, we run 48 and 44 processes per node (PPN) on bare metal. The legend `VM.44.144.LatSens.RoCE.SRIOV` means the virtual machine uses 44 vCPUs, has 144GB memory, sets latency sensitivity to high, and uses RoCE SR-IOV. The legend format is also used in the later HPC application tests.

Figure 23 shows that RoCE SR-IOV can achieve near bare metal performance on all message sizes for the aggregate bandwidth/message rate test.

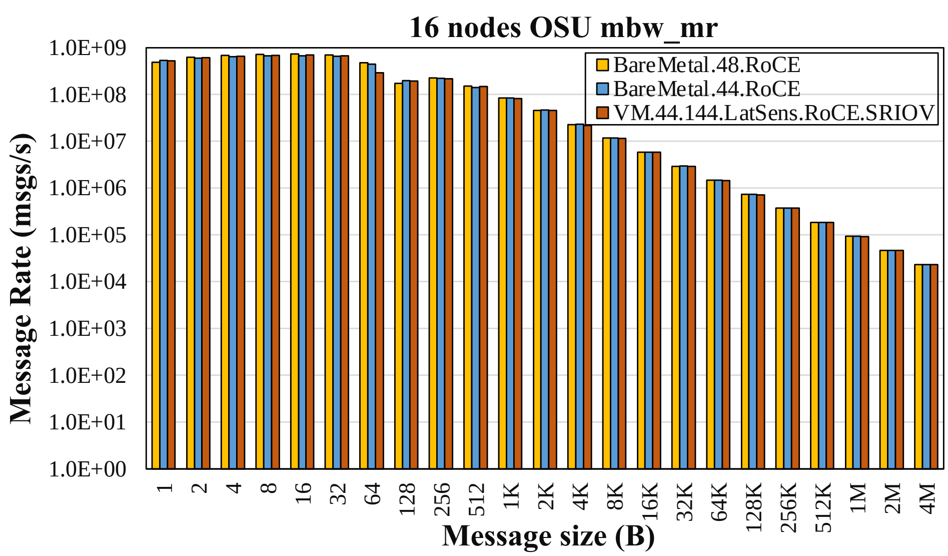


Figure 23: OSU MBW_MR Test on 16 nodes.

In **Figure 24**, we notice `BareMetal.48.RoCE` has an average of 22% and 2% higher all_to_all latency than `BareMetal.44.RoCE` and `VM.44.144.LatenSen.RoCE.SRIOV` on all message sizes, respectively. This is because more communication is involved in the 16 nodes * 48 PPN = 768 processes than in the 16 nodes * 44 PPN = 704 processes. Since the 16 nodes all_to_all test is the pure and intensive communication test, we will continue working on further improvements to narrow down the virtual overhead.

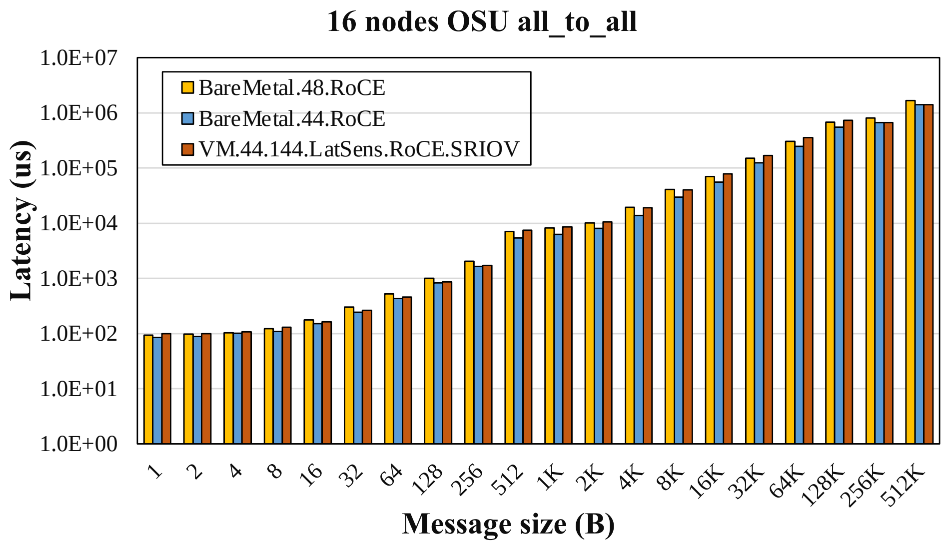


Figure 24: OSU All-to-All on 16 nodes.

