VMware Paravirtual RDMA for High Performance Computing

PVRDMA Setup and Performance Study - November 29, 2021

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

Table of Contents

Introduction

High Performance Computing (HPC) workloads require ever-increasing demands for high throughput and low-latency networking. InfiniBand and Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) have played a role in enabling these workloads to scale with superior performance to typical TCP-based solutions. Such devices can be configured to work with a virtual machine (VM) on ESXi through VMDirectPath I/O (passthrough) mode, but doing so disables certain features for the VM, such as vSphere HA, DRS, and vMotion.

Through VMware PVRDMA—the paravirtual RDMA device that takes advantage of any RoCE-capable hardware on an ESXi host—ESXi can support running workloads with high performance while still maintaining the flexibility offered by virtualization, including operational flexibility, better resilience, higher availability, and lifecycle management capabilities. This high performance is possible because PVRDMA allows for direct memory transfers (DMA) between guest memory to the RoCE-capable hardware without CPU intervention. Using the RDMA protocol enables the potential for memory-to-memory zero-copy transfers between guests on two different hosts.

In this paper, we first show you how to configure a PVRDMA device in a vSphere environment, including the ESXi host setup, VM setup, and guest operating system setup. We then demonstrate the performance benefits of PVRDMA over TCP and show close performance to RDMA-capable hardware used in passthrough mode, all while supporting vSphere HA, DRS, and vMotion.

PVRDMA Setup

PVRDMA is a virtual device that you can add to a VM. You need to set this up on both vCenter Server and the guest operating system. The instructions from VMware Docs have been replicated below with screenshots to follow along more easily. Please also see the KB article for supported operations and operating systems, limitations, and new features.

vCenter Server Setup

To create a distributed switch (DVS) with RoCE-capable NICs as uplinks:

1. Go to the **Networking** tab in the menu on the left, right-click on your datacenter, and go to **Distributed Switch > New Distributed Switch...** and click on it:

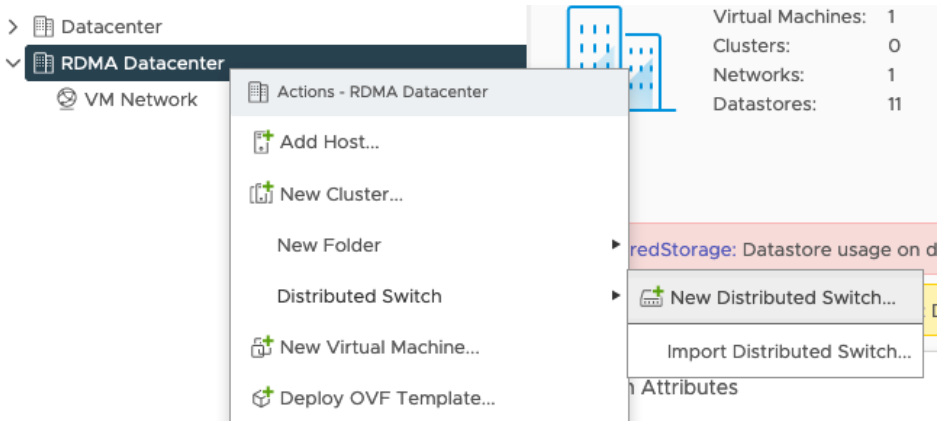


Figure 1. Choose Datacenter > Distributed Switch > New Distributed Switch

2. After you create the distributed switch, right click on it, and then click on **Add and Manage Hosts…**:

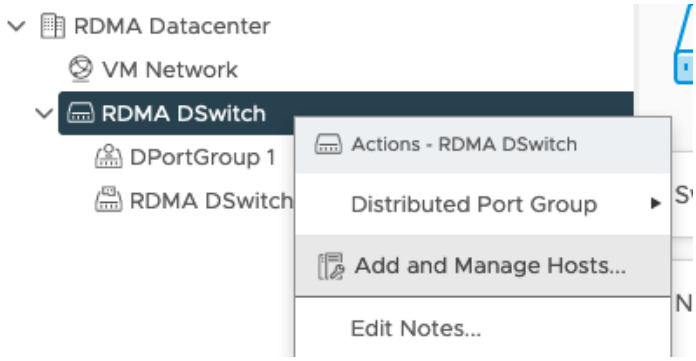


Figure 2. Click the new switch and click on Add and Manage Hosts …

3. Click on **Add hosts**, then go the next page.
4. Click on **New hosts…** and select the hosts you want to add to the distributed switch, then go to the next page.
5. Select the vmnic representing the RoCE-capable NIC you want to use as the uplink for the PVRDMA device.
    a. if you are unsure of which uplink you need to select, you can click on the host in the **Hosts and Clusters** menu.
    b. Click on the **Configure** tab.
    c. Under **Networking,** look at **Physical adapters.**
    d. By clicking on the listed vmnics, you can view under the **RDMA** tab if the vmnic supports RDMA. The vmnic needs to support either the **RoCE v1** or **RoCE v2** protocols.

e. Click on **Assign uplink**, select **Uplink 1**, then click on **OK**. The desired NIC should say **(Assigned)** and show **Uplink 1** under the **Uplink** column.

