# QEMU虚拟存储的几种访问形式

[mnstory.net](http://mnstory.net)

## 存储相关命令

查看是否有FC存储：

# lspci -nn | grep "Fibre Channel"

81:00.0 Fibre Channel [0c04]: Brocade Communications Systems, Inc. 425/825/42B/82B 4Gbps/8Gbps PCIe dual port FC HBA [1657:0013] (rev 01)

81:00.1 Fibre Channel [0c04]: Brocade Communications Systems, Inc. 425/825/42B/82B 4Gbps/8Gbps PCIe dual port FC HBA [1657:0013] (rev 01)

其中ID为81:00.0 的 vendor & device ID 为 1657: 0013（上面中括号内），通过device ID，可以查找是否已经有设备驱动存在：

# grep 1657 /lib/modules/`uname -r`/modules.\* | grep 0013

/lib/modules/3.10.0/modules.alias:alias pci:v00001657d00000013sv\*sd\*bc\*sc\*i\* bfa

上面命令最后的bfa为此HBA的驱动名称，查看是否已经加载驱动：

# lsmod | grep bfa

bfa 2300947 6

scsi\_transport\_fc 55172 1 bfa

如果没有加载，可以通过modprobe -v bfa手动加载：

# modinfo bfa

filename: /lib/modules/3.10.0/kernel/drivers/scsi/bfa/bfa.ko

version: 3.2.3.0

author: Brocade Communications Systems, Inc.

description: Brocade Fibre Channel HBA Driver fcpim ipfc

license: GPL

rhelversion: 7.1

查看fc adapters详细信息：

# ls -l /sys/class/fc\_host/

total 0

lrwxrwxrwx 1 root root 0 May 13 11:49 host11 -> ../../devices/pci0000:80/0000:80:01.0/0000:81:00.0/host11/fc\_host/host11/

lrwxrwxrwx 1 root root 0 May 13 11:49 host12 -> ../../devices/pci0000:80/0000:80:01.0/0000:81:00.1/host12/fc\_host/host12/

# ls -lh /sys/class/fc\_host/host11/

total 0

-r--r--r-- 1 root root 4.0K May 17 14:38 active\_fc4s

lrwxrwxrwx 1 root root 0 May 17 14:38 device -> ../../../host11/

-rw-r--r-- 1 root root 4.0K May 17 14:38 dev\_loss\_tmo

-r--r--r-- 1 root root 4.0K May 17 14:38 fabric\_name

--w------- 1 root root 4.0K May 17 14:38 issue\_lip

-r--r--r-- 1 root root 4.0K May 17 14:38 maxframe\_size

-r--r--r-- 1 root root 4.0K May 17 14:38 max\_npiv\_vports

-r--r--r-- 1 root root 4.0K May 13 11:52 node\_name

-r--r--r-- 1 root root 4.0K May 17 14:38 npiv\_vports\_inuse

-r--r--r-- 1 root root 4.0K May 17 14:38 port\_id

-r--r--r-- 1 root root 4.0K May 13 11:52 port\_name

-r--r--r-- 1 root root 4.0K May 17 14:38 port\_state

-r--r--r-- 1 root root 4.0K May 17 14:38 port\_type

drwxr-xr-x 2 root root 0 May 17 14:38 power/

-r--r--r-- 1 root root 4.0K May 17 14:38 speed

drwxr-xr-x 2 root root 0 May 17 14:38 statistics/

lrwxrwxrwx 1 root root 0 May 13 11:49 subsystem -> ../../../../../../../class/fc\_host/

-r--r--r-- 1 root root 4.0K May 17 14:38 supported\_classes

-r--r--r-- 1 root root 4.0K May 17 14:38 supported\_fc4s

-r--r--r-- 1 root root 4.0K May 17 14:38 supported\_speeds

-r--r--r-- 1 root root 4.0K May 17 14:38 symbolic\_name

-rw-r--r-- 1 root root 4.0K May 17 14:38 tgtid\_bind\_type

-rw-r--r-- 1 root root 4.0K May 13 11:49 uevent

--w------- 1 root root 4.0K May 17 14:38 vport\_create

--w------- 1 root root 4.0K May 17 14:38 vport\_delete

# cat /sys/class/fc\_host/host11/speed

unknown

# cat /sys/class/fc\_host/host11/node\_name

0x20008c7cff65a8e4

# cat /sys/class/fc\_host/host11/port\_name

0x10008c7cff65a8e4

# cat /sys/class/fc\_host/host11/port\_id

0x000000

# cat /sys/class/fc\_host/host11/port\_type

Unknown

如果安装了systool也可以用如下命令查看详细信息：

# systool -c fc\_host

# systool -c fc\_host -v host11

查看当前的MUTLIPATH情况：

# multipath -ll

36f01faf000dcec22000048b25549e4f6 dm-3 DELL,MD36xxf

size=500G features='0' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

`- 12:0:1:10 sdi 8:128 active ready running

36f01faf000dcec22000048b45549e501 dm-69 DELL,MD36xxf

size=500G features='0' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

`- 12:0:1:11 sdj 8:144 active ready running

36f01faf000dcec2200004bd4555467b9 dm-4 DELL,MD36xxf

size=1.3T features='0' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

`- 12:0:1:4 sdg 8:96 active ready running

36f01faf000dcec22000048b05549e4ea dm-2 DELL,MD36xxf

size=500G features='0' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

`- 12:0:1:9 sdh 8:112 active ready running

36f01faf000dcec22000048a85549e49d dm-0 DELL,MD36xxf

size=500G features='0' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

`- 12:0:1:1 sdf 8:80 active ready running

使用lsblk查看当前block设备情况：

*NAME : 这是块设备名*

*MAJ:MIN : 本栏显示主要和次要设备号*

*RM : 本栏显示设备是否可移动设备。注意，在本例中设备sdb和sr0的RM值等于1，这说明他们是可移动设备*

*SIZE : 本栏列出设备的容量大小信息。例如298.1G表明该设备大小为298.1GB，而1K表明该设备大小为1KB*

*RO : 该项表明设备是否为只读。在本案例中，所有设备的RO值为0，表明他们不是只读的*

*TYPE :本栏显示块设备是否是磁盘或磁盘上的一个分区*

*MOUNTPOINT : 本栏指出设备挂载的挂载点*

# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sdf 8:80 0 500G 0 disk

├─sdf1 8:81 0 492.2G 0 part

└─36f01faf000dcec22000048a85549e49d (dm-0) 253:0 0 500G 0 mpath

└─36f01faf000dcec22000048a85549e49d-part1 (dm-69) 253:69 0 492.2G 0 part

sdg 8:96 0 1.3T 0 disk

└─36f01faf000dcec2200004bd4555467b9 (dm-1) 253:1 0 1.3T 0 mpath

sdh 8:112 0 500G 0 disk

└─36f01faf000dcec22000048b05549e4ea (dm-2) 253:2 0 500G 0 mpath /36f01fa

sdi 8:128 0 500G 0 disk

└─36f01faf000dcec22000048b25549e4f6 (dm-3) 253:3 0 500G 0 mpath /36f01fa

sdj 8:144 0 500G 0 disk

└─36f01faf000dcec22000048b45549e501 (dm-4) 253:4 0 500G 0 mpath

查看某一块设备信息：

# lsblk -b /dev/sdf

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sdf 8:80 0 536870912000 0 disk

