昵称：SammyLiu
园龄：2年6个月
荣誉：推荐博客
粉丝：470
关注：30
+加关注

KVM 介绍（3）：I/O 全虚拟化和准虚拟化 [KVM I/O QEMU Full-Virtualizaiton Para-virtualization]

学习 KVM 的系列文章：

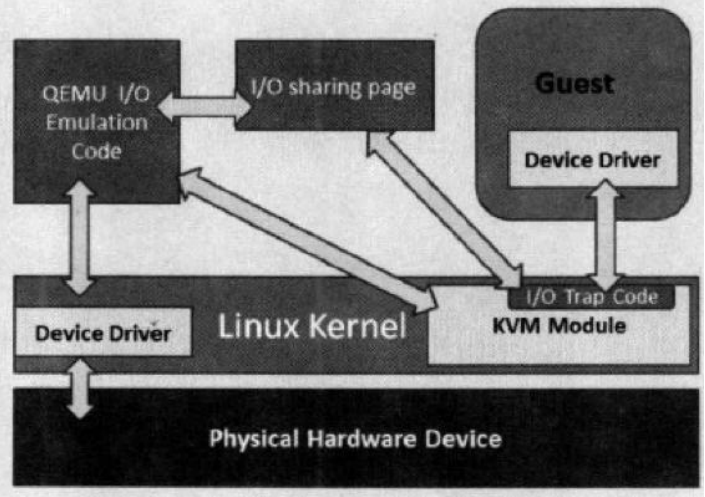
- （1）介绍和安装
- （2）CPU 和 内存虚拟化
- （3）I/O QEMU 全虚拟化和准虚拟化（Para-virtulizaiton）
- （4）I/O PCI/PCIe设备直接分配和 SR-IOV
- （5）libvirt 介绍
- （6）Nova 通过 libvirt 管理 QEMU/KVM 虚机
- （7）快照（snapshot）
- （8）迁移（migration）

在 QEMU/KVM 中，客户机可以使用的设备大致可分为三类：
1. 模拟设备：完全由 QEMU 纯软件模拟的设备。
2. Virtio 设备：实现 VIRTIO API 的半虚拟化设备。
3. PCI 设备直接分配（PCI device assignment）。

1. 全虚拟化 I/O 设备

KVM 在 IO 虚拟化方面，传统或者默认的方式是使用 QEMU 纯软件的方式来模拟 I/O 设备，包括键盘、鼠标、显示器，硬盘 和 网卡 等。模拟设备可能会使用物理的设备，或者使用纯软件来模拟。模拟设备只存在于软件中。

1.1 原理



过程：

1. 客户机的设备驱动程序发起 I/O 请求操作请求
2. KVM 模块中的 I/O 操作捕获代码拦截这次 I/O 请求
3. 经过处理后将本次 I/O 请求的信息放到 I/O 共享页（sharing page），并通知用户空间的 QEMU 程序。
4. QEMU 程序获得 I/O 操作的具体信息之后，交由硬件模拟代码来模拟出本次 I/O 操作。
5. 完成之后，QEMU 将结果放回 I/O 共享页，并通知 KMV 模块中的 I/O 操作捕获代码。
6. KVM 模块的捕获代码读取 I/O 共享页中的操作结果，并把结果放回客户机。

注意：当客户机通过DMA（Direct Memory Access）访问大块I/O时，QEMU 模拟程序将不会把结果放进共享页中，而是通过内存映射的方式将结果直接写到客户机的内存中共，然后通知KVM模块告诉客户机DMA操作已经完成。

这种方式的优点是可以模拟出各种各样的硬件设备；其缺点是每次 I/O 操作的路径比较长，需要多次上下文切换，也需要多次数据复制，所以性能较差。

1.2 QEMU 模拟网卡的实现

Qemu 纯软件的方式来模拟I/O设备，其中包括经常使用的网卡设备。Guest OS启动命令中没有传入的网络配置时，QEMU默认分配

rtl8139 类型的虚拟网卡类型，使用的是默认用户配置模式，这时候由于没有具体的网络模式的配置，Guest的网络功能是有限的。
全虚拟化情况下，KVM虚机可以选择的网络模式包括：

1. 默认用户模式（User）；
2. 基于网桥(Bridge)的模式；
3. 基于NAT(Network Address Translation)的模式；

2017年5月 (1)
2017年3月 (1)
2017年1月 (1)
2016年10月 (7)
2016年9月 (5)
2016年8月 (4)
2016年7月 (1)
2016年6月 (5)
2016年5月 (1)
2016年4月 (1)
2016年3月 (9)
2016年2月 (4)
2016年1月 (2)
2015年12月 (7)
2015年11月 (7)
2015年10月 (4)
2015年9月 (4)
2015年8月 (5)
2015年7月 (9)
2015年6月 (10)
2015年5月 (3)
2015年4月 (11)
2015年3月 (2)
2015年2月 (6)
2015年1月 (5)
2014年12月 (6)

文章分类(21)

Ceph(1)
GlusterFS
Web 服务器(2)
操作系统(1)
存储
大数据(2)
分布式系统
服务器(1)
网络(11)
虚拟化(3)
云

文章档案(42)

2016年10月 (2)
2016年9月 (1)
2016年6月 (1)
2016年5月 (3)
2015年12月 (4)
2015年10月 (5)
2015年9月 (2)
2015年6月 (1)
2015年4月 (23)

积分与排名

积分 - 286831
排名 - 535

最新评论

1. Re:Neutron 理解 (1): Neutron 所实现的虚拟化网络 [How Netruon Virtualizes Network]
eth1 - 公共网络（untagged），管理网络（tag=102），存储网络（tag=103）不好意思，大家共用同一个eth1端口的时候，请问这里交换机端口是配置为tagged还是

untagged.....

