Mellanox OpenStack Solution Reference Architecture

Rev 1.3

January 2014

www.mellanox.com

NOTE:

THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT (“PRODUCT(S)”) AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES “AS-IS” WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCT(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Mellanox Technologies, Ltd.
Beit Mellanox
PO Box 586 Yokneam 20692
Israel
www.mellanox.com
Tel: +972 (0)74 723 7200
Fax: +972 (0)4 959 3245

Contents

List of Figures

Preface

About this Document

This reference design presents the value of using Mellanox interconnect products and describes how to integrate the OpenStack solution (Havana release or later) with the end-to-end Mellanox interconnect products.

Audience

This reference design is intended for server and network administrators.

The reader must have experience with the basic OpenStack framework and installation.

References

For additional information, see the following documents:

Table 1: Related Documentation

Reference	Location
Mellanox OFED User Manual	www.mellanox.com > Products > Adapter IB/VPI SW > Linux SW/Drivers http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=26&menu_section=34
Mellanox software source packages	https://github.com/mellanox-openstack
OpenStack Website	www.openstack.org
Mellanox OpenStack wiki page	https://wiki.openstack.org/wiki/Mellanox-OpenStack
Mellanox Ethernet Switch Systems User Manual	http://www.mellanox.com/related-docs/user_manuals/SX10XX_User_Manual.pdf
Mellanox Ethernet adapter cards	http://www.mellanox.com/page/ethernet_cards_overview
Solutions space on Mellanox community	http://community.mellanox.com/community/support/solutions
OpenStack RPM package	http://community.mellanox.com/docs/DOC-1187
Mellanox eSwitchd Installation for OpenFlow and OpenStack	http://community.mellanox.com/docs/DOC-1126

Reference	Location
Troubleshooting	http://community.mellanox.com/docs/DOC-1127
Mellanox OFED Driver Installation and Configuration for SR-IOV	http://community.mellanox.com/docs/DOC-1317

Revision History

Table 2: Document Revision History

Revision	Date	Changes
1.3	Jan. 2014	Removed OpenFlow sections. Added related topics for OpenStack Havana release.
1.2	Sep. 2013	Minor editing
1.1	June 2013	Added OpenFlow feature
1.0	May 2013	Initial revision

1 Solution Overview

Deploying and maintaining a private or public cloud is a complex task, with various vendors developing tools to address the different aspects of the cloud infrastructure, management, automation, and security. These tools tend to be expensive and create integration challenges for customers when they combine parts from different vendors. Traditional offerings suggest deploying multiple network and storage adapters to run management, storage, services, and tenant networks. These also require multiple switches, cabling, and management infrastructure, which increases both up front and maintenance costs.

Other, more advanced offerings provide a unified adapter and first level ToR switch, but still run multiple and independent core fabrics. Such offerings tend to suffer from low throughput because they do not provide the aggregate capacity required at the edge or in the core; and because they deliver poor application performance due to network congestion and lack of proper traffic isolation.

Several open source "cloud operating system" initiatives have been introduced to the market, but none has gained sufficient momentum to succeed. Recently OpenStack has managed to establish itself as the leading open source cloud operating system, with wide support from major system vendors, OS vendors, and service providers. OpenStack allows central management and provisioning of compute, networking, and storage resources, with integration and adaptation layers allowing vendors and/or users to provide their own plug-ins and enhancements.

Mellanox Technologies offers seamless integration between its products and OpenStack layers and provides unique functionality that includes application and storage acceleration, network provisioning, automation, hardware-based security, and isolation. Furthermore, using Mellanox interconnect products allows cloud providers to save significant capital and operational expenses through network and I/O consolidation and by increasing the number of virtual machines (VMs) per server.

Mellanox provides a variety of network interface cards (NICs) supporting one or two ports of 10GbE, 40GbE, or 56Gb/s InfiniBand. These adapters simultaneously run management, network, storage, messaging, and clustering traffic. Furthermore, these adapters create virtual domains within the network that deliver hardware-based isolation and prevent cross-domain traffic interference.

