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

Using NVIDIA ConnectX adapter cards for HPC and machine learning workloads on vSphere – August 31, 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

Over the last two decades, InfiniBand (IB) and remote direct memory access (RDMA) over Converged Enhanced Ethernet (RoCE) have become increasingly popular for deploying modern high-performance computing and machine learning (HPC/ML) clusters. NVIDIA Mellanox ConnectX adapter cards support both InfiniBand Enhanced Data Rate and Ethernet network connectivity and provide low latency, high message rate, and other features in high-performance computing (HPC), cloud, and storage environments today [2].

There's no shortage of instruction about how to set up these systems. On the contrary, the challenge for users is to navigate website changes and product updates to ensure they are acting on the most up-to-date information.

In particular, VMware offers documentation [1][7][8] on how to configure a virtual machine (VM) to use SR-IOV hardware standard devices. Likewise, Mellanox offers documentation [2][3][4][5][9][13]on how to set up and configure the firmware and driver of Mellanox ConnectX adapter cards in vSphere environments.

To the best of our knowledge, there is currently no single document that consolidates content and the necessary steps from both sites to create a complete configuration workflow—what customers need to take an SR-IOV device from scratch to a ready-to-use state. Therefore, the VMware OCTO team is authoring a series of technical guides to fill the gap. We aim to present the best practices for customers and HPC administrators to use SR-IOV and other virtualization techniques in an HPC/ML environment.

In this document, we walk through the steps to enable IB SR-IOV on a dual-port Mellanox ConnectX-5 VPI adapter card in vSphere 7.x. We build on the most current VMware and Mellanox documentation and cover the steps from BIOS, ESXi, and vCenter to the functionality test on the VM guest operating system by using ibverbs, Intel cluster checker, and the Ohio State University (OSU) microbenchmark suite. We also introduce how to use the vHPC toolkit, an open-source tool developed by VMware, to speed up the deployment of an HPC cluster in vSphere.

Finally, we present a performance study of five HPC applications across multiple vertical domains, all concerned with dynamic systems (including manufacturing, weather forecasting, and the life sciences). We conclude that virtual HPC clusters with VMware vSphere perform nearly as well as bare-metal HPC clusters while offering all the advantages of virtualization with vSphere like increased IT agility, flexibility, scalability, and cost savings of hardware.

2 Configuration Workflow

According to the SR-IOV specification, each Virtual Function (VF) has a restricted set of configuration resources and can be provisioned as a separate device for each VM. For example, NVIDIA ConnectX family adapter cards can offer up to 127 VFs depending on the firmware capabilities. The VFs are connected to the Physical Function (PF), which owns the same resources shared by all VFs and manages the global functions—for example, moving data in and out of the device. SR-IOV is typically used with an SR-IOV-enabled hypervisor such as vSphere to provide a VM direct hardware access to physical resources, hence increasing its utilization and performance.

For simplicity, Figure 1 illustrates the general idea of the IB SR-IOV configuration on two VMs. We enable SR-IOV functionality on the physical adapters, then attach the VFs to VMs in vSphere, and use the virtual distributed switch (VDS) for the communication between them. When we benchmark and test performance, we will use the example of 16 VMs on 16 servers.

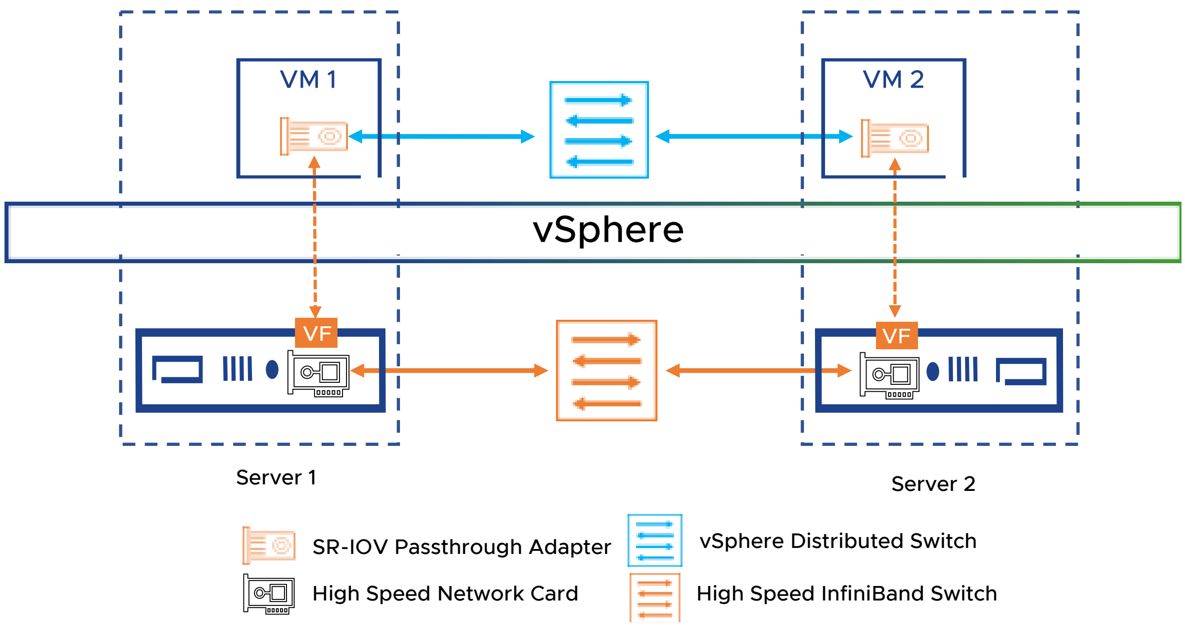


Figure 1. Illustration of IB SR-IOV configuration

Figure 2 presents the flow chart to enable IB SR-IOV. The configuration workflow is generally divided into four stages, from BIOS, ESXi, and vCenter to the VM guest.

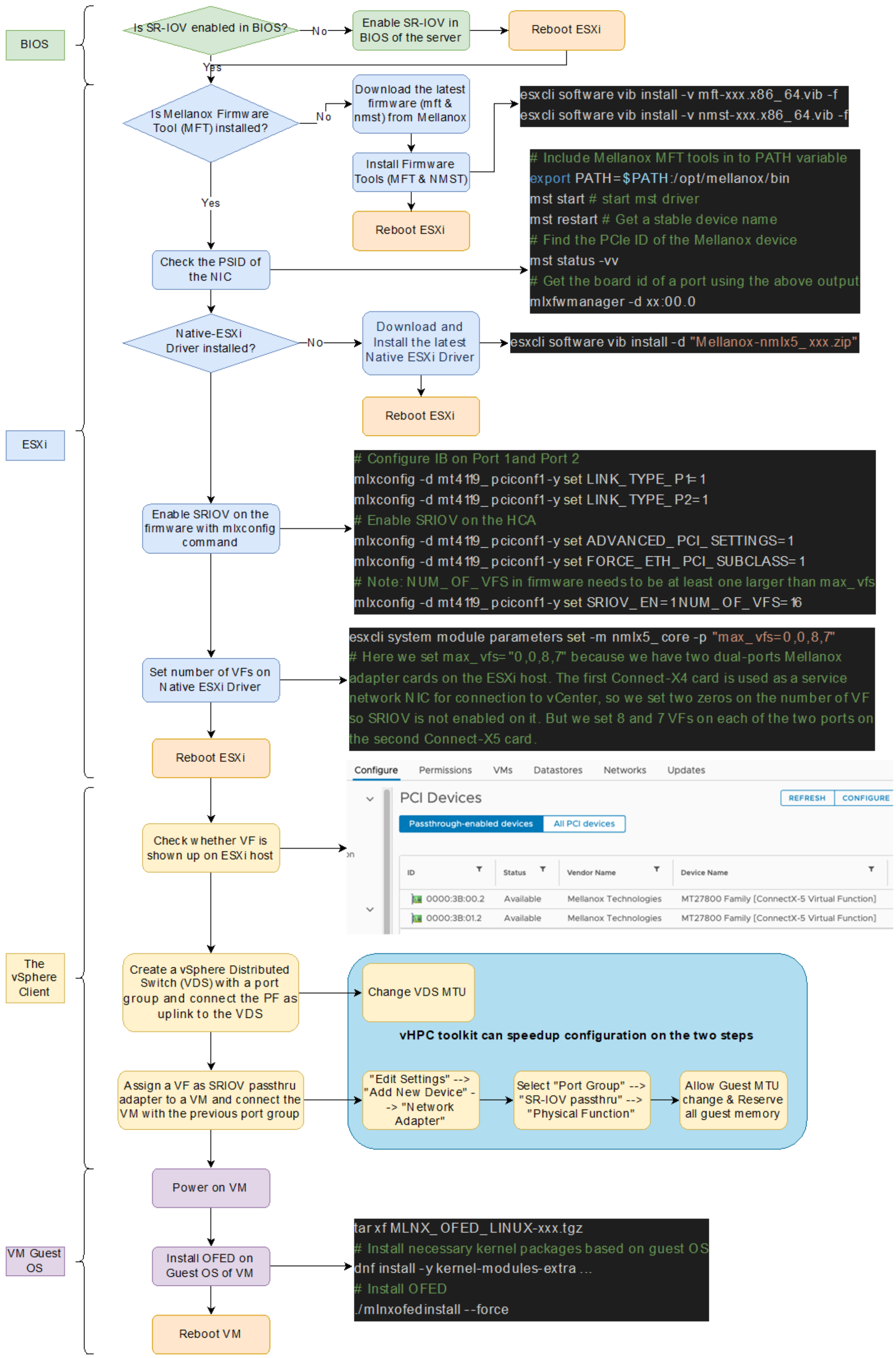


Figure 2. Flow chart to enable IB SR-IOV on NVIDIA Mellanox ConnectX-5 in ESXi 7.x.