--xianke9

2. Re:理解Docker（5）：Docker 网络
1.12版本上网络的表现如何？

--幽灵狼

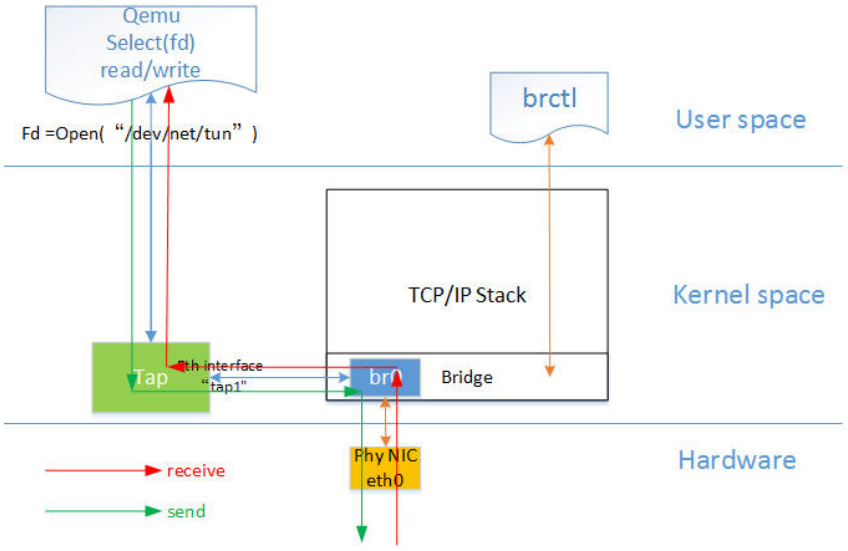
3. Re:理解Docker（5）：Docker 网络
我想请问一下运行docker quickstart terminal时一直卡在"waiting for an IP"应该如何解决呢？希望楼主能解答一下。

--silentbell

分别使用的 qemu-kvm 参数为：

- **-net user[,vlan=n]：**使用用户模式网络堆栈,这样就不需要管理员权限来运行.如果没有指 定-net选项,这将是默认的情况.**-net tap[,vlan=n][,fd=h]**
- **-net nic[,vlan=n][,macaddr=addr]：**创建一个新的网卡并与VLAN n(在默认的情况下n=0)进行连接。作为可选项的项目,MAC地址可以进行改变.如果 没有指定-net选项,则会创建一个单一的NIC.
- **-net tap[,vlan=n][,fd=h][,ifname=name][,script=file]：**将TAP网络接口 name 与 VLAN n 进行连接,并使用网络配置脚本文件进行 配置。默认的网络配置脚本为/etc/qemu-ifup。如果没有指定name,OS 将会自动指定一个。fd=h可以用来指定一个已经打开的TAP主机接口的句柄。

网桥模式是目前比较简单，也是用的比较多的模式，下图是网桥模式下的 VM的收发包的流程。



如图中所示，红色箭头表示数据报文的入方向，步骤：

1. 网络数据从 Host 上的物理网卡接收，到达网桥；
2. 由于 eth0 与 tap1 均加入网桥中，根据二层转发原则，br0 将数据从 tap1 口转发出去，即数据由 Tap设备接收；
3. Tap 设备通知对应的 fd 数据可读；
4. fd 的读动作通过 tap 设备的字符设备驱动将数据拷贝到用户空间，完成数据报文的前端接收。

（引用自 http://luoye.me/2014/07/17/netdev-virtual-1/ ）

1.3 RedHat Linux 6 中提供的模拟设备

- 模拟显卡：提供2块模拟显卡。
- 系统组件：
  - ntel i440FX host PCI bridge
  - PIIX3 PCI to ISA bridge
  - PS/2 mouse and keyboard
  - EvTouch USB Graphics Tablet
  - PCI UHCI USB controller and a virtualized USB hub
  - Emulated serial ports
  - EHCI controller, virtualized USB storage and a USB mouse
- 模拟的声卡：intel-hda
- 模拟网卡：e1000，模拟 Intel E1000 网卡；rtl8139，模拟 RealTeck 8139 网卡。
- 模拟存储卡：两块模拟 PCI IDE 接口卡。KVM 限制每个虚拟机最多只能有4块虚拟存储卡。还有模拟软驱。

