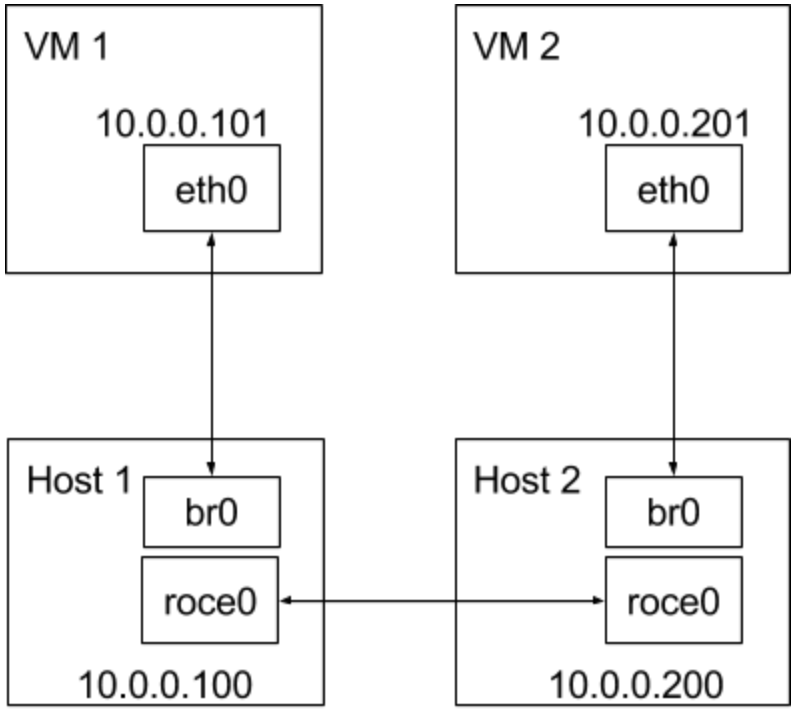
Instructions on How to Run PerfTest with vRDMA

1 Basic Network Structure

The examples in this document are demonstrated based on the following network structure:



Linux bridges are created over the RoCE port on each host, each VM is started with a virtio-net (eth0) device combined to the virtual bridge on the host. VMs are able to communicate via TCP/IP, and also RDMA verbs when the vRDMA drivers are correctly loaded in the VM (not explicitly presented as a device).

2 Preparing the Host Drivers and Network

First, load kernel modules and prepare the host side (the same for host 1 and host 2)

```
host1:~# sudo service opensm stop
host1:~# sudo rmmod rdma_ucm
host1:~# sudo rmmod ib_ucm
host1:~# sudo rmmod ib_ipoib
host1:~# sudo rmmod vhost_rdmacm
host1:~# sudo rmmod rdma_cm
```

```
host1:~# sudo rmmod ib_cm
host1:~# sudo rmmod mlx4_ib
host1:~# sudo rmmod ib_sa
host1:~# sudo rmmod iw_cm
host1:~# sudo rmmod ib_umad
host1:~# sudo rmmod ib_mad
host1:~# sudo rmmod ib_uverbs
host1:~# sudo rmmod ib_ipoib
host1:~# sudo rmmod vhost_rdmacm
host1:~# sudo rmmod rdma_cm
host1:~# sudo rmmod mlx4_ib
host1:~# sudo rmmod mlx5_ib
host1:~# sudo rmmod vhost_hyv
host1:~# sudo rmmod ib_core
host1:~# sudo modprobe ib_core
host1:~# # load vhost and necessary host modules
host1:~# sudo modprobe vhost
host1:~# sudo insmod /path/to/vhost_hyv.ko
host1:~# sudo modprobe mlx4_ib
host1:~# sudo modprobe rdma_cm
host1:~# sudo modprobe rdma_ucm
host1:~# sudo insmod /path/to/vhost_rdmacm.ko

host1:~# # used by ibv_devinfo
host1:~# sudo modprobe ib_uverbs

host1:~# # configure the ports as needed, both RoCE and InfiniBand are supported
host1:~# sudo connectx_port_config

ConnectX PCI devices :
|----------------------------|
| 1             0000:82:00.0 |
|----------------------------|

Before port change:
auto (ib)
auto (ib)

|----------------------------|
| Possible port modes:       |
| 1: Infiniband              |
| 2: Ethernet                |
| 3: AutoSense               |
|----------------------------|
Select mode for port 1 (1,2,3): 2
Select mode for port 1 (1,2,3): 1

host1:~# # start opensm service
host1:~# sudo modprobe ib_umad
host1:~# sudo service opensm start

host1:~# # create a linux bridge over roce port
host1:~# sudo brctl addbr br0
host1:~# sudo brctl addif br0 roce0
host1:~# sudo ifconfig roce0 0
host1:~# sudo ifconfig br0 10.0.0.100/24

host1:~# # enable IPoIB for the ib port
host1:~# sudo modprobe ib_ipoib
host1:~# sudo ifconfig ib0 10.0.1.100/24
```

3 Run Perftest on OSv guest

We use perftest to test network bandwidth and latency. First we need to install the perftest and build vrdma-ulibs that we need

```
host1:~# scripts/build image=perftest,open-mpi,open-mpi-hello,netpipe,vrdma-ulibs,cli -j16
```

Configure the configuration file of the RDMA device, the file should contain the GUID of the device, for example:

```
host1:~# cat /path/to/hyv_config

f452:1403:0022:b970
```

Next, we start the server application listening on RDMA port 2 on one of the hosts:

```
host1:~# ./scripts/run.py -d -e "/tools/ib_write_bw.so -i 2" -V --pass-args '--name osv1
--device virtio-hyv-pci,config_path=/path/to/hyv_config -netdev bridge,id=eth0 -device
virtio-net-pci,netdev=eth0,id=br0,mac=52:54:00:12:34:56'
```

The above command will start OSv with perftest server, and get an IP from DHCP. In the example, the VM1 on host1 has an IP address 10.0.0.101. VM2 on host 2 has an IP address 10.0.0.201.