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.

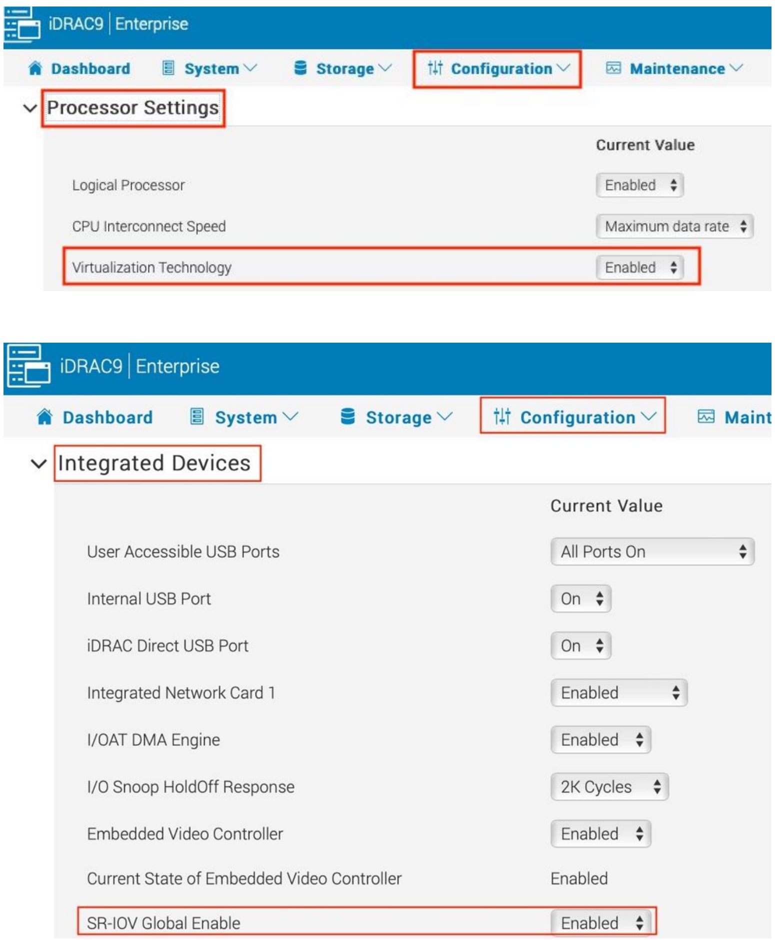


Figure 3. Enable Virtualization and SR-IOV Global in BIOS of iDrac in Dell R740.

> **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**.

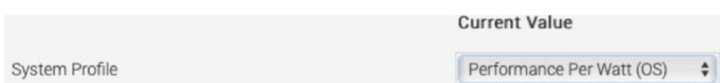


Figure 4. Power profile in BIOS

2.2 ESXi configuration

After enabling SR-IOV in the BIOS, we need to configure the adapter card in ESXi by configuring its firmware and the native ESXi driver. We also need to download and install any missing software. Three reboots of the ESXi host will be required in this section. (Most of the steps in this section refer to Native ConnectX Driver for VMware ESXi Server [5], and you can check additional information there.)

2.2.1 Install Mellanox firmware tools and the first reboot

First, we need to check whether the latest firmware tools (NVIDIA Firmware Tools (MFT) Documentation v4.18.1 [3]) are installed on our ESXi host. Figure 5 shows that there are two packages included: NMST and MFT.

> **Best Practice:** We recommend downloading them [14] 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.

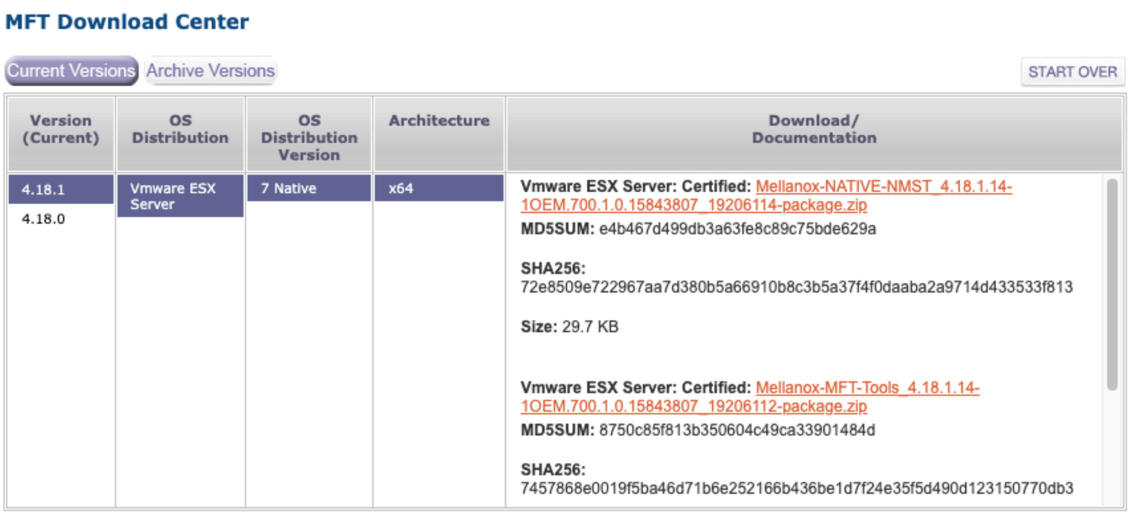


Figure 5. Download firmware tools from the NVIDIA Mellanox website

After extracting the two zip files, we use the following commands to install them on the ESXi host in 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 firmware tools to check whether they function well—for example, by querying the status, firmware version, and board id and 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, as 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 the vSphere Client (vCenter) and vSAN, while the `ConnectX5`, on which we intend to enable IB 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, 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 on the NVIDIA website in Figure 8. If online updating of the firmware is not available, we can choose to manually burn the firmware with the flint command [5].

Figure 8. The latest firmware version of our HPC NIC

2.2.2 Install 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, you can download it at 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 if it were an Ethernet device so that vSphere can detect it. This webpage directs to a VMware site to download the nmlx_core driver.