├─sdf1 8:81 0 528449564160 0 part

└─36f01faf000dcec22000048a85549e49d (dm-0) 253:0 0 536870912000 0 mpath

└─36f01faf000dcec22000048a85549e49d-part1 (dm-1) 253:1 0 528449564160 0 part

-S参数，输出scsi info，TRAN栏为fc，表示是FC存储：

# lsblk -S

NAME HCTL TYPE VENDOR MODEL REV TRAN

sda 0:0:10:0 disk ATA INTEL SSDSC2BB48 0140

sdb 0:0:11:0 disk ATA INTEL SSDSC2BB48 0140

sdc 0:0:12:0 disk ATA INTEL SSDSC2BB48 0140

sdd 0:0:13:0 disk ATA INTEL SSDSC2BB48 0140

sde 5:0:0:0 disk ATA FORESEE 128GB SS N053 sata

sdf 12:0:1:1 disk DELL MD36xxf 0784 fc

sdg 12:0:1:4 disk DELL MD36xxf 0784 fc

sdh 12:0:1:9 disk DELL MD36xxf 0784 fc

sdi 12:0:1:10 disk DELL MD36xxf 0784 fc

sdj 12:0:1:11 disk DELL MD36xxf 0784 fc

## 共享外置存储在两台机的差异

共享外置映射到两台机器上，本身HBA卡不一致，by path路径有细微的变化，by id的无变化：

node1 # lspci -nn

81:00.0 Fibre Channel [0c04]: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PCI Express HBA [1077:2532] (rev 02)

81:00.1 Fibre Channel [0c04]: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PCI Express HBA [1077:2532] (rev 02)

node2 # lspci -nn

81:00.0 Fibre Channel [0c04]: Brocade Communications Systems, Inc. 425/825/42B/82B 4Gbps/8Gbps PCIe dual port FC HBA [1657:0013] (rev 01)

81:00.1 Fibre Channel [0c04]: Brocade Communications Systems, Inc. 425/825/42B/82B 4Gbps/8Gbps PCIe dual port FC HBA [1657:0013] (rev 01)

node1 # ll /dev/disk/by-path/

pci-0000:81:00.0-fc-0x2012f01fafdcec22-lun-9 -> ../../sdh

node2 # ll /dev/disk/by-path/

pci-0000:81:00.1-fc-0x2012f01fafdcec22-lun-9 -> ../../sdh

node1 # ll /dev/disk/by-id/

scsi-36f01faf000dcec22000048b05549e4ea -> ../../dm-2

wwn-0x6f01faf000dcec22000048b05549e4ea -> ../../sdh

node2 # ll /dev/disk/by-id/

scsi-36f01faf000dcec22000048b05549e4ea -> ../../dm-2

wwn-0x6f01faf000dcec22000048b05549e4ea -> ../../sda

node1 # multipath -ll

36f01faf000dcec22000048b05549e4ea dm-2 DELL,MD36xxf

size=500G features='0' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

`- 1:0:1:9 sdh 8:112 active ready running

node2 # multipath -ll

36f01faf000dcec22000048b05549e4ea dm-2 DELL,MD36xxf

size=500G features='0' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

`- 12:0:1:9 sdh 8:112 active ready running

## QEMU虚拟存储的几种访问形式

### 将HOST机QCOW2镜像，映射给GUEST机作为IDE磁盘

-drive file=/data/images/f9263e855786/vm-disk-1.qcow2,if=none,id=drive-ide0,cache=none -device ide-hd,bus=ide.0,unit=0,drive=drive-ide0,id=ide0

### 将HOST机QCOW2镜像，映射给GUEST机作为SCSI磁盘

-device lsi,id=scsihw0,bus=pci.0,addr=0x5 -drive file=/data/images/f9263e855786/vm-disk-2.qcow2,if=none,id=drive-scsi0,cache=none -device scsi-hd,bus=scsihw0.0,scsi-id=0,drive=drive-scsi0,id=scsi0

### 将HOST机QCOW2镜像，映射给GUEST机作为VIRTIO-BLK磁盘

-drive file=/data/images/f9263e855786/vm-disk-1.qcow2,if=none,id=drive-virtio1,cache=none,aio=native,cache.direct=on -device virtio-blk-pci,drive=drive-virtio1,id=virtio1,bus=pci.0,addr=0xb

此种方式（代号qcow2-on-fc），虚拟机内部跑IO，会反映到HOST机，但是虚拟机IO WAIT和IO UTIL都比HOST高，因为有虚拟机一层的损耗，损耗还挺高，例如：

GUEST： %iowait 60% %util 100%

HOST： %iowait 0% %util 80%

### 将HOST机存储设备，映射给GUEST机为IDE磁盘

-drive file=/dev/mapper/36f01faf000dcec22000048a85549e49d,if=none,id=drive-ide1,cache=none,aio=native -device ide-hd,bus=ide.0,unit=1,drive=drive-ide1,id=ide1

GUEST机(redhat 5)内部，lspci显示：

00:01.1 IDE interface [0101]: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II] [8086:7010]

fdisk -l发现有新增的磁盘 /dev/hdb

这种方式，磁盘类型和ID均会发生变化，HOST机里面查看到的是：

host # ls -l /dev/mapper/36f01faf000dcec22000048a85549e49d

/dev/mapper/36f01faf000dcec22000048a85549e49d -> ../dm-0

host # ls -l /dev/disk/by-id/

scsi-36f01faf000dcec22000048a85549e49d -> ../../dm-0

wwn-0x6f01faf000dcec22000048a85549e49d -> ../../sdf

host # lspci

81:00.0 Fibre Channel: Brocade Communications Systems, Inc. 425/825/42B/82B 4Gbps/8Gbps PCIe dual port FC HBA (rev 01)