Next step is to start a perftest client on the same RDMA port with the server IP (for example 10.0.0.201)

```
host2:~# ./scripts/run.py -d -e "/tools/ib_write_bw.so -i 2 10.0.0.201" -V --pass-args
'--name osv2 --device virtio-hyv-pci,config_path=/path/to/hyv_config -netdev bridge,id=eth0
-device virtio-net-pci,netdev=eth0,id=br0,mac=52:54:00:12:34:56'
```

The above command will test the write bandwidth between server and client. Sample output should look like:

```
vRDMA: udata_create
vRDMA: do_hcall, npargs: 1
vRDAM: pargds[0]: size: 8, addr: 0xffffa00001716870
vRDMA: got hcall ack.
vRDMA: async hcall.
ibv_cmd_modify_qp
attr_mask: 12e01
cmd.attr_mask: 0
vrdma_modify_qp
attr_mask: 77313
hqp->ibqp.qp_type: 2
vRDMA: udata_create
```

```
vRDMA: do_hcall, npargs: 1
vRDAM: pargds[0]: size: 8, addr: 0xffffa00001716870
vRDMA: got hcall ack.
vRDMA: async hcall.
---------------------------------------------------------------------------------------
 #bytes     #iterations    BW peak[MB/sec]    BW average[MB/sec]   MsgRate[Mpps]
 2          5000             12.34              12.21                6.399550
 4          5000             25.05              24.78                6.494802
 8          5000             50.23              49.45                6.481639
 16         5000             100.21             99.33                6.509434
 32         5000             200.41             198.76               6.512843
 64         5000             401.84             386.90               6.339011
 128        5000             803.68             791.92               6.487402
 256        5000             1603.30            1586.13              6.496783
 512        5000             3206.59            3175.03              6.502456
 1024       5000             5518.93            5482.74              5.614331
 2048       5000             5810.08            5803.33              2.971305
 4096       5000             5898.02            5897.34              1.509719
 8192       5000             5964.02            5963.18              0.763287
 16384      5000             5992.16            5989.90              0.383354
 32768      5000             5997.93            5997.07              0.191906
 65536      5000             6001.77            6001.20              0.096019
```

3 Run Perftest on Linux guest

After the host is configured as shown in step 2, now we can start Linux guests 1 and 2 by

```
host1:~# qemu-system-x86_64 -enable-kvm -display sdl -boot d -drive file=/path/to/image -m
4096M -cpu Nehalem -smp 2 -name "VM1-3.18" -device
virtio-hyv-pci,config_path="/path/to/hyv_config" -netdev bridge,id=hn0 -device
virtio-rdmacm-pci -device virtio-net-pci,netdev=hn0,id=br0,mac=52:54:00:12:34:57 &

host2:~# qemu-system-x86_64 -enable-kvm -display sdl -boot d -drive file=/path/to/image -m
4096M -cpu Nehalem -smp 2 -name "VM1-3.18" -device
virtio-hyv-pci,config_path="/path/to/hyv_config" -netdev bridge,id=hn0 -device
virtio-rdmacm-pci -device virtio-net-pci,netdev=hn0,id=br0,mac=52:54:00:12:34:56 &
```

After the VMs are started, prepare the guest 1 and 2 using the following commands to load the guest vRDMA drivers (the same on both guests):

```
guest1:~# sudo rmmod virtmlx4_ib
guest1:~# sudo rmmod virtio_hyv
guest1:~# sudo rmmod virtio_rdmacm
guest1:~# sudo rmmod ib_uverbs
guest1:~# sudo rmmod ibcore
```

```
guest1:~# sudo modprobe ib_core
guest1:~# sudo modprobe ib_uverbs
guest1:~# sudo insmod /path/to/virtio_hyv.ko
guest1:~# sudo insmod /path/to/virtmlx4_ib.ko
```

For Linux VM, a specific Perftest 3.0-0.18 has to be configured and installed, which can be downloaded from:
https://www.openfabrics.org/downloads/perftest/perftest-3.0-0.18.gb464d59.tar.gz

Now we are ready to conduct the performance tests.To test write bandwidth, first start a request on the server side (VM 1):

```
guest1:~# ib_write_bw -i 2
```

Then, use the following command on the client side with the server's IP (for example 10.0.0.201):

```
guest2:~# ib_write_bw -i 2 -a 10.0.0.201
```

To test write latency, use the following commands on the server and client side respectively.

```
guest1:~# ib_write_lat -i 2
guest2:~# ib_write_lat -i 2-a 10.0.0.201
```

To test read bandwidth:

```
guest1:~# ib_read_bw -i 2
guest2:~# ib_read_bw -i 2-a 10.0.0.201
```

To test read latency:

```
guest1:~# ib_read_lat -i 2
guest2:~# ib_read_lat -i 2 -a 10.0.0.201
```

Optionally, one can specify and pin a process to particular CPU by using taskset. For example, when we specify the write bandwidth test to run on core 13 and 14, one can use

```
guest1:~# taskset 0x03000 ib_write_bw -i 2
```