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（4）：I/O 设备直接分配和 SR-IOV [KVM PCI/PCIe Pass-Through SR-IOV]

学习 KVM 的系列文章：

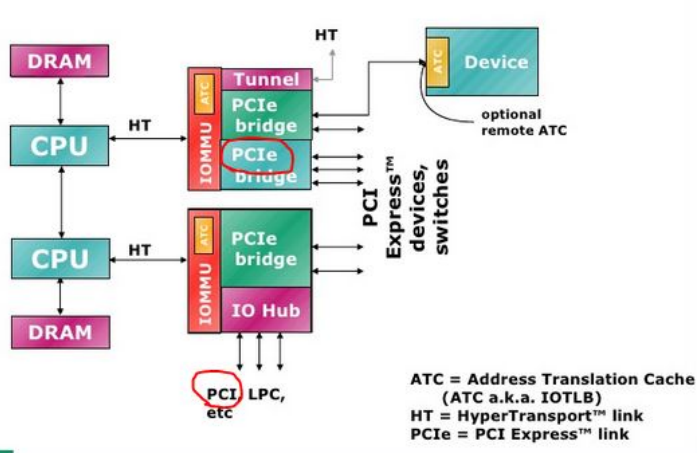
- （1）介绍和安装
- （2）CPU 和 内存虚拟化
- （3）I/O QEMU 全虚拟化和准虚拟化（Para-virtulizaiton）
- （4）I/O PCI/PCIe设备直接分配和 SR-IOV
- （5）libvirt 介绍
- （6）Nova 通过 libvirt 管理 QEMU/KVM 虚机
- （7）快照（snapshot）
- （8）迁移（migration）

本文将分析 PCI/PCIe 设备直接分配（Pass-through）和 SR-IOV， 以及三种 I/O 虚拟化方式的比较。

1. PCI/PCI-E 设备直接分配给虚机（PCI Pass-through）

设备直接分配（Device assignment）也称为 Device Pass-Through。

先简单看看PCI 和 PCI-E 的区别（AMD CPU）：



（简单点看，PCI 卡的性能没有 PCI-E 高，因为 PCI-E 是直接连在 IOMMU 上，而 PCI 卡是连在一个 IO Hub 上。）

主要的 PCI 设备类型：

- Network cards (wired or wireless)
- SCSI adapters
- Bus controllers: USB, PCMCIA, I2C, FireWire, IDE
- Graphics and video cards
- Sound cards

1.1 PCI/PCIe Pass-through 原理

这种方式，允许将宿主机中的物理 PCI 设备直接分配给客户机使用。较新的x86平台已经支持这种类型，Intel 定义的 I/O 虚拟化技术成为 VT-d，AMD 的称为 AMD-V。KVM 支持客户机以独占方式访问这个宿主机的 PCI/PCI-E 设备。通过硬件支持的 VT-d 技术将设备分给客户机后，在客户机看来，设备是物理上连接在PCI或者PCI-E总线上的，客户机对该设备的I/O交互操作和实际的物理设备操作完全一样，不需要或者很少需要 KVM 的参与。运行在 VT-d 平台上的 QEMU/KVM，可以分配网卡、磁盘控制器、USB控制器、VGA 显卡等设备供客户机直接使用。

几乎所有的 PCI 和 PCI-E 设备都支持直接分配，除了显卡以外（显卡的特殊性在这里）。PCI Pass-through 需要硬件平台 Intel VT-d 或者 AMD IOMMU 的支持。这些特性必须在 BIOS 中被启用。Red Hat Enterprise Linux 6.0 及以上版本支持热插拔的 PCI 设备直接分配到虚拟机。

网卡直接分配：

性能(4)
虚拟化(7)
原理(22)
云Cloud(29)

随笔档案(121)

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)

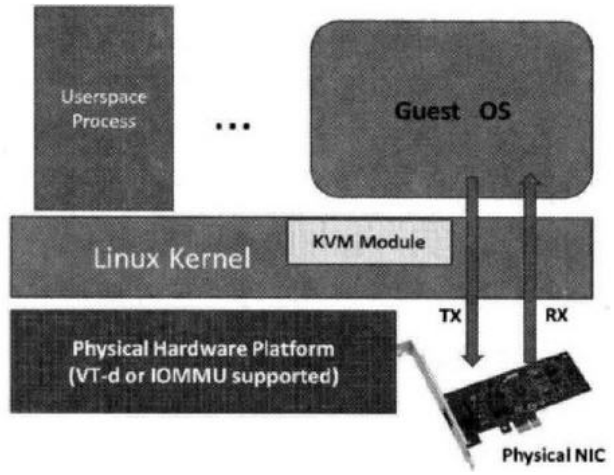
2016年10月