注意：RedHat Linux KVM 不支持 SCSI 模拟。

在不显式指定使用其它类型设备的情况下，KVM 虚机将使用这些默认的虚拟设备。比如上面描述的默认情况下 KVM 虚机默认使用 rtl8139网卡。比如，在 RedHat Linxu 6.5 主机上启动KVM RedHat Linux 6.4 虚机后，登录虚机，查看 pci 设备，可以看到这些模拟设备：

```
[root@rh64-1 ~]# lspci
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: Cirrus Logic GD 5446
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controll
er (rev 03)
```

当使用 “-net nic,model=e1000” 指定网卡model 为 e1000 时，

```
[root@rh64-1 ~]# ethtool -i eth1
driver: e1000
version: 7.3.21-k8-NAPI
firmware-version:
bus-info: 0000:00:03.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: no
```

1.4 qemu-kvm 关于磁盘设备和网络的主要选项

类型	选项

磁盘设备（软盘、硬盘、CDROM等）	**-drive** option[,option[,option[,...]]]：定义一个硬盘设备；可用子选项有很多。 file=/path/to/somefile：硬件映像文件路径； if=interface：指定硬盘设备所连接的接口类型，即控制器类型，如ide、scsi、sd、mtd、floppy、pflash及virtio等； index=index：设定同一种控制器类型中不同设备的索引号，即标识号； media=media：定义介质类型为硬盘(disk)还是光盘(cdrom)； format=format：指定映像文件的格式，具体格式可参见qemu-img命令； **-boot** [order=drives][,once=drives][,menu=on|off]：定义启动设备的引导次序，每种设备使用一个字符表示；不同的架构所支持的设备及其表示字符不尽相同，在x86 PC架构上，a、b表示软驱、c表示第一块硬盘，d表示第一个光驱设备，n-p表示网络适配器；默认为硬盘设备(-boot order=dc,once=d)
网络	**-net nic**[,vlan=n][,macaddr=mac][,model=type][,name=name][,addr=addr][,vectors=v]：创建一个新的网卡设备并连接至vlan n中；PC架构上默认的NIC为e1000，macaddr用于为其指定MAC地址，name用于指定一个在监控时显示的网上设备名称；emu可以模拟多个类型的网卡设备；可以使用“qemu-kvm -net nic,model=?”来获取当前平台支持的类型； **-net tap**[,vlan=n][,name=name][,fd=h][,ifname=name][,script=file][,downscript=dfile]：通过物理机的TAP网络接口连接至vlan n中，使用script=file指定的脚本(默认为/etc/qemu-ifup)来配置当前网络接口，并使用downscript=file指定的脚本(默认为/etc/qemu-ifdown)来撤消接口配置；使用script=no和downscript=no可分别用来禁止执行脚本； **-net user**[,option][,option][,...]：在用户模式配置网络栈，其不依赖于管理权限；有效选项有： vlan=n：连接至vlan n，默认n=0； name=name：指定接口的显示名称，常用于监控模式中； net=addr[/mask]：设定GuestOS可见的IP网络，掩码可选，默认为10.0.2.0/8； host=addr：指定GuestOS中看到的物理机的IP地址，默认为指定网络中的第二个，即x.x.x.2； dhcpstart=addr：指定DHCP服务地址池中16个地址的起始IP，默认为第16个至第31个，即x.x.x.16-x.x.x.31； dns=addr：指定GuestOS可见的dns服务器地址；默认为GuestOS网络中的第三个地址，即x.x.x.3； tftp=dir：激活内置的tftp服务器，并使用指定的dir作为tftp服务器的默认根目录； bootfile=file：BOOTP文件名称，用于实现网络引导GuestOS；如：qemu -hda linux.img -boot n -net user,tftp=/tftpserver/pub,bootfile=/pxelinux.0

对于网卡来说，你可以使用 modle 参数指定虚拟网络的类型。 RedHat Linux 6 所支持的虚拟网络类型有：

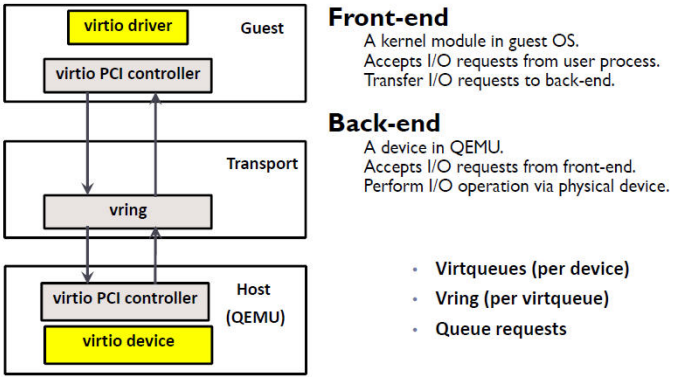
```
[root@rh65 isoimages]# kvm -net nic,model=?
qemu: Supported NIC models: ne2k_pci,i82551,i82557b,i82559er,rtl8139,e1000,pcnet,virtio
```