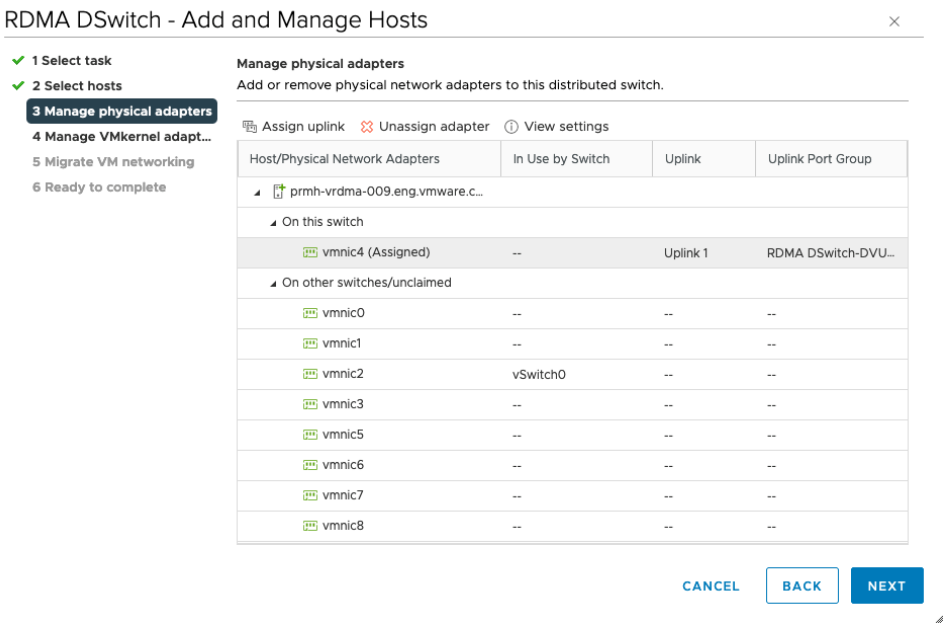


Figure 3. Add and Manage Hosts screen

6. Keep going to the next page until you finish creating a distributed switch.

> **Note:** Adding more than one uplink to the distributed switch is not supported. Please have only a single uplink per host in the distributed switch.

ESXi Host Setup

PVRDMA needs to use a VMKernel network interface (vmknic) to send and receive control data with other PVRDMA devices. In order to do so, you need to set a vmknic for use by PVRDMA.

To set a vmknic that PVRDMA can use:

1. Select the host you need to configure the vmknic for PVRDMA on, go to the **Configure** tab, and select **Advanced System Settings**.

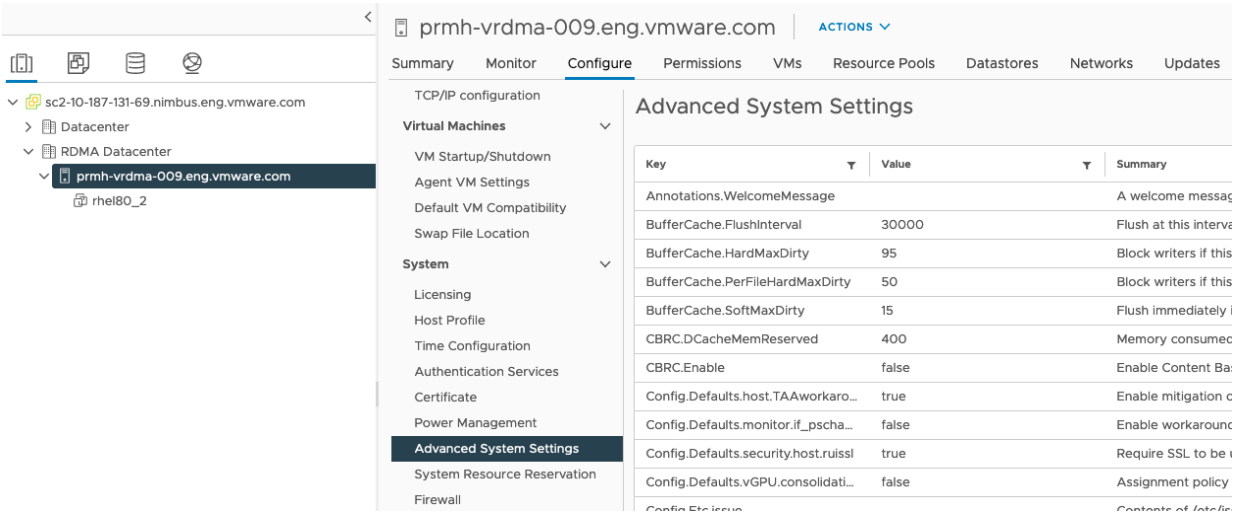


Figure 4. Advanced System Settings screen

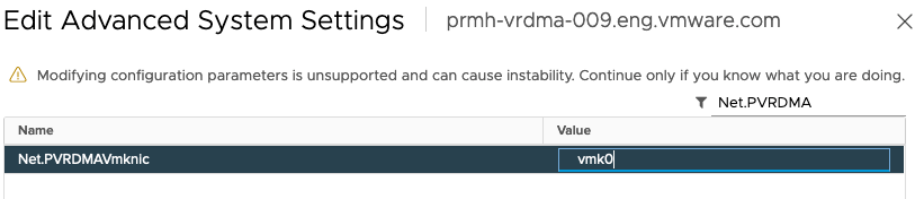


Figure 5. Edit Advanced System Settings screen

Note: The vmknics of hosts on which VMs with PVRDMA reside must be able to reach each other in order for the PVRDMA devices to communicate over the RDMA-capable NIC.

To edit firewall settings to allow PVRDMA control data traffic, follow these steps in the UI or enter the commands as shown at the end of the steps:

1. Select the host you need to edit the firewall settings on.
2. Go to the **Configure** tab.
3. Select **System > Firewall**
4. Click the **Edit...** button in the top right corner.
5. Filter for **pvrdma**.
6. Make sure the checkbox is checked.
7. Click **OK**.

