UNIVERSITÉ DE NANTES
DIRECTION DES SYSTÈMES
D'INFORMATION ET DU NUMÉRIQUE

SR-IOV and KVM virtual machines under GNU/Linux Debian Operating System

Yoann Juet @ University of Nantes, France
Information Technology Services

Version 1.0 (28 Mar 2014)

Our goal

- Virtualize high-performance servers, firewalls requiring:
  - Low network latency and jitter
  - Low processor impact (I/O)
  - High throughput (10Gbps)
- Solution: Single Root - IO Virtualization (SR-IOV)
  - A single PCI card is showed up as multiple virtual PCI cards
  - Exposes *n* virtual interfaces from a single physical interface
    - > No miracle, shared bandwidth

Prerequisites

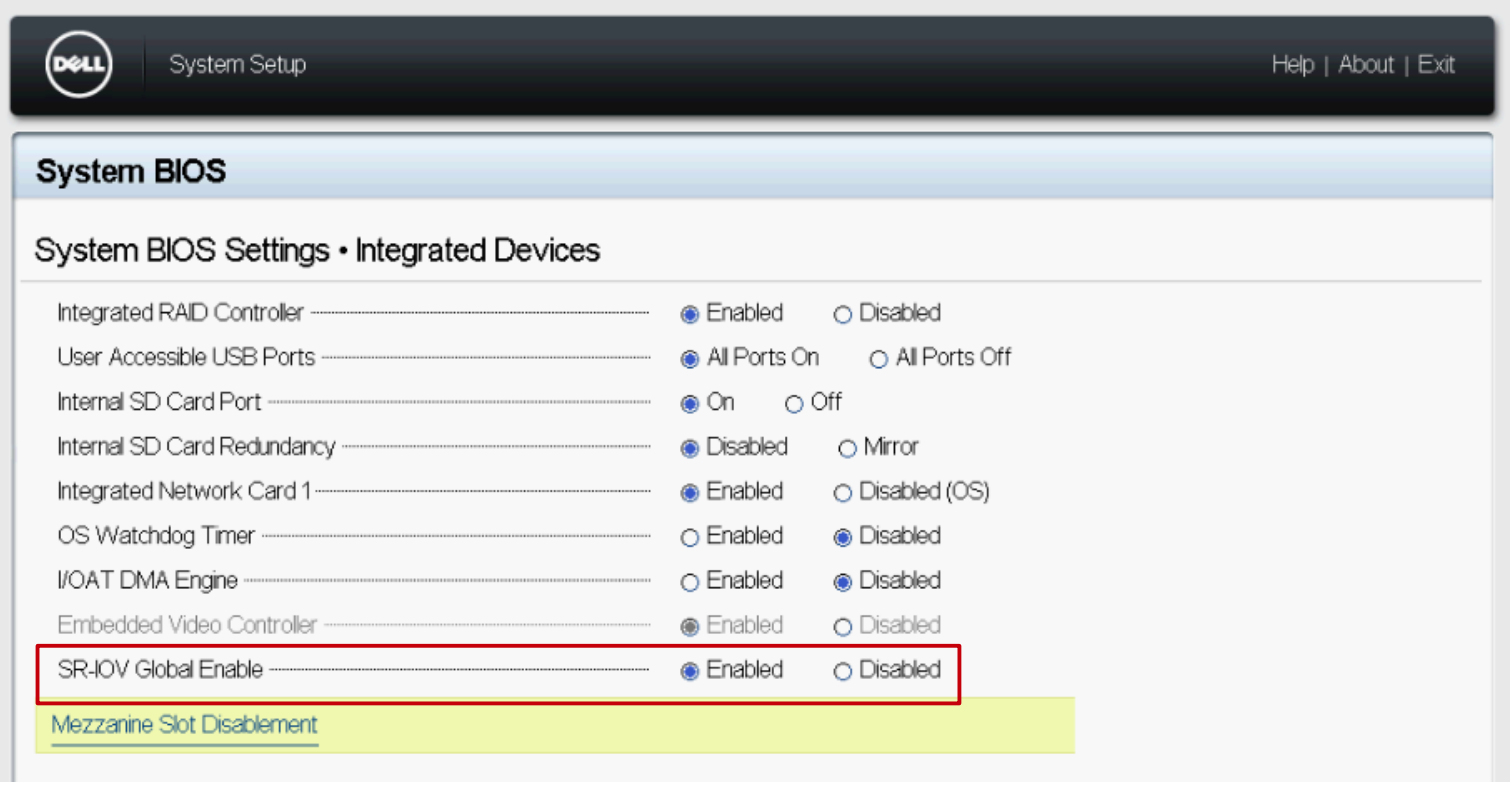
- Virtualization Technology for Directed I/O: Intel VT-d or AMD-Vi
  - Must be supported by both the CPU and the chipset
  - Guest machines gain direct memory access (DMA) to PCI(e) devices, such as Ethernet cards
- PCI-SIG Single Root I/O Virtualization: SR-IOV
  - Must be supported by both the Ethernet cards and the BIOS
  - Guest machines are able to achieve ~ bare metal performance