2. 准虚拟化（Para-virtualizaiton）I/O 驱动 virtio

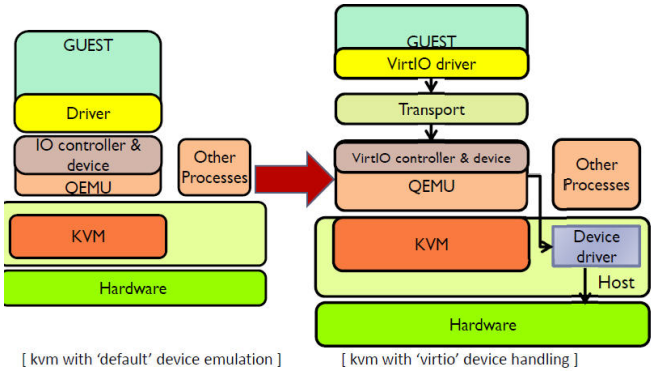
在 KVM 中可以使用准虚拟化驱动来提供客户机的I/O 性能。目前 KVM 采用的的是 virtio 这个 Linux 上的设备驱动标准框架，它提供了一种 Host 与 Guest 交互的 IO 框架。

2.1 virtio 的架构

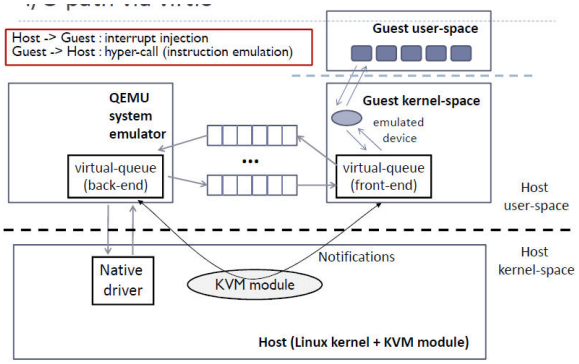
KVM/QEMU 的 vitio 实现采用在 Guest OS 内核中安装前端驱动（Front-end driver）和在 QEMU 中实现后端驱动（Back-end）的方式。前后端驱动通过 vring 直接通信，这就绕过了经过 KVM 内核模块的过程，达到提高 I/O 性能的目的。



纯软件模拟的设备和 Virtio 设备的区别：virtio 省去了纯模拟模式下的异常捕获环节，Guest OS 可以和 QEMU 的 I/O 模块直接通信。



使用 Virtio 的完整虚机 I/O流程：

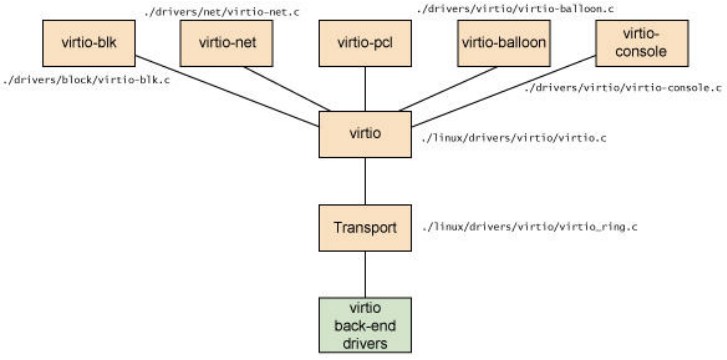


Host 数据发到 Guest：
1. KVM 通过中断的方式通知 QEMU 去获取数据，放到 virtio queue 中
2. KVM 再通知 Guest 去 virtio queue 中取数据。

2.2 Virtio 在 Linux 中的实现

Virtio 是在半虚拟化管理程序中的一组通用模拟设备的抽象。这种设计允许管理程序通过一个应用编程接口（API）对外提供一组通用模拟设备。通过使用半虚拟化管理程序，客户机实现一套通用的接口，来配合后面的一套后端设备模拟。后端驱动不必是通用的，只要它们实现了前端所需的行为。因此，Virtio 是一个在 Hypervisor 之上的抽象API接口，让客户机知道自己运行在虚拟化环境中，进而根据 virtio 标准与 Hypervisor 协作，从而客户机达到更好的性能。

- 前端驱动：客户机中安装的驱动程序模块
- 后端驱动：在 QEMU 中实现，调用主机上的物理设备，或者完全由软件实现。
- virtio 层：虚拟队列接口，从概念上连接前端驱动和后端驱动。驱动可以根据需要使用不同数目的队列。比如 virtio-net 使用两个队列，virtio-block只使用一个队列。该队列是虚拟的，实际上是使用 virtio-ring 来实现的。
- virtio-ring：实现虚拟队列的环形缓冲区



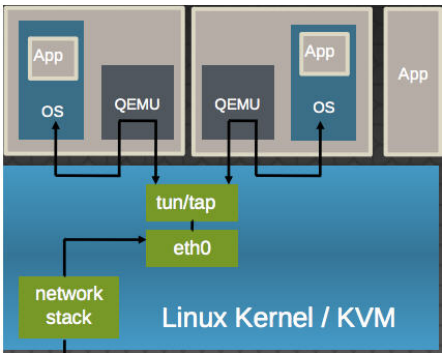
Linux 内核中实现的五个前端驱动程序：

- 块设备（如磁盘）
- 网络设备
- PCI 设备
- 气球驱动程序（动态管理客户机内存使用情况）
- 控制台驱动程序

Guest OS 中，在不使用 virtio 设备的时候，这些驱动不会被加载。只有在使用某个 virtio 设备的时候，对应的驱动才会被加载。每个前端驱动器具有在管理程序中的相应的后端的驱动程序。