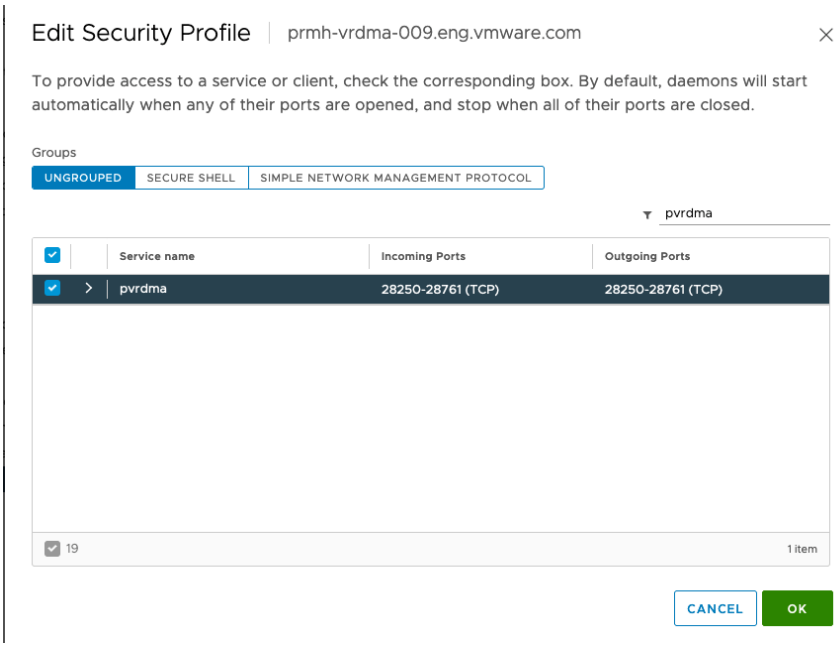


Figure 6. Edit Security Profile screen

The above steps can be done using the ESXi shell instead. Just run the two commands below:

```
esxcli system settings advanced set -o /Net/PVRDMAVmknic -s vmk0
esxcli network firewall ruleset set -e true -r pvrdma
```

Virtual Machine Setup

PVRDMA devices are supported starting on virtual machine hardware version 13 and onwards. Make sure the VM you would like to add the PVRDMA device to is at least **HWv13 or higher**. **If it is not:**

1. Right click on the VM.
2. Go to **Compatibility > Upgrade VM Compatibility…**
3. Upgrade the VM to at least HWv13 (vSphere 6.5+).

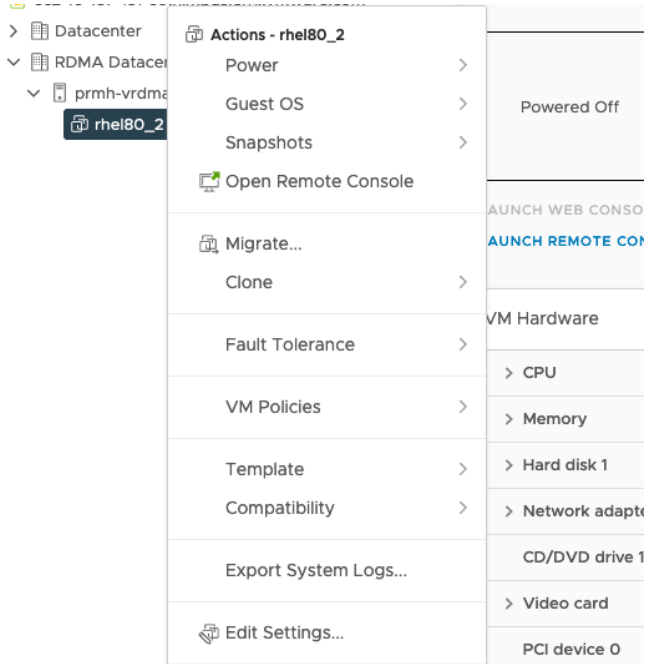


Figure 7. Actions screen

Now, you can add the PVRDMA device to the virtual machine:

1. Right-click on the VM and click on **Edit Settings...**.
2. Click on **Add New Device** and under **Network**, select **Network Adapter**.

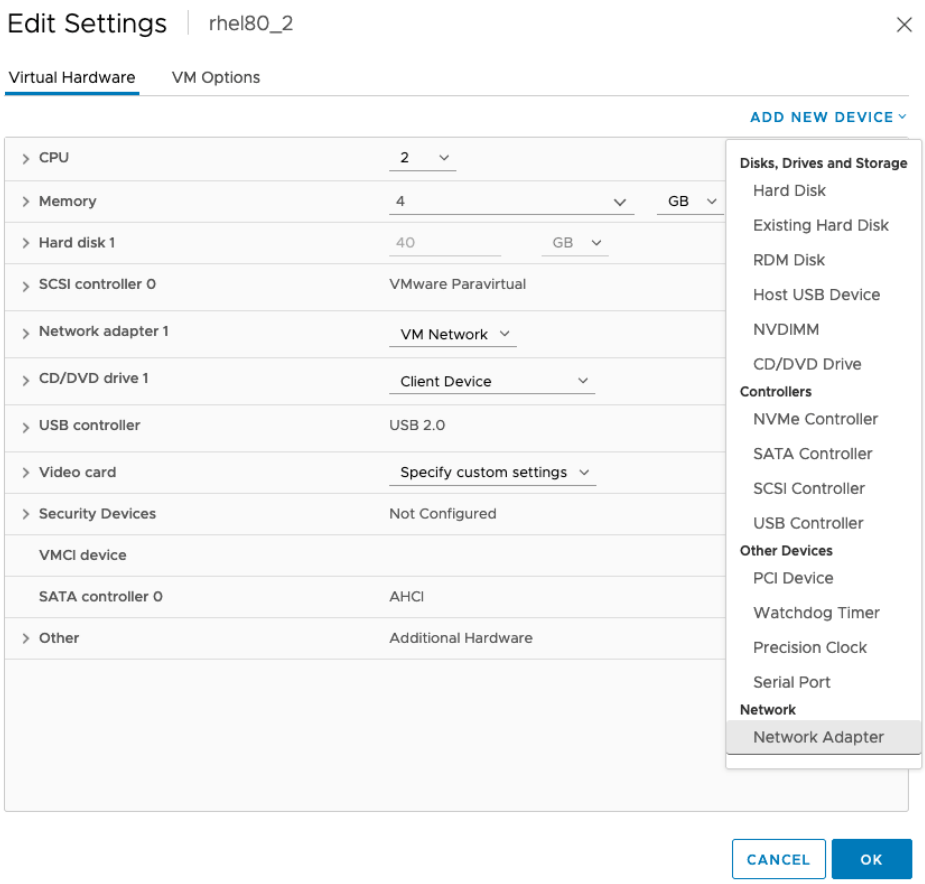


Figure 8. Add a new network adapter

3. Change the adapter type to **PVRDMA** and click on the dropdown menu for **New Network,** and then click on **Browse….**

