Progress Report of RDMA Performance in Multi-Tenant Virtualization Environment – VMs and Containers

ZHU, ZHUANGDI

zhuangdizhu@yahoo.com

Supervised by

Salman Baset, Michael Hines and Alex Liu

Introduction

This project is focusing on the measurement study of RDMA performance in multi-tenant environment for VMs and Containers. The following are some of my thoughts and questions I expect to answer in this project.

- **Why is this topic meaningful or important?**
  RDMA-enabled devices, like InfiniBand, provides high-speed (40~100 Gb/sec bandwidth from QDR, FDR to EDR) network for high performance computing systems. However, it is still rare to be utilized in the mainstream cloud systems. In recent years, as the price of IB devices are much cheaper [16], it has already been a trend to see more RDMA-enabled Ethernet and IB adapters deployed in commodity computing systems. With the development of virtualization-enabled technologies, like SR-IOV, it is of practical significance to set up cloud systems based on high performance RDMA-enabled network to provide virtualized systems low-latency interconnects to users. On the other side, docker/container provides a lightweight virtualization technique compared to VM. Hence, it may be a wise option to set up an RDMA-enabled cloud system based on dockers/LXCs rather than VMs, especially for multi-tenant environment. Above all, this study would give an evaluation of these techniques and develop a guide for others in deploying such a cloud system in practice.

- **What are the challenges of this project?**
  From my experience in last two months, the key part of this study is setting up the correct software stack on the suitable hardware devices. For example, I failed to install OFA/OFED, the supporting software for RDMA, on my VM on VirtualBox of my Macbook last December purely due to the OS and OFED compatibility issues. Another example is that I realized that the CPU does not support Intel VT-D that results that SR-IOV cannot be used for multi-tenant VMs after many efforts in passthrough of VMs. Its complexity lies on the compatibility and configuration of the entire software stack, from BIOS settings, IB drivers, compatible Linux OS versions, Mellanox or OFA OFED, supporting system libraries (e.g., libvirt, etc.), virtualization hypervisor (KVM or Docker), compatible Guest OS or image, configuration of network on VM/container, programming API to utilize either TCP or RDMA for transferring data between hosts/VMs/containers and the toolkit/software/program for performance measurement.

Proposed Experiments

The proposed experimentation paths are as the following table. A series of 10 groups of performance experiments are going to measure both the latency and bandwidth for transferring the same amount of data (say 1GB) in different settings.

	Party A		Party B		Protocol	Transfer
1	Host	1	Host	1	TCP	single transfer and four transfers
2	Host	1	Host	1	RDMA	single transfer and four transfers
3	VM	1	VM	1	TCP	single transfer and four transfers
4	VM	1	VM	1	RDMA	single transfer and four transfers
5	LXC	1	LXC	1	TCP	single transfer and four transfers
6	LXC	1	LXC	1	RDMA	single transfer and four transfers
7	VM	4	VM	4	TCP	single transfer from 1 VM to 1 VM
8	VM	4	VM	4	RDMA	single transfer from 1 VM to 1 VM
9	LXC	4	LXC	4	TCP	single transfer from 1 container to 1 container
10	LXC	4	LXC	4	RDMA	single transfer from 1 container to 1 container

During the work, I have used different approaches/programs in evaluating the network performance, as follows.

- **qperf**
  qperf is a Linux utility to measure bandwidth and latency between two nodes [1]. It can work over TCP/IP as well as RDMA transports. Hence it is a convenient tool for performance testing in our experiments.
- **rdma_cm server/client program**
  rdma_cm server/client program is adjusted form a sample code[4] for the transfer of large messages between hosts using the InfiniBand verbs library[2]. The inner basic flow-control protocol breaks messages into segments and then uses the RDMA-write-with-immediate-data (IBV_WR_RDMA_WRITE_WITH_IMM) operation to transfer these segments[3]. And I may evaluate/consider whether this is useful for future tests.
- **MPI send/receive program in C**
  MPI programming provides a high-level communication interface for operating multiple hosts/VMs to transfer messages concurrently. As the proposed experiments are finally carried out for multi-tenants, it is convenient to have tests using MPI libraries. Basic MPI send/receive codes are used in the earlier tests. And I may valuate/consider whether this is useful for future tests as well.