以 virtio-net 为例，解释其原理：

（1）virtio-net 的原理：

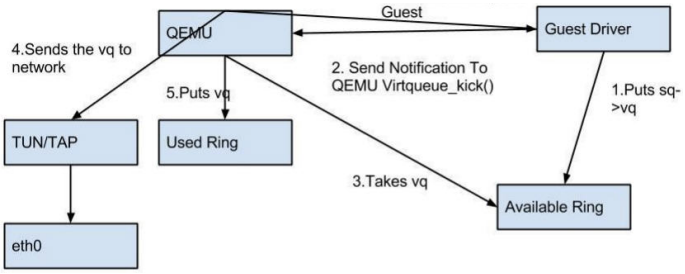


它使得：

1. 多个虚机共享主机网卡 eth0
2. QEMU 使用标准的 tun/tap 将虚机的网络桥接到主机网卡上
3. 每个虚机看起来有一个直接连接到主机PCI总线上的私有 virtio 网络设备
4. 需要在虚机里面安装 virtio驱动

（2）virtio-net 的流程：

6. Sends Interrupt to



总结 Virtio 的优缺点：

- 优点：更高的IO性能，几乎可以和原生系统差不多。
- 缺点：客户机必须安装特定的 virtio 驱动。一些老的 Linux 还没有驱动支持，一些 Windows 需要安装特定的驱动。不过，较新的和主流的 OS都有驱动可以下载了。Linux 2.6.24+ 都默认支持 virtio。可以使用 lsmod | grep virtio 查看是否已经加载。

2.3 使用 virtio 设备（以 virtio-net 为例）

使用 virtio 类型的设备比较简单。较新的 Linux 版本上都已经安装好了 virtio 驱动，而 Windows 的驱动需要自己下载安装。

（1）检查主机上是否支持 virtio 类型的网卡设备

```
[root@rh65 isoimages]# kvm -net nic,model=?
qemu: Supported NIC models: ne2k_pci,i82551,i82557b,i82559er,rtl8139,e1000,pcnet,virtio
```

（2）指定网卡设备model 为 virtio，启动虚机

```
[root@rh65 domains]# kvm -smp 2 -cpu host -m 2048 -drive file=./rh64-6.qcow2,if=ide,media=disk,format=qcow2 -
boot c -name rh64-6  -net nic,model=virtio
```

（3）通过 vncviewer 登录虚机，能看到被加载了的 virtio-net 需要的内核模块

```
[root@rh64-1 ~]# lsmod | grep virtio
virtio_net             16824  0
virtio_pci              6985  0
virtio_ring             8301  2 virtio_net,virtio_pci
virtio                  4977  2 virtio_net,virtio_pci
```

