

# To study Linux and Hardware with QEMU

Dongli Zhang

July 2, 2019

## When to study and debug SCSI ...

When to study the concept of SCSI `<adapter, channel, id, lun>` ...

- There is **no** SCSI hardware available
- To reboot the test server is **time-consuming**
- It is hard to customize **SCSI topology** (e.g., `#lun` or `#target`)
- It is hard to customize **SCSI config** (e.g., `#queue` for `mq`)

We may want to study more **Linux and Hardware features...**

- SCSI
- NVMe
- NVDIMM
- Virtio
- Ethernet
- PCI and PCIe
- BIOS
- IOMMU
- NUMA
- CPU Hotplug
- Memroy Hotplug
- PM Suspend

# Why QEMU

QEMU is a generic and open source machine emulator and virtualizer

- QEMU can emulate lots of hardware
- QEMU can boot from Linux kernel on host
  - It is time-consuming to build and install kernel in VM

This tutorial is **NOT** to...

- teach how to use QEMU cmdline
- how to build and debug kernel inside guest
- how to debug generic features like buddy allocator or CFS scheduler
- how to debug advanced features (e.g., qlogic) with QEMU
- what is SCSI, NVMe, NVDIMM ...

- QEMU version in the tutorial: commit **076243ffe6c1**
- Linux version in the tutorial: tag **v5.2-rc4**

To build Linux on host (All CONFIG is 'Y'):

```
# make defconfig  
# make menuconfig  
# make -j8 > /dev/null
```

The output is something like:

```
/.../linux/arch/x86_64/boot/bzImage
```

To build QEMU:

```
# ./configure --target-list=x86_64-softmmu  
# make -j8 > /dev/null
```

We directly use output w/o 'make install':

```
./x86_64-softmmu/qemu-system-x86_64
```

# Boot QEMU and Guest Linux

- No need to work with guest IP, but only `<host_ip>:5022`
- Connect to guest via **VNC**
- Serial console output is redirected to **stdio**

To boot guest with Linux kernel locating on host:

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -serial stdio -smp 4 -m 4096M \
-net nic -net user,hostfwd=tcp::5022-:22 \
-kernel /home/user/linux/arch/x86_64/boot/bzImage \
-append "root=/dev/sda1 init=/sbin/init text console=ttyS0" \
-hda /home/user/img/boot.qcow2
```

To login to guest in another shell:

```
# ssh user@<host_ip> -p 5022
```

or

```
# vncviewer <host_ip>
```

- SeaBIOS is the default BIOS for QEMU and KVM
- <https://git.seabios.org/seabios.git>
- SeaBIOS version for the tutorial: commit **6e56ed129c97**

The below options are enabled to dump debug message to serial port:

```
CONFIG_DEBUG_LEVEL=8  
CONFIG_DEBUG_SERIAL=y  
CONFIG_DEBUG_SERIAL_PORT=0x3f8  
CONFIG_DEBUG_IO=y
```

To build SeaBIOS:

```
# make menuconfig  
# make
```

The output is at:

```
/.../seabios/out/bios.bin
```

**-serial stdio** is used to dump debug message to stdio

To boot guest with Linux kernel and BIOS:

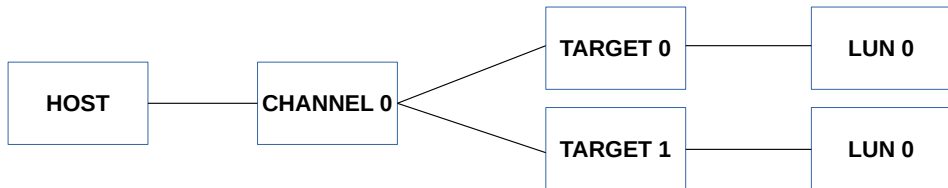
```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-bios /home/user/seabios/out/bios.bin -serial stdio
```



2 targets (each with a lun) on the same <adapter, channel>

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-device megasas,id=scsi0 \  
-device scsi-hd,drive=drive0,bus=scsi0.0,channel=0,scsi-id=0,lun=0,bootindex=1 \  
-drive file=/home/user/img/boot.qcow2,if=none,id=drive0 \  
-device scsi-hd,drive=drive1,bus=scsi0.0,channel=0,scsi-id=1,lun=0 \  
-drive file=/home/user/img/disk.qcow2,if=none,id=drive1
```