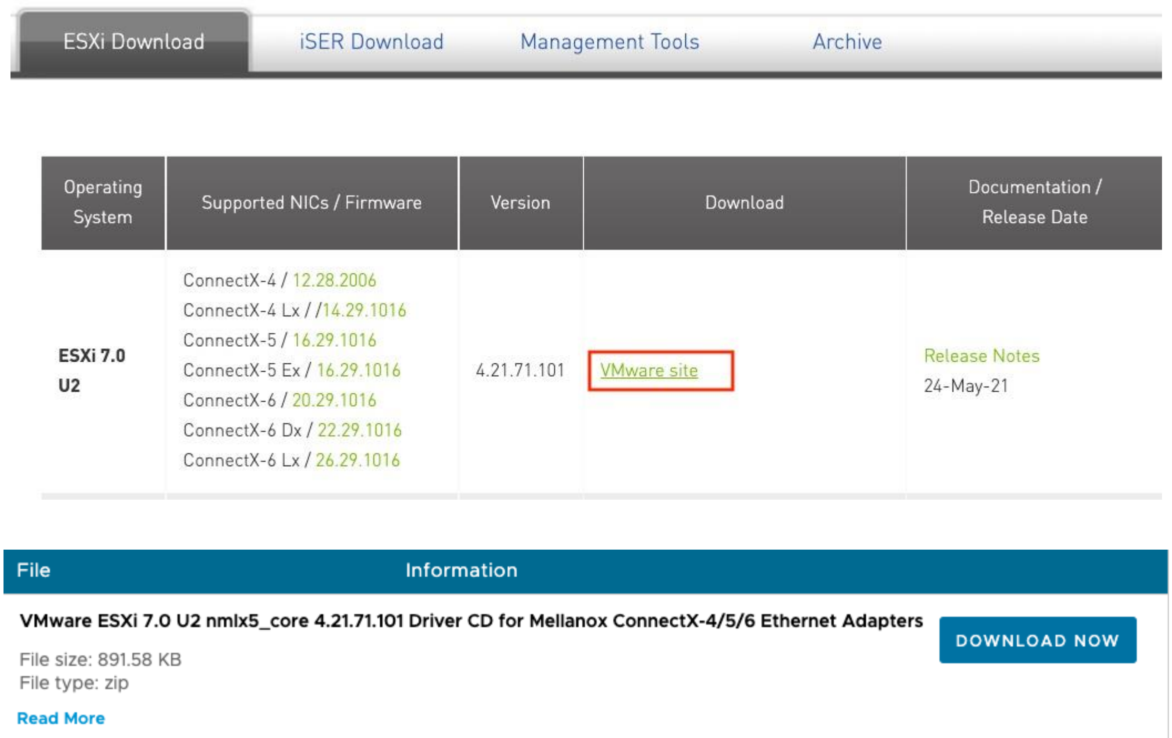


Figure 9. Download the native ESXi driver

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

Then 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 IB 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 IB on the firmware of Port 1 and Port 2 of ConnectX-5
2 [esxi]$ mlxconfig -d mt4119_pciconf1 -y set LINK_TYPE_P1=1
3 [esxi]$ mlxconfig -d mt4119_pciconf1 -y set LINK_TYPE_P2=1
4
5 # Enable SRIOV on the firmware of ConnectX-5
6 [esxi]$ mlxconfig -d mt4119_pciconf1 -y set ADVANCED_PCI_SETTINGS=1
7 [esxi]$ mlxconfig -d mt4119_pciconf1 -y set FORCE_ETH_PCI_SUBCLASS=1
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 “max_vfs=0,0,8,7”
13
14 # Reboot the host for the third time
```

Figure 11. Commands to enable IB SR-IOV, the firmware, and the nmlx driver

In lines 6 and 7, we set `FORCE_ETH_PCI_SUBCLASS=1`, so that the IB adapter can be treated as an Ethernet adapter in vSphere, but the underlying fabric protocol is still IB. In line 9, we set `SRIOV_EN=1` and `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 is 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, so we set the first two numbers to zero. But we would like to create eight VFs on the first port and seven VFs on the second port on our ConnectX-5, so we set the last two numbers to 8 and 7.

2.3 vCenter 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 status **Down** → **Edit**. (Since the IB adapters (vmnic2) in Figure 12 are feigned as Ethernet adapters in vSphere, the actual speed is shown as **Down**, but we can check that the SR-IOV status shows **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** → **Management** → **Hardware** → **PCI Devices** → **Configure SR-IOV** → Select the physical adapters in the list → Change the number of Virtual Functions.)