（4）查看 pci 设备

```
[root@rh64-1 ~]# lspci
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: Cirrus Logic GD 5446
00:03.0 Ethernet controller: Red Hat, Inc Virtio network device
```

```
[root@rh64-1 ~]# lspci -vv -s 00:03.0
00:03.0 Ethernet controller: Red Hat, Inc Virtio network device
        Subsystem: Red Hat, Inc Device 0001
        Physical Slot: 3
        Control: I/O+ Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Step
ping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort-
<MAbort- >SERR- <PERR- INTx-
        Interrupt: pin A routed to IRQ 11
        Region 0: I/O ports at c020 [size=32]
        Region 1: Memory at f2020000 (32-bit, non-prefetchable) [size=4K]
        Expansion ROM at f2030000 [disabled] [size=64K]
        Capabilities: [40] MSI-X: Enable+ Count=3 Masked-
                Vector table: BAR=1 offset=00000000
                PBA: BAR=1 offset=00000800
        Kernel driver in use: virtio-pci
        Kernel modules: virtio_pci
```

```
[root@rh64-1 ~]# ethtool -i eth1
driver: virtio_net
version:
firmware-version:
bus-info: virtio0
supports-statistics: no
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

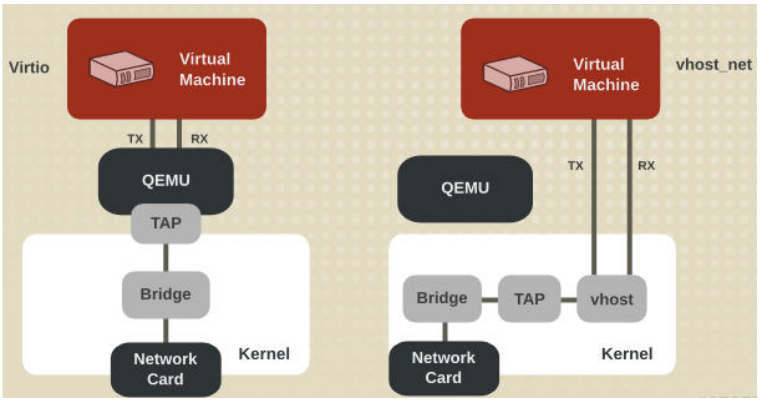
其它 virtio 类型的设备的使用方式类似 virtio-net。

2.4 vhost-net （kernel-level virtio server）

前面提到 virtio 在宿主机中的后端处理程序（backend）一般是由用户空间的QEMU提供的，然而如果对于网络 I/O 请求的后端处理能够在在内核空间来完成，则效率会更高，会提高网络吞吐量和减少网络延迟。在比较新的内核中有一个叫做“vhost-net” 的驱动模块，它是作为一个内核级别的后端处理程序，将virtio-net的后端处理任务放到内核空间中执行，减少内核空间到用户空间的切换，从而提高效率。

根据 KVM 官网的这篇文章，vhost-net 能提供更低的延迟（latency）（比 e1000 虚拟网卡低 10%），和更高的吞吐量（throughput）（8倍于普通 virtio，大概 7~8 Gigabits/sec）。

vhost-net 与 virtio-net 的比较：



vhost-net 的要求：

- qemu-kvm-0.13.0 或者以上
- 主机内核中设置 CONFIG_VHOST_NET=y 和在虚机操作系统内核中设置 CONFIG_PCI_MSI=y（Red Hat Enterprise Linux 6.1 开始支持该特性）
- 在客户机内使用 virtion-net 前段驱动
- 在主机内使用网桥模式，并且启动 vhost_net

qemu-kvm 命令的 -net tap 有几个选项和 vhost-net 相关的: -net tap,[,vnet_hdr=on|off][,vhost=on|off][,vhostfd=*h*][,vhostforce=on|off]

- **vnet_hdr =on|off：**设置是否打开TAP设备的“IFF_VNET_HDR”标识。“vnet_hdr=off”表示关闭这个标识；“vnet_hdr=on”则强制开启这个标识，如果没有这个标识的支持，则会触发错误。IFF_VNET_HDR是tun/tap的一个标识，打开它则允许发送或接受大数据包时仅仅做部分的校验和检查。打开这个标识，可以提高virtio_net驱动的吞吐量。
- **vhost=on|off：**设置是否开启vhost-net这个内核空间的后端处理驱动，它只对使用MIS-X中断方式的virtio客户机有效。
- **vhostforce=on|off：**设置是否强制使用 vhost 作为非MSI-X中断方式的Virtio客户机的后端处理程序。
- **vhostfs=h：**设置为去连接一个已经打开的vhost网络设备。

vhost-net 的使用实例：

（1）确保主机上 vhost-net 内核模块被加载了

```
[root@rh65 domains]# modprobe vhost-net
[root@rh65 domains]# lsmod | grep vhost
vhost_net              30520  0
macvtap                10039  1 vhost_net
tun                    17095  2 vhost_net
```

（2）启动一个虚拟机，在客户机中使用 -net 定义一个 virtio-net 网卡，在主机端使用 -netdev 启动 vhost

```
[root@rh65 domains]# kvm -smp 2 -m 2048 -drive file=../domains/rh64-9.qcow2,if=i
de,media=disk,format=qcow2 -boot c -name rh64-7 -net nic,model=virtio -netdev ta
p,vhost=on,id=guest0
Warning: vlan 0 is not connected to host network
Warning: netdev guest0 has no peer
VNC server running on `::1:5900'
```

（3）在虚拟机端，看到 virtio 网卡使用的 TAP 设备为 tap0。

```
monitor console
QEMU 0.12.1 monitor - type 'help' for more information
(qemu) info network
VLAN 0 devices:
  virtio-net-pci.0: model=virtio-net-pci,macaddr=52:54:00:12:34:56
Devices not on any VLAN:
  guest0: ifname=tap0,script=/etc/qemu-ifup,downscript=/etc/qemu-ifdown
```

（4）在宿主机中看 vhost-net 被加载和使用了，以及 Linux 桥 br0，它连接物理网卡 eth1 和 客户机使用的 TAP 设备 tap0