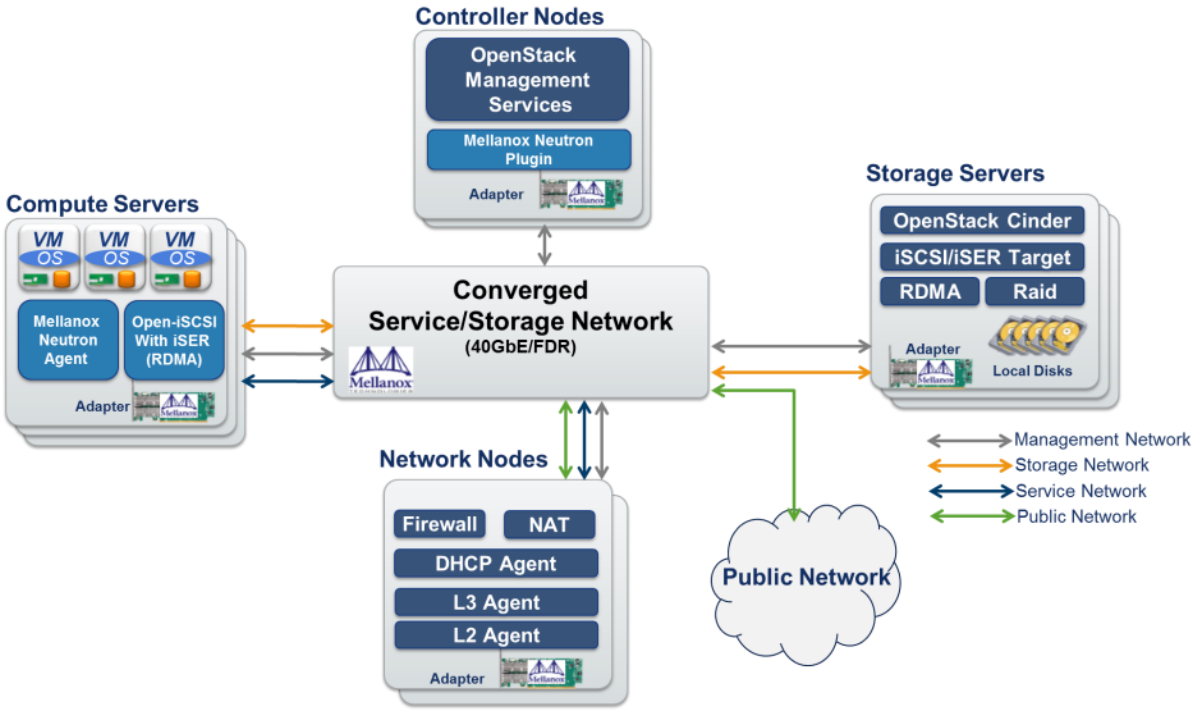
In addition, Mellanox Virtual Protocol Interconnect (VPI) switches deliver the industry's most cost-effective and highest capacity switches (supporting up to 36 ports of 56Gb/s). When deploying large-scale, high-density infrastructures, leveraging Mellanox converged network VPI solutions translates into fewer switching elements, far fewer optical cables, and simpler network design.

Mellanox plugins are inbox for Havana release. The Havana release includes out of the box support for InfiniBand and Ethernet Mellanox components for Nova, Cinder and Neutron.

Mellanox integration with OpenStack provides the following benefits:

- Cost-effective and scalable infrastructure that consolidates the network and storage to a highly efficient flat fabric, increases the VM density, commoditizes the storage infrastructure, and linearly scales to thousands of nodes
- Delivers the best application performance with hardware-based acceleration for messaging, network traffic, and storage
- Easy to manage via standard APIs. Native integration with OpenStack Neutron (network) and Cinder (storage) provisioning APIs
- Provides tenant and application security/isolation, end-to-end hardware-based traffic isolation, and security filtering
- Mellanox designed its end-to end OpenStack cloud solution to offer seamless integration between its products and OpenStack services.
- By using Mellanox 10/40GbE and FDR 56Gb/s adapters and switches with OpenStack Havana release, customers can gain significant improvement in block storage access performance with Cinder. In addition, customers can deploy an embedded virtual switch to run virtual machine traffic with bare-metal performance, provide hardened security and QoS, all with simple integration.
- Mellanox has partnered with the leading OpenStack distributions to allow customers to confidently deploy an OpenStack cloud, with proven interoperability and integrated support. For more information click here.

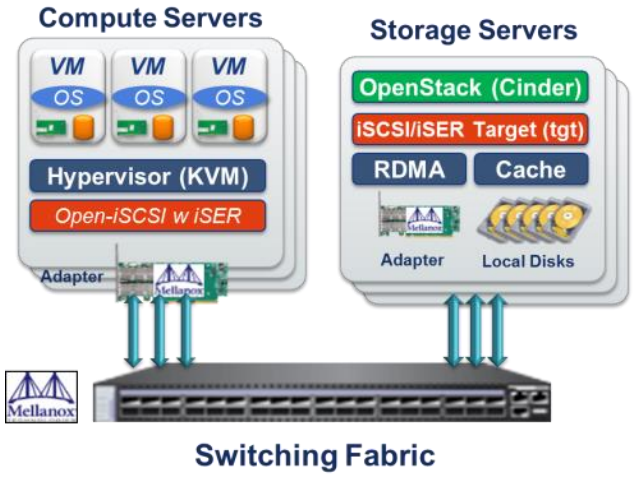
Figure 1: Mellanox OpenStack Architecture



2 Accelerating Storage

Data centers rely on communication between compute and storage nodes, as compute servers read and write data from the storage servers constantly. In order to maximize the server's application performance, communication between the compute and storage nodes must have the lowest possible latency, highest possible bandwidth, and lowest CPU utilization.

Figure 2: OpenStack Based IaaS Cloud POD Deployment Example



Storage applications rely on iSCSI over TCP communications protocol stack continuously interrupt the processor in order to perform basic data movement tasks (packet sequence and reliability tests, re-ordering, acknowledgements, block level translations, memory buffer copying, etc). This causes data center applications that rely heavily on storage communication to suffer from reduced CPU efficiency, as the processor is busy sending data to and from the storage servers rather than performing application processing. The data path for applications and system processes must wait in line with protocols such as TCP, UDP, NFS, and iSCSI for their turn using the CPU. This not only slows down the network, but also uses system resources that could otherwise have been used for executing applications faster.

Mellanox OpenStack solution extends the Cinder project by adding iSCSI running over RDMA (iSER). Leveraging RDMA Mellanox OpenStack delivers 6X better data throughput (for example, increasing from 1GB/s to 5GB/s) and while simultaneously reducing CPU utilization by up to 80% (see Figure 3).