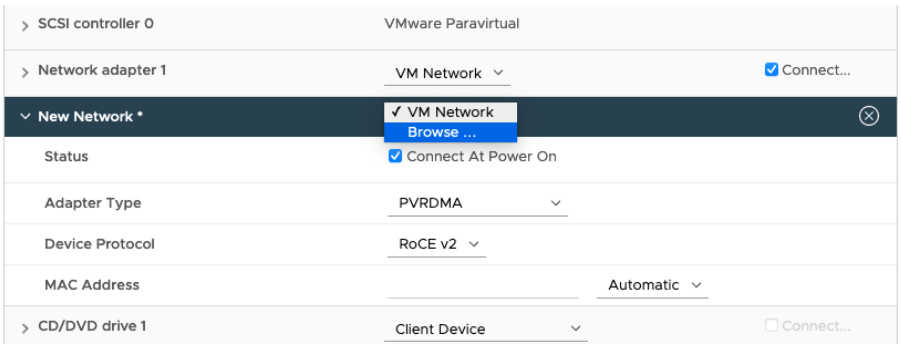


Figure 9. Adapter Type: PVRDMA > New Network > Browse

4. Select the distributed port group corresponding to the distributed switch that you created and click **OK**. Then click **OK** again to finish editing the VM settings.

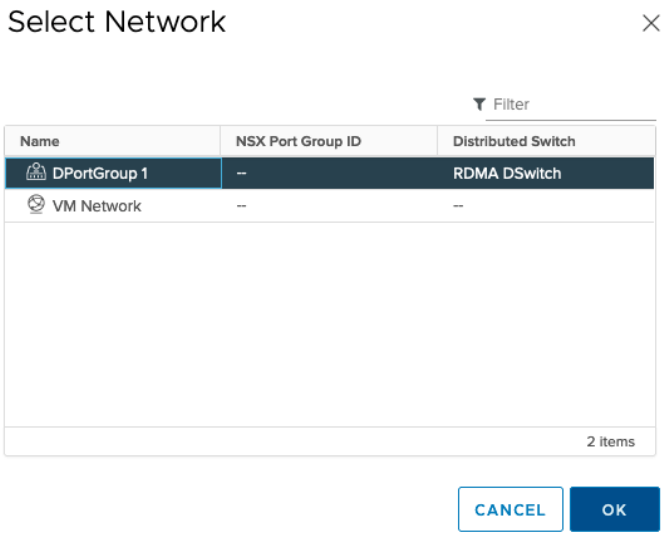


Figure 10. Select Network screenshot

Guest Operating System Setup

PVRDMA requires driver support in the guest operating system. Below are the supported operating systems for the two different protocols supported by PVRDMA:

- **In RoCEv1:** PVRDMA supports CentOS 7.2 or later, RHEL 7.2 or later, SLES 12 SP1 or later, Oracle Linux 7 UEKR4 or later, Ubuntu LTS Releases 14.04 or later, and Photon OS 3.0 or later.
- **In RoCEv2:** PVRDMA supports CentOS 7.3 or later, RHEL 7.3 or later, SLES 12 SP3 or later, Oracle Linux 7 UEKR5 or later, Ubuntu LTS Releases 16.04.2 or later, and Photon OS 3.0 or later.

Userspace applications also require userspace libraries for PVRDMA. The easiest way to install this is to install rdma-core. You can also install rdma-core packages using your operating system's package manager.

Troubleshooting

To confirm that PVRDMA is using the underlying RoCE-capable NIC, you can check application performance from the VM or inspect the PVRDMA device backend settings from the ESXi shell.

1. Running a simple server-client bandwidth test can tell you if the PVRDMA device is operating as expected. One set of microbenchmarks for RDMA is perftests. If `ib_send_bw` gives a

bandwidth close to line rate on larger packet sizes, the PVRDMA device should be configured correctly. For example, see the output of `ib_send_bw` on a 40G RoCE-capable NIC:

```
[root@prmh-lab-net3-dhcp232-029 ~]# ib_send_bw -x 1 192.168.51.31
---------------------------------------------------------------------------------------
                    Send BW Test
Dual-port       : OFF          Device         : vmw_pvrdma0
Number of qps   : 1            Transport type : IB
Connection type : RC           Using SRQ      : OFF
PCIe relax order: ON
ibv_wr* API     : OFF
TX depth        : 128
CQ Moderation   : 1
Mtu             : 1024[B]
Link type       : Ethernet
GID index       : 1
Max inline data : 0[B]
rdma_cm QPs     : OFF
Data ex. method : Ethernet
---------------------------------------------------------------------------------------
local address: LID 0000 QPN 0x2a61 PSN 0x244ed6
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:51:32
remote address: LID 0000 QPN 0x0002 PSN 0x9a7160
GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:51:31
---------------------------------------------------------------------------------------
#bytes     #iterations    BW peak[MB/sec]    BW average[MB/sec]   MsgRate[Mpps]
65536      1000             4407.61            4395.71            0.070331
---------------------------------------------------------------------------------------
```

Figure 11. Output of ib_ send_ bw on a 40G ROCE-capable NIC

2. If you have access to the ESXi shell, you can use the `vsish` command to check if the PVRDMA device is correctly configured to use the RDMA-capable NIC. The command to do so is:

   ```
   vsish -e cat /vmkModules/vrdma/pvrdmaDevices/<VM World ID>_<Adapter Index>/properties
   ```

   with <VM World ID> found by running `esxcli vm process list`, and <Adapter Index> likely to be `0`, unless there are multiple PVRDMA devices added to the virtual machine. If `Physical HCA available` is `1` in the output, that means the RDMA-capable NIC is in use by the PVRDMA device.