Experimental Systems

I have used several systems in different stages as following. Each of them has some reasons not satisfying the needs to perform all tests. The first one is my own laptop with quite limited memory to start 4 VMs. The second one is the resource of ANU when I was an exchange student in Australia. The pros are the VMs and HPC system are equipped with the latest IB FDR devices with SR-IOV support enabled so that I could finish a complete MPI performance test with 2 nodes and 2 VMs. But I cannot perform tests with 8 VMs limited by resource and there is no docker/container support. Since my ANU email account will expire from March, I may not be able to access them. The third systems include two pure metal servers that I have full access and control with IB QDR device installed. The problem is its CPU (Intel i7-3770K) does not support Intel VT-d and hence SR-IOV is not possible. I haven't started to use the last system, IBM's openstack cloud yet and that would be the platform for my future work.

Table 1. MacBook Air

Item	Device
Processor	Intel(R) Core i5 Dual Core @ 1.4GHz
Memory	4 GB
Operating System	Mac OS X 10.9.5
VM Software	VirtualBox 4.3.20
VM Guest OS	Linux CentOS 6.5
VM Kernel Version	2.6.32-431.el6.x86_64
Guest Memory	1 GB
System Location	My personal computer

Table 2. ANU's OpenStack Cloud System [17]

Item	Device
Processor	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz
Memory	128 GB
Network	Mellanox ConnectX-3 VPI (MCX383A-FCxx) Single FDR IB (56Gb/s)
Operating System	Linux CentOS 6.5
Kernel Linux Version	2.6.32-431.20.3.el6.x86_64
OFED	Mellanox OFED-3.12
VM Hypervisor	OpenStack (KVM)
VCPUS	4
Guest Memory	8 GB
VM Guest OS	Cent OS 6.5
SR-IOV support	Enabled with 32 VFs per node
System Location	Australian National University, AUSTRALIA

Table 3. Two Acer Desktop Computers with InfiniBand in Nanjing University

Item	Device
Processor	Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz Intel VT-D (not supported)
Memory	8 GB
Number of Servers	2
Network	Mellanox ConnectX-3 VPI (MCX353A-QCBT) Single QDR 40Gb/s Copper Direct Connection between two servers
Operating System	Cent OS 6.5
Kernel Linux Version	2.6.32-431.el6.x86_64
OFED	Mellanox OFED-3.12
Docker Version	1.3.2
Docker Image	Ubuntu 14.04
System Location	Nanjing University, NANJING

Table 4. IBM OpenLab OpenStack Beijing

Status	Applied account already but not used for any real testing yet.

Work Progress

1. **2014/12/01-2015/12/20 Australian National University**
   - Installed VirtualBox 4.3.20 on my MacBook and created two VMs with CentOS6.5 OS. Tried to install Open Fabrics Alliance OFED(3.12/3.5) but failed due to incompatibility to Linux version(2.6.32-431.el6.x86_64)
   - Finished and compared experiments of host-host single/four data transfer using RDMA with those using TCP on ANU's HPC system.
   - Finished and compared experiments of VM-VM single/four data transfer using RDMA with those using TCP on ANU's OpenStack cloud system.

2. **2015/01/06-Now Nanjing University**
   - Successfully installed InfiniBand adapters and IB cable directly connected two new servers
   - Set up OFA OFED and enabled RDMA to work successfully on two Linux hosts (CentOS 6.5)
   - Uninstalled OFA OFED and changed to MLNX_OFED - 3.12 to enable SR-IOV [6], installed KVMs (for VMs) and Dockers (for containers).
   - Tried to enable SR-IOV on InfiniBand adapter so that it can be used by multi-VMs but failed since CPU does not support VT-d[7][8][9].
   - Measured and compared the host-host bandwidth and latency of TCP and RDMA using qperf

- Tried to test TCP and RDMA host-host single/four data transfer using MPI programming but failed, and I think it is due to mis-configuration[10].
- Performed the test of RDMA host-host single file transfer using RDMA/CM server/client program [3].
- Created two containers (Ubuntu 14.04 images) and virtualized InfiniBand cards on containers [13][14].
- Measured and compared the container-container bandwidth and latency of RDMA using qperf.