Technical environment

- Dell Blade Servers M420
  - Intel Xeon CPU E5-2407
  - Dual Broadcom NetXtreme II BCM57810 10Gbps cards
  - Operating Systems Debian 7 (code name "Wheezy")
    - On hosts as well as guests machines

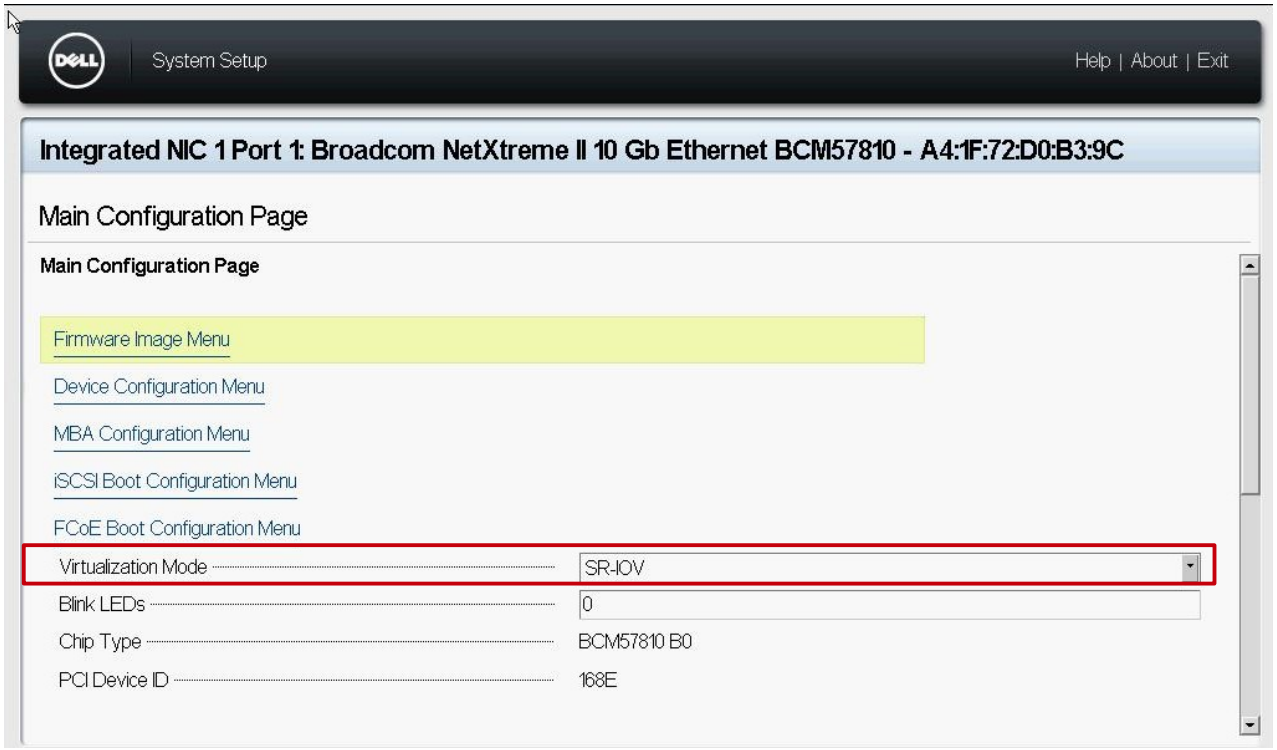
BIOS

- Ensure SR-IOV BIOS option is enabled
  - System BIOS > Integrated Devices > SR-IOV Global Enable



BIOS

- Ensure SR-IOV mode is set on both Ethernet cards
  - Device Settings > Integrated NIC 1 Port {1|2} > Virtualization Mode > SR-IOV



Debian: Starting with SR-IOV

- Some Kernel Requirements:

CONFIG_PCI_IOV=y

CONFIG_BNX2X_SRIOV=y

CONFIG_PCI_STUB=y

CONFIG_VFIO_IOMMU_TYPE1=y

CONFIG_VFIO=y

CONFIG_VFIO_PCI=y

→ Default Debian 7 kernel is not recommended for use with SR-IOV feature. Rather, prefer a recent kernel (at this time 3.13.6) that fixes important bugs related to SR-IOV such as “VLAN configuration for VFs”.

