Virtio驱动安装适配及功能分析

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 修订者 | 修订描述 |
| 2017.10.26 | v1 | 林 清 | 初稿 |
| 2017.10.27 | V1.1 | 林 清 | 修改、整理 |
|  |  |  |  |
|  |  |  |  |

[1.1 介绍 2](#_Toc496880897)

[1.1.1 Virtio模块 2](#_Toc496880898)

[1.1.2 版本要求 2](#_Toc496880899)

[1.2 应用场景 2](#_Toc496880900)

[1.3 在客户机上安装virtio驱动 3](#_Toc496880901)

[1.3.1 切换I/O虚拟化的工作方式 3](#_Toc496880902)

[1.3.2 制作Windows客户机镜像 3](#_Toc496880903)

[1.3.3 Linux客户机安装驱动 4](#_Toc496880904)

[1.4 I/O虚拟化方式介绍 5](#_Toc496880905)

[1.4.1 通过QEMU方式模拟虚拟机I/O 5](#_Toc496880906)

[1.4.2 通过Virtio模拟虚拟机I/O 6](#_Toc496880907)

[1.5 参考资料 7](#_Toc496880908)

## 介绍

KVM-QEMU 虚拟化方案中，I/O虚拟化传统方式是使用QEMU纯软件的方式来模拟I/O设备，效率并不非常高。

在新版本的KVM方案中，可以在客户机中使用半虚拟化驱动（Paravirtualized Drivers，PV Drivers）Virtio来提高客户机的性能（特别是I/O性能，包括：磁盘、网络、GPU）。

Virtio作为一种I/O半虚拟化解决方案，需要对Guest操作系统进行修改才能正常工作。本文将简要介绍Virtio，以及如何在Linux &Windows客户机中安装Virtio。

### Virtio模块

在客户机中，virtio由以下模块组成：

|  |  |  |
| --- | --- | --- |
| 子模块 | 模块功能 | 备注 |
| virtio-net | 网络模块 |  |
| virtio-blk | 块设备模块 | 磁盘相关的模块，但是virtio\_blk模拟的并非是一个真正的scsi设备，并且对部分设备不支持。virtio\_scsi是前者的增强，性能和前者一致，适用范围更广。目前Linux默认使用的是virtio-blk，而Windows下只有virtio-scsi。 |
| virtio-scsi | SCSI设备模块 |
| virtio-serial | 总线模块 | 包括virtio\_console、virtio\_pci ?? |
| virtio-ballon | 内存模块 | 主要用来动态调整虚拟机的内存，已经采集虚机中的内存信息 |
| virtio-rng | 随机数模块 |  |
| virtio-9p | 模拟9P文件系统 |  |
| 其他模块 | 显卡和输入设备模块 | virtio-vga/ virtio-input都在开发中 |

### 版本要求

* KVM & Libvirt版本要求libvirt >= 0.4.4，KVM版本大于60。
* Linux客户机要求内核版本大于2.6.25（通过uname –a查看，查了一下这个版本是2008年发布的https://kernelnewbies.org/Linux\_2\_6\_25），Fedora发行版要求版本至少大于等于Fedora 9 。
* Windows操作系统有virtio-win工具覆盖绝大多数的Windows版本。

## 应用场景

当前Openstack常用的Guest操作系统主要是Linux 和Windows，两种操作系统需要安装对应版本的Virtio。在Linux中2.6.25以上的内核版本默认集成了virtio 模块，Windows版本大多数情形下不包含Virtio驱动需要手动安装。因此对部门的vTool工具来说，主要要解决的问题是如何在Windows 虚机中安装Virtio。有以下两个场景：

**场景一：**

直接自作带Virtio驱动的Glance镜像。

**场景二：**

Windows虚拟机已经启动，工作在I/O全虚拟化模式下。在虚拟机外部，如何安装Virtio驱动，并且切换到VirtIO准虚拟化模式？

## 在客户机上安装virtio驱动

### 切换I/O虚拟化的工作方式

当前部门环境中，Nova Compute默认情形下都是以virtio作为I/O设备驱动的（包括：任意类型的存储、网络、内存），而且没有配置项可以修改为全虚拟化I/O。