## SCSI: megasas 2/2

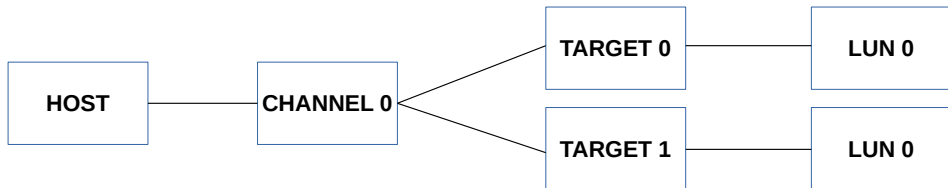


```
[ 0.626341] scsi host0: Avago SAS based MegaRAID driver
[ 0.644708] scsi 0:2:0:0: Direct-Access QEMU QEMU HARDDISK 2.5+ PQ: 0 ANSI: 5
[ 0.646012] scsi 0:2:1:0: Direct-Access QEMU QEMU HARDDISK 2.5+ PQ: 0 ANSI: 5
[ 0.671123] sd 0:2:0:0: Attached scsi generic sg0 type 0
[ 0.671710] sd 0:2:1:0: Attached scsi generic sg1 type 0
[ 0.673409] sd 0:2:1:0: [sdb] Attached SCSI disk
[ 0.680489] sd 0:2:0:0: [sda] Attached SCSI disk
```

2 targets (each with a lun) on the same <adapter, channel>

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-device lsi53c895a,id=scsi0 \  
-device scsi-hd,drive=drive0,bus=scsi0.0,channel=0,scsi-id=0,lun=0,bootindex=1 \  
-drive file=/home/user/img/boot.qcow2,if=none,id=drive0 \  
-device scsi-hd,drive=drive1,bus=scsi0.0,channel=0,scsi-id=1,lun=0 \  
-drive file=/home/user/img/disk.qcow2,if=none,id=drive1
```

## SCSI: lsi53c895a 2/2

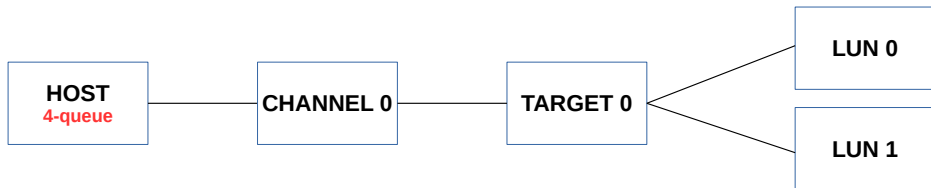


```
[ 0.610488] scsi host0: sym-2.2.3  
[ 3.603414] scsi 0:0:0:0: Direct-Access QEMU QEMU HARDDISK 2.5+ PQ: 0 ANSI: 5  
[ 3.613141] scsi 0:0:1:0: Direct-Access QEMU QEMU HARDDISK 2.5+ PQ: 0 ANSI: 5  
[ 3.623833] sd 0:0:0:0: Attached scsi generic sg0 type 0  
[ 3.624993] sd 0:0:1:0: Attached scsi generic sg1 type 0  
[ 3.632309] sd 0:0:0:0: [sda] Attached SCSI disk  
[ 3.641668] sd 0:0:1:0: [sdb] Attached SCSI disk
```

## 2 lun on the same <adapter, channel, id>

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-device virtio-scsi-pci,id=scsi0,num_queues=4 \  
-device scsi-hd,drive=drive0,bus=scsi0.0,channel=0,scsi-id=0,lun=0,bootindex=1 \  
-drive file=/home/user/img/boot.qcow2,if=none,id=drive0 \  
-device scsi-hd,drive=drive1,bus=scsi0.0,channel=0,scsi-id=0,lun=1 \  
-drive file=/home/user/img/disk.qcow2,if=none,id=drive1
```

## SCSI: virtio\_scsi 2/3



```
[ 0.604610] scsi host0: Virtio SCSI HBA
[ 0.606220] scsi 0:0:0:0: Direct-Access QEMU QEMU HARDDISK 2.5+ PQ: 0 ANSI: 5
[ 0.607168] scsi 0:0:0:1: Direct-Access QEMU QEMU HARDDISK 2.5+ PQ: 0 ANSI: 5
[ 0.617180] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 0.618537] sd 0:0:0:1: [sdb] Attached SCSI disk
[ 0.619014] sd 0:0:0:1: Attached scsi generic sg1 type 0
[ 0.625877] sd 0:0:0:0: [sda] Attached SCSI disk
```

# SCSI: virtio\_scsi 3/3

```
# ls /sys/block/sda/mq
```

```
0 1 2 3
```

```
# ls /sys/block/sdb/mq
```

```
0 1 2 3
```

```
# cat /proc/interrupts | grep virtio
```

24:	0	0	0	0 PCI-MSI 65536-edge	virtio0-config
25:	0	0	0	0 PCI-MSI 65537-edge	virtio0-control
26:	0	0	0	0 PCI-MSI 65538-edge	virtio0-event
27:	1171	0	0	0 PCI-MSI 65539-edge	virtio0-request
28:	0	1180	0	0 PCI-MSI 65540-edge	virtio0-request
29:	0	0	831	0 PCI-MSI 65541-edge	virtio0-request
30:	0	0	0	1636 PCI-MSI 65542-edge	virtio0-request

## NVMe device with 8 hardware queues

- Customize **num\_queues** to test how NVMe driver works with different #queues

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device nvme,drive=nvme0,serial=deadbeaf1,num_queues=8 \  
-drive file=/home/user/img/disk.qcow2,if=none,id=nvme0
```