4 Performance Study of HPC Applications

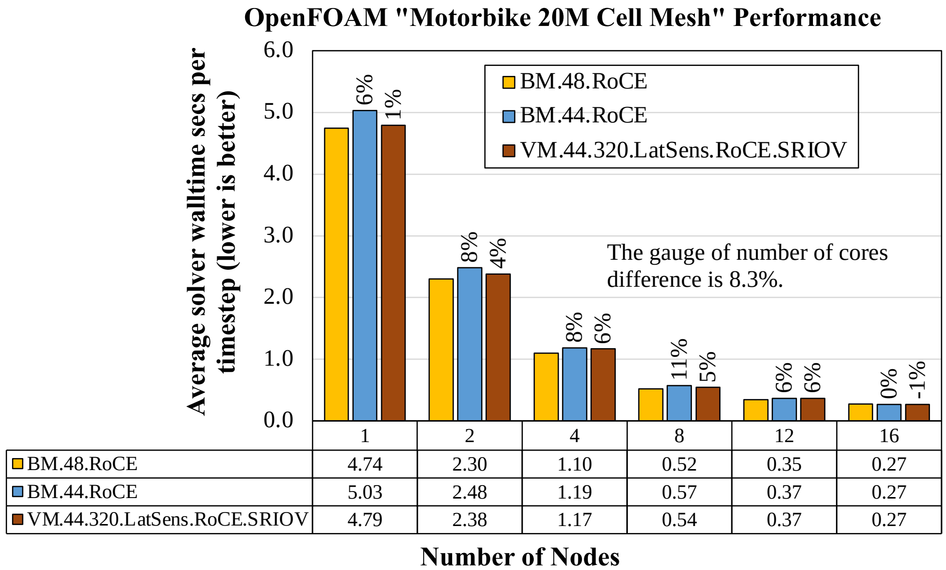
In this section, we compare the performance and strong scalability between the bare metal and virtual systems by using a range of different HPC applications across multiple vertical domains along with the benchmark datasets shown in Table 2. We use the tuning best practice in Performance Study of HPC Scale-Out Workloads on VMware vSphere 7 [10] to achieve MPI application performance running in a virtualized infrastructure that is close to the performance observed for the bare metal infrastructure. Since 48 PPN in bare metal uses 8.3% more cores than 44 PPN in virtual, we use this number as a gauge. Thus, if the performance delta falls within 8.3%, we consider this acceptable since vSphere offers other features like vSAN, vMotion, high availability, security, isolation, and more.

Table 2: Application and Benchmark Details.

Application	Vertical Domain	Benchmark Dataset	Version
OpenFOAM	Manufacturing – Computational Fluid Dynamics (CFD)	Motorbike 20M cell mesh	9
Weather Research and Forecasting (WRF)	Weather and Environment	Conus 2.5KM	3.9.1.1
Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS)	Molecular Dynamics	EAM Metallic Solid Benchmark	20210310
GROMACS	Life Sciences – Molecular Dynamics	HECBioSim BenchPEP 12M Atoms	2020.5
Nanoscale Molecular Dynamics (NAMD)	Life Sciences – Molecular Dynamics	STMV – 8M Atoms	2.14