Debian: Starting with SR-IOV

- At this step, SR-IOV is not yet configured. Two PCIe network adapter cards are visible
- Check for SR-IOV hardware support:

```
# lspci -v
...
00:05.0 System peripheral: Intel Corporation Xeon E5/Core i7 Address Map, VTd_Misc, System Management (rev 07)
...
01:00.0 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet (rev 10)
[...]
Capabilities: [1c0] Single Root I/O Virtualization (SR-IOV)
Kernel driver in use: bnx2x

01:00.1 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet (rev 10)
[...]
Capabilities: [1c0] Single Root I/O Virtualization (SR-IOV)
Kernel driver in use: bnx2x
...
```

Debian: Starting with SR-IOV

- Kernel 3.8+ brings sysfs interface support for getting the maximal number of VF for a given PF, as well as for getting and setting the current number of VF:

```
# echo 8 > /sys/bus/pci/devices/0000\:01\:00.1/sriov_numvfs
# lspci
...
01:00.0 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet (rev 10)
01:00.1 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet (rev 10)
01:09.0 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
01:09.1 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
01:09.2 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
01:09.3 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
01:09.4 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
01:09.5 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
01:09.6 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
01:09.7 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
...
```

SR-IOV feature is now activated on the second 10Gbps card, eth1 (**here 8 VFs per PF → 64 max**)

Debian: Starting with SR-IOV

- Each VF appears as a traditional network interface (eth2 to eth9)

 # ip link show | grep mtu

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT

2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master bond0 state UP mode DEFAULT qlen 1000

3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000

23: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000

24: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000

25: eth4: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000

26: eth5: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000

27: eth6: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000

28: eth7: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000

29: eth8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000

30: eth9: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000

Debian: PCI passthrough with libvirt

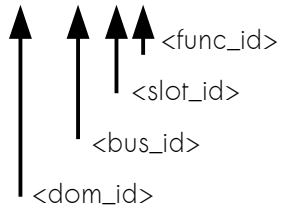
- First method: Assignment with <hostdev> block

```
<hostdev mode='subsystem' type='pci' managed='yes'>
   <source>
      <address domain='<dom_id>' bus='<bus_id>' slot='<slot_id>' function='<func_id>'/>
   </source>
</hostdev>
```

Excerpt from guest XML file

Where <dom_id>, <bus_id>, <slot_id> and <func_id> are given by:

```
# lspci -D
0000:01:09.0 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
...
0000:01:09.7 Ethernet controller: Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet Virtual Function
```



- First virtual PCIe device (VF0): <address domain='0x0000' bus='0x01' slot='0x09' function='0x0'/>
- Last virtual PCIe device (VF7): <address domain='0x0000' bus='0x01' slot='0x09' function='0x7'/>

Debian: PCI passthrough with libvirt

→ Second method: Assignment with <interface type='hostdev'> block

```
<interface type='hostdev' managed='yes'>
  <mac address='<virtual_mac_address>'/>
  <source>
    <address domain='<dom_id>' bus='<bus_id>' slot='<slot_id>' function='<func_id>'/>
  </source>
</interface>
```

Excerpt from guest XML file

Where <virtual_mac_address>' is the guest interface virtual mac address. <dom_id>, <bus_id>, <slot_id>, <func_id> are defined in the previous slide.

Unfortunately, such an assignment method doesn't work on a standard Debian 7 distro (qemu-kvm 1.1.2, libvirt 0.9.12) → need to upgrade qemu-kvm to version 1.3 or later

```
# virsh define 01-test.xml
Domain 01-test defined from 01-test.xml

# virsh start 01-test
error: Failed to start domain 01-test
error: An error occurred, but the cause is unknown
```

Debian: PCI passthrough with libvirt

→ Third method: Assignment from a pool of VFs

```
<network>
<name>sriov</name>
<forward mode='hostdev' managed='yes'>
  <driver name='vfio'/>
  <pf dev='<iface>'/>
</forward>
</network>
```

Network XML file
Directory /etc/libvirt/qemu/networks/

```
<interface type='network'>
  <source network='sriov'/>
  <vlan>
    <tag id='<vlan_id>'/>
  </vlan>
</interface>
```

Excerpt from guest XML file

Again, such an assignment method is currently unsupported on Debian 7 → need to upgrade libvirt to version 0.10.0 or later

Debian: Vlan isolation

- Assumption: use case based on a standard Debian 7
  - No choice, first assignment method for libvirt is mandatory
  - No vlan declaration within the guest XML file
- Use 'ip link' to configure vlan on VF interfaces
  - Should be done on the host before the guest is up

ip link set vf <vf_id> vlan <vlan_id> dev <iface>

Example: ip link set vf 0 vlan 403 dev eth1

Where:

- **<vf_id>** is the Virtual Function Identifier, starting from 0 to 7 (or more),
- **<vlan_id>** is the vlan identifier to be allowed,
- **<iface>** is the physical interface associated to the VF

Debian: MAC address

- Other consequence of the first assignment method:
  - No provision of VF MAC address within the guest XML file
  - Should be done with 'ip link' before the guest is up

ip link set <iface> vf <vf_id> mac <vf_mac>

Example: ip link set eth1 vf 0 mac de:ad:fe:ed:ff:01

Where **<vf_mac>** is the virtual mac address associated to the VF

Then deactivate/reactivate SR-IOV for effective use (to be scripted once the host is running):

echo 0 > /sys/bus/pci/devices/0000\:01\:00.1/sriov_numvfs

echo 8 > /sys/bus/pci/devices/0000\:01\:00.1/sriov_numvfs

University of Nantes - IT Services

Yoann (dot) Juet (at) univ-nantes.fr

Questions