```
[ 0.576209] nvme nvme0: pci function 0000:00:04.0  
[ 0.620458] nvme nvme0: 4/0/0 default/read/poll queues
```

```
# ls /dev/nvme0  
nvme0 nvme0n1
```

```
# cat /proc/interrupts | grep nvme
```

24:	11	0	0	0	PCI-MSI 65536-edge	<b>nvme0q0</b>
25:	40	0	0	0	PCI-MSI 65537-edge	<b>nvme0q1</b>
26:	0	0	0	0	PCI-MSI 65538-edge	<b>nvme0q2</b>
27:	0	0	41	0	PCI-MSI 65539-edge	<b>nvme0q3</b>
28:	0	0	0	0	PCI-MSI 65540-edge	<b>nvme0q4</b>

## NVDIMM: Non-Volatile Dual In-line Memory Module

- **pmem** and **blk** types
- QEMU supports only **pmem** type

```
# qemu-system-x86_64 -vnc :0 -smp 4 \  
-machine pc,nvdimmm,accel=kvm \  
-m 2G,maxmem=10G,slots=4 \  
-object memory-backend-file,share,id=mem1,mem-path=nvdimmm.img,size=16G \  
-device nvdimmm,memdev=mem1,id=nvdimmm1 \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-hda /home/user/img/boot.qcow2 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text"
```

# Install utility library for managing the libnvdimm

- `sudo apt-get install libndctl ndctl`
- <https://github.com/pmem/ndctl>

```
CONFIG_BLK_DEV_RAM_DAX=y
CONFIG_FS_DAX=y
CONFIG_X86_PMEM_LEGACY=y
CONFIG_LIBNVDIMM=y
CONFIG_BLK_DEV_PMEM=m
CONFIG_ARCH_HAS_PMEM_API=y
CONFIG_TRANSPARENT_HUGEPAGE=y
CONFIG_MEMORY_HOTPLUG=y
CONFIG_MEMORY_HOTREMOVE=y
CONFIG_ZONE_DEVICE=y
CONFIG_FS_DAX_PMD=y
CONFIG_ACPI_NFIT=y
```

```
# ndctl list
"dev":" namespace0.0",
"mode":" raw",
"size":17179869184,
"sector_size":512,
"blockdev":" pmem0",
"numa_noe":0
```

## Virtio Block device with 4 hardware queues

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device virtio-blk-pci,drive=drive0,id=virtblk0,num-queues=4 \  
-drive file=/home/user/img/disk.qcow2,if=none,id=drive0
```

## Virtio Block 2/2

```
# ls /dev/vda  
/dev/vda
```

```
#ls /sys/block/vda/mq/  
0 1 2 3
```

```
#cat /proc/interrupts | grep virtio
```

24:	0	0	0	0	PCI-MSI 65536-edge	virtio0-config
25:	1	0	0	0	PCI-MSI 65537-edge	virtio0-req.0
26:	0	30	0	0	PCI-MSI 65538-edge	virtio0-req.1
27:	0	0	33	0	PCI-MSI 65539-edge	virtio0-req.2
28:	0	0	0	0	PCI-MSI 65540-edge	virtio0-req.3

# QEMU Tap Bridge Helper Script

- The script bridges **tap** created by QEMU to **host bridge** (e.g., br0)
- Used by QEMU **-netdev** during VM creation

```
# cat /home/user/qemu-ifup  
  
#!/bin/sh  
# Script to bring a network (tap) device for qemu up.  
  
br="br0"  
  
ifconfig $1 up  
brctl addif $br "$1"  
  
exit
```

# Virtio Net 1/2

- To create Virtio Net device with **4** queues (consuming 9 vectors)
- **qemu-ifup** is from previous slide
- The **#vector** should be configured correctly ( $4(\text{TX})+4(\text{RX})+1(\text{Conf})=9$ )

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 -m 4096M \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device virtio-net-pci,netdev=tapnet,mq=true,vectors=9 \  
-netdev tap,id=tapnet,ifname=tap0,\  
script=/home/user/qemu-ifup,downscript=no,queues=4,vhost=off
```

## Virtio Net 2/2

```
host# ip addr | grep tap0
```

```
34: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br0  
state UNKNOWN group default qlen 1000
```

```
vm# cat /proc/interrupts | grep virtio
```

24:	0	0	0	0	PCI-MSI 49152-edge	virtio0-config
25:	57	1	0	0	PCI-MSI 49153-edge	<b>virtio0-input.0</b>
26:	0	0	1	0	PCI-MSI 49154-edge	<b>virtio0-output.0</b>
27:	0	110	0	1	PCI-MSI 49155-edge	<b>virtio0-input.1</b>
28:	1	0	0	0	PCI-MSI 49156-edge	<b>virtio0-output.1</b>
29:	0	1	135	0	PCI-MSI 49157-edge	<b>virtio0-input.2</b>
30:	0	0	1	0	PCI-MSI 49158-edge	<b>virtio0-output.2</b>
31:	0	0	0	49	PCI-MSI 49159-edge	<b>virtio0-input.3</b>
32:	0	0	0	0	PCI-MSI 49160-edge	<b>virtio0-output.3</b>



The **e1000e** can be substituted by:

- **r8169**(8169cp), **vmxnet3**(vmxnet3), **i82550**(e100), **e1000**(e1000)

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 -m 4096M \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device e1000e,netdev=tapnet \  
-netdev tap,id=tapnet,ifname=tap0,script=/home/user/qemu-ifup,downscript=no
```

```
vm# ethtool -i enp0s3 | grep driver  
driver: e1000e
```