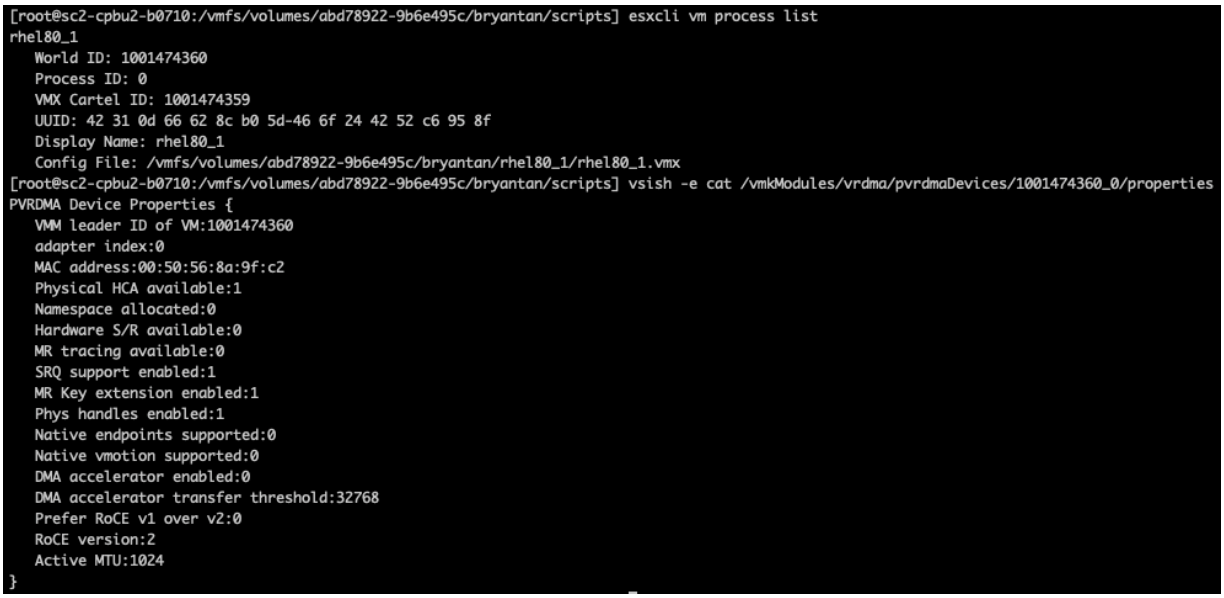


Figure 12. Output of vsish command

Through the ESXi shell, you can also view the available RDMA devices using `esxcli rdma device list` and view device stats of each RDMA device using `esxcli rdma device stats get -d vmrdmaX`. RDMA device stats only works on PCI device physical functions and may not work with certain RDMA or PVRDMA configurations.

Performance Testing with HPC Application OpenFOAM

OpenFOAM is a popular open-source software used for solving computational fluid dynamics (CFD) problems, including aerodynamics, simulation of industrial flows, combustion systems, and electronic design automation. CFD is the largest user of HPC infrastructures in engineering. OpenFOAM uses MPI for parallel distributed workloads.

Message passing interface (MPI) is a standard for multiple processor programming of HPC code that runs on a variety of machines: from small clusters to giant supercomputers. Numerous scientific applications that run in a distributed way use the MPI library and take advantage of its communication operations.

By demonstrating PVRDMA performance and utility in OpenFOAM with MPI, we show that it is possible to achieve good performance with virtualization in applications commonly used in the HPC space.