以磁盘IO的配置为例，virtio虚拟化配置为：

<disk type='file' device='disk'>

<driver name='qemu' type='qcow2' cache='none'/>

……

<target dev='vda' bus='virtio'/>

<alias name='virtio-disk0'/>

……

</disk>

全虚拟化配置为：

<disk type='file' device='disk'>

<driver name='qemu' type='qcow2' cache='none'/>

……

<target dev='hda' bus='ide'/>

<alias name='disk0'/>

……

</disk>

切换虚拟化配置以后，部分设备KVM是不支持热生效的（比如磁盘设备），需要重启虚拟机。

### 制作Windows客户机镜像

对Windows操作系统来说，官方的ISO不能直接用来启动虚拟机，因此一般要先制作qcow2格式的Windows镜像。制作完成镜像以后以ide方式将虚拟机启动，在虚拟机中安装VirtIO驱动，完成后导出该镜像。

#### 制作Windows镜像

制作一个Windows的qcow2镜像过程大致如下：

* 下载ISO镜像；
* 创建空白qcow2文件，作为虚拟机的硬盘；
* 从Virsh直接创建虚机，在配置文件中指定从光驱启动（光驱指向下载的ISO镜像）；
* 启动虚机将Windows安装到空白qcow2文件中；
* 从硬盘重启虚机，光驱指向Virtio的文件，进入虚机将Virtio安装好；

#### 获取Virtio-Win驱动

在fedora上可以获取带数字签名的客户机驱动程序（Windows管理驱动时，需要驱动带数字签名）。

# 添加fedora源

sudo wget https://fedorapeople.org/groups/virt/virtio-win/virtio-win.repo -O /etc/yum.repos.d/virtio-win.repo

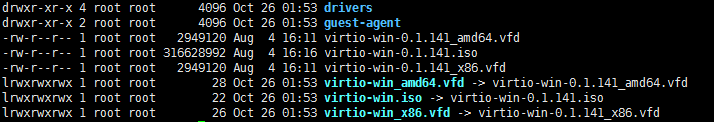
# 安装virtio-win-stable

sudo yum install virtio-win

# 安装virtio-win-latest

sudo yum --enablerepo=virtio-win-latest update virtio-win

安装之后可以在/usr/share/virtio-win/目录下看到安装文件（下面的这些文件也可以到<http://fedoraproject.org/wiki/Windows_Virtio_Drivers>直接下载）：