## Create 2 PCI-2-PCI bridge's secondary bus

- The 1st secondary bus is with 1 E1000 NIC
- The 2nd secondary bus is with 2 E1000 NIC

```
# qemu-system-x86_64 -machine pc,accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device pci-bridge,id=bridge0,chassis_nr=1 \  
    -device e1000,bus=bridge0,addr=0x3 \  
-device pci-bridge,id=bridge1,chassis_nr=2 \  
    -device e1000,bus=bridge1,addr=0x3 \  
    -device e1000,bus=bridge1,addr=0x4
```

## PCI Bridge 2/3

00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)  
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]  
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]  
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)  
00:02.0 VGA compatible controller: Device 1234:1111 (rev 02)  
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)  
00:04.0 PCI bridge: Red Hat, Inc. **QEMU PCI-PCI bridge**  
00:05.0 PCI bridge: Red Hat, Inc. **QEMU PCI-PCI bridge**  
01:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)  
02:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)  
02:04.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)

# PCI Bridge 3/3

00:04.0 PCI bridge: Red Hat, Inc. QEMU PCI-PCI bridge

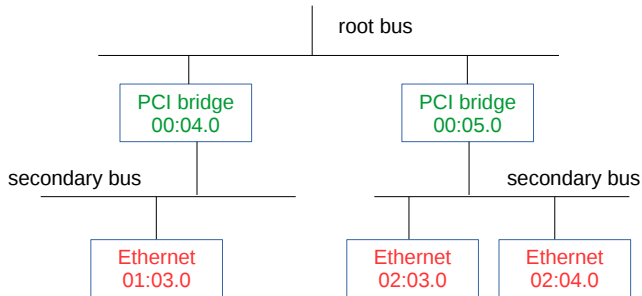
00:05.0 PCI bridge: Red Hat, Inc. QEMU PCI-PCI bridge

01:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)

02:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)

02:04.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)

```
# lspci -t
-[0000:00]--+-00.0
              +-01.0
              +-01.1
              +-01.3
              +-02.0
              +-03.0
              +-04.0-[01]----03.0
                  \-05.0-[02]--+-03.0
                              \-04.0
```



### Create 2 PCI Expander Bridge (PXB)'s primary bus

- The 1st primary bus is with 1 E1000 NIC
- The 2nd primary bus is with 2 E1000 NIC

```
# qemu-system-x86_64 -machine pc,accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device pxb,id=bridge1,bus=pci.0,bus_nr=3 \  
    -device e1000,bus=bridge1,addr=0x3 \  
-device pxb,id=bridge2,bus=pci.0,bus_nr=8 \  
    -device e1000,bus=bridge2,addr=0x3 \  
    -device e1000,bus=bridge2,addr=0x4
```

## PCI Root Bus 2/3

00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)  
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]  
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]  
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)  
00:02.0 VGA compatible controller: Device 1234:1111 (rev 02)  
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)  
00:04.0 Host bridge: Red Hat, Inc. QEMU **PCI Expander bridge**  
00:05.0 Host bridge: Red Hat, Inc. QEMU **PCI Expander bridge**  
03:00.0 PCI bridge: Red Hat, Inc. QEMU **PCI-PCI bridge**  
04:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)  
08:00.0 PCI bridge: Red Hat, Inc. QEMU **PCI-PCI bridge**  
09:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)  
09:04.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)

# PCI Root Bus 3/3

00:04.0 Host bridge: Red Hat, Inc. QEMU PCI Expander bridge

00:05.0 Host bridge: Red Hat, Inc. QEMU PCI Expander bridge

03:00.0 PCI bridge: Red Hat, Inc. QEMU PCI-PCI bridge

04:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)

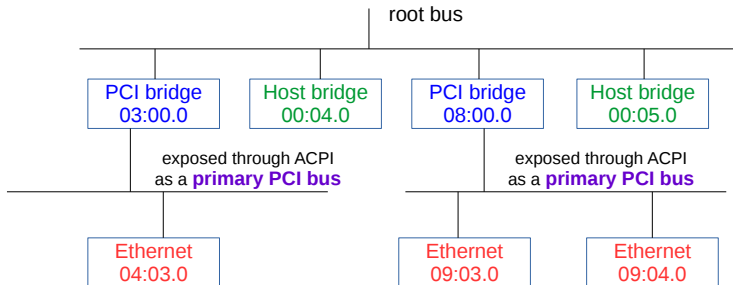
08:00.0 PCI bridge: Red Hat, Inc. QEMU PCI-PCI bridge

09:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)

09:04.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 03)

```
# lspci -t
```

```
-+-[0000:08]---00.0-[09]--+-03.0
|                               \-04.0
+-[0000:03]---00.0-[04]----03.0
\-[0000:00]--+-00.0
              +-01.0
              +-01.1
              +-01.3
              +-02.0
              +-03.0
              +-04.0
              \-05.0
```



# PCI Express Root Complex 1/3

## Create 2 extra PCI Express Root Complex

- The 1st PCI Express Root Complex is with 2 E1000 NIC
- The 2nd PCI Express Root Complex is with 1 E1000 NIC

```
# qemu-system-x86_64 -machine q35,accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device pxb-pcie,id=pcie.1,bus_nr=2,bus=pcie.0 \  
  -device ioh3420,id=pcie_pci_bridge1,bus=pcie.1,chassis=1 \  
    -device e1000e,bus=pcie_pci_bridge1 \  
  -device ioh3420,id=pcie_pci_bridge2,bus=pcie.1,chassis=2 \  
    -device e1000e,bus=pcie_pci_bridge2 \  
-device pxb-pcie,id=pcie.2,bus_nr=8,bus=pcie.0 \  
  -device ioh3420,id=pcie_pci_bridge3,bus=pcie.2,chassis=3 \  
    -device e1000e,bus=pcie_pci_bridge3
```



## PCI Express Root Complex 2/3

00:00.0 Host bridge: Intel Corporation 82G33/G31/P35/P31 Express DRAM Controller  
00:01.0 VGA compatible controller: Device 1234:1111 (rev 02)  
00:02.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection  
00:03.0 Host bridge: Red Hat, Inc. QEMU PCIe Expander bridge  
00:04.0 Host bridge: Red Hat, Inc. QEMU PCIe Expander bridge  
00:1f.0 ISA bridge: Intel Corporation 82801IB (ICH9) LPC Interface Controller (rev 02)  
00:1f.2 SATA controller: Intel Corporation 82801IR/IO/IH (ICH9R/DO/DH) 6 port SATA Controller [AHCI mode] (rev 02)  
00:1f.3 SMBus: Intel Corporation 82801I (ICH9 Family) SMBus Controller (rev 02)  
02:00.0 PCI bridge: Intel Corporation 7500/5520/5500/X58 I/O Hub PCI Express Root Port 0  
02:01.0 PCI bridge: Intel Corporation 7500/5520/5500/X58 I/O Hub PCI Express Root Port 0  
03:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection  
04:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection  
08:00.0 PCI bridge: Intel Corporation 7500/5520/5500/X58 I/O Hub PCI Express Root Port 0  
09:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection

# PCI Express Root Complex 3/3

00:03.0 Host bridge: Red Hat, Inc. QEMU PCIe Expander bridge

00:04.0 Host bridge: Red Hat, Inc. QEMU PCIe Expander bridge

02:00.0 PCI bridge: Intel Corporation 7500/5520/5500/X58 I/O Hub PCI Express Root Port 0 (rev 02)

02:01.0 PCI bridge: Intel Corporation 7500/5520/5500/X58 I/O Hub PCI Express Root Port 0 (rev 02)

03:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection

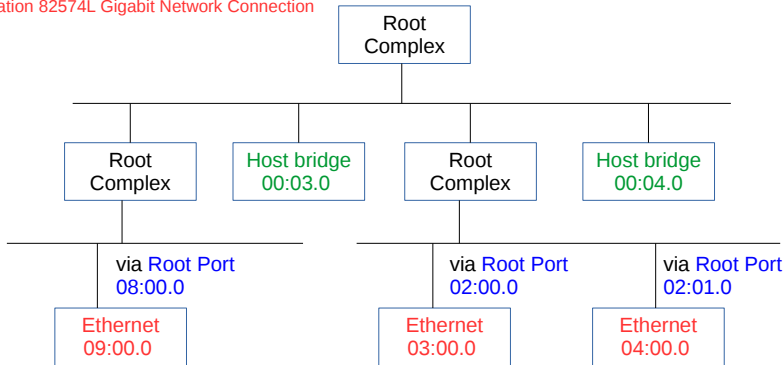
04:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection

08:00.0 PCI bridge: Intel Corporation 7500/5520/5500/X58 I/O Hub PCI Express Root Port 0 (rev 02)

09:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection

```
# lspci -t
```

```
--[0000:08]---00.0-[09]---00.0
+-[0000:02]-+00.0-[03]---00.0
|           \-01.0-[04]---00.0
\-[0000:00]-+00.0
            +-01.0
            +-02.0
            +-03.0
            +-04.0
            +-1f.0
            +-1f.2
            \-1f.3
```