Experimental Results

1. RDMA /TCP host-host single/four data transfer on ANU's HPC using MPI programming.

Table 5. MPI Testing Results on Host to Host (on ANU's HPC)

host-host										
Protocol	RDMA	**TCP**	**TCP**	TCP	TCP	TCP	RDMA	RDMA	RDMA	RDMA
Num of transfer	**1**	**1**	**4**				**4**			
			Process0	**Process2**	**Process4**	**Process6**	**Process0**	**Process2**	**Process4**	**Process6**
No.1	0.174	1.021	1.602	1.002	1.287	1.010	0.405	0.438	0.411	0.414
No.2	0.172	1.385	1.438	0.772	1.394	1.005	0.427	0.427	0.422	0.432
No.3	0.174	0.942	1.548	0.936	1.367	0.873	0.338	0.351	0.339	0.336
No.4	0.171	1.260	1.438	1.017	1.511	0.991	0.456	0.472	0.444	0.470
No.5	0.172	0.913	1.357	1.024	1.461	1.156	0.400	0.410	0.394	0.408
No.6	0.175	1.336	1.599	0.933	1.293	1.034	0.412	0.427	0.428	0.426
No.7	0.174	0.951	1.502	1.037	1.487	0.856	0.452	0.448	0.440	0.462
No.8	0.173	0.829	1.403	1.028	1.770	1.132	0.415	0.424	0.434	0.449
No.9	0.175	1.286	1.364	0.774	1.591	0.934	0.418	0.417	0.427	0.427
No.10	0.171	1.049	1.414	1.229	1.601	1.170	0.416	0.408	0.404	0.405
Average	0.173	1.097	1.467	0.975	1.476	1.016	0.414	0.422	0.414	0.423

2. RDMA /TCP VM-VM single/four data transfer on ANU’s HPC using MPI programming.

Table 6. MPI Testing Results on VM to VM(on ANU’s OpenStack Cloud)

VM-VM										
Protocol	RDMA	TCP	TCP	TCP	TCP	TCP	RDMA	RDMA	RDMA	RDMA
Num of transfer	**1**	**1**	**4**				**4**			
			process0	**process2**	**process4**	**process6**	**process0**	**process2**	**process4**	**process6**
No.1	0.230	1.009	1.056	1.402	1.426	1.067	0.599	0.600	0.583	0.593
No.2	0.230	1.022	1.257	1.074	1.224	1.085	0.592	0.594	0.601	0.615
No.3	0.230	0.957	1.181	1.304	1.277	1.016	0.741	0.737	0.763	0.702
No.4	0.230	0.966	1.131	1.248	1.323	1.080	0.568	0.569	0.579	0.595
No.5	0.230	1.030	1.415	1.140	0.994	1.393	0.580	0.584	0.593	0.583
No.6	0.230	0.992	1.103	1.291	1.295	1.117	0.583	0.552	0.565	0.560
No.7	0.230	1.026	1.325	0.980	1.307	1.069	0.571	0.570	0.570	0.576
No.8	0.230	0.899	1.131	1.356	1.054	1.363	0.552	0.555	0.563	0.558
No.9	0.230	1.036	1.001	1.337	1.257	1.049	0.634	0.634	0.634	0.632
No.10	0.230	0.880	0.983	1.274	1.004	1.388	0.615	0.620	0.618	0.657
Average	0.230	0.982	1.158	1.241	1.216	1.163	0.603	0.601	0.607	0.607