\*.iso包含所有的模块驱动（网络、磁盘、内存、RNG等），支持的Windows操作系统版本覆盖Win10 到WinXP。

### Linux客户机安装驱动

Linux内核以模块形式集成了Virtio，借助一些Linux平台的模块管理工具，可以进行对Linux 内核中的模块进行管理。

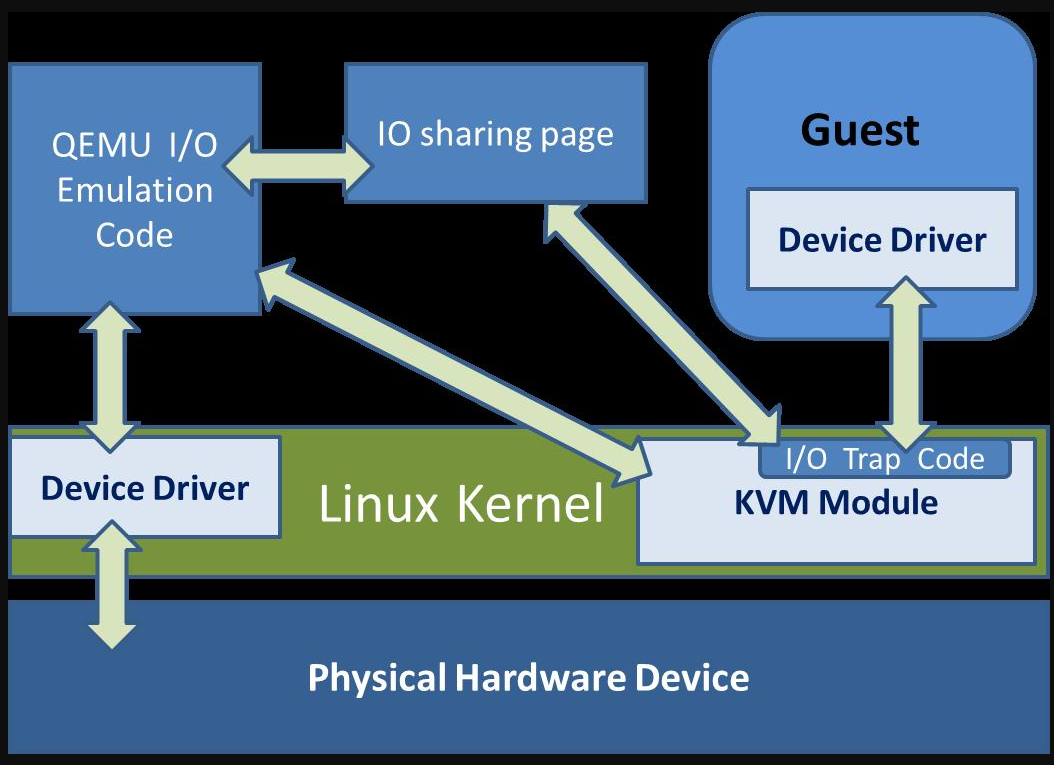
使用modprobe、modinfo、lsmod可以查看模块信息，以及动态加载卸载某些模块。

在/etc/modules-load.d/目录下可以配置开机自动加载模块（参考man modules-load.d）。通过modprobe黑名单功能可以禁用某些自动加载的模块。