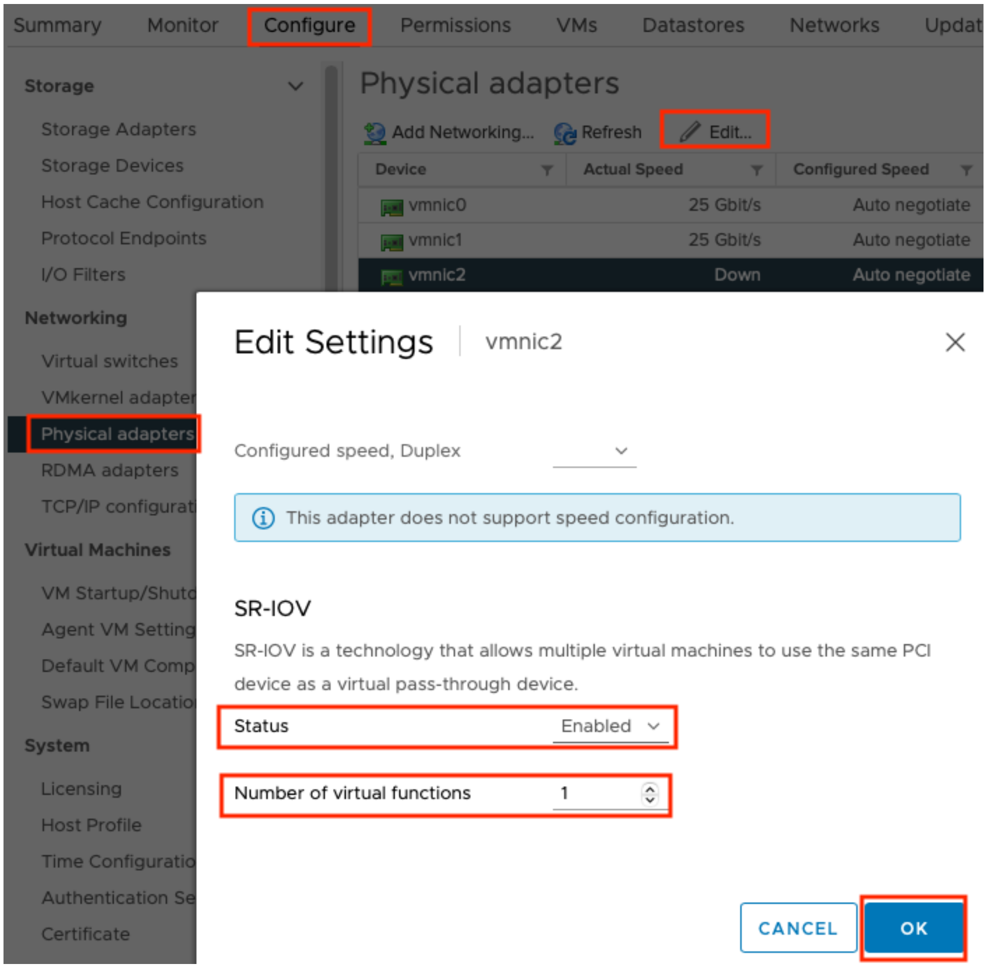


Figure 12. Edit the number of VFs in the vSphere Client or in the ESXi host

Next, we can view the VFs shown as **Passthrough-enabled devices** in Figure 13 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.

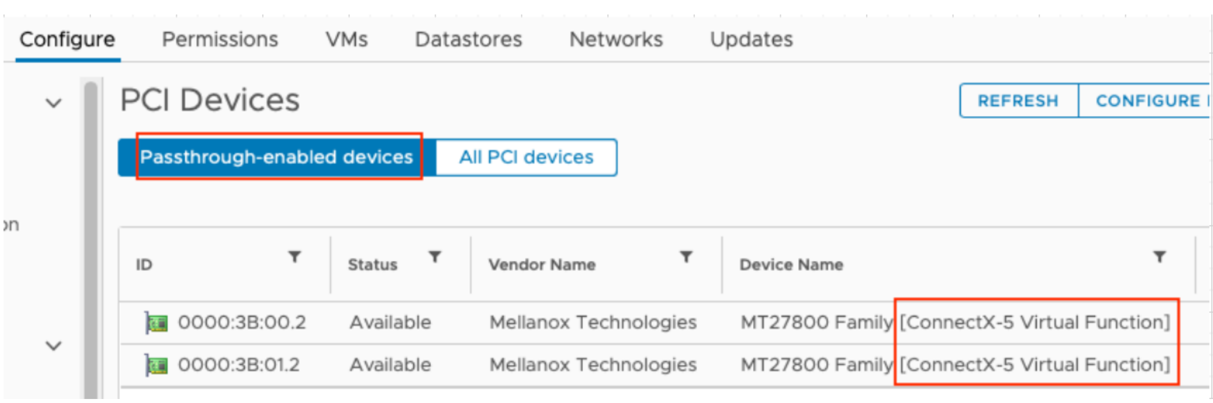


Figure 13. VFs are shown as PCI **Passthrough-enabled devices** in the vSphere Client

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**.

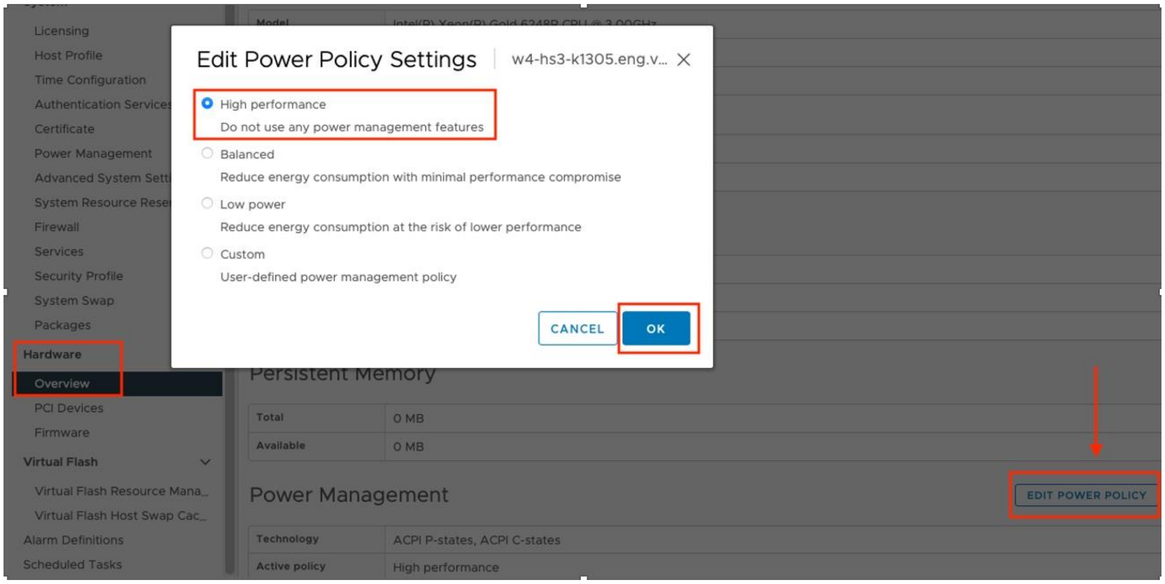


Figure 14. Choose **High Performance** as the power policy for the ESXi host

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 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-IB-DVS"
[vhpc]$ sriov_dvs_pg="SRIOV-IB-DVS-PG"
[vhpc]$ PF_ID="0000:3b:00.0"
[vhpc]$ physical_adapter="vmnic2"
[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 high-speed communication requirement for an HPC workload (Figure 16).

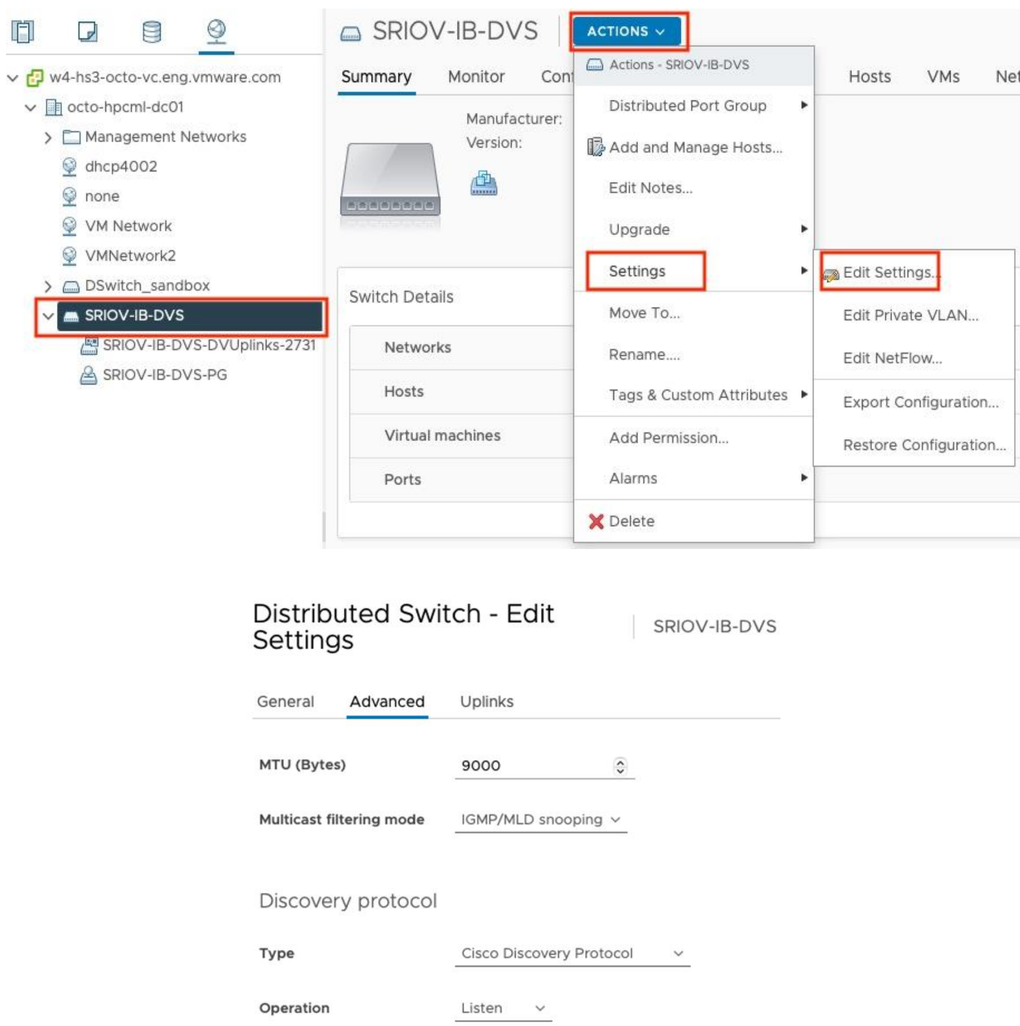


Figure 16. Change MTU=9000 on the VDS

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] using the vSphere Client. Note that the VM requires reserved memory, and the selection of **Allow the Guest MTU** is changed to **Allow** SR-IOV.

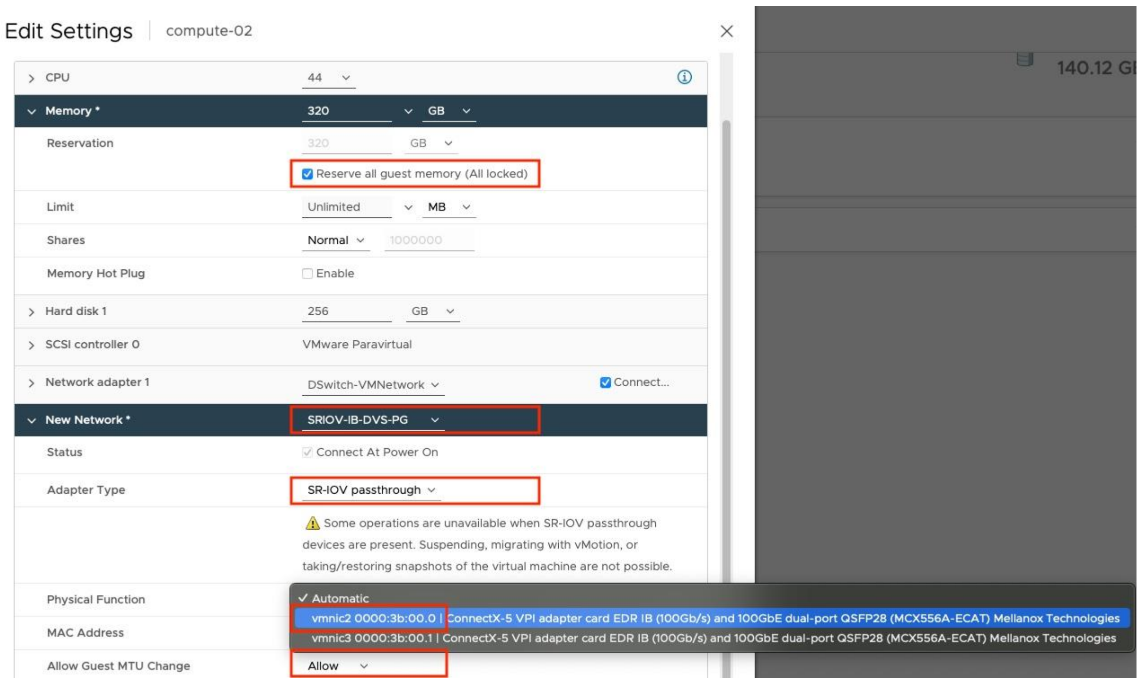


Figure 17. Use the vSphere Client to assign a VF as an SR-IOV passthrough adapter to a VM

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