Mellanox ConnectX®-3 adapters bypass the operating system and CPU by using RDMA, allowing much more efficient data movement. iSER capabilities are used to accelerate hypervisor traffic, including storage access, VM migration, and data and VM replication. The use of RDMA shifts data movement processing to the Mellanox ConnectX-3 hardware, which provides zero-copy message transfers for SCSI packets to the application, producing significantly faster performance, lower network latency, lower access time, and lower CPU overhead. iSER can provide 6X faster performance than traditional TCP/IP based iSCSI. The

iSER protocol unifies the software development efforts of both Ethernet and InfiniBand communities, and reduces the number of storage protocols a user must learn and maintain.

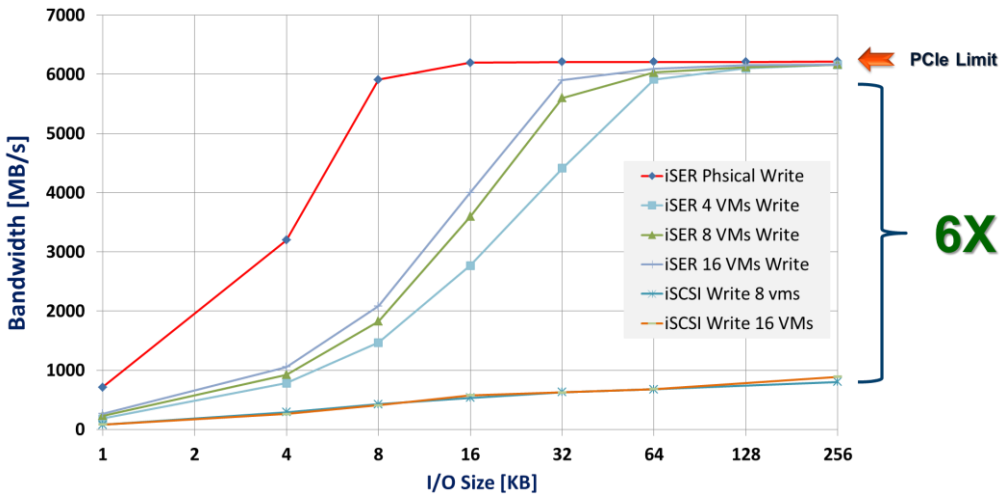
RDMA bypass allows the application data path to effectively skip to the front of the line. Data is provided directly to the application immediately upon receipt without being subject to various delays due to CPU load-dependent software queues. This has three effects:

- There is no waiting, which means that the latency of transactions is incredibly low.
- Because there is no contention for resources, the latency is deterministic, which is essential for offering end users a guaranteed SLA.
- Bypassing the OS, using RDMA results in significant savings in CPU cycles. With a more efficient system in place, those saved CPU cycles can be used to accelerate application performance.

In the following diagram, it is clear that by performing hardware offload of the data transfers using the iSER protocol, the full capacity of the link is utilized to the maximum of the PCIe limit.

To summarize, network performance is a significant element in the overall delivery of data center services and benefits from high speed interconnects.Unfortunately the high CPU overhead associated with traditional storage adapters prevents systems from taking full advantage of these high speed interconnects. The iSER protocol uses RDMA to shift data movement tasks to the network adapter and thus frees up CPU cycles that would otherwise be consumed executing traditional TCP and iSCSI protocols. Hence, using RDMA-based fast interconnects significantly increases data center application performance levels.

Figure 3: RDMA Acceleration



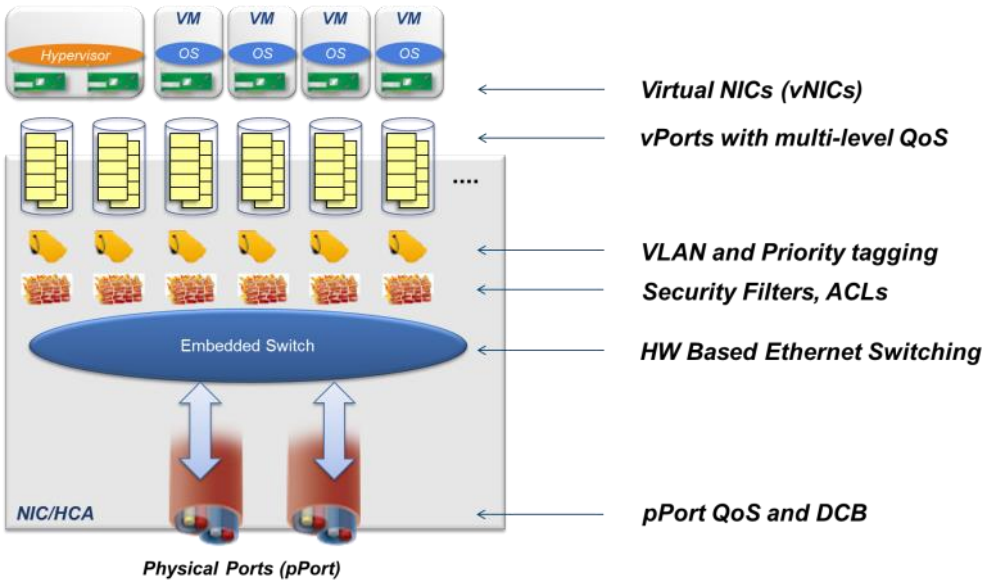
3 Network Virtualization on Mellanox Adapters

Single Root IO Virtualization (SR-IOV) allows a single physical PCIe device to present itself as multiple devices on the PCIe bus. Mellanox ConnectX®-3 adapters are capable of exposing up to 127 virtual instances called Virtual Functions (VFs). These virtual functions can then be provisioned separately. Each VF can be viewed as an additional device associated with the Physical Function. It shares the same resources with the Physical Function, and its number of ports equals those of the Physical Function.