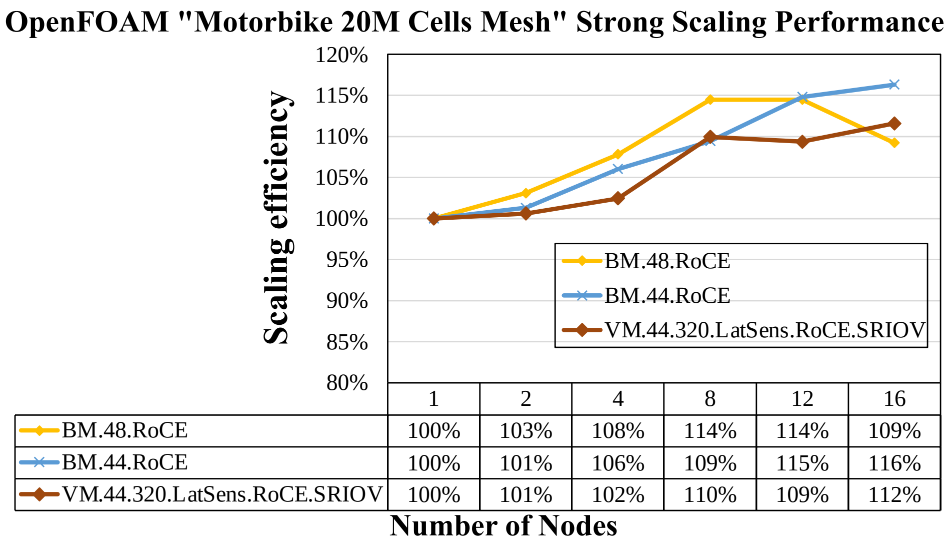
4.1 OpenFOAM

We begin with the OpenFOAM software for computational fluid dynamics. Since the 20M Motorbike benchmark needs a larger memory than 144GB to run, we expand the VM's memory to 320GB only in this HPC application. We use the `BM.48.RoCE` as the baseline, so the percentage number on the top of the columns `BM.44.RoCE` and `VM.44.320.LatSens.RoCE.SRIOV` in **Figure 25** shows the performance delta compared to the base. We can observe that VM.44 has at most a 6% delta compared to BM.48 using 4 and 12 nodes and performs better than BM.44 on all node counts.



	1	2	4	8	12	16
BM.48.RoCE	4.74	2.30	1.10	0.52	0.35	0.27
BM.44.RoCE	5.03	2.48	1.19	0.57	0.37	0.27
VM.44.320.LatSens.RoCE.SRIOV	4.79	2.38	1.17	0.54	0.37	0.27

Figure 25: OpenFOAM performance comparison between virtual and bare metal systems.

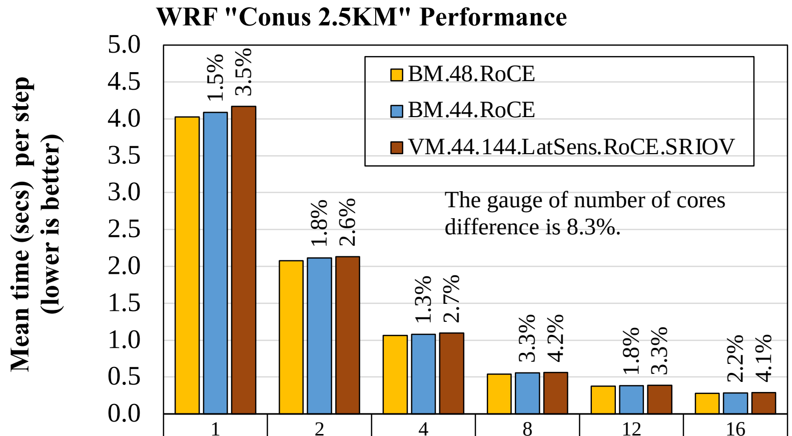


	1	2	4	8	12	16
BM.48.RoCE	100%	103%	108%	114%	114%	109%
BM.44.RoCE	100%	101%	106%	109%	115%	116%
VM.44.320.LatSens.RoCE.SRIOV	100%	101%	102%	110%	109%	112%

Figure 26: OpenFOAM strong scaling comparison between virtual and bare metal systems.

4.2 WRF

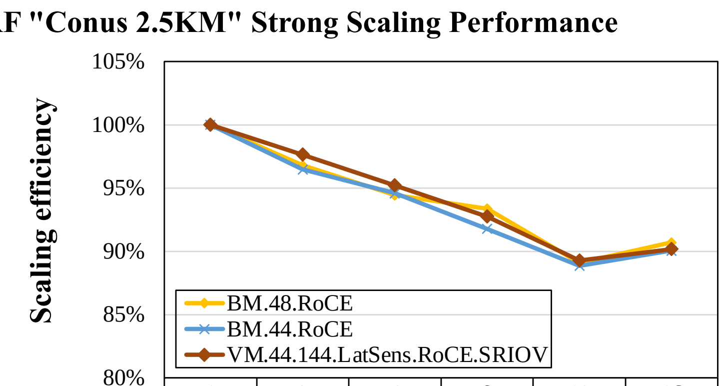
For our following example, we try the WRF model, a numerical weather prediction system used in atmospheric research and other applications. Here, we can observe that VM.44 has at most a 4.2% performance delta on 16 nodes than BM.48 (Figure 27 and Figure 28). Other node counts still present the performance delta within the 8.3% gauge.