```
[vhpc]$ sriov_dvs=”SRIOV-IB-DVS”
[vhpc]$ sriov_dvs_pg=”SRIOV-IB-DVS-PG”
[vhpc]$ PF_ID=”0000:3b:00.0”
[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 OS OFED **Download** tab [9]. Figure 19 shows the command to install OFED. A reboot of the VM is required after the OFED 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

> **Note:** The OFED user manual [13] mentions using one of the compute nodes as the IB subnet manager, but that is not required here since we have an IB switch and can enable the subnet manager on the switch. Since we updated our NIC adapters to the latest firmware, the latest MLX OS on the IB switch is also required to ensure the activity of the IB network. To enable the subnet manager and set MTU on the IB switch, refer to the documentation of your product vendor.

After OFED is installed on the guest, we first need to force restart the OFED driver, and then we can check its version with `ofed_info -s`. With `ip a` to list the network interface, we see ib0 displays. If you want to ping the IB interface on other hosts, an IP address is required to add to the interface (that is, ib0). Then we can use `ibv_devinfo` or `ibstatus` to check the status of the IB port. Figure 20 shows that the port `mlx5_0` is in the active state with `active_MTU=4096` and is using **InfiniBand** 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, ib0 is the interface of IB VF
[guest OS]$ # ip a
. . .
ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 4092 qdisc mq state UP group default qlen 256
     link/infiniband 00:00:00:e9:fe:80:00:00:00:00:00:00:00:50:56:ff:fe:b3:9e:c1 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
# If you want to ping the interface, IP address is required to add to ib0 with command
ip addr add XXX.XXX.XXX.XX/XX dev ib0

# Check device information
[guest OS]$ ibv_devinfo
hca_id: mlx5_0
    transport:          InfiniBand (0)
    fw_ver:             16.32.1010
    node_guid:          00xx:xxxx:xxxx:xxxx
    sys_image_guid:     0xxx:xxxx:xxxx:xxxx
    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:         1
            port_lid:       205
            port_lmc:       0x00
            link_layer:     InfiniBand
```

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

3 Functionality Evaluation

In this section, we evaluate the functionality of the IB SR-IOV that we configured using three tests: the ibverbs utility test, the Intel cluster checker, 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.

Environment		Bare Metal	Virtual Machine
Hardware	Server	PowerEdge R740 vSAN ReadyNode	
	Processor	2 x Intel Xeon Gold 6248R @ 3.00GHz	
	HPC InfiniBand Network NIC	100 GbE NVIDIA Mellanox ConnectX-5 VPI Dual Ports	
	Service Network NIC	10/25 GbE NVIDIA Mellanox ConnectX-4 Dual Ports	
	HPC InfiniBand Network Switch	Mellanox SB7800 100 Gb IB switch	
	Service Network Switch	Dell PowerSwitch S5248F-ON	
	ConnectX-5 firmware	16.32.1010	
BIOS	Power Profile	Performance Per Watt (OS controlled)	
	Hyperthreading	Enabled	
	Virtualization	Intel VT-d Enabled	
Cores		All 48 cores used	44 vCPU reserved, High Latency sensitivity
Memory		24 * 16GB RDIMM, All 384 GB used	144 GB reserved for the VM
Operating system	Host	RHEL 8.1	VMware vSphere 7.0U2, Guest OS: RHEL 8.1
	Power Policy	Default	High Performance
	Mellanox Firmware tools	MFT & NMST 4.18.1	
	Native Mellanox (NMLX) Driver	N/A	4.21.71.101
	OFED	5.4-3.0.3.0	
Build Libraries	Compiler	GCC 9.3.0	
	MPI	OpenMPI 4.1.2	
	UCX	1.12.0	
	Intel One API / Cluster Checker	2022.2 / 2021 Update 6 (build 20220318)	
	Spack	0.17.1	
	OSU MicroBenchmark	5.7.1	

Table 1. Testbed details of the virtual clusters

3.1 ibverbs utility test

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

```
[root@compute-02 ~]# ib_send_bw -a -report_gbits -d mlx5_0 compute-03

                    Send BW Test
 Dual-port       : OFF      Device         : mlx5_0
 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       : IB
 Max inline data : 0[B]
 rdma_cm QPs     : OFF
 Data ex. Method : Ethernet

 local address: LID 0xcd QPN 0x00cf PSN 0x363fd8
 remote address: LID 0xcc QPN 0x00cf PSN 0xd8c77e

 #bytes     #iterations  BW peak[Gb/sec]  BW average[Gb/sec]  MsgRate[Mpps]
 2          1000         0.087386         0.083431            5.214466
 4          1000         0.17             0.16                5.148076
 8          1000         0.33             0.33                5.215248
 16         1000         0.67             0.67                5.215975
 32         1000         1.34             1.33                5.214121
 64         1000         2.67             2.67                5.206085
 128        1000         5.34             5.33                5.209293
 256        1000         10.66            10.64               5.197154
 512        1000         21.25            21.21               5.177948
 1024       1000         42.06            41.92               5.117584
 2048       1000         73.08            72.99               4.455041
 4096       1000         91.74            91.73               2.799474
 8192       1000         94.35            94.32               1.439255
 16384      1000         94.32            94.32               0.719622
 32768      1000         94.71            94.70               0.361266
 65536      1000         94.87            94.87               0.180948
 131072     1000         95.12            95.06               0.090655
 262144     1000         94.93            94.85               0.045228
 524288     1000         94.40            94.00               0.022410
 1048576    1000         94.81            94.09               0.011216
 2097152    1000         95.23            95.05               0.005665
 4194304    1000         95.41            95.11               0.002835
 8388608    1000         95.48            95.29               0.001420
```

Figure 21. ibverbs bandwidth test

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

```
[root@compute-02 ~]# ib_send_lat -a -report_gbits -d mlx5_0 compute-03

                   Send Latency Test
 Dual-port       : OFF      Device         : mlx5_0
 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       : IB
 Max inline data : 236[B]
 rdma_cm QPs     : OFF
 Data ex. Method : Ethernet

 local address: LID 0xcd QPN 0x00cb PSN 0xda6976
 remote address: LID 0xcc QPN 0x00cc PSN 0xa10b25

#bytes
#iterations  t_min[usec]  t_max[usec]  t_typical[usec]  t_avg[usec]  t_stdev[usec]  99%[usec]  99.9%[usec]
 2       1000        1.12         6.55         1.17             1.19         0.16           1.41       6.55
 4       1000        1.11         3.38         1.15             1.16         0.09           1.31       3.38
 8       1000        1.10         4.01         1.15             1.16         0.11           1.30       4.01
 16      1000        1.11         3.73         1.15             1.16         0.10           1.31       3.73
 32      1000        1.14         4.47         1.19             1.20         0.14           1.37       4.47
 64      1000        1.23         4.38         1.26             1.27         0.11           1.43       4.38
 128     1000        1.26         3.43         1.30             1.31         0.11           1.46       3.43
 256     1000        1.67         4.42         1.70             1.72         0.12           1.90       4.42
 512     1000        1.71         3.82         1.74             1.76         0.12           1.93       3.82
 1024    1000        1.83         4.34         1.89             1.92         0.12           2.16       4.34
 2048    1000        2.07         3.94         2.11             2.13         0.11           2.32       3.94
 4096    1000        2.56         4.52         2.64             2.67         0.14           3.37       4.52
 8192    1000        3.26         5.82         3.37             3.39         0.09           3.63       5.82
 16384   1000        4.64         6.52         4.87             4.89         0.15           5.50       6.52
 32768   1000        6.82         10.03        7.03             7.04         0.13           7.64       10.03
 65536   1000        9.53         12.35        9.75             9.76         0.14           10.11      12.35
 131072  1000        14.94        17.06        15.40            15.42        0.19           15.94      17.06
 262144  1000        26.80        29.58        27.54            27.62        0.37           28.58      29.58
 524288  1000        50.49        54.00        51.66            51.66        0.47           52.73      54.00
 1048576 1000        96.70        103.54       98.62            98.65        0.67           100.52     103.54
 2097152 1000        184.85       190.65       186.96           186.92       0.72           188.57     190.65
 4194304 1000        360.54       365.81       362.85           362.88       0.78           364.86     365.81
 8388608 1000        712.22       724.09       714.70           714.71       0.98           717.16     724.09
```

Figure 22. ibverbs latency test

Figure 22 shows that the latency of `ib_send_lat` is averaging 1.2 microseconds for small messages. Switch latency is documented to be 90 nanoseconds for the IB Mellanox SB7800 switch [12] and accounts for most of the latency for small message transfers.

3.2 Use Intel Cluster Checker to validate 16 VMs in the virtual cluster

We next use Intel Cluster Checker **Error! Reference source not found.** to validate the virtual cluster of 16 VMs. We have passed three health checks—`health_base`, `health_extended_user`, and `intel_hpc_platform_compat-hpc-cluster-2.0`—on the 16 VMs, and the final simulation and modeling test `select_solutions_sim_mod_user_plus_2021.0` on four groups of four VMs in parallel. We will demonstrate how to use it in more detail in a separate document.