SR-IOV is commonly used in conjunction with an SR-IOV enabled hypervisor to provide virtual machines with direct hardware access to network resources, thereby improving performance.

Mellanox ConnectX-3 adapters equipped with onboard embedded switch (eSwitch) are capable of performing layer-2 switching for the different VMs running on the server. Using the eSwitch will gain even higher performance levels and in addition improve security, and isolation.

Figure 4: eSwitch Architecture



eSwitch main capabilities and characteristics:

- Virtual switching: creating multiple logical virtualized networks. The eSwitch offload engines handle all networking operations up to the VM, thereby dramatically reducing software overheads and costs.
- Performance: The switching is handled in hardware, as opposed to other applications that use a software-based switch. This enhances performance by reducing CPU overhead.

- Security: The eSwitch enables network isolation (using VLANs) and anti-MAC spoofing.
- Monitoring: Port counters are supported.

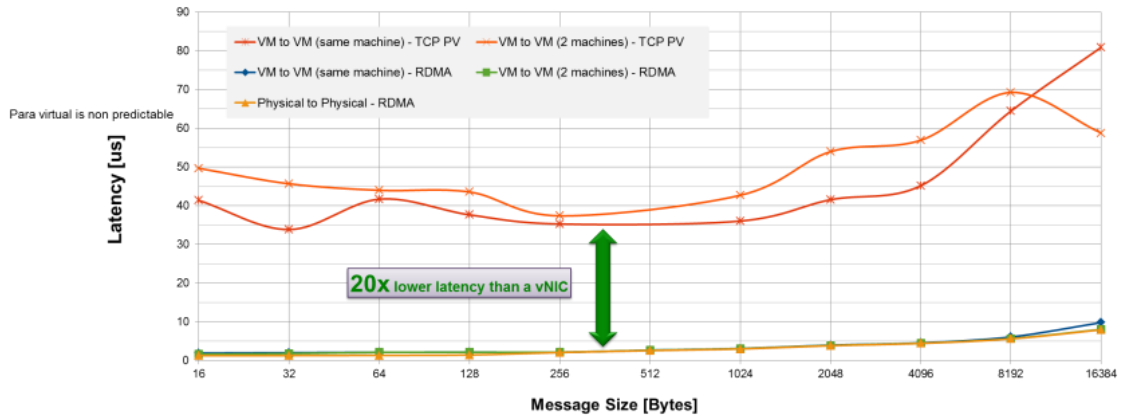
3.1 Performance Measurements

Many data center applications benefits from low latency network communication while others require deterministic latency. Using regular TCP connectivity between VMs can create high latency and unpredictable delay behavior.

Figure 5 shows the dramatic difference (20X improvement) delivered by SR-IOV connectivity running RDMA compared to para-virtualized vNIC running a TCP stream.

Using the direct connection of the SR-IOV and the ConnectX-3 hardware eliminates the software processing that adds an unpredictable delay to packet data movement. The result is a consistently low latency that allows application software to rely on deterministic packet transfer times.

Figure 5: Latency Comparison

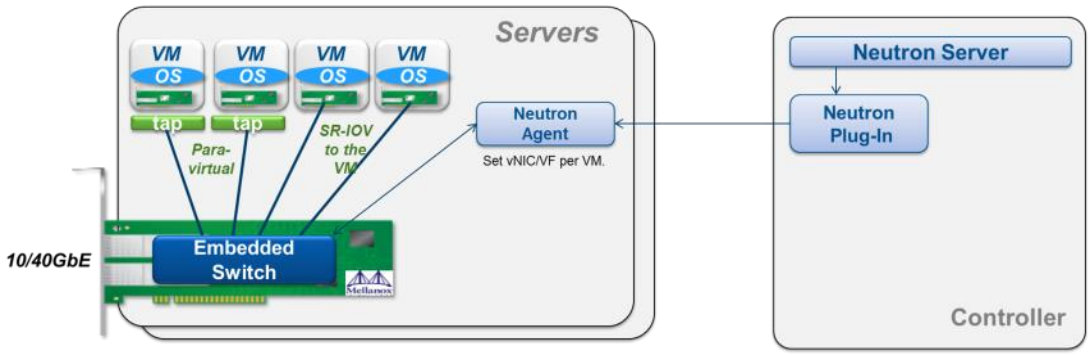


3.2 Seamless Integration

The eSwitch configuration is transparent to the OpenStack controller administrator. The installed eSwitch daemon on the server is responsible for hiding the low-level configuration. The administrator will use the standard OpenStack dashboard APIs REST interface for the fabric management.

The Neutron agent configures the eSwitch in the adapter card.

Figure 6: Network Virtualization



4 Setup and Installation

The OpenStack environment should be installed according to the OpenStack documentation package.