81:00.1 Fibre Channel: Brocade Communications Systems, Inc. 425/825/42B/82B 4Gbps/8Gbps PCIe dual port FC HBA (rev 01)

host # multipath -ll

36f01faf000dcec22000048a85549e49d dm-0 DELL,MD36xxf

size=500G features='0' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

`- 12:0:1:1 sdf 8:80 active ready running

从GUEST机查看磁盘ID已经变化，且类似不再是SCSI，而是IDE：

guest # ls -l /dev/disk/by-id/

ata-SANGFOR\_HARDDISK\_7961700010988-L -> ../../hdb

### 将HOST机存储设备，映射给GUEST机为VIRTIO-BLK磁盘

-drive file=/dev/mapper/36f01faf000dcec2200004bd4555467b9,if=none,id=drive-virtio3,cache=none,aio=native,cache.direct=on -device virtio-blk-pci,drive=drive-virtio3,id=virtio3,bus=pci.0,addr=0xd

GUEST机(redhat 5)内部，lspci显示：

00:0d.0 SCSI storage controller [0100]: Red Hat, Inc Virtio block device [1af4:1001]

fdisk -l发现有新增的磁盘 /dev/vdc

此种方式（代号map-dev-vdx），磁盘类型和ID均会发生变化，和上面的差别是使用了VIRTIO-BLK替代了IDE。虚拟机内部跑IO，会反映到HOST机，但是虚拟机IO WAIT和IO UTIL都比HOST高，因为有虚拟机一层的损耗，损耗还挺高，例如：

GUEST： %iowait 20% %util 100%

HOST： %iowait 0% %util 75%

### 将HOST机存储设备，透传给GUEST机为SCSI磁盘

-device lsi,id=scsi0,bus=pci.0,addr=0xb -drive file=/dev/sdh,if=none,id=drive-scsi0-0-0-0,format=raw -device scsi-block,bus=scsi0.0,channel=0,scsi-id=0,lun=0,drive=drive-scsi0-0-0-0,id=scsi0-0-0-0

GUSET机(redhat 5)内部，lspci显示：

00:0b.0 SCSI storage controller: LSI Logic / Symbios Logic 53c895a

fdisk -l发现有新增的磁盘 /dev/sda，参数里的/dev/sdh和/dev/disk/by-path/pci-0000:81:00.1-fc-0x2012f01fafdcec22-lun-9等路径等效

# ls -l /dev/disk/by-id/

scsi-36f01faf000dcec22000048b05549e4ea -> ../../sda

### 将HOST机存储设备，透传给GUEST机为VIRTIO-SCSI磁盘

-device virtio-scsi-pci,id=scsi0,bus=pci.0,addr=0xb -drive file=/dev/disk/by-id/wwn-0x6f01faf000dcec22000048b05549e4ea,if=none,id=drive-scsi-dev0,format=raw -device scsi-block,bus=scsi0.0,channel=0,scsi-id=0,lun=9,drive=drive-scsi-dev0,id=scsi-dev0

等价，但是推荐用前者，因为ID是不会变的但PATH是会变的：

-device virtio-scsi-pci,id=scsi0,bus=pci.0,addr=0xb -drive file=/dev/disk/by-path/pci-0000:81:00.1-fc-0x2012f01fafdcec22-lun-9,if=none,id=drive-scsi-dev0,format=raw -device scsi-block,bus=scsi0.0,channel=0,scsi-id=0,lun=9,drive=drive-scsi-dev0,id=scsi-dev0

GUSET机(redhat 5)内部，由于无驱动，lspci显示：

00:0b.0 SCSI storage controller: Red Hat, Inc Device 1004

而且，fdisk -l也看不到任何磁盘。

并且，lsmod | grep virtio 看并没有virtio\_scsi驱动, grep VIRTIO /boot/config-`uname -r` 输出里面找不到CONFIG\_SISC\_VIRTIO

GUSET机(centos 6.3)内部，默认有驱动，lspci显示：

00:0b.0 SCSI storage controller: LSI Logic / Symbios Logic 53c895a

fdisk -l显示/dev/sda

查看HOST机，我们之前映射的是：

host # realpath /dev/disk/by-path/pci-0000:81:00.1-fc-0x2012f01fafdcec22-lun-9

/dev/sdh

host # ll /dev/disk/by-id/

scsi-36f01faf000dcec22000048b05549e4ea -> ../../dm-2

wwn-0x6f01faf000dcec22000048b05549e4ea -> ../../sdh

查看GUEST机，映射后scsi的ID直接从HOST机透进来了：

guest # ll /dev/disk/by-id/

scsi-36f01faf000dcec22000048b05549e4ea -> ../../sda

wwn-0x6f01faf000dcec22000048b05549e4ea -> ../../sda

**在aSV里面修改配置映射匹配：**

vi `find /cfs/ -name 3816770497938.conf`

在类似：

ide0: 36f01faf000dcec22000048b25549e4f6:vm-disk-1.qcow2,cache=directsync,preallocate=off,forecast=disable,cache\_size=256,size=30G

的IDE磁盘选项下面添加：

scsi0: 36f01faf000dcec22000048b25549e4f6:file:/dev/sdf

scsi0: 3600a0980006c8a6d000003a0574733a7:file:/dev/disk/by-id/wwn-0x600a0980006c888a0000047c5746b94c,iothread=on

其中file:/dev/sdf是你要映射的HOST机fc磁盘路径。启动虚拟机后，在GUEST机内部可以lspci或fdisk -l看到新增的磁盘。但是不是透传，而是映射

-drive file=/dev/sdf,if=none,id=drive-virtio0,cache=none,aio=native -device virtio-blk-pci,drive=drive-virtio0,id=virtio0,bus=pci.0,addr=0xa

添加：

scsihw: virtio-scsi-pci

scsi0: 36f01faf000dcec22000048b25549e4f6:file:/dev/sdf

得到：

-device virtio-scsi-pci,id=virtioscsi0,bus=pci.3,addr=0x1 -drive file=/dev/sdf,if=none,id=drive-scsi0,cache=none,aio=native,cache.direct=on -device scsi-block,bus=virtioscsi0.0,channel=0,scsi-id=0,lun=0,drive=drive-scsi0,id=scsi0

和实际想要的匹配：

-device virtio-scsi-pci,id=scsi0,bus=pci.0,addr=0xb -drive file=/dev/sdf,if=none,id=drive-scsi-dev0,format=raw -device scsi-block,bus=scsi0.0,channel=0,scsi-id=0,lun=9,drive=drive-scsi-dev0,id=scsi-dev0

此种方式（代号pt-dev-vscsi），虚拟机内部跑IO，HOST机是看不到IO波动的。

### 将HOST机HBA卡(pci)透传给GUEST机(利用VT-d)

1. 确保BIOS里面VT-d已经开启(一般情况下都是开启的)

2. 修改内核启动项，设置 intel\_iommu=on iommu=pt

这两个参数的含义：

intel\_iommu=on

Enable intel iommu driver.

iommu=pt

This option enables Pass Through in context mapping if

Pass Through is supported in hardware. With this option

DMAR is disabled in kernel and kernel uses swiotlb, but

KVM can still uses VT-d IOTLB hardware.