```
Intel® Cluster Checker 2021 Update 6
17:20:59 June 3 2022 UTC
Nodefile used: nodelist
Databases used: select_solutions_sim_mod_user_plus_2018.0.db
Reports1. CPU:
  [CPU Model Name: In®(R®eon(R) Gold 6248R CPU@3.00GHz; Sockets:2][nodes: compute-[01-04]]

  2. DGEMM:
  [DGEMM Performance: 3081.050 GFLOP/s.][nodes: compute-03]
  [DGEMM Performance: 3090.500 GFLOP/s.][nodes: compute-04]
  [DGEMM Performance: 3104.410 GFLOP/s.][nodes: compute-01]
  [DGEMM Performance: 3122.590 GFLOP/s.][nodes: compute-02]

  3. HPCG_CLUSTER:
  [HPCG 4-Node Performance: 169.95 GFLOP/s.][nodes: compute-[01-04]]

  4. HPCG_SINGLE:
  [HPCG 1-Node Performance: 43.13 GFLOP/s.][nodes: compute-01]
  [HPCG 1-Node Performance: 43.22 GFLOP/s.][nodes: compute-05]
  [HPCG 1-Node Performance: 43.23 GFLOP/s.][nodes: compute-14]
  [HPCG 1-Node Performance: 43.28 GFLOP/s.][nodes: compute-03]
  [HPCG 1-Node Performance: 43.30 GFLOP/s.][nodes: compute-16]
  [HPCG 1-Node Performance: 43.31 GFLOP/s.][nodes: compute-15]
  [HPCG 1-Node Performance: 43.32 GFLOP/s.][nodes: compute-13]
  [HPCG 1-Node Performance: 43.33 GFLOP/s.][nodes: compute-02, compute-[10-11]]
  [HPCG 1-Node Performance: 43.35 GFLOP/s.][nodes: compute-06]
  [HPCG 1-Node Performance: 43.37 GFLOP/s.][nodes: compute-07, compute-12]
  [HPCG 1-Node Performance: 43.40 GFLOP/s.][nodes: compute-08]
  [HPCG 1-Node Performance: 43.42 GFLOP/s.][nodes: compute-09]
  [HPCG 1-Node Performance: 43.44 GFLOP/s.][nodes: compute-04]

  5. HPL:
  [HPL 4-Node Performance: 10908.20 GFLOP/s.][nodes: compute-[01-04]]

  6. IMB_PINGPONG:
  [IMB pingpong bandwidth: 12.407 GB/s and latency: 1.050 mcseconds.][nodes:compute-[02-03]]
  [IMB pingpong bandwidth: 12.439 GB/s and latency: 1.050 mcseconds.][nodes:compute-[03-04]]
  [IMB pingpong bandwidth: 12.454 GB/s and latency: 1.050 mcseconds.][nodes:compute-[01-02]]