For OpenStack Havana release the following installation changes should be applied:

- A Neutron server should be installed with the Mellanox Neutron plugin.
- Mellanox Neutron agent, eSwitch daemon, and Nova VIF driver should be installed on the compute notes.

For OpenStack Grizzly release the following installation changes should be applied:

- A Neutron server should be installed with the Mellanox Neutron plugin.
- A Cinder patch should be applied to the storage servers (for iSER support).
- Mellanox Neutron agent, eSwitch daemon, and Nova VIF driver should be installed on the compute notes.

4.1 Hardware Requirements

- Mellanox ConnectX®-3 adapter cards
- 10GbE or 40GbE Ethernet switches
- Cables required for the ConnectX-3 card (typically using SFP+ connectors for 10GbE or QSFP connectors for 40GbE)
- Server nodes complying with OpenStack requirements
- Compute nodes with SR-IOV capability (BIOS and OS support)

In terms of adapters, cables, and switches, many variations are possible. Visit www.mellanox.com for more information.

Figure 7: Mellanox MCX314A-BCBT, ConnectX-3 40GbE Adapter

Figure 8: Mellanox SX1036, 36x40GbE

Figure 9: Mellanox 40GbE, QSFP Copper Cable

4.2 Software Requirements

- Supported OS
- RHEL 6.4 or higher
- Mellanox OFED 2.0.3 (SR-IOV support) or higher.
- KVM hypervisor – complying with OpenStack requirements

4.3 Prerequisites

1. Hardware is set up.
   - To reduce the number of ports in the network, two different subnets can be mapped to the same physical interface on two different VLANs.
2. Mellanox OFED 2.0.3 (SR-IOV enabled) is installed on each of the network adapters.
   - For Mellanox OFED installation refer to Mellanox OFED User Manual (Installation chapter). http://www.mellanox.com/page/products_dyn?product_family=26
   - Visit Mellanox Community – for verification options and adaptation. http://community.mellanox.com/docs/DOC-1317
3. The OpenStack packages are installed on all network elements.
4. EPEL repository is enabled. (http://fedoraproject.org/wiki/EPEL).

4.4 OpenStack Software Installation

For Mellanox OpenStack installation visit the Mellanox OpenStack wiki pages

at https://wiki.openstack.org/wiki/Mellanox-OpenStack.

4.5 Troubleshooting

Troubleshooting actions for OpenStack installation with Mellanox plugins can be found at http://community.mellanox.com/docs/DOC-1127.

5 Setting Up the Network

5.1 Configuration Examples

Once installation is completed, the network must be set up.

Setting up a network consists of the following steps:

1. Creating a network.
2. Creating a VM instance.

   Two types of instances can be created:

   i. Para-virtualized vNIC.

   ii. SR-IOV direct path connection.
3. Creating disk volume.
4. Binding the disk volume to the instance created.

5.1.1 Creating a Network

Use the commands `neutron net-create` and `neutron subnet-create` to create a new network and a subnet (“net-example” in the example).

```
$neutron net-create net-example
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 16b790d6-4f5a-4739-a190-7598f331b696 |
| name                      | net-example                          |
| provider:network_type     | vlan                                 |
| provider:physical_network | default                              |
| provider:segmentation_id  | 4                                    |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 679545ff6c1e4401adcafa0857aefe2e     |
+---------------------------+--------------------------------------+
```

```
$neutron subnet-create net-example 192.168.199.0/24
Created a new subnet:
+-----------------+------------------------------------------------------+
| Field           | Value                                                |
+-----------------+------------------------------------------------------+
| allocation_pools | {"start": "192.168.199.2", "end": "192.168.199.254"} |
| cidr            | 192.168.199.0/24                                     |
| dns_nameservers |                                                      |
| enable_dhcp     | True                                                 |
| gateway_ip      | 192.168.199.1                                        |
| host_routes     |                                                      |
| id              | 3c9ff1ae-218d-4020-b065-a2991d23bb72                 |
| ip_version      | 4                                                    |
| name            |                                                      |
| network_id      | 16b790d6-4f5a-4739-a190-7598f331b696                 |
| tenant_id       | 679545ff6c1e4401adcafa0857aefe2e                     |
+-----------------+------------------------------------------------------+
```

5.1.2 Creating a Para-Virtualized vNIC Instance

1. Using the OpenStack Dashboard, launch an VM instance using the Launch Instance button.
2. Insert all the required parameters and click Launch.

   This operation creates a macvtap interface on top of a Virtual Function (VF).