更多内核参数内容可以[参考](http://www.jinbuguo.com/kernel/boot_parameters.html)。

aSV可以使用如下命令简便设置：

sed -i 's/intel\_iommu=off/intel\_iommu=on/g' /boot/boot/grub/grub.cfg

然后重启系统，重启后可查看dmesg日志，有"Intel-IOMMU: enabled"，aSV的日志显示类似：

# cat /sf/log/blackbox/today/LOG\_dmesg.txt | grep -e DMAR -e IOMMU

ACPI: DMAR 000000006fc3bd48 00162 (v01 ALASKA A M I 00000001 INTL 20091013)

Intel-IOMMU: enabled

dmar: IOMMU 0: reg\_base\_addr fbffc000 ver 1:0 cap d2078c106f0466 ecap f020de

dmar: IOMMU 1: reg\_base\_addr c7ffc000 ver 1:0 cap d2078c106f0466 ecap f020de

IOAPIC id 10 under DRHD base 0xfbffc000 IOMMU 0

IOAPIC id 8 under DRHD base 0xc7ffc000 IOMMU 1

IOAPIC id 9 under DRHD base 0xc7ffc000 IOMMU 1

IOMMU 0 0xfbffc000: using Queued invalidation

IOMMU 1 0xc7ffc000: using Queued invalidation

IOMMU: hardware identity mapping for device 0000:ff:08.0

IOMMU: hardware identity mapping for device 0000:ff:08.2

IOMMU: Setting RMRR:

IOMMU: Setting identity map for device 0000:03:00.0 [0x7261f000 - 0x7a65efff]

IOMMU: Prepare 0-16MiB unity mapping for LPC

3. 确保pci\_stub驱动存在，执行 modprobe pci\_stub

4. 更换驱动

查看当前需要映射的PCI，例如FC的HBA卡：

# lspci -nn | grep HBA

81:00.0 Fibre Channel [0c04]: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PCI Express HBA [1077:2532] (rev 02)

81:00.1 Fibre Channel [0c04]: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PCI Express HBA [1077:2532] (rev 02)

以 81:00.0 为例，我们要将其使用的qla2xxx驱动替换成pci-stub驱动，映射前：

# lspci -nn -v

81:00.0 Fibre Channel [0c04]: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PCI Express HBA [1077:2532] (rev 02)

Subsystem: QLogic Corp. Device [1077:015d]

Physical Slot: 5

Flags: bus master, fast devsel, latency 0, IRQ 33

I/O ports at f100 [size=256]

Memory at fbe84000 (64-bit, non-prefetchable) [size=16K]

Memory at fbd00000 (64-bit, non-prefetchable) [size=1M]

Expansion ROM at fbe40000 [disabled] [size=256K]

Capabilities: [44] Power Management version 3

Capabilities: [4c] Express Endpoint, MSI 00

Capabilities: [88] MSI: Enable- Count=1/32 Maskable- 64bit+

Capabilities: [98] Vital Product Data

Capabilities: [a0] MSI-X: Enable+ Count=32 Masked-

Capabilities: [100] Advanced Error Reporting

Capabilities: [138] Power Budgeting <?>

Kernel driver in use: qla2xxx

81:00.0 的 vendor & device ID 为 1077:2532，上面中括号内。

为pci-stub设置新的vendor和device ID

# echo "1077 2532" > /sys/bus/pci/drivers/pci-stub/new\_id

和原来驱动解绑

# echo "0000:81:00.0" > /sys/bus/pci/devices/0000:81:00.0/driver/unbind

绑定pci-stub驱动

# echo "0000:81:00.0" > /sys/bus/pci/drivers/pci-stub/bind

再次查看 81:00.0 详细信息，发现驱动变成pci-stub了，而且HOST机不能再使用此设备

# lspci -nn -v

81:00.0 Fibre Channel [0c04]: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PCI Express HBA [1077:2532] (rev 02)

Subsystem: QLogic Corp. Device [1077:015d]

Physical Slot: 5

Flags: fast devsel, IRQ 33

I/O ports at f100 [size=256]

Memory at fbe84000 (64-bit, non-prefetchable) [size=16K]

Memory at fbd00000 (64-bit, non-prefetchable) [size=1M]

Expansion ROM at fbe40000 [disabled] [size=256K]

Capabilities: [44] Power Management version 3

Capabilities: [4c] Express Endpoint, MSI 00

Capabilities: [88] MSI: Enable- Count=1/32 Maskable- 64bit+

Capabilities: [98] Vital Product Data

Capabilities: [a0] MSI-X: Enable- Count=32 Masked-

Capabilities: [100] Advanced Error Reporting

Capabilities: [138] Power Budgeting <?>

Kernel driver in use: pci-stub

5. 给虚拟机启动参数添加需要透传的pci设备，例如透传刚才的81:00.0，则加入

-device pci-assign,host=81:00.0

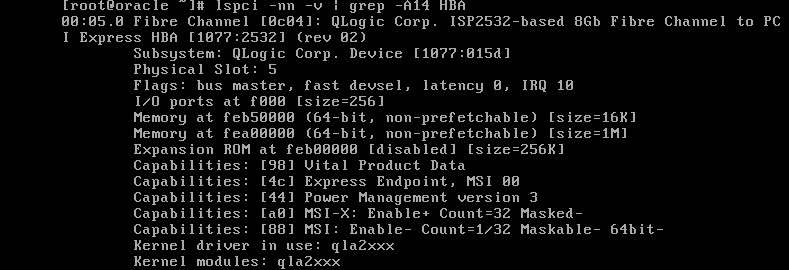
然后启动虚拟机，如果运气好就能启动成功，如果运气不好，例如遇到slot冲突，例如，日志提示：

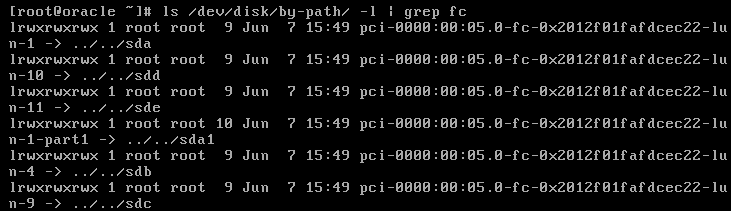
kvm: -device pci-bridge,id=pci.3,chassis\_nr=3,bus=pci.0,addr=0x5: PCI: slot 5 function 0 not available for pci-bridge, in use by kvm-pci-assign

从lspci看 81:00.0 设备的 Physical Slot是5，与之前的命令行冲突了

修改命令，将其他device的addr修改为其他不冲突的值，再次启动即可。

启动虚拟机后，会看到和HOST机一样的HBA卡，而HOST不能再使用此HBA，如果此HBA驱动也存在，即可正常识别，下为GUEST机内识别HBA卡截图：





以上步骤，写成脚本为：pci-redirect.sh

#!/bin/bash

# redirect pci, write by mnstory.net@20160607

main()