Testbed Setup

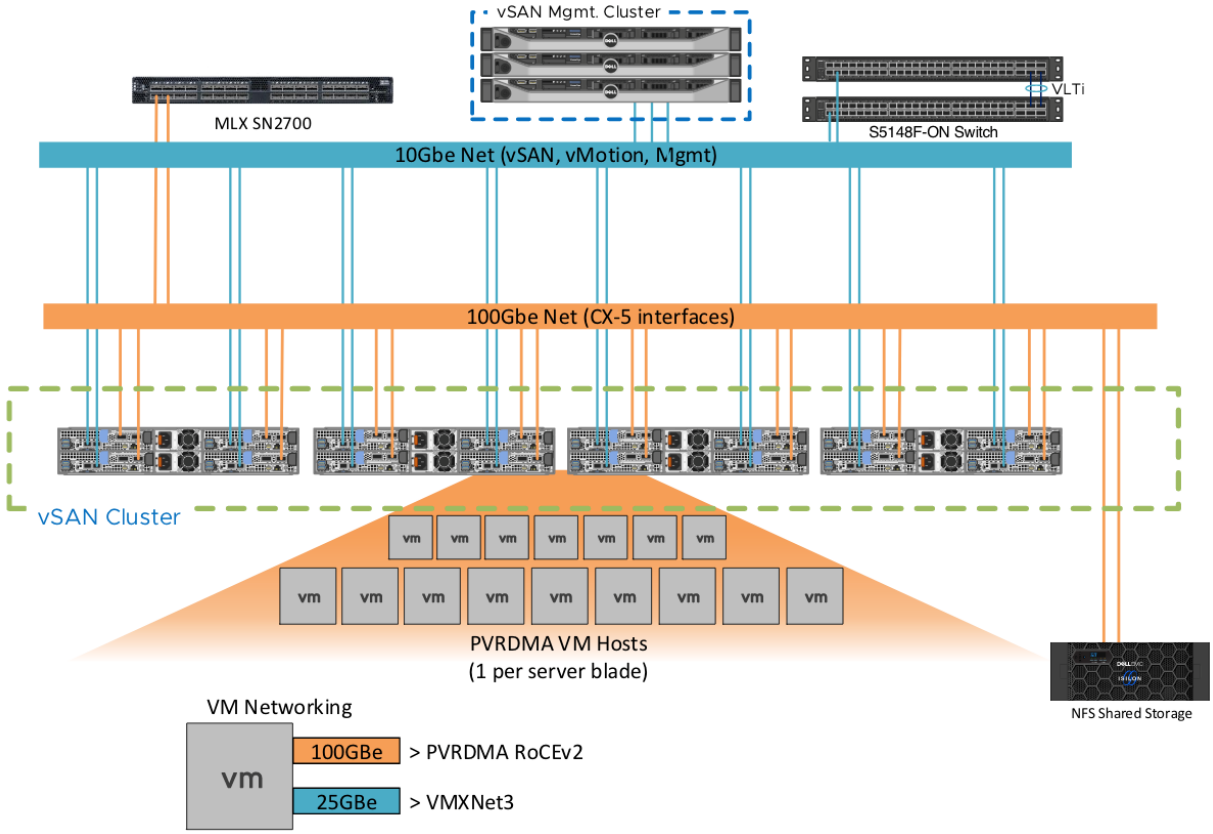


Figure 13. Cluster and OpenFOAM setup

The testbed consisted of:

- 8 ESXi 7.0 hosts on PowerEdge C6420 with Intel Xeon Gold 6148 CPU (20 core/40 thread), 200GB RAM, NVIDIA Mellanox ConnectX-5 Ex 100GbE NIC for RDMA
- NVIDIA Mellanox ConnectX-5 Ex NICs connected using a 100GbE NVIDIA Mellanox switch
- CentOS 7.6, 20 vCPUs, 100GB RAM, VM on VSAN datastore, one VM per ESXi host
- OpenMPI version 4.1.0, using openib BTL for RDMA transport
- OpenFOAM version 8, running the cavityFine experiment from the OpenFOAM user guide

  The cavityFine test runs a fluid dynamics simulation with the given parameters. The simulated case is a two-dimensional square with stationary walls on 3 sides and a moving wall on the fourth.

Results

We ran the case for the cavityFine experiment with a start time of 0.5 seconds and an end time of 0.50003 seconds. We aimed to keep the amount of work done per node equal as we scaled up the number of nodes, a method known as *weak scaling*, similar to what customers do by running larger problems or the same problem with higher resolution when scaling up on larger clusters. To do so, we had to double the mesh size to be solved and halved the time delta for the solver when the number of processes was doubled. We ran a single VM on each ESXi host, with each VM running 20 MPI processes, and tested with 1, 2, 4, and 8 VMs ranging from 20 MPI processes to 160 MPI processes.

The below table and graphs show the performance of PVRDMA, and its comparison with passthrough (using NVIDIA Mellanox ConnectX-5 Ex NIC with SR-IOV virtual function) and TCP. The PVRDMA device is configured to use RDMA over the NVIDIA Mellanox ConnectX-5 Ex NIC, and TCP was run over the NVIDIA Mellanox ConnectX-5 Ex NIC as well, without RDMA but still taking advantage of the NIC's higher performance. Case names encode the number of hosts used for the test (n) and the number of MPI processes run per host (np).

Note: The 1 node, 20 MPI processes run does not use the underlying transport and had an average run time of 119.61 seconds.

Cases	Mesh Size	Time Delta (s)	PVRDMA (s)	Passthrough (s)	TCP (s)
n2np40	(2000 2000 1)	0.0000012	336.49	279.78	501.19
n4np80	(4000 2000 1)	0.0000006	768.74	601.52	1014.25
n8np160	(4000 4000 1)	0.0000003	1583.22	1307.51	2878.94

Table 1. OpenFOAM weak scaling performance on different transports, measured in seconds (s)