Figure 10: OpenStack Dashboard Instances

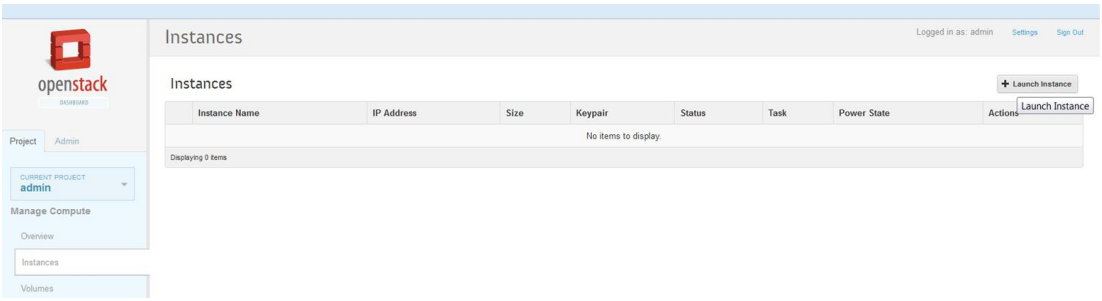
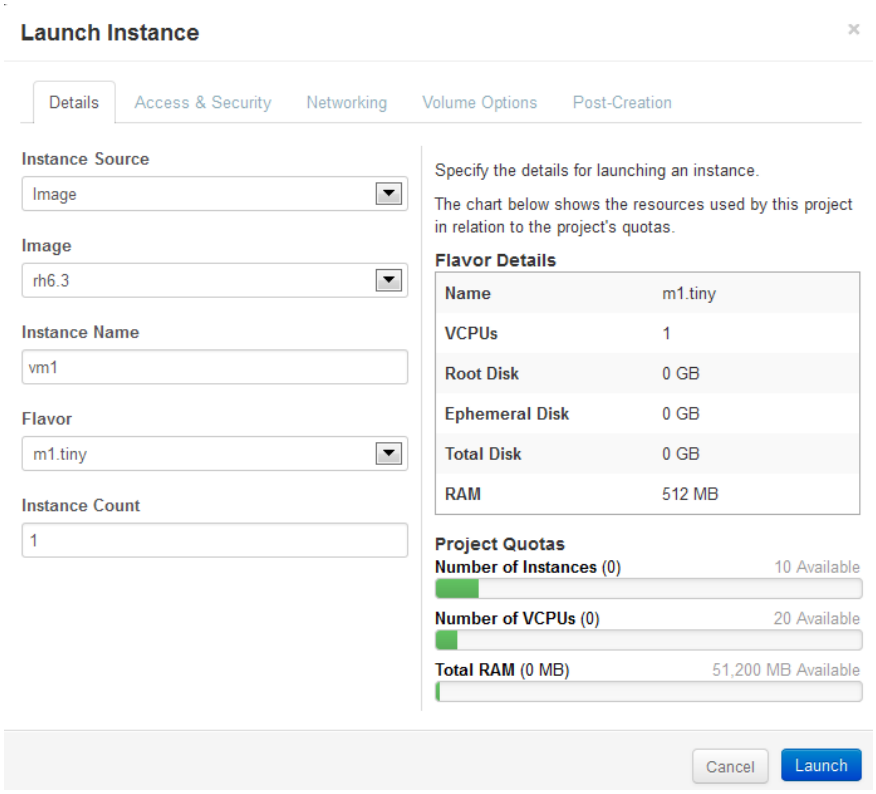
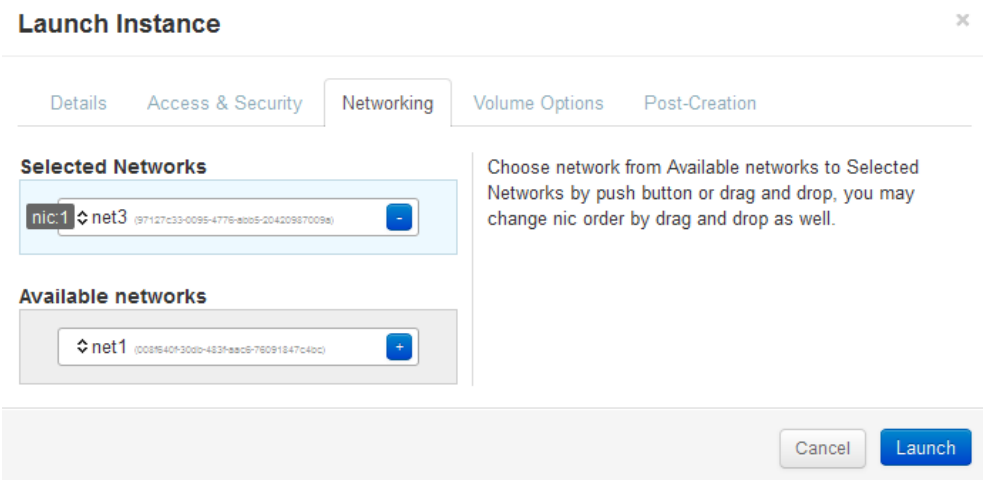


Figure 11: OpenStack Dashboard, Launch Instance



3. Select the desired network for the vNIC ("net3" in the example).

Figure 12: OpenStack Dashboard, Launch Interface – Select Network



5.1.3 Creating an SR-IOV Instance

1. Use the command `neutron port-create` for the selected network (‘net3’ in the example) to create a port with ‘vnic_type=hostdev’.

```
$neutron port-create net-example --binding:profile type=dict vnic_type=hostdev
Created a new port:
+-----------------------+-----------------------------------------------------------+
| Field                 | Value                                                     |
+-----------------------+-----------------------------------------------------------+
| admin_state_up        | True                                                      |
| binding:capabilities  | {"port_filter": false}                                    |
| binding:host_id       |                                                           |
| binding:profile       | {"physical_network": "default"}                           |
| binding:vif_type      | hostdev                                                   |
| device_id             |                                                           |
| device_owner          |                                                           |
| fixed_ips             | {"subnet_id": "3c9ff1ae-218d-4020-b065-a2991d23bb72",     |
                          "ip_address": "192.168.199.2"}                            |
| id                    | a43d35f3-3870-4ae1-9a9d-d2d341b693d6                      |
| mac_address           | fa:16:3e:67:ad:ef                                         |
| name                  |                                                           |
| network_id            | 16b790d6-4f5a-4739-a190-7598f331b696                      |
| status                | DOWN                                                      |
| tenant_id             | 679545ff6c1e4401adcafa0857aefe2e                          |
+-----------------------+-----------------------------------------------------------+
```