{

    if ! grep -P 'intel\_iommu=on|iommu=pt' /proc/cmdline >/dev/null; then

        echo "please switch on iommu in kernel params first"

        echo "for aSV, execute follow commands: "

        echo "    sed -i 's/intel\_iommu=off/intel\_iommu=on/g' /boot/boot/grub/grub.cfg"

        echo "    reboot -f"

        return 1

    fi

    local filter="HBA"

    if [ -n "$1" ]; then

        echo "use '$1' replace filter '$filter'"

        filter="$1"

    else

        echo "use '$filter' as default filter, you can pass your own"

    fi

    modprobe pci\_stub

    local args=""

    for id in $(lspci -nn -D | grep "$filter" | awk '{print $1}'); do

        local vender="$(lspci -s $id -n | awk '{print $3}' | awk -F: '{print $1,$2}')"

        echo $vender > /sys/bus/pci/drivers/pci-stub/new\_id

        echo $id > /sys/bus/pci/devices/$id/driver/unbind

        echo $id > /sys/bus/pci/drivers/pci-stub/bind

        if lspci -s $id -nnvD | grep pci-stub >/dev/null; then

            echo -e "parse "$(lspci -s $id -nnD) "\033[01;32mSUCCESS\033[00m"

            args="$args -device pci-assign,host=$id"

        fi

    done

    if [ -z "$args" ]; then

        echo "no target device found filter by $filter"

        return 1

    fi

    echo "add follow args as your QEMU params(filter by $filter)"

    echo -e "\033[1m   $args\033[00m"

    return 0

}

main "$@"

此种方式（代号pt-pci），虚拟机内部跑IO，HOST机是看不到IO的。

## 性能验证

### 测试环境

**HOST机配置：**

CPU： Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz，2x10物理核，开超频

MEM： 128G

**GUEST机配置：**

CPU： 虚拟Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz，1x20虚拟核

MEM： 64G

**存储：**

81:00.0 Fibre Channel [0c04]: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PCI Express HBA [1077:2532] (rev 02)

36f01faf000dcec2200004bd4555467b9 dm-1 DELL,MD36xxf

size=1.3T features='0' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

`- 2:0:0:4 sdb 8:16 active ready running

**测试脚本fio-stress.sh：**

#!/bin/bash

filename=/dev/disk/by-id/wwn-0x6f01faf000dcec2200004bd4555467b9

runtime=1

group="default"

uniform()

{

    local val="$1"

    local extra=""

    if [ "${val%KB\*}" != "$val" ]; then

        echo "scale=2; ${val%KB\*}/1024" | bc -l

        return 0

    fi

    if [ "${val%MB\*}" != "$val" ]; then

        echo ${val%MB\*}

        return 0

    fi

    if [ "${val%GB\*}" != "$val" ]; then

        echo "scale=2; ${val%GB\*}\*1024" | bc -l

        return 0

    fi

    if [ "${val%B\*}" != "$val" ]; then

        echo "scale=2; ${val%B\*}/1024/1024" | bc -l

        return 0

    fi

    echo $val

}

parseRW()

{

    local action="$1"

    shift

    local blk=$(echo "$@" | grep -P -A1 "$action\s\*:")

    local io="0"

    local bw="0"

    local iops="0"

    local latmin="0"

    local latmax="0"

    local latavg="0"

    if [ -n "$blk" ]; then

        io=$(echo "$blk" | sed -n  's/.\*io=\([^,]\*\).\*/\1/p')

        io=$(uniform $io)

        bw=$(echo "$blk" | sed -n  's/.\*bw=\([^,]\*\).\*/\1/p')

        bw=$(uniform $bw)

        iops=$(echo "$blk" | sed -n  's/.\*iops=\([^,]\*\).\*/\1/p')

        latmin=$(echo "$blk" | sed -n  's/.\*min=\([^,]\*\).\*/\1/p')

        latmax=$(echo "$blk" | sed -n  's/.\*max=\([^,]\*\).\*/\1/p')

        latavg=$(echo "$blk" | sed -n  's/.\*avg=\([^,]\*\).\*/\1/p')

    fi

    echo "$io $bw $iops $latmin $latmax $latavg"

}

parseCPU()

{

    local blk=$(echo "$@" | grep -P "cpu\s\*:")

    local usr="0"

    local sys="0"

    local ctx="0"

    if [ -n "$blk" ]; then

        usr=$(echo "$blk" | sed -n  's/.\*usr=\([^,]\*\).\*/\1/p')

        sys=$(echo "$blk" | sed -n  's/.\*sys=\([^,]\*\).\*/\1/p')

        ctx=$(echo "$blk" | sed -n  's/.\*ctx=\([^,]\*\).\*/\1/p')

    fi

    echo "${usr%\%\*} ${sys%\%\*} $ctx"

}

run()

{

    local ioengine="$1"

    local direct="$2"

    local rw="$3"

    local bs="$4"

    local iodepth="$5"

    local numjobs="$6"

    local name="$ioengine-$direct-$rw-$bs-$iodepth-$numjobs"

    local cmd="fio -filename=$filename -rw=$rw -bs=$bs -iodepth=$iodepth -direct=$direct -time\_based -thread -ioengine=$ioengine -numjobs=$numjobs -runtime=$runtime -group\_reporting -name=name"

    echo -e -n "$group $name $@ \t"

    local res="$($cmd)"

    # usr sys ctx      r-io r-bw r-iops r-latmin r-latmax r-latavg     w-io w-bw w-iops w-latmin w-latmax w-latavg

    echo -e "$(parseCPU "$res") \t$(parseRW 'read' "$res") \t\t$(parseRW 'write' "$res")"

}

v1()

{

    for ioengine in "sync" "libaio"; do

        for direct in "0" "1"; do

            for rw in "read" "randread" "write" "randwrite" "randrw"; do

                for bs in "4k" "8k" "32k" "2048k"; do

                    for iodepth in 1 32 64 128; do

                        for numjobs in 1 8 32 64; do

                            run "$ioengine" "$direct" "$rw" "$bs" "$iodepth" "$numjobs"

                        done

                    done

                done

            done

        done

    done

}

v2()

{

    for ioengine in "sync" "libaio"; do

        for direct in "1"; do

            for rw in "read" "randread" "write" "randwrite"; do

                for bs in "4k" "2048k"; do

                    for iodepth in 128; do

                        for numjobs in 8 32; do

                            run "$ioengine" "$direct" "$rw" "$bs" "$iodepth" "$numjobs"

                        done

                    done

                done

            done

        done

    done

}

main()