## I/O虚拟化方式介绍

### 通过QEMU方式模拟虚拟机I/O

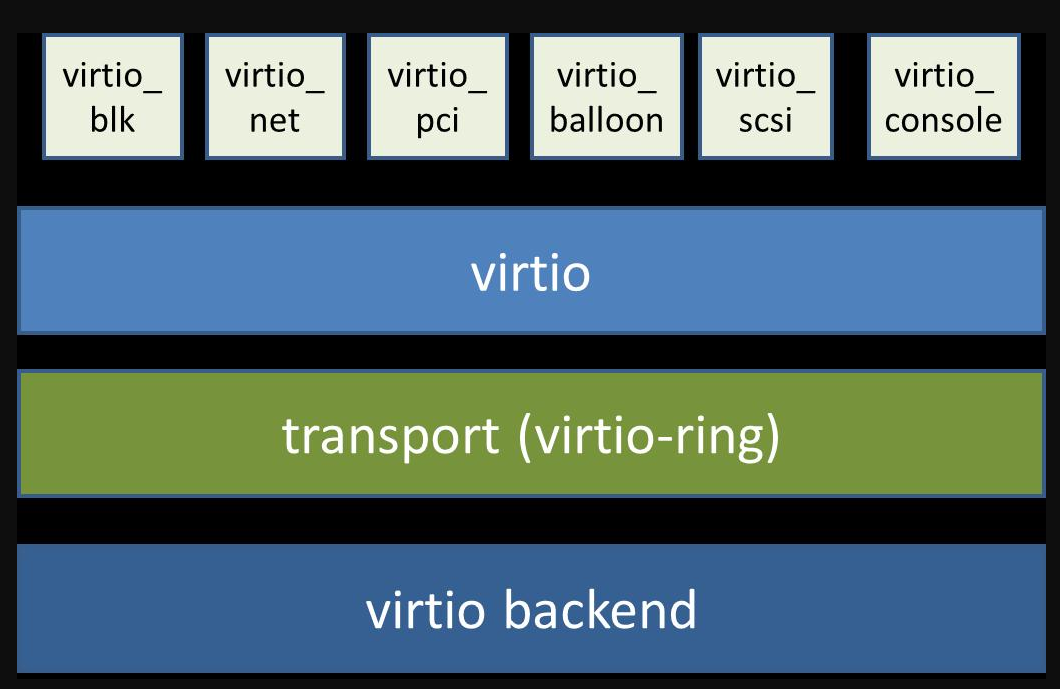
QEMU纯软件方式模拟I/O设备的基本过程模型如下图。



使用QEMU模拟I/O的情况下，当客户机中的设备驱动程序（device driver）发起I/O操作请求之时，KVM模块中的I/O操作捕获代码会拦截这次I/O请求，然后经过处理后将本次I/O请求的信息存放到I/O共享页，并通知用户控件的QEMU程序。QEMU模拟程序获得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的I/O操作，完成之后，将结果放回到I/O共享页，并通知KVM模块中的I/O操作捕获代码。最后，由KVM模块中的捕获代码读取I/O共享页中的操作结果，并把结果返回到客户机中。

QEMU模拟I/O设备的方式，其优点是可以通过软件模拟出各种各样的硬件设备。而它的缺点是，每次I/O操作的路径比较长，有较多的VMEntry、VMExit发生，需要多次上下文切换（context switch），也需要多次数据复制，所以它的性能较差。

### 通过Virtio模拟虚拟机I/O



其中前端驱动（frondend，如virtio-blk、virtio-net等）是在客户机中存在的驱动程序模块，而后端处理程序（backend）是在QEMU中实现的。

在这前后端驱动之间，还定义了两层来支持客户机与QEMU之间的通信：

virtio层是虚拟队列接口，它在概念上将前端驱动程序附加到后端处理程序。一个前端驱动程序可以使用0个或多个队列，具体数量取决于需求。例如，virtio-net网络驱动程序使用两个虚拟队列（一个用于接收，另一个用于发送），而virtio-blk块驱动程序仅使用一个虚拟队列。虚拟队列实际上被实现为跨越客户机操作系统和hypervisor的衔接点，但它可以通过任意方式实现，前提是客户机操作系统和virtio后端程序都遵循一定的标准，以相互匹配的方式实现它。

virtio-ring层实现了环形缓冲区（ring buffer），用于保存前端驱动和后端处理程序执行的信息，并且它可以一次性保存前端驱动的多次I/O请求，并且交由后端去动去批量处理，最后实际调用宿主机中设备驱动实现物理上的I/O操作，这样做就可以根据约定实现批量处理而不是客户机中每次I/O请求都需要处理一次，从而提高客户机与hypervisor信息交换的效率。

Virtio半虚拟化驱动的方式，可以获得很好的I/O性能，其性能几乎可以达到和native（即：非虚拟化环境中的原生系统）差不多的I/O性能。所以，在使用KVM之时，如果宿主机内核和客户机都支持virtio的情况下，一般推荐使用virtio达到更好的性能。当然，virtio的也是有缺点的，它必须要客户机安装特定的Virtio驱动使其知道是运行在虚拟化环境中，且按照Virtio的规定格式进行数据传输，不过客户机中可能有一些老的Linux系统不支持virtio和主流的Windows系统需要安装特定的驱动才支持Virtio。不过，较新的一些Linux发行版（如RHEL 6.3、Fedora 17等）默认都将virtio相关驱动编译为模块，可直接作为客户机使用virtio，而且对于主流Windows系统都有对应的virtio驱动程序可供下载使用。

## 参考资料

<http://fedoraproject.org/wiki/Windows_Virtio_Drivers> （获取安装包）

<http://www.linux-kvm.org/page/WindowsGuestDrivers/viostor/installation> （在Windows中安装）

<https://wiki.archlinux.org/index.php/Kernel_modules_(%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)> （Linux的模块操作）

<https://www.ibm.com/developerworks/cn/linux/l-virtio/> （virtio虚拟化介绍）