2. Use the command `nova boot` to launch an instance with the created port attached.

```
$nova boot --flavor m1.small --image rh6.4p --nic
port-id=a43d35f3-3870-4ae1-9a9d-d2d341b693d6 vm3
+--------------------------------------+--------------------------------------+
| Property                             | Value                                |
+--------------------------------------+--------------------------------------+
| OS-EXT-STS:task_state                | scheduling                           |
| image                                | rh6.4p                               |
| OS-EXT-STS:vm_state                  | building                             |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000042                    |
| OS-SRV-USG:launched_at               | None                                 |
| flavor                               | m1.small                             |
| id                                   | 161da6a9-6508-4e23-9f6f-881383461ab4 |
| security_groups                      | [{u'name': u'default'}]              |
| user_id                              | b94edf2504c84223b58e254314528902     |
| OS-DCF:diskConfig                    | MANUAL                               |
| accessIPv4                           |                                      |
| accessIPv6                           |                                      |
| progress                             | 0                                    |
| OS-EXT-STS:power_state               | 0                                    |
| OS-EXT-AZ:availability_zone          | nova                                 |
| config_drive                         |                                      |
| status                               | BUILD                                |
| updated                              | 2013-12-19T07:32:42Z                 |
| hostId                               |                                      |
| OS-EXT-SRV-ATTR:host                 | None                                 |
| OS-SRV-USG:terminated_at             | None                                 |
| key_name                             | None                                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                 |
| name                                 | vm3                                  |
| adminPass                            | tiTE37tQrNBn                         |
| tenant_id                            | 679545ff6c1e4401adcafa0857aefe2e     |
| created                              | 2013-12-19T07:32:41Z                 |
| os-extended-volumes:volumes_attached | []                                   |
| metadata                             | {}                                   |
+--------------------------------------+--------------------------------------+
```

5.1.4 Creating a Volume

Create a volume using the Volumes tab on the OpenStack dashboard. Click the Create Volume button.

Figure 13: OpenStack Dashboard, Volumes

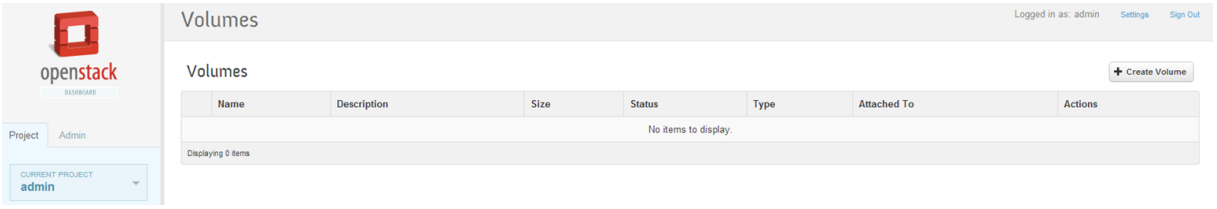


Figure 14: OpenStack Dashboard, Create Volumes

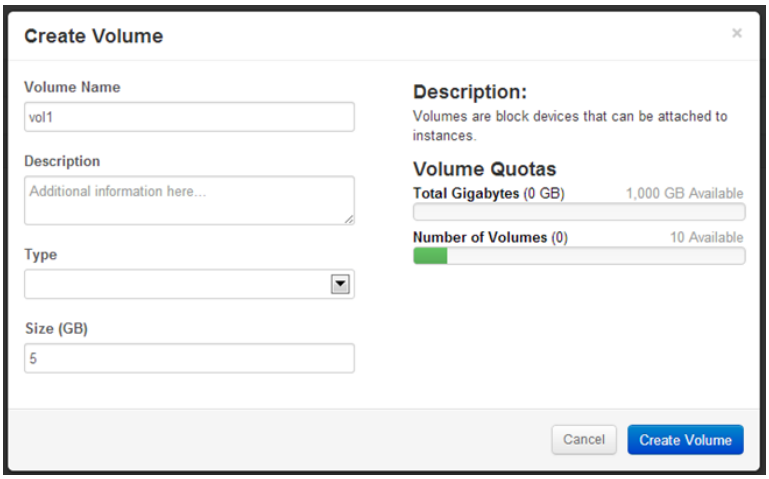
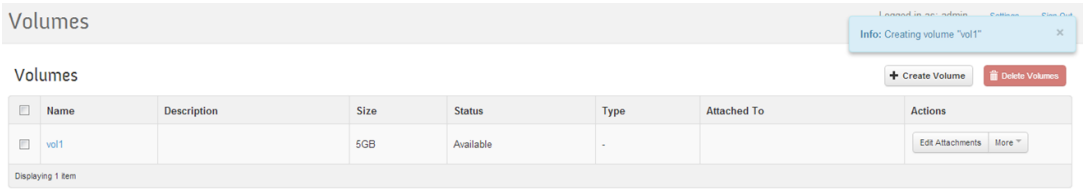