  7. STREAM:
  [STREAM Performance: 176.854 GB/s.][nodes: compute-01]
  [STREAM Performance: 177.399 GB/s.][nodes: compute-02]
  [STREAM Performance: 177.536 GB/s.][nodes: compute-03]
  [STREAM Performance: 177.556 GB/s.][nodes: compute-04]
```

Figure 23. Simulation and modeling test on one set of four VMs by Intel Cluster Checker

3.3 OSU microbenchmark test

Since our server has been configured as dual-boot (bare metal and ESXi), we use the OSU microbenchmarks to compare the communication performance—first on the 16 bare-metal nodes, then on the 16 VMs. We run multiple bandwidth/message rate benchmark (mbw_mr) in Figure 24 and collective benchmark (all_to_all) in Figure 25 with first 2, then 4, 8, 12, and 16 VMs. Each datapoint uses the average of five runs. Since the VMs are using 44 vCPUs, for fair comparison, we run 48 and 44 processes per node (PPN) on `BareMetal (BM)` nodes. The legend `VM.44.144.LatSense.IB.SRIOV` means the virtual machine uses 44 vCPUs, 144 GB memory, sets latency sensitivity to high and uses IB SR-IOV. The legend format is also used in the later HPC application test.

Figure 24 shows that IB SR-IOV can achieve near bare-metal performance on all message sizes for the aggregate bandwidth/message rate test.

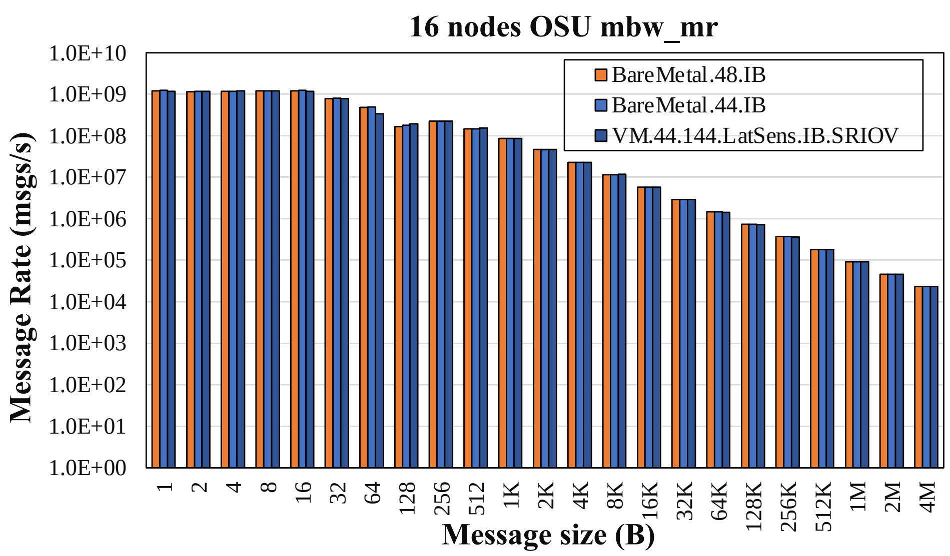


Figure 24. OSU MBW_MR test on 16 nodes

In Figure 25, we notice `BareMetal.48.IB` has an average of 24% and 10% higher all_to_all latency than the `BareMetal.44.IB and VM.44.IB.SRIOV` on various 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.

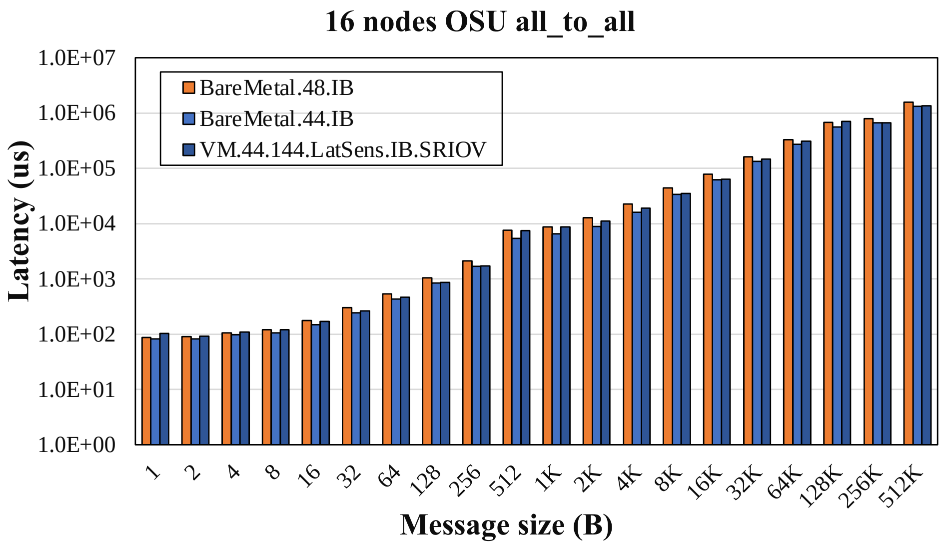


Figure 25. OSU all-to-all on 16 nodes

4 Performance Study of HPC Applications

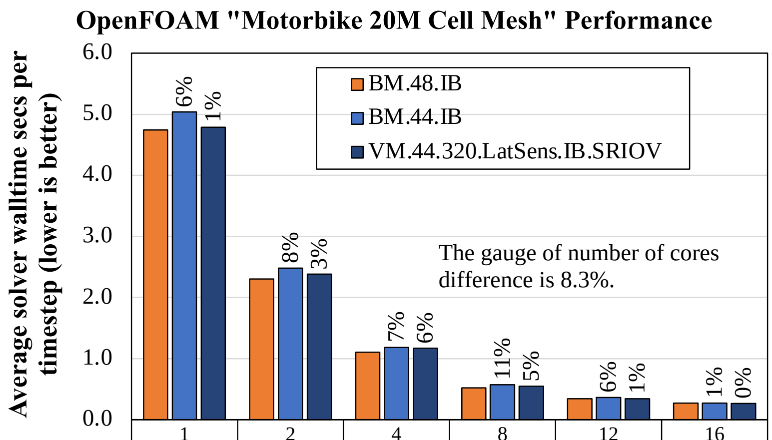
In this section, we compare the performance and strong scalability between the bare metal and virtual systems by using a range of different HPC applications across multiple vertical domains along with the benchmark datasets used in **Error! Reference source not found.**. We use the tuning best practice in [10] to achieve MPI application performance 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.

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

Table 2. Application and benchmark details

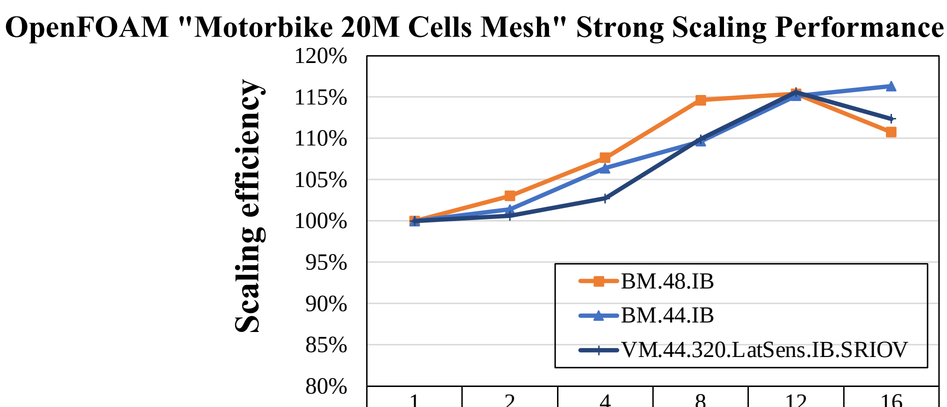
4.1 OpenFOAM

We begin with the OpenFOAM software for computational fluid dynamics. Since the 20 million cell Motorbike benchmark needs a larger memory than 144 GB to run, we expand the VM's memory to 320 GB for only this application. We use the `BM.48.IB` as the baseline, so the percentage number on the top of the columns `BM.44.IB` and `VM.44.320.LatSens.IB.SRIOV` in Figure 26 shows the performance delta compared to the baseline. We observe that VM.44 has at most a 6% delta compared to BM.48 using 4 nodes (176 cores) and performs better than BM.44 on all node counts.



	1	2	4	8	12	16
BM.48.IB	4.74	2.30	1.10	0.52	0.34	0.27
BM.44.IB	5.03	2.48	1.18	0.57	0.36	0.27
VM.44.320.LatSens.IB.SRIOV	4.79	2.38	1.17	0.54	0.35	0.27

Figure 26. OpenFOAM performance comparison between virtual and bare-metal systems

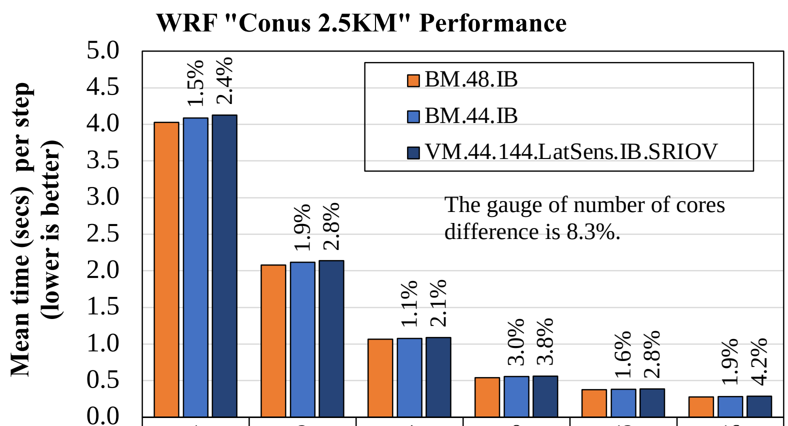


	1	2	4	8	12	16
BM.48.IB	100%	103%	108%	115%	115%	111%
BM.44.IB	100%	101%	106%	110%	115%	116%
VM.44.320.LatSens.IB.SRIOV	100%	101%	103%	110%	116%	112%

Figure 27. OpenFOAM strong scaling comparison between virtual and bare-metal systems

4.2 WRF

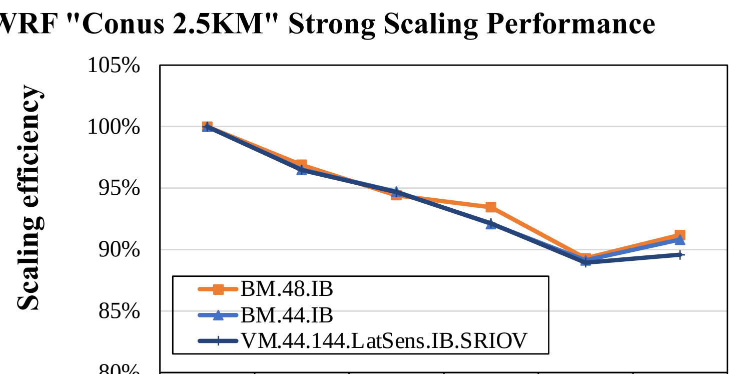
For our following example, we try the WRF model, a numerical weather prediction system used in atmospheric research and other applications. Here, we observe that VM.44 has at most a 4.2% performance delta on 16 nodes compared to BM.48 in Figure 28 andFigure 29Figure 29. Other node counts still present the performance delta within the 8.3% gauge.



	1	2	4	8	12	16
BM.48.IB	4.03	2.08	1.07	0.54	0.38	0.28
BM.44.IB	4.09	2.12	1.08	0.55	0.38	0.28
VM.44.144.LatSens.IB.SRIOV	4.12	2.14	1.09	0.56	0.39	0.29

Number of nodes

Figure 28. WRF performance comparison between virtual and bare-metal systems