{

    if [ -n "$1" ]; then

        group="$1"

    fi

    if [ -n "$2" ]; then

        filename="$2"

    fi

    if [ -n "$3" ]; then

        runtime="$3"

    fi

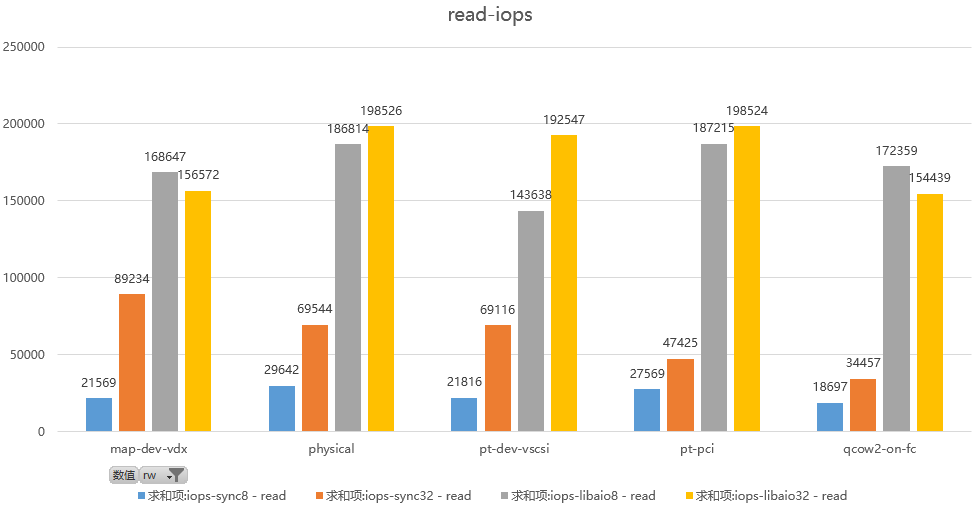
    echo -e "group name ioengine direct rw bs iodepth numjobs \tusr sys ctx \t\tr-io r-bw r-iops r-latmin r-latmax r-latavg \tw-io w-bw w-iops w-latmin w-latmax w-latavg"

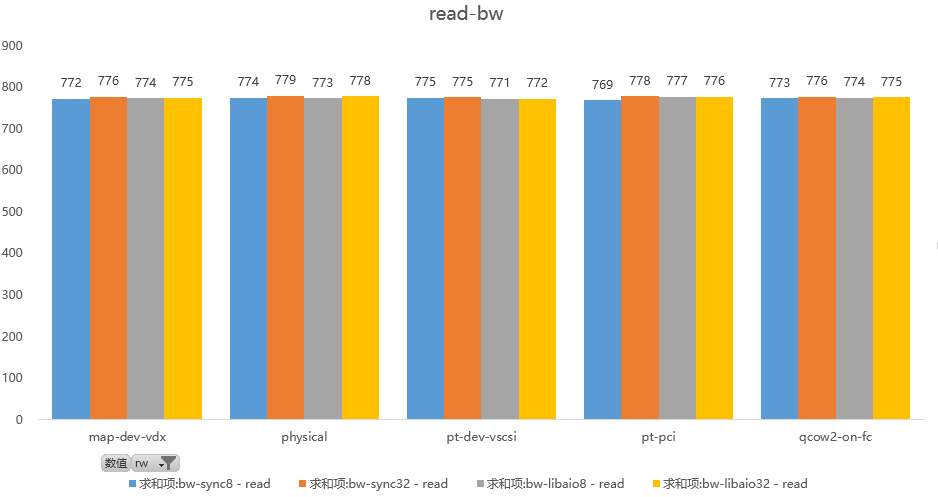
    v2

}

main "$@"

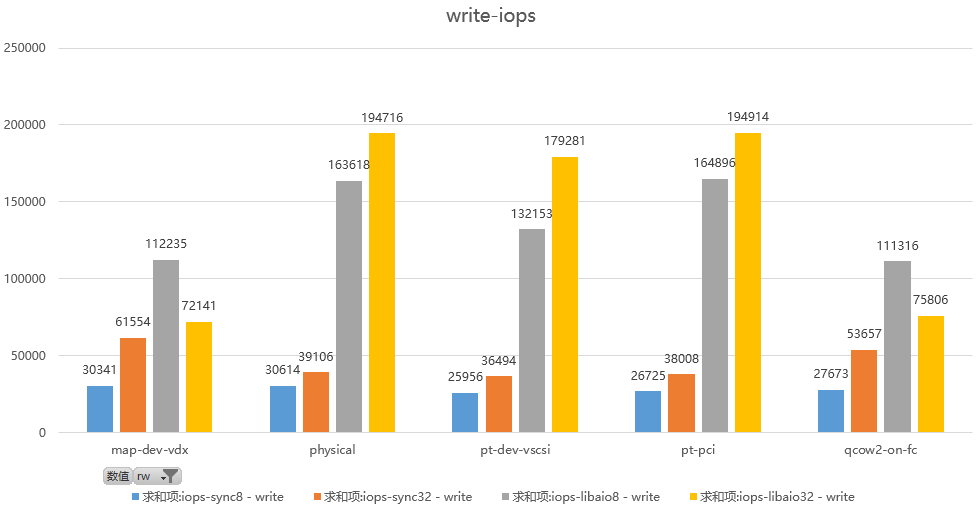
### read

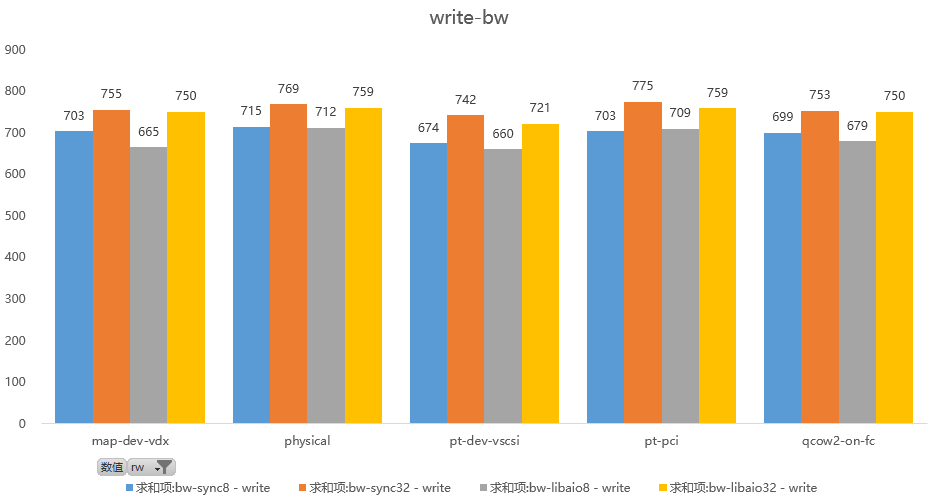




同步读，bw性能都是一致的,IOPS性能，以最高性能libaio为基准数据，其中map-dev-vdx和qcow2-on-fc性能相当，比physical差15%左右，pt-dev-vscsi比physical差13%，pt-pci和physical相当，符合理论。