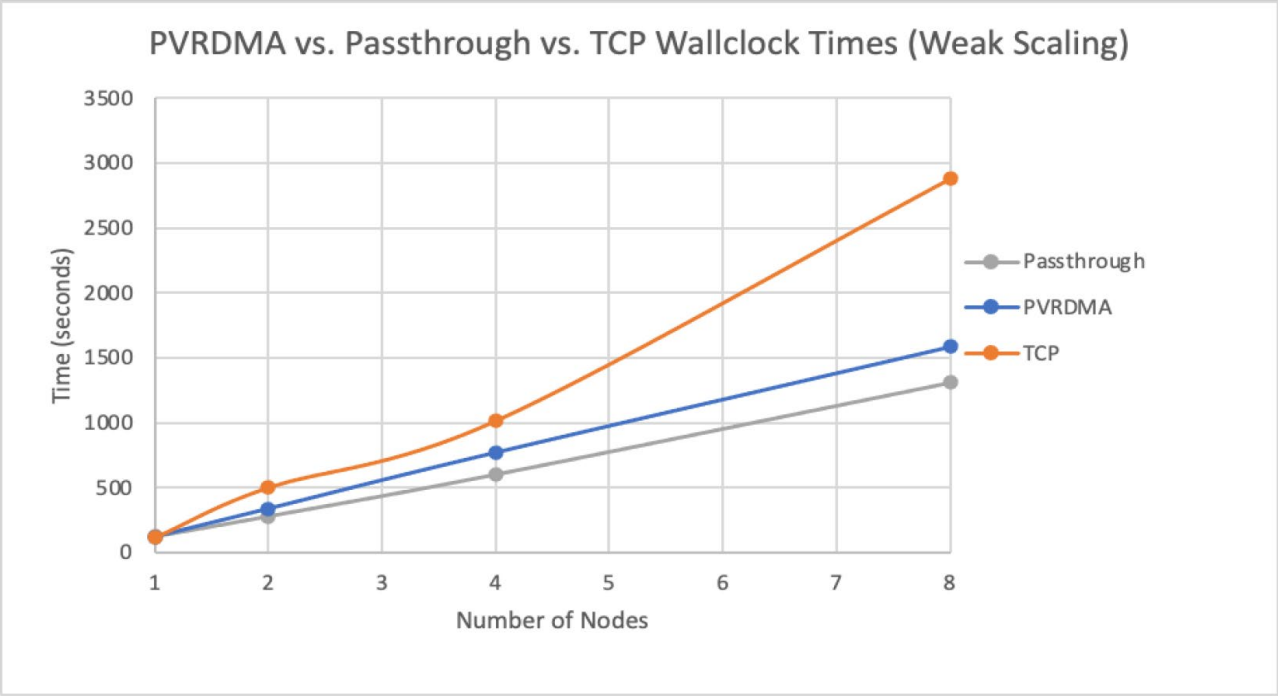


Figure 14. Weak scaling performance comparison between passthrough, PVRDMA, and TCP (lower is better)

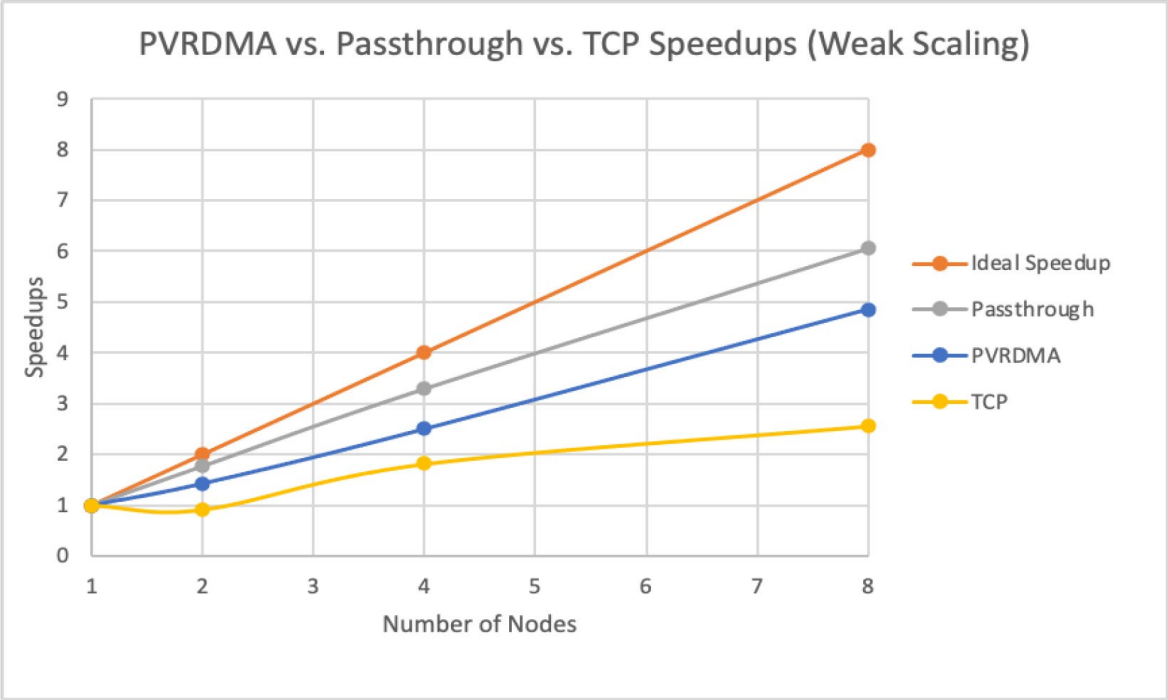
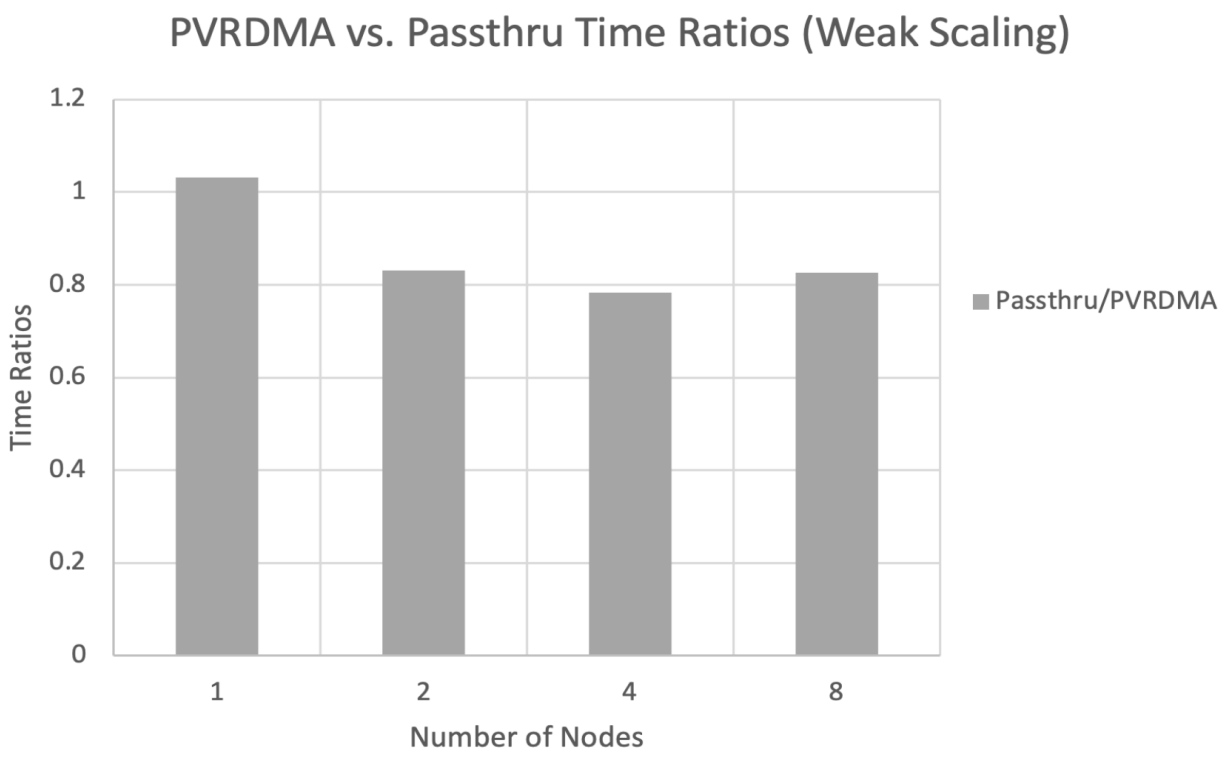
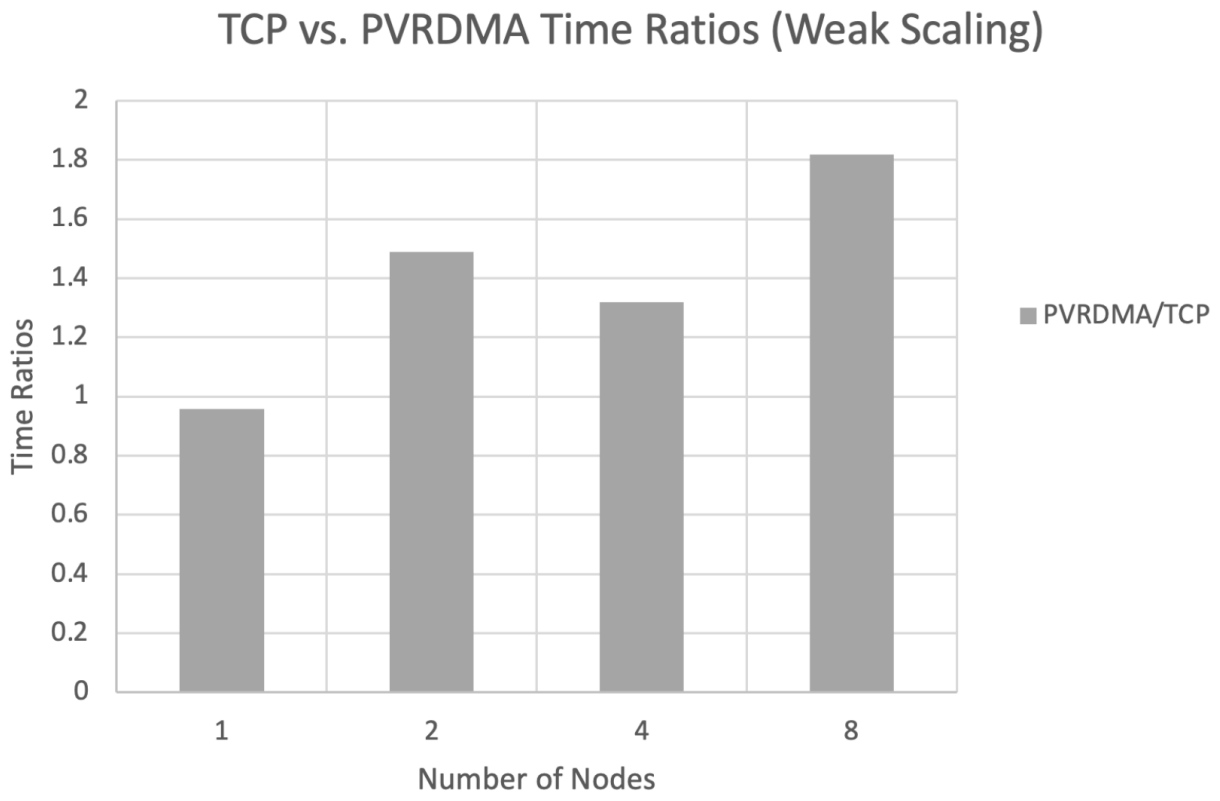