## Create 1 PCI Express Switch

- There is 1 Upstream Port connecting to 2 Downstream Ports
- Each Downstream Port is connected to 1 E1000e NIC

```
# qemu-system-x86_64 -machine q35,accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device ioh3420,id=root_port1,bus=pcie.0 \  
  -device x3130-upstream,id=upstream1,bus=root_port1 \  
    -device xio3130-downstream,id=downstream1,bus=upstream1,chassis=9 \  
      -device e1000e,bus=downstream1 \  
    -device xio3130-downstream,id=downstream2,bus=upstream1,chassis=10 \  
      -device e1000e,bus=downstream2
```

## PCI Express Switches 2/3

00:00.0 Host bridge: Intel Corporation 82G33/G31/P35/P31 Express DRAM Controller  
00:01.0 VGA compatible controller: Device 1234:1111 (rev 02)  
00:02.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection  
**00:03.0 PCI bridge: Intel Corporation 7500/5520/5500/X58 I/O Hub PCI Express Root Port 0**  
00:1f.0 ISA bridge: Intel Corporation 82801IB (ICH9) LPC Interface Controller (rev 02)  
00:1f.2 SATA controller: Intel Corporation 82801IR/IO/IH (ICH9R/DO/DH) 6 port SATA Controller [AHCI mode] (rev 02)  
00:1f.3 SMBus: Intel Corporation 82801I (ICH9 Family) SMBus Controller (rev 02)  
**01:00.0 PCI bridge: Texas Instruments XIO3130 PCI Express Switch (Upstream) (rev 02)**  
**02:00.0 PCI bridge: Texas Instruments XIO3130 PCI Express Switch (Downstream) (rev 01)**  
**02:01.0 PCI bridge: Texas Instruments XIO3130 PCI Express Switch (Downstream) (rev 01)**  
03:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection  
04:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection

# PCI Express Switches 3/3

00:03.0 PCI bridge: Intel Corporation 7500/5520/5500/X58 I/O Hub PCI Express Root Port 0 (rev 02)

01:00.0 PCI bridge: Texas Instruments XIO3130 PCI Express Switch (Upstream) (rev 02)

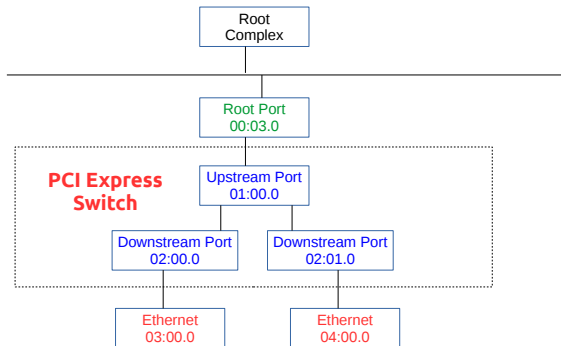
02:00.0 PCI bridge: Texas Instruments XIO3130 PCI Express Switch (Downstream) (rev 01)

02:01.0 PCI bridge: Texas Instruments XIO3130 PCI Express Switch (Downstream) (rev 01)

03:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection

04:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network Connection

```
# lspci -t
-[0000:00]--+-00.0
             +-01.0
             +-02.0
             +-03.0-[01-04]----00.0-[02-04]--+-00.0-[03]----00.0
             |                               \-01.0-[04]----00.0
             +-1f.0
             +-1f.2
             \-1f.3
```



## Intel VT-d with Interrupt Remapping (IR) enabled

- Only **q35** machine supports virtual IOMMU
- **intel\_iommu=on** should be added to kernel cmdline

```
# qemu-system-x86_64 -vnc :0 -smp 4 -m 4096M \  
-machine q35,accel=kvm,kernel_irqchip=split \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text intel_iommu=on" \  
-hda /home/user/img/boot.qcow2 \  
-device nvme,drive=nvme0,serial=deadbeaf1,num_queues=8 \  
-drive file=/home/user/img/disk.qcow2,if=none,id=nvme0 \  
-device intel_iommu,intremap=on
```

```
# dmesg | egrep "IOMMU|iommu"
```

```
[ 0.000000] DMAR: IOMMU enabled  
[ 0.003000] DMAR-IR: IOAPIC id 0 under DRHD base 0xfed90000 IOMMU 0  
[ 0.477614] pci 0000:00:00.0: Adding to iommu group 0  
[ 0.478078] pci 0000:00:01.0: Adding to iommu group 1  
[ 0.478517] pci 0000:00:02.0: Adding to iommu group 2  
[ 0.478963] pci 0000:00:03.0: Adding to iommu group 3  
[ 0.479421] pci 0000:00:1f.0: Adding to iommu group 4  
[ 0.479857] pci 0000:00:1f.2: Adding to iommu group 4  
[ 0.480316] pci 0000:00:1f.3: Adding to iommu group 4
```

## Machine of 2 NUMA node

- 2 memory NUMA node (1st=2048MB, 2nd=256MB)
- 2 CPU socket and each has 2 cores (of 1 thread)

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-smp 4,sockets=2,cores=2,threads=1 -m 2304M \  
-numa node,mem=2048,cpus=0-1 \  
-numa node,mem=256,cpus=2-3
```



# NUMA 2/2