Figure 15: OpenStack Dashboard, Volumes

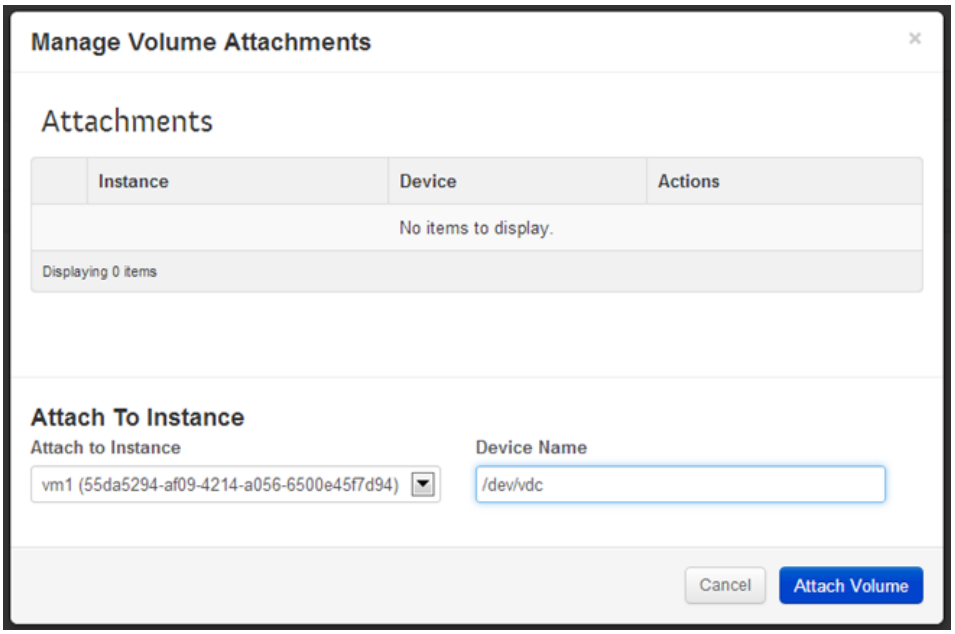


5.1.5 Attaching a Volume

Attach a volume to the desired instance.

The device name should be /dev/dv<letter>. E.g. "/dev/vdc"

Figure 16: OpenStack Dashboard, Manage Volume Attachments

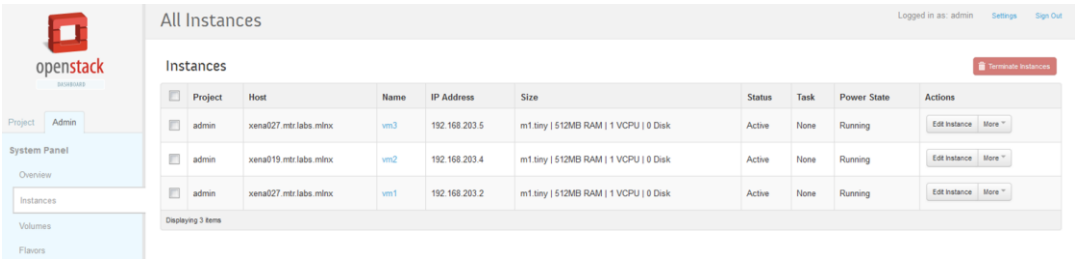


5.2 Verification Examples

5.2.1 Instances Overview

Use the OpenStack Dashboard to view all configured instances.

Figure 17: VM Overview

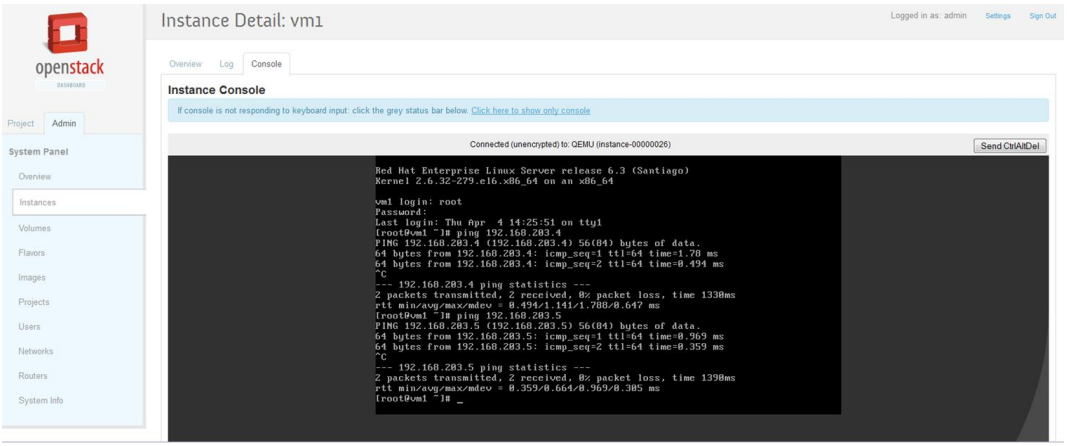


5.2.2 Connectivity Check

There are many options for checking connectivity between instances, one of which is to simply open a remote console and ping the required host.

To launch a remote console for a specific instance, select the Console tab and launch the console.

Figure 18: Remote Console Connectivity



5.2.3 Volume Check

To verify that the created volume is attached to a specific instance, click the Volumes tab.

Figure 19: OpenStack Dashboard, Volumes

In addition, run the `fdisk` command from the instance console to see the volume details.

Figure 20: OpenStack Dashboard, Console