### write

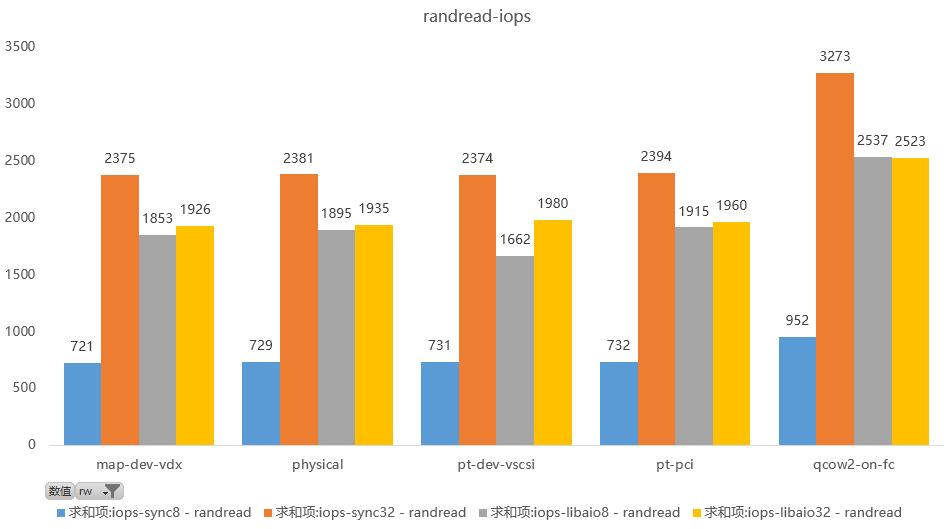


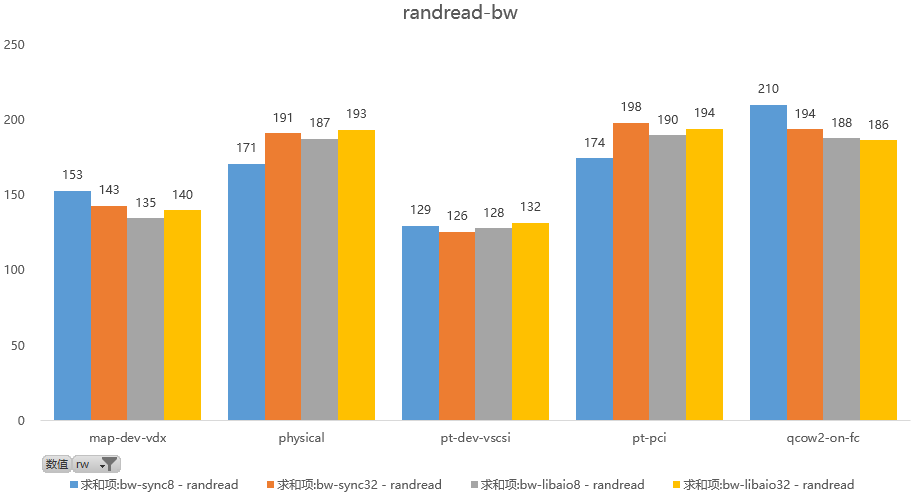


write，顺序写的情况libaio能充分发挥异步多深度(128 iodepth)批量提交的优势，比sync方式效果高很多。

同步写，bw性能都是一致的,IOPS性能，以最高性能libaio为基准数据，其中map-dev-vdx和qcow2-on-fc性能相当，比physical差48%左右，pt-dev-vscsi比physical差13%，pt-pci和physical相当，符合理论。

### randread



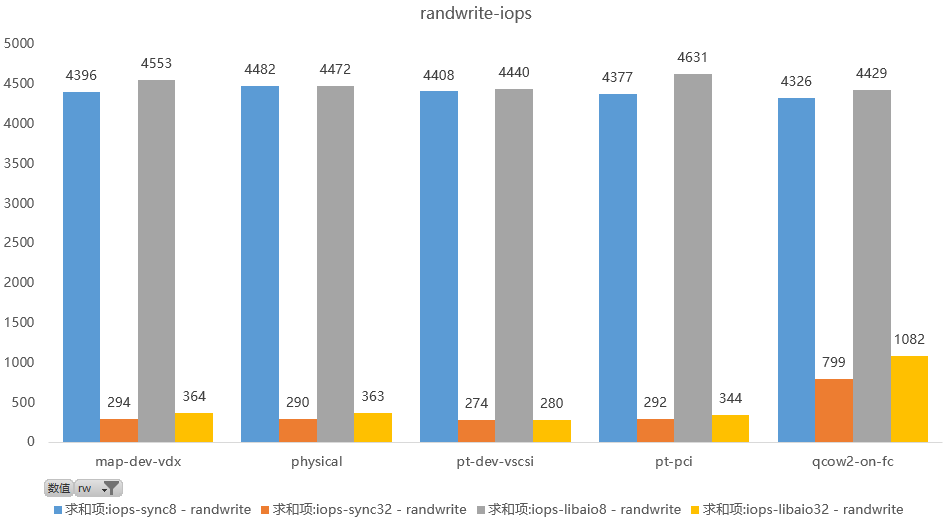


多jobs(32 vs. 8)在randread的情况下比randwrite好。

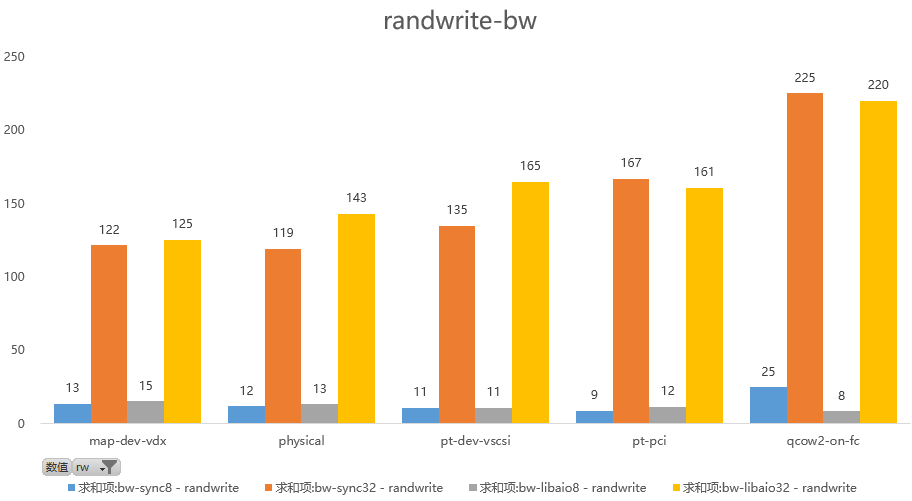
随机读的bw，不论jobs多说还是同步异步，效果都差不多，看来压力已经足够。

随机读， IOPS性能，以最高性能sync为基准数据，除qcow2-on-fc性能最好，超过physical 36%外，其他几种方式与physical齐平，libaio情况，pt-dev-vscsi比physical差5%。