```
[root@rh65 ~]# lsmod | grep vhost
vhost_net              30520  1
macvtap                10039  1 vhost_net
tun                    17095  4 vhost_net
[root@rh65 ~]# rmmod vhost_net
ERROR: Module vhost_net is in use
[root@rh65 ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.3440b5d905ee       no              eth1
                                                        tap0
virbr0          8000.5254006e2086       yes             virbr0-nic
```

一般来说，使用 vhost-net 作为后端处理驱动可以提高网络的性能。不过，对于一些网络负载类型使用 vhost-net 作为后端，却可能使其性能不升反降。特别是从宿主机到其中的客户机之间的UDP流量，如果客户机处理接受数据的速度比宿主机发送的速度要慢，这时就容易出现性能下降。在这种情况下，使用vhost-net将会是UDP socket的接受缓冲区更快地溢出，从而导致更多的数据包丢失。故这种情况下，不使用vhost-net，让传输速度稍微慢一点，反而会提高整体的性能。

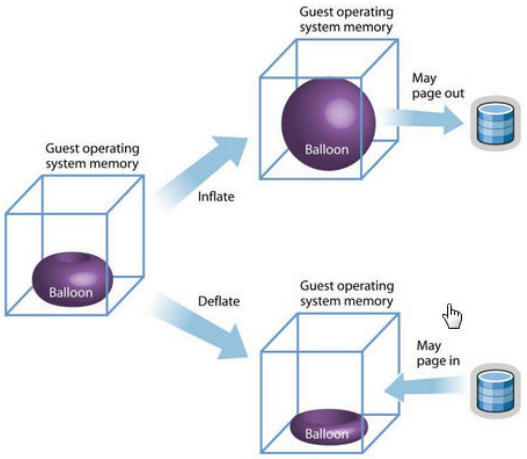
使用 qemu-kvm 命令行，加上“vhost=off”（或没有vhost选项）就会不使用vhost-net，而在使用libvirt时，需要对客户机的配置的XML文件中的网络配置部分进行如下的配置，指定后端驱动的名称为“qemu”（而不是“vhost”）。

```
<interface type="network">
...
  <model type="virtio"/>
  <driver name="qemu"/>
...
</interface>
```

2.6 virtio-balloon

另一个比较特殊的 virtio 设备是 virtio-balloon。通常来说，要改变客户机所占用的宿主机内存，要先关闭客户机，修改启动时的内存配置，然后重启客户机才可以实现。而 内存的 ballooning（气球）技术可以在客户机运行时动态地调整它所占用的宿主机内存资源，而不需要关闭客户机。该技术能够：

- 当宿主机内存紧张时，可以请求客户机回收利用已分配给客户机的部分内存，客户机就会释放部分空闲内存。若其内存空间不足，可能还会回收部分使用中的内存，可能会将部分内存换到交换分区中。
- 当客户机内存不足时，也可以让客户机的内存气球压缩，释放出内存气球中的部分内存，让客户机使用更多的内存。



目前很多的VMM，包括 KVM, Xen，VMware 等都对 ballooning 技术提供支持。其中，KVM 中的 Ballooning 是通过宿主机和客户机协同来实现的，在宿主机中应该使用 2.6.27 及以上版本的 Linux内核（包括KVM模块），使用较新的 qemu-kvm（如0.13版本以上），在客户机中也使用 2.6.27 及以上内核且将“CONFIG_VIRTIO_BALLOON”配置为模块或编译到内核。在很多Linux发行版中都

已经配置有“CONFIG_VIRTIO_BALLOON=m”，所以用较新的Linux作为客户机系统，一般不需要额外配置virtio_balloon驱动，使用默认内核配置即可。

原理：

1. KVM 发送请求给 VM 让其归还一定数量的内存给KVM。
2. VM 的 virtio_balloon 驱动接到该请求。
3. VM 的驱动是客户机的内存气球膨胀，气球中的内存就不能被客户机使用。
4. VM 的操作系统归还气球中的内存给VMM
5. KVM 可以将得到的内存分配到任何需要的地方。
6. KM 也可以将内存返还到客户机中。

优势和不足：

优势	不足
1. ballooning 可以被控制和监控 2. 对内存的调节很灵活，可多可少。 3. KVM 可以归还内存给客户机，从而缓解其内存压力。	1. 需要客户机安装驱动 2. 大量内存被回收时，会降低客户机的性能。 3. 目前没有方便的自动化的机制来管理 ballooning，一般都在 QEMU 的 monitor 中执行命令来实现。 4. 内存的动态增加或者减少，可能是内存被过度碎片化，从而降低内存使用性能。

在QEMU monitor中，提供了两个命令查看和设置客户机内存的大小。

- (qemu) info balloon　#查看客户机内存占用量（Balloon信息）
- (qemu) balloon *num*　#设置客户机内存占用量为*num*MB

使用实例：

（1）启动一个虚机，内存为 2048M，启用 virtio-balloon

```
[root@rh65 domains]# kvm -smp 2 -cpu host -m 2048 -drive file=./rh64-6.qcow2,if=ide,media=disk,format=qcow2 -
boot c -name rh64-6  -net nic,model=e1000 -balloon virtio
```

（2）通过 vncviewer 进入虚机，查看 pci 设备

```
[root@rh64-1 ~]# lsmod | grep virtio
virtio_balloon          4798  0
virtio_pci              6985  0
virtio_ring             8301  2 virtio_balloon,virtio_pci
virtio                  4977  2 virtio_balloon,virtio_pci
[root@rh64-1 ~]# lspci
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: Cirrus Logic GD 5446
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controll
```

```
er (rev 03)
00:04.0 RAM memory: Red Hat, Inc Virtio memory balloon
```

（3）看看内存情况，共 2G 内存