	1	2	4	8	12	16
BM.48.RoCE	4.02	2.08	1.07	0.54	0.38	0.28
BM.44.RoCE	4.08	2.12	1.08	0.56	0.38	0.28
VM.44.144.LatSens.RoCE.SRIOV	4.17	2.13	1.09	0.56	0.39	0.29

Number of nodes

Figure 27: WRF performance comparison between virtual and bare metal systems.



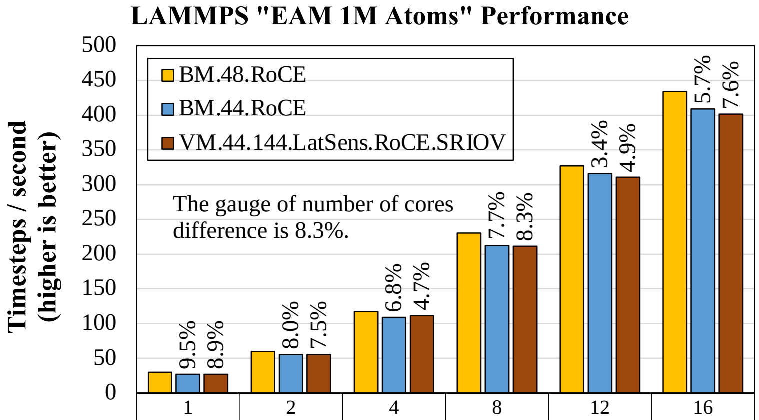
	1	2	4	8	12	16
BM.48.RoCE	100%	97%	94%	93%	89%	91%
BM.44.RoCE	100%	96%	95%	92%	89%	90%
VM.44.144.LatSens.RoCE.SRIOV	100%	98%	95%	93%	89%	90%

Number of Nodes

Figure 28: WRF strong scaling comparison between virtual and bare metal systems.

4.3 LAMMPS

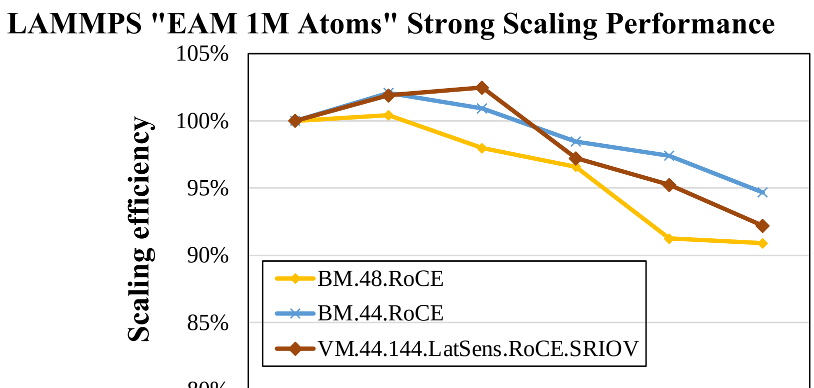
Next, we use the molecular dynamics simulator LAMMPS. **Figure 29** and **Figure 30** show that VM.44 has the largest delta of 8.9% on the single node. But BM.44 also has a delta of 9.5%, which we attribute to the input data decomposition. Other node counts still present the performance delta within the 8.3% gauge.



	1	2	4	8	12	16
BM.48.RoCE	30	60	117	231	327	434
BM.44.RoCE	27	55	109	213	316	409
VM.44.144.LatSens.RoCE.SRIOV	27	55	112	212	311	401

Number of Nodes

Figure 29: LAMMPS performance comparison between virtual and bare metal systems.



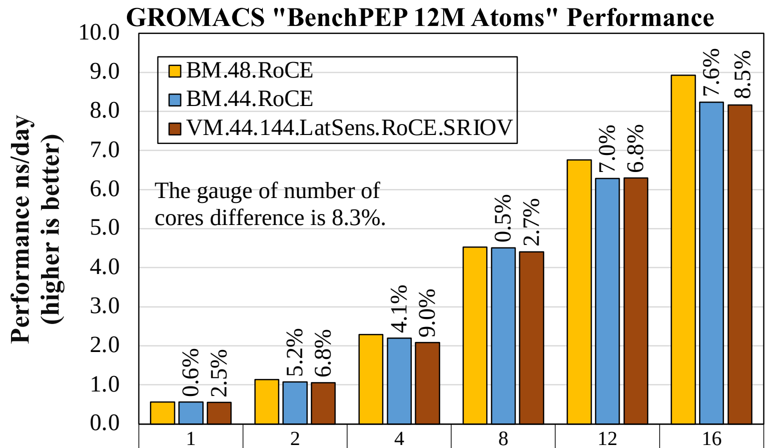
	1	2	4	8	12	16
BM.48.RoCE	100%	100%	98%	97%	91%	91%
BM.44.RoCE	100%	102%	101%	98%	97%	95%
VM.44.144.LatSens.RoCE.SRIOV	100%	102%	102%	97%	95%	92%

Number of nodes

Figure 30: LAMMPS strong scaling comparison between virtual and bare metal systems.