### randwrite



1. randwrite 应该看8job的数据，因为8job数据好于32job太多，可以看出，当写线程增加，randwrite性能急剧下降。几种方式的性能可以认为无差别。
2. randwrite，存储寻道成为瓶颈，libaio和sync并无太明显差异。



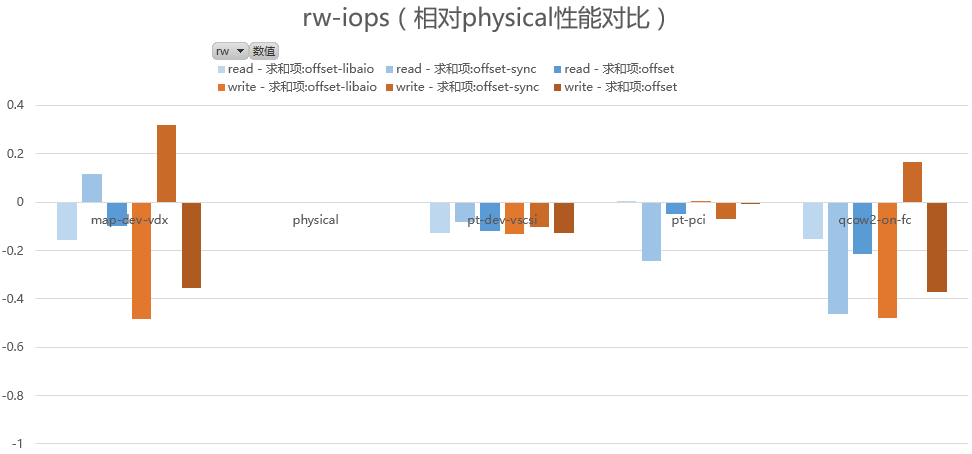
随机写的两幅图，非常有意思，因为随着jobs增加，性能变化却巨大，randwrite-iops图，bs为4K的情况下计算IOPS，Jobs为8的性能远远高于jobs为32的，说明写IO的jobs越多，会导致写排队和竞争增加，性能大幅度下降。

但是从randwrite-bw图看，在bs为2048K时好像结论相反，其实结论也能正确理解，当bs增大，iops会相应降低，写入寻道和写入竞争相对会降低，所以bw符合常规，还是jobs越大，bw越大。

两幅图综合说明，IO性能，真的需要找到一个平衡点，压力太小，效果不好，压力太大，效果急剧下降。

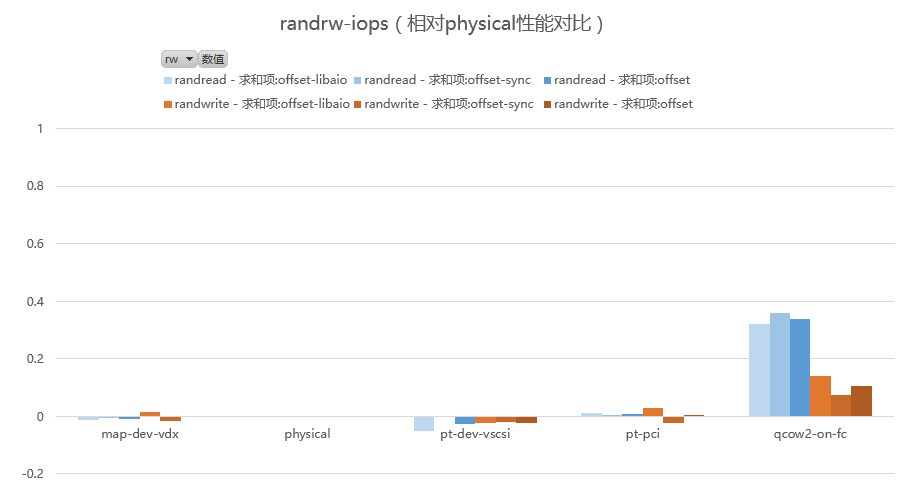
随机读， IOPS性能，以最高性能libaio为基准数据，除qcow2-on-fc性能最好，超过physical 14%外，其他几种方式与physical齐平，相差不过2%。

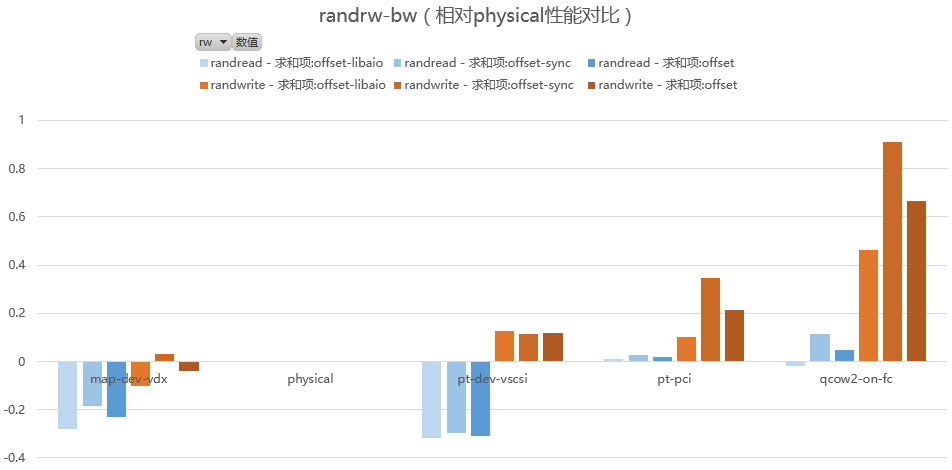
### 总结





如果排除开挂的sync write（这是由于QEMU IO路径实现影响），可以看出，顺序IOPS从高到底为 physical, pt-pci, pt-dev-vscisi,qcow2-on-fc,map-dev-vdx；bw可以认为相同。





随机IO，排除qcow2-on-fc由于缓存等因素影响，导致随机IO性能颇高，其他的，性能差不多。bw，以pt-pci最接近physical，qcow2-on-fc随机IO吞吐最高。

虚拟化场景，QCOW2是个性能波动较大的实现，好的时候太好，差的时候太差；如果追求稳定IO，建议选择pt-pci，如果追求稳定和兼容的平衡，建议使用pt-dev-vscisi方式。

2016/6/17