Figure 15. Weak scaling speedup comparison between passthrough, PVRDMA, TCP, and the ideal case (higher is better)

In figures 14 and 15 we can see that PVRDMA runs closely to passthrough, and even though there is some cost for virtualization, it offers vSphere features like consolidation, HA, vMotion, and more.





From figure 16, we can see that PVRDMA performance is approximately 80% of the passthrough performance using the same RoCE-capable NIC, and from figure 17 the PVRDMA performance is 30%-80% better than the TCP performance. While using an RoCE-capable NIC in passthrough mode is still the most performant, PVRDMA bridges the gap between TCP and passthrough, while still providing the benefits of virtualization.

Conclusion

Through the experiments with OpenFOAM, PVRDMA is shown to achieve good performance on an HPC application. The performance of PVRDMA exceeds that of TCP, and while using an RDMA-capable NIC in passthrough mode allows better performance, PVRDMA can take full advantage of virtualization benefits, such as vSphere HA, DRS, and vMotion. In the future, we also plan to demonstrate the performance and functionality benefits of using PVRDMA for machine learning/deep learning workloads.

About the Authors

Bryan Tan is a senior member of technical staff working on virtualized Remote Direct Memory Access (RDMA) and other virtual devices within VMware vSphere.

Acknowledgements

Chris Gully and Dell Technologies Solutions brought up the OpenFOAM lab environment. Vishnu Dasa assisted with bringing up the setup, running OpenFOAM on various configurations, and editing and reviewing this writeup. Na Zhang and the rest of the CTO Advanced Technologies Group gave input regarding OpenFOAM scaling experiments. Na also helped edit and clarify parts of this writeup.