```
[root@rh64-1 ~]# free -m
             total       used       free     shared    buffers     cached
Mem:          1877        183       1693          0         11         65
-/+ buffers/cache:        107       1770
Swap:         4031          0       4031
```

（4）进入 QEMU Monitor，调整 balloon 内存为 500M

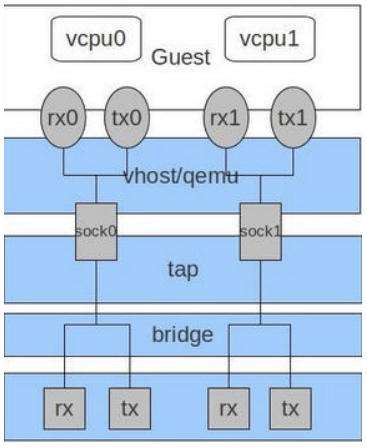
```
monitor console
QEMU 0.12.1 monitor - type 'help' for more information
(qemu) info balloon
balloon: actual=2048
(qemu) balloon 512
(qemu) info balloon
balloon: actual=512
```

（5）回到虚机，查看内存，变为 500 M

```
[root@rh64-1 ~]# free -m
             total       used       free     shared    buffers     cached
Mem:           341        183        157          0         11         65
-/+ buffers/cache:        107        234
Swap:         4031          0       4031
```

2.7 RedHat 的 多队列 Virtio（multi-queue）

目前的高端服务器都有多个处理器，虚拟使用的虚拟CPU数目也不断增加。默认的 virtio-net 不能并行地传送或者接收网络包，因为 virtio_net 只有一个TX 和 RX 队列。而多队列 virtio-net 提供了一个随着虚机的虚拟CPU增加而增强网络性能的方法，通过使得 virtio 可以同时使用多个 virt-queue 队列。



它在以下情况下具有明显优势：

1. 网络流量非常大
2. 虚机同时有非常多的网络连接，包括虚拟机之间的、虚机到主机的、虚机到外部系统的等
3. virtio 队列的数目和虚机的虚拟CPU数目相同。这是因为多队列能够使得一个队列独占一个虚拟CPU。

注意：对队列 virtio-net 对流入的网络流工作得非常好，但是对外发的数据流偶尔会降低性能。打开对队列 virtio 会增加中的吞吐量，这相应地会增加CPU的负担。 在实际的生产环境中需要做必须的测试后才确定是否使用。

在 RedHat 中，要使用多队列 virtio-net，在虚机的 XML 文件中增加如下配置：

```
<interface type='network'>
      <source network='default'/>
      <model type='virtio'/>
      <driver name='vhost' queues='N'/>
</interface>
```

然后在主机上运行下面的命令：

```
ethtool -L eth0 combined M ( 1 <= M <= N)
```

2.8 Windows 客户机的 virtio 前端驱动

Windows 客户机下的 virtio 前端驱动必须下载后手工安装。 RedHat Linux 这篇文章 说明了在 Windows 客户机内安装virtio 驱动的方法。

参考文档：

- http://linux.web.cern.ch/linux/centos7/docs/rhel/Red_Hat_Enterprise_Linux-7-Virtualization_Tuning_and_Optimization_Guide-en-US.pdf

- http://toast.djw.org.uk/qemu.html
- KVM 官方文档
- KVM 虚拟化技术实战与解析 任永杰、单海涛 著
- RedHat Linux 6 官方文档
- http://www.slideshare.net 中关于 KVM 的一些文档
- http://www.linux-kvm.org/page/Multiqueue
- 以及部分来自于网络，比如 http://smilejay.com/2012/11/use-ballooning-in-kvm/

分类: KVM,虚拟化

SammyLiu
关注 - 30
粉丝 - 470

荣誉：推荐博客

« 上一篇：KVM 介绍（2）：CPU 和内存虚拟化
» 下一篇：KVM 介绍（4）：I/O 设备直接分配和 SR-IOV [KVM PCI/PCIe Pass-Through SR-IOV]

posted on 2015-06-03 13:10 SammyLiu 阅读(6241) 评论(5) 编辑 收藏

评论:

#1楼 2015-06-03 16:01 | Edison Chou
点个赞，持续关注！期待 libvirt API 的介绍文章

支持(0) 反对(0)

#2楼 2015-07-01 00:20 | jython.li
顶一下

支持(0) 反对(0)

#3楼 2015-07-27 13:40 | jusonalien
写的很详实~

支持(0) 反对(0)

#4楼 2015-12-29 00:26 | liunian0o0
楼主，xml 文件里的pci slot号，能和虚拟机内部的对应起来吗？

支持(0) 反对(0)

#5楼[楼主] 2015-12-29 11:26 | SammyLiu
@ liunian0o0
虚机中的pci信息是类似这样的 <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2'/>，在虚机中 lspci 的结果是类似这样的
00:01.1 ..
00:02.0 ..
00:03.0 ..
，可见两者是能对应的，lspci 中的 <bus>:<slot>.<function>。具体你再查查吧。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

» 更多新闻...

最新知识库文章:

· 程序员的工作、学习与绩效
· 软件开发为什么很难
· 唱吧DevOps的落地，微服务CI/CD的范本技术解读
· 程序员，如何从平庸走向理想？
· 我为什么鼓励工程师写blog

» 更多知识库文章...