4.4 GROMACS

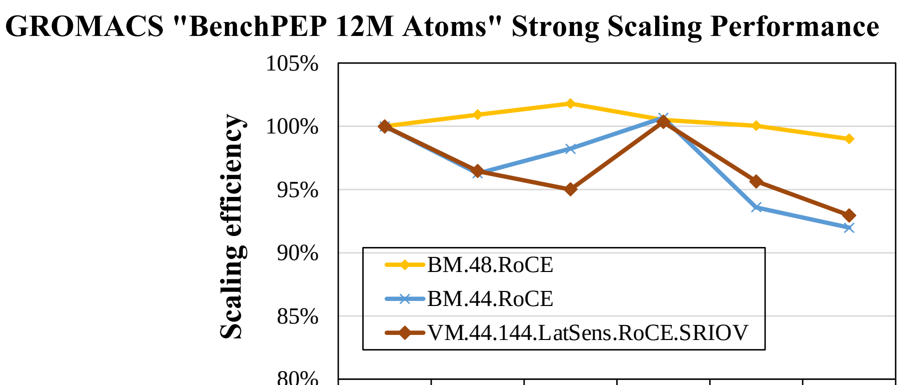
Next, we use GROMACS, a simulator often used to study biomolecules. Here, the largest delta between VM.44 and BM.48 is 8.5% on 16 nodes. Since BM.44 also has a 7.6% delta compared to BM.48, we consider this delta acceptable since it is related to the difference in domain decomposition.



	1	2	4	8	12	16
BM.48.RoCE	0.56	1.14	2.29	4.53	6.76	8.92
BM.44.RoCE	0.56	1.08	2.20	4.51	6.29	8.24
VM.44.144.LatSens.RoCE.SRIOV	0.55	1.06	2.09	4.41	6.30	8.17

Number of Nodes

Figure 31: GROMACS performance comparison between virtual and bare metal systems.



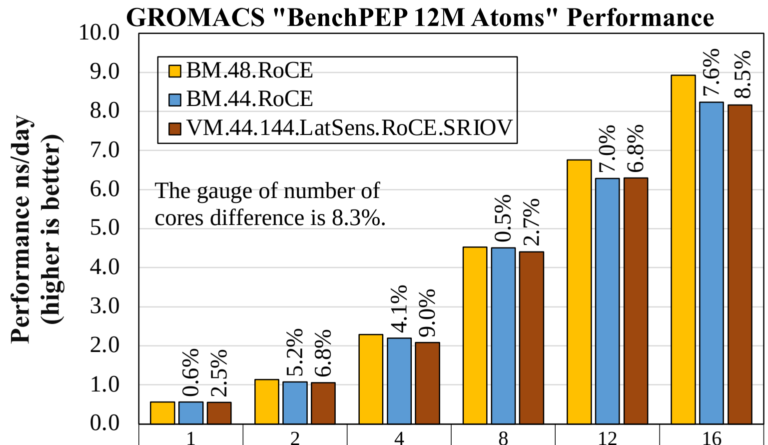
	1	2	4	8	12	16
BM.48.RoCE	100%	101%	102%	101%	100%	99%
BM.44.RoCE	100%	96%	98%	101%	94%	92%
VM.44.144.LatSens.RoCE.SRIOV	100%	96%	95%	100%	96%	93%

Number of Nodes

Figure 32: GROMACS strong scaling comparison between virtual and bare metal systems.