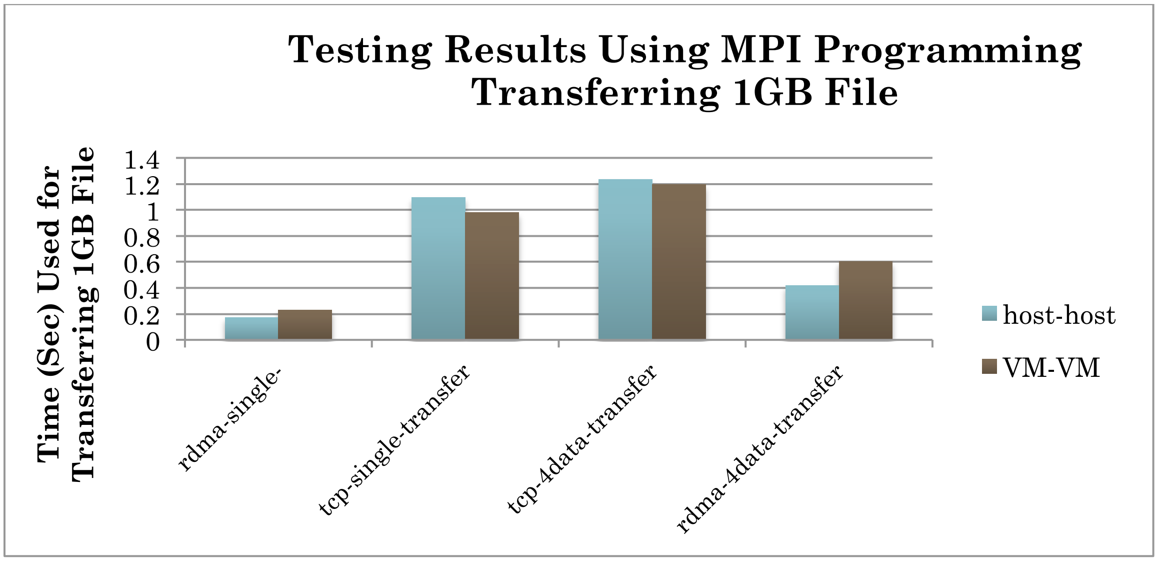

Figure 1. Measurement and comparison of host-host/VM-VM single/four data transfer

3. RDMA host-host single transfer on hosts in Nanjing University using rdma_cm server/client program.

Table 7. rdma_cm Server/Client Program Testing Results on Host to Host

Host-Host	No.1	No.2	No.3	No.4	No.5	No.6	No.7	No.8	No.9	No.10	Avg
Single-transfer	1.154	1.157	1.155	1.142	1.151	1.154	1.157	1.155	1.159	1.216	1.160

4. RDMA/TCP host-host qperf test on hosts in Nanjing University

Table 8. qperf Testing Results on Host to Host

Host - Host	rc_bw	rc_rdma_read_bw	rc_rdma_write_bw	rc_lat	rc_rdma_read_lat	rc_rdma_write_lat	rc_bi_bw
	GB/sec	GB/sec	GB/sec	us	us	us	us
1	3.71	3.78	3.72	12.30	4.24	12.00	6.30
2	3.71	3.75	3.72	12.30	4.11	12.00	6.28
3	3.71	3.74	3.72	12.20	4.11	12.10	6.32
4	3.71	3.74	3.72	12.30	4.13	12.00	6.35
5	3.71	3.75	3.71	12.30	4.15	12.00	6.35
6	3.69	3.74	3.71	12.20	4.12	12.10	6.30
7	3.71	3.74	3.71	12.30	4.11	12.10	6.33
8	3.72	3.75	3.74	12.20	4.13	12.00	6.32
9	3.71	3.75	3.73	12.30	4.10	11.80	6.34
10	3.71	3.75	3.74	12.30	4.10	12.00	6.36
average	**3.71**	**3.75**	**3.72**	**12.27**	**4.13**	**12.01**	**6.33**