```
# ls /sys/devices/system/node/node0 | grep cpu[0-9]
```

```
cpu0 cpu1
```

```
# ls /sys/devices/system/node/node1 | grep cpu[0-9]
```

```
cpu2 cpu3
```

```
[ 0.007388] Initmem setup node 0 [mem 0x0000000000001000-0x000000007fffffff]
```

```
[ 0.007390] On node 0 totalpages: 524190
```

```
[ 0.015744] Initmem setup node 1 [mem 0x0000000080000000-0x000000008ffdf000]
```

```
[ 0.015746] On node 1 totalpages: 65504
```

# CPU Hotplug 1/3

- Can be used to debug how block/net drivers work with hotplug
- Init #cpu is 2 while the max #cpu is 4

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -m 4096M \  
-smp 2,maxcpus=4 \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device nvme,drive=nvme0,serial=deadbeaf1,num_queues=8 \  
-drive file=/home/user/img/disk.qcow2,if=none,id=nvme0 \  
-monitor stdio  
QEMU 4.0.50 monitor - type 'help' for more information  
(qemu)
```

## CPU Hotplug 2/3

To add new vcpu from QEMU:

```
(qemu) device_add qemu64-x86_64-cpu,id=core1,socket-id=2,core-id=0,thread-id=0
```

To add online new vcpu by VM:

```
vm# echo 1 > /sys/devices/system/cpu/cpu2/online
```

```
vm# dmesg
```

```
[ 1021.173154] CPU2 has been hot-added  
[ 1029.524516] smpboot: Booting Node 0 Processor 2 APIC 0x2  
[ 1029.604423] Will online and init hotplugged CPU: 2
```

To offline new vcpu by VM:

```
vm# echo 0 > /sys/devices/system/cpu/cpu2/online
```

```
vm# dmesg
```

```
[ 1354.176282] smpboot: CPU 2 is now offline
```

# CPU Hotplug 3/3

- A block-mq cpu hotplug bug reproduced by QEMU: <https://patchwork.kernel.org/patch/10889307>
- Inflight requests on software queue is spliced to the incorrect hardware queue during cpu offline

When a cpu is offline, `blk_mq_hctx_notify_dead()` is called once for each hctx for the offline cpu.

While `blk_mq_hctx_notify_dead()` is used to splice all `ctx->rq_lists[type]` to `hctx->dispatch`, it never checks whether the ctx is already mapped to the hctx.

For example, on a VM (with nvme) of 4 cpu, to offline cpu 2 out of the 4 cpu (0-3), `blk_mq_hctx_notify_dead()` is called once for each io queue hctx:

```
1st: blk_mq_ctx->cpu = 2 for blk_mq_hw_ctx->queue_num = 3
2nd: blk_mq_ctx->cpu = 2 for blk_mq_hw_ctx->queue_num = 2
3rd: blk_mq_ctx->cpu = 2 for blk_mq_hw_ctx->queue_num = 1
4th: blk_mq_ctx->cpu = 2 for blk_mq_hw_ctx->queue_num = 0
```

Although `blk_mq_ctx->cpu = 2` is only mapped to `blk_mq_hw_ctx->queue_num = 2` in this case, its `ctx->rq_lists[type]` will however be moved to `blk_mq_hw_ctx->queue_num = 3` during the 1st call of `blk_mq_hctx_notify_dead()`.

This patch would return and go ahead to next call of `blk_mq_hctx_notify_dead()` if ctx is not mapped to hctx.

# Memory Hotplug 1/2

Boot with:

- initial 2048MB memory
- extra 4 slots to hotplug memory up to extra 4096MB-2048MB=2048MB

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 \  
-m 2048M,slots=4,maxmem=4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-monitor stdio  
QEMU 4.0.50 monitor - type 'help' for more information  
(qemu)
```

# Memory Hotplug 2/2

**Before** memory hotplug:

```
# cat /proc/meminfo | grep MemTotal
```

```
MemTotal:      1972380 kB
```

To add 1024MB memory:

```
(qemu) object_add memory-backend-ram,id=mem1,size=1024M
```

```
(qemu) device_add pc-dimm,id=dimm1,memdev=mem1
```

```
# dmesg
```

```
[ 99.324281] Built 1 zonelists, mobility grouping on. Total pages: 523480
```

```
[ 99.324282] Policy zone: Normal
```

**After** memory hotplug (more 'memory<section>' available under [/sys/devices/system/memory/](#)):

```
# cat /proc/meminfo | grep MemTotal
```

```
MemTotal:      3020956 kB
```

# PM Suspend 1/3

- To debug how each kernel component works during PM freezing, e.g., **unlike jbd2, o2hb\_thread does NOT freeze itself proactively**
- To debug how each driver (e.g, nvme or virtio) works with PM suspend

```
# qemu-system-x86_64 -machine accel=kvm -vnc :0 -smp 4 -m 4096M \  
-net nic -net user,hostfwd=tcp::5022-:22 \  
-kernel /home/user/linux/arch/x86_64/boot/bzImage \  
-append "root=/dev/sda1 init=/sbin/init text" \  
-hda /home/user/img/boot.qcow2 \  
-device nvme,drive=nvme0,serial=deadbeaf1,num_queues=8 \  
-drive file=/home/user/img/disk.qcow2,if=none,id=nvme0 \  
-monitor stdio \  
QEMU 4.0.50 monitor - type 'help' for more information \  
(qemu)
```

# PM Suspend 2/3