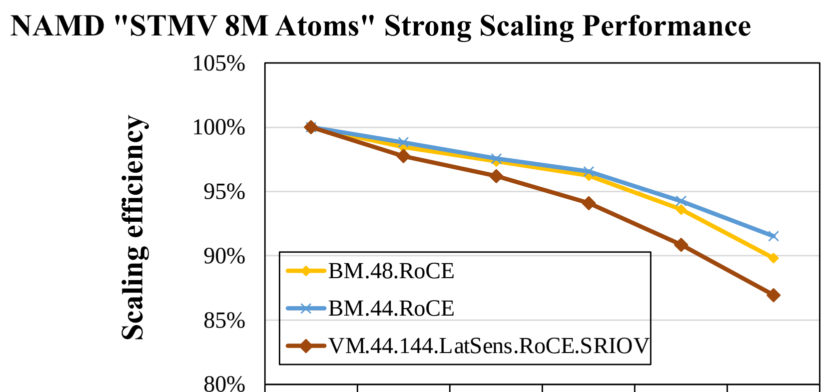
4.5 NAMD

Last, we use NAMD, a simulator of large biomolecular systems. We run NAMD in a hybrid mode, such as for a 44 PPN, 1 MPI process with 43 computing threads, and 1 communication thread launched on a node. Currently, we see an 8.5% performance delta on the 16 nodes comparing VM.44 and BM.48. Since our servers don't enable Sub-NUMA Clustering (SNC), we plan to investigate further the performance improvement with SNC enabled in the future.



Number of Nodes	1	2	4	8	12	16
BM.48.RoCE	0.56	1.14	2.29	4.53	6.76	8.92
BM.44.RoCE	0.56	1.08	2.20	4.51	6.29	8.24
VM.44.144.LatSens.RoCE.SRIOV	0.55	1.06	2.09	4.41	6.30	8.17

Figure 33: NAMD performance comparison between virtual and bare metal systems.



Number of Nodes	1	2	4	8	12	16
BM.48.RoCE	100%	98%	97%	96%	94%	90%
BM.44.RoCE	100%	99%	98%	97%	94%	92%
VM.44.144.LatSens.RoCE.SRIOV	100%	98%	96%	94%	91%	87%

Figure 34: NAMD strong scaling comparison between virtual and bare metal systems.

5 Summary

In this document, we walked through the steps to configure RoCE SR-IOV on NVIDIA Mellanox ConnectX-5 adapter cards in vSphere 7.x. We evaluated this setup's functionality with two benchmarks and studied its performance on five typical HPC applications. In all cases, our virtual HPC cluster approached the performance of a bare metal cluster.

We wrote this technical guide so that it would remain useful even when software versions and products evolve in the future. We hope you found it insightful. We will return as we continue this series of technical guides on other topics, including DirectPath I/O of IB and RoCE and the performance differences when using IB and RoCE for HPC workloads.

6 References

[1] An Evaluation of Ethernet Performance for Scientific Workloads

[2] TOP500.org The List

[3] NVIDIA Firmware Tools (MFT) Documentation v4.18.1

[4] Native ConnectX Driver for VMware ESXi Server

[5] NVIDIA ConnectX-4 onwards NICs NATIVE ESXi Driver for VMware vSphere User Manual v4.19.71.1

[6] vHPC-toolkit github page

[7] Create a vSphere Distributed Switch

[8] Assign a Virtual Function as SR-IOV Passthrough Adapter to a Virtual Machine

[9] Guest OFED Download link

[10] Performance Study of HPC Scale-Out Workloads on VMware vSphere 7

[11] Intel Cluster Checker

[12] The Eternal Battle Between InfiniBand and Ethernet in HPC

[13] Research and Evaluation of RoCE in IHEP Data Center

[14] NVIDIA ConnectX-5 InfiniBand Adapter Cards

[15] Avoid Assigning Multiple NICs in the Same Computer to the Same Subnet

[16] NVIDIA Firmware Tools (MFT)

[17] ConnectX-5 VPI/InfiniBand Firmware Download Center

About the Author

Yuankun Fu has been a Senior member of technical staff in the HPC/ML group of VMware OCTO since July 2021. He focuses on HPC/ML application performance on the VMware platform. He works on a wide variety of HPC projects, from creating technical guides and performance best practices to root-causing performance challenges when running highly technical workloads on customer platforms. Previously, he was a research assistant at Purdue University and interned at the Los Alamos National Lab. He holds a Ph.D. degree in Computer Science from Purdue University.

Acknowledgments

The author thanks Ramesh Radhakrishnan, Chris Gully, and Michael Cui from VMware, Rizwan Ali and Martin Hilgeman from Dell Technologies, and Martin Feyereisen for their contribution to the study. The author also thanks Pamela Gorder and Julie Brodeur for editing this document.