5. RDMA container-container qperf test on hosts in Nanjing University

Table 9. qperf Testing Results on Container to Container

Container-Container	rc_bw	rc_rdma_read_bw	rc_rdma_write_bw	rc_lat	rc_rdma_read_lat	rc_rdma_write_lat	rc_bi_bw
	GB/sec	GB/sec	GB/sec	us	us	us	
1	3.73	3.67	3.73	11.30	4.10	11.20	6.39
2	3.72	3.68	3.74	11.30	4.10	11.20	6.41
3	3.73	3.67	3.74	11.30	4.20	11.20	6.42
4	3.73	3.68	3.74	11.20	4.14	11.20	6.41
5	3.72	3.69	3.74	11.30	4.10	11.20	6.41
6	3.73	3.70	3.74	11.20	4.12	11.20	6.37
7	3.72	3.68	3.74	11.30	4.10	11.20	6.42
8	3.73	3.67	3.74	11.50	4.11	11.20	6.42
9	3.72	3.72	3.74	11.30	4.11	11.20	6.42
10	3.73	3.70	3.74	11.30	4.11	11.20	6.43
average	**3.73**	**3.69**	**3.74**	**11.30**	**4.12**	**11.20**	**6.41**

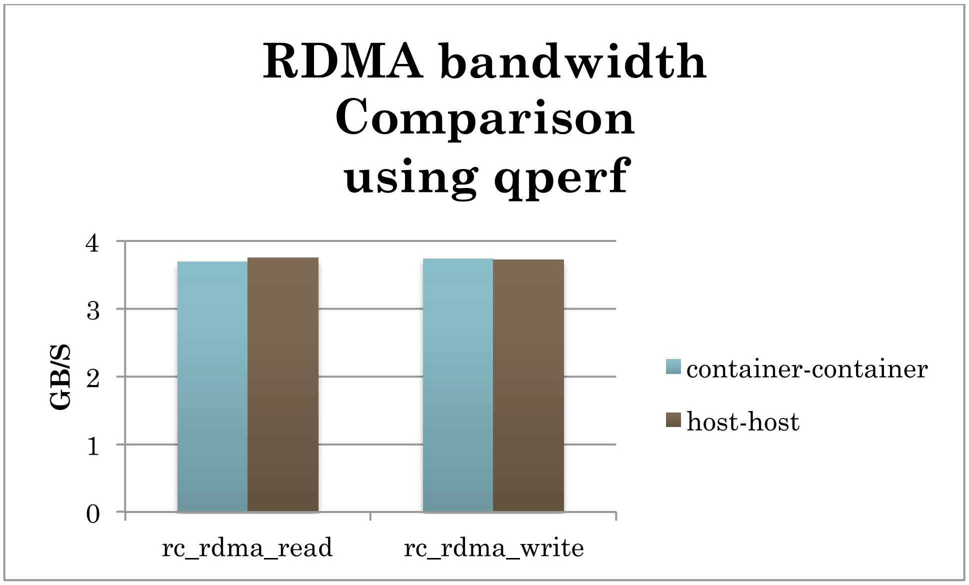

Figure 2. qperf bandwidth results

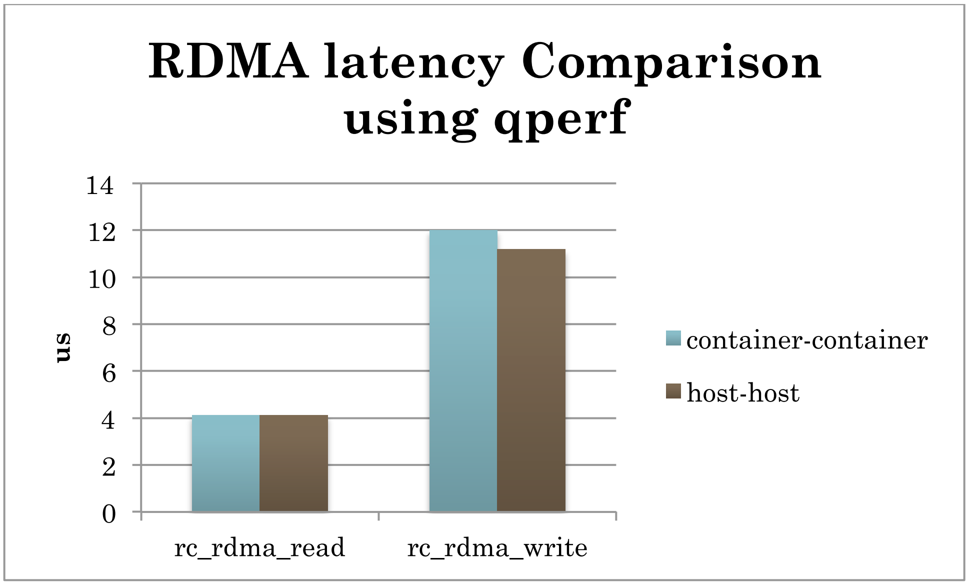

Figure 3. qperf latency results

Difficulties

1. Performance testing methods:
   qperf [1] utility is easy to use but I don't know whether it can work for evaluating the transferring performance on multiple hosts/VMs/containers. MPI program is convenient for operating many processes on multiple hosts/VMs/containers but needs to install and set up its environment on every host/VM/container. Another approach may be write my own program using TCP/IP socket programming for TCP testing and RDMA/CM API for RDMA testing.

2. SR-IOV and IOMMU Support
   Current CPU (Intel i7 3770K) on Acer Servers in Nanjing University does not provide support to Intel VT-d (Virtualization Technology on Directed I/O) that failed to support IOMMU [7]. Without these support, SR-IOV is not available. Hence, all experiments for single or multi-tenant VMs are not possible (even directly using passthrough to attach PCI devices to one VM is still impossible because it needs VT-d enabled CPU [12]).

3. Comparison of TCP and RDMA tests
   Currently the Infiniband Adapters (Mellanox ConnectX-3 VPI (MCX383A-FCxx) Single FDR IB (56Gb/s)) [11] on hosts are directly connected via a copper cable, but there is only one Ethernet Adapter on each host and they are not connected directly. The TCP performance over Ethernet is very slow. To fairly compare the difference of TCP and RDMA tests I will need two dedicated NICS (10Gb/s).