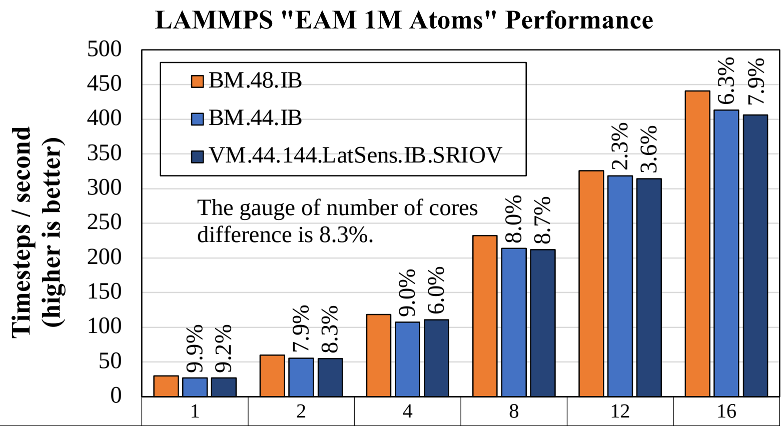
	1	2	4	8	12	16
BM.48.IB	100%	97%	94%	93%	89%	91%
BM.44.IB	100%	96%	95%	92%	89%	91%
VM.44.144.LatSens.IB.SRIOV	100%	96%	95%	92%	89%	90%

Number of Nodes

Figure 29. WRF strong scaling comparison between virtual and bare-metal systems

4.3 LAMMPS

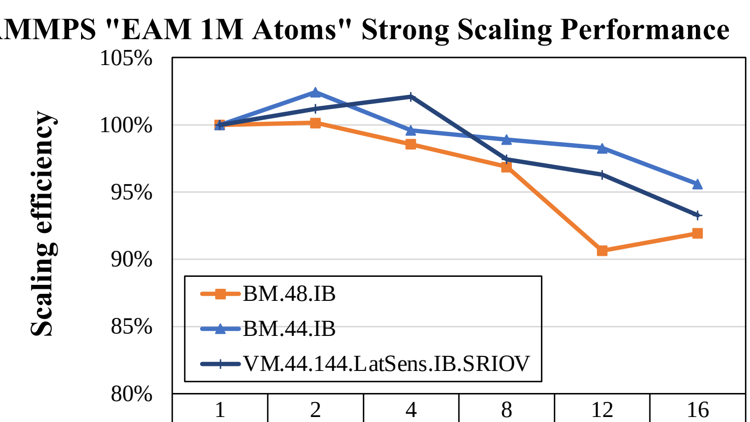
Next, we use the molecular dynamics simulator LAMMPS. Figure 30 and Figure 31 show that VM.44 has the largest delta—9.2%—on the single node. But BM.44 also has a delta of 9.9%, which we attribute to the input data decomposition. Other node counts still present the performance delta within the 8.3% gauge.



	1	2	4	8	12	16
BM.48.IB	30	60	118	232	326	441
BM.44.IB	27	55	108	214	318	413
VM.44.144.LatSens.IB.SRIOV	27	55	111	212	314	406

Number of Nodes

Figure 30. LAMMPS performance comparison between virtual and bare-metal systems



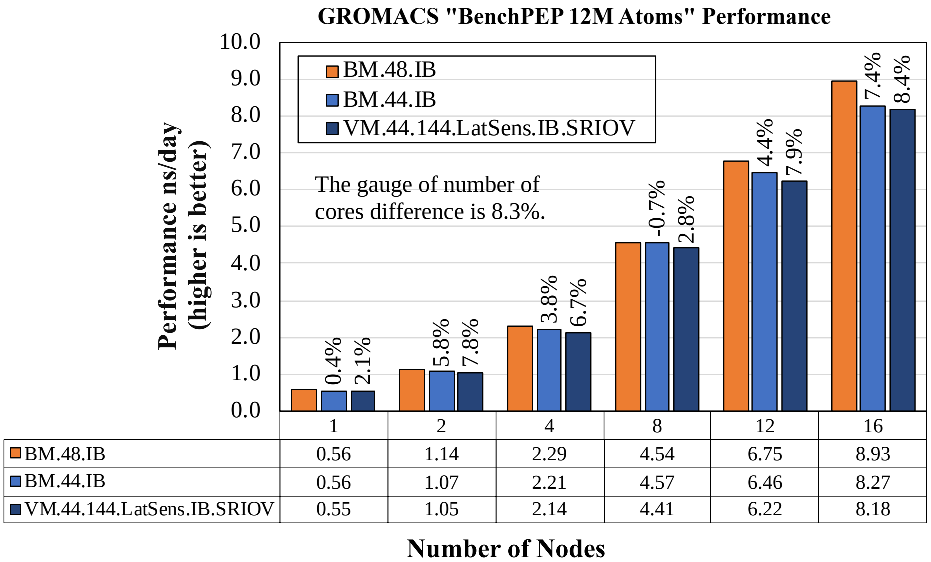
	1	2	4	8	12	16
BM.48.IB	100%	100%	99%	97%	91%	92%
BM.44.IB	100%	102%	100%	99%	98%	96%
VM.44.144.LatSens.IB.SRIOV	100%	101%	102%	97%	96%	93%

Number of nodes

Figure 31. LAMMPS strong scaling comparison between virtual and bare-metal systems

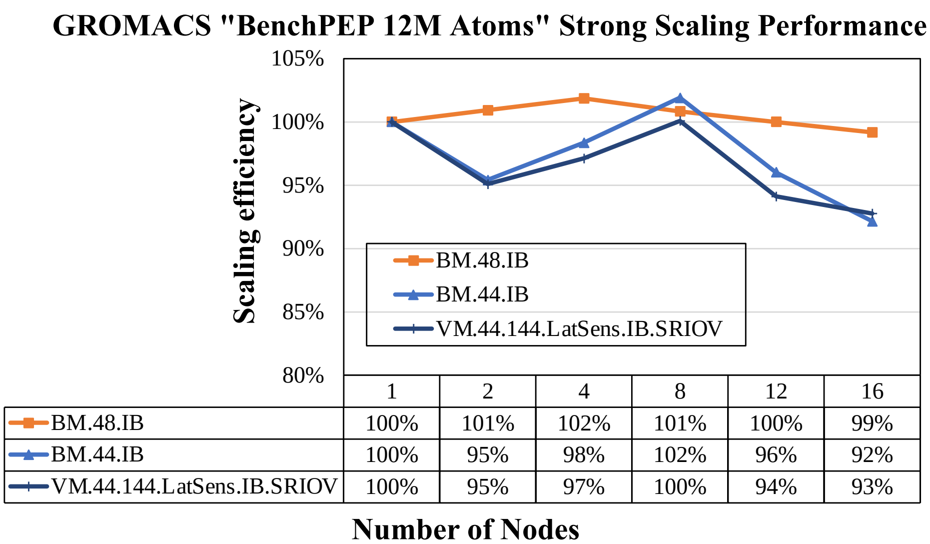
4.4 GROMACS

Next, we use GROMACS, a simulator often used to study biomolecules. Here we see that the largest delta between VM.44 and BM.48 is 8.4% on the 16 nodes. Since BM.44 also has a 7.4% delta compared to BM.48, we consider this delta to be acceptable.



	1	2	4	8	12	16
BM.48.IB	0.56	1.14	2.29	4.54	6.75	8.93
BM.44.IB	0.56	1.07	2.21	4.57	6.46	8.27
VM.44.144.LatSens.IB.SRIOV	0.55	1.05	2.14	4.41	6.22	8.18

Figure 32. GROMACS performance comparison between virtual and bare-metal systems

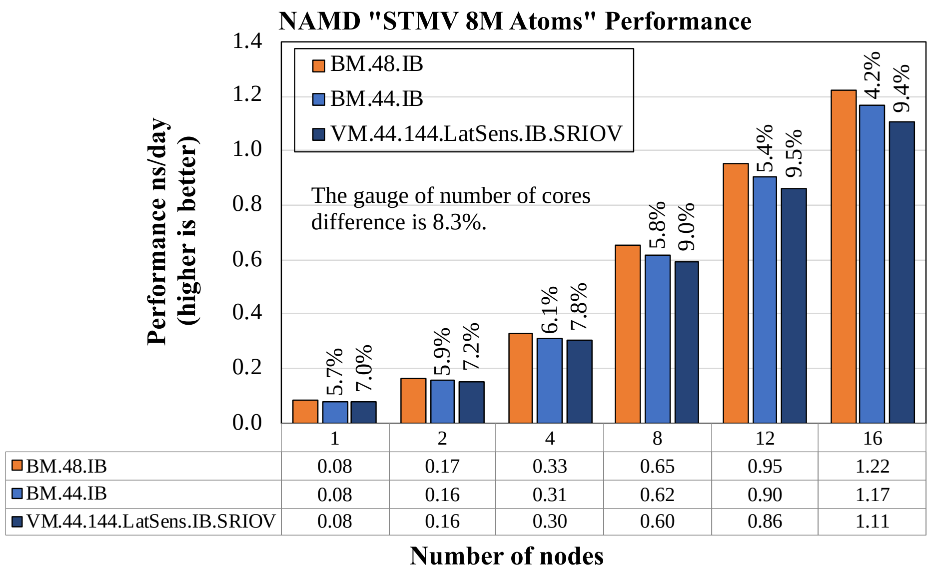


	1	2	4	8	12	16
BM.48.IB	100%	101%	102%	101%	100%	99%
BM.44.IB	100%	95%	98%	102%	96%	92%
VM.44.144.LatSens.IB.SRIOV	100%	95%	97%	100%	94%	93%

Figure 33. GROMACS strong scaling comparison between virtual and bare-metal systems

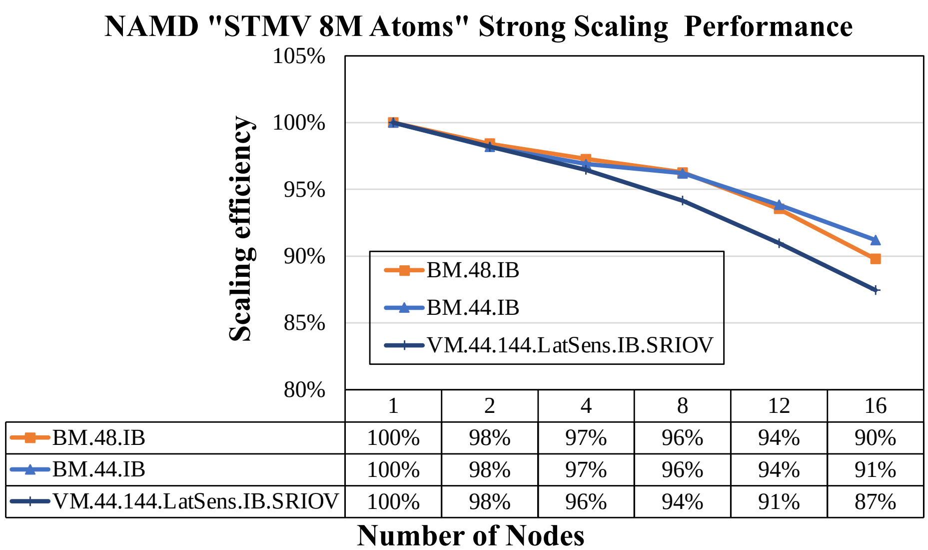
4.5 NAMD

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 one communication thread is launched on a node. Currently, we see a 9.4% performance delta on the 16 nodes comparing virtual and bare metal in Figure 34. The strong scaling efficiency between bare metal and virtual is, at most, a 3% delta.



	1	2	4	8	12	16
BM.48.IB	0.08	0.17	0.33	0.65	0.95	1.22
BM.44.IB	0.08	0.16	0.31	0.62	0.90	1.17
VM.44.144.LatSens.IB.SRIOV	0.08	0.16	0.30	0.60	0.86	1.11

Figure 34. NAMD performance comparison between virtual and bare-metal systems



	1	2	4	8	12	16
BM.48.IB	100%	98%	97%	96%	94%	90%
BM.44.IB	100%	98%	97%	96%	94%	91%
VM.44.144.LatSens.IB.SRIOV	100%	98%	96%	94%	91%	87%

Figure 35. NAMD strong scaling comparison between virtual and bare-metal systems

5 Summary

In this document, we walked through the steps to configure IB SR-IOV on NVIDIA Mellanox ConnectX-5 adapter cards in vSphere 7.x. We evaluated this setup's functionality with three 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 aimed to construct this technical guide to remain useful even when software versions and products evolve in the future. We hope you found it insightful and will return as we expand this series of guides to other topics, including how to enable RoCE SR-IO, DirectPath I/O of IB and RoCE, and the performance differences when using IB and RoCE for HPC workloads.

6 References

[1] Single Root I/O Virtualization (SR-IOV)

[2] NVIDIA ConnectX-5 InfiniBand adapter cards

[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 OS OFED Download link

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

[11] Intel Cluster Checker

[12] SB7800 InfiniBand EDR 100Gb/s Switch System

[13] NVIDIA MLNX_OFED Documentation Rev 5.7-1.0.2.0

[14] NVIDIA Firmware Tools (MFT)

About the Author

Yuankun Fu has been a 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 PhD 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 contributions to the study. The author also thanks Pamela Gorder and Julie Brodeur for editing this document.