```
# echo freeze > /sys/power/state —> to suspend from VM  
(qemu) system_powerdown —> to resume from QEMU
```

```
# dmesg
```

```
[ 84.198422] PM: suspend entry (s2idle)  
[ 85.249993] Filesystems sync: 1.051 seconds  
[ 85.252942] Freezing user space processes ... (elapsed 0.001 seconds) done.  
[ 85.254433] OOM killer disabled.  
[ 85.254434] Freezing remaining freezable tasks ... (elapsed 0.000 seconds) done.  
[ 85.255212] printk: Suspending console(s) (use no_console_suspend to debug)  
[ 85.261298] sd 0:0:0:0: [sda] Synchronizing SCSI cache  
[ 85.283587] sd 0:0:0:0: [sda] Stopping disk  
[ 105.107310] sd 0:0:0:0: [sda] Starting disk  
[ 105.115072] nvme nvme0: 4/0/0 default/read/poll queues  
[ 105.261509] ata2.01: NODEV after polling detection  
[ 105.261896] ata1.01: NODEV after polling detection  
[ 105.265826] OOM killer enabled.  
[ 105.265827] Restarting tasks ... done.  
[ 105.273076] PM: suspend exit  
[ 107.172694] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
```



# PM Suspend 3/3

- Sample kernel warning at [kernel/irq/chip.c:210 irq\\_startup+0xd6/0xe0](#)
- Bug reported: <http://lists.infradead.org/pipermail/linux-nvme/2019-April/023234.html>
- How I reproduce with QEMU: <http://lists.infradead.org/pipermail/linux-nvme/2019-April/023237.html>

On 04/04/2019 04:55 PM, Ming Lei wrote:

On Thu, Apr 04, 2019 at 08:23:59AM +0000, fin4478 fin4478 wrote:

Hi,

I do not use suspend/resume but noticed this kernel warning when testing it. This warning is present in earlier kernels too. My system works fine after resume. If there is a patch to fix this, I can test it.

```
[ 53.403033] PM: suspend entry (deep)
[ 53.403034] PM: Syncing filesystems ... done.
[ 53.404775] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 53.405972] OOM killer disabled.
[ 53.405973] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
[ 53.407036] printk: Suspending console(s) (use no_console_suspend to debug)
[ 53.407491] ACPI Debug: "RRIO"
[ 53.407505] serial 00:03: disabled
[ 53.407560] r8169 0000:07:00.0 enp7s0: Link is Down
[ 53.415042] sd 5:0:0:0: [sda] Synchronizing SCSI cache
[ 53.415065] sd 5:0:0:0: [sda] Stopping disk
[ 53.428943] WARNING: CPU: 10 PID: 3127 at kernel/irq/chip.c:210 irq_startup+0xd6/0xe0
```

Looks the 'WARN\_ON\_ONCE(force)' in irq\_startup() is a bit too strict.

irq\_build\_affinity\_masks() doesn't guarantee that each IRQ's affinity can include at least one online CPU.

# Do not always trust QEMU

- QEMU can have bug: <https://www.spinics.net/lists/linux-block/msg37936.html>
- Fixed in QEMU commit [9d6459d21a6e](#) ("nvme: fix write zeroes offset and count")

Hi,

It is observed that ext4 is corrupted easily by running some workloads on QEMU NVMe, such as:

- 1) `mkfs.ext4 /dev/nvme0n1`
- 2) `mount /dev/nvme0n1 /mnt`
- 3) `cd /mnt; git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git`
- 4) then the following error message may show up:

```
[ 1642.271816] EXT4-fs error (device nvme0n1): ext4_mb_generate_buddy:747: group 0, block bitmap and bg descriptor inconsistent: 32768 vs 23513 free clusters
```

Or `fsck.ext4` will complain after running `'umount /mnt'`

The issue disappears by reverting [6e02318eaea53eaafe6](#) ("nvme: add support for the Write Zeroes command").

QEMU version:

```
QEMU emulator version 2.10.2(qemu-2.10.2-1.fc27)
Copyright (c) 2003-2017 Fabrice Bellard and the QEMU Project developers
```

Thanks,  
Ming

How to **effectively** setup **debug/study** environment with QEMU:

- Build and run Linux kernel from host
- Use QEMU but not libvirt

Components and Features to debug:

- SCSI (megasas, lsi53c895a, virtio\_scsi), NVMe, NVDIMM
- Virtio Block and Virtio Net
- Ethernet Card (e1000e, e1000, e100, 8139cp, vmxnet3)
- PCI Bus, PCIe Root Complex and PCIe Switch
- BIOS (seabios), IOMMU (intel), PM Suspend
- NUMA, Memory Hotplug, CPU Hotplug