References

1. qperf manpage. URL: http://linux.die.net/man/1/qperf
2. T. Bedeir. Building an RDMA-Capable Application with IB Verbs. Technical re- port, HPC Advisory Council, 2010. URL: http://www.hpcadvisorycouncil.com/pdf/ building-an-rdma-capable-application-with-ib-verbs.pdf.
3. T. Bedeir. RDMA Read and Write with IB Verbs. Technical report, HPC Advisory Council, 2010. URL: http://www.hpcadvisorycouncil.com/pdf/rdma-read-and-write-with-ib-verbs.pdf.
4. RDMA Flow Control Basics Sample Code [online]. 2013. URL: https://sites.google.com/a/bedeir.com/home/rdma-file-transfer.tar.gz.
5. Mellanox OFED Driver Installation and Configuration for SR-IOV. URL: https://community.mellanox.com/docs/DOC-1317
6. MLNX_OFED User Manual. URL: http://www.mellanox.com/related-docs/prod_software/Mellanox_OFED_Linux_User_Manual_v2.4-1.0.0.pdf
7. SR-IOV features. URL: http://fedoraproject.org/wiki/Features/SR-IOV
8. CPU information. URL: http://ark.intel.com/products/65523/Intel-Core-i7-3770K-Processor-8M-Cache-up-to-3_90-GHz?q=%20i7-3770K
9. VT-d supported information. URL: http://ark.intel.com/search/advanced?VTD=true
10. OpenMPI Frequently Asked Questions [online]. URL: http://www.open-mpi.org/faq/?category=openfabrics#ib-locked-pages-user.
11. Infiniband Adapters information. URL: http://www.mellanox.com/related-docs/prod_adapter_cards/PB_ConnectX3_VPI_Card.pdf
12. PCI passthrough. URL: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Virtualization/chap-Virtualization-PCI_passthrough.html
13. Docker userguide. URL: https://docs.docker.com/userguide/
14. Introduction to Control Groups. URL: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch01.html
15. Emulex and Mellanox Drive Down 10 Gig Cost. URL: http://www.networkcomputing.com/networking/emulex-and-mellanox-drive-down-10-gig-cost/a/d-id/1233397
16. ANU's OpenStack Cloud System Configuration. URL: http://nci.org.au/nci-systems/national-facility/other-compute-resources/nectar-compute-cloud/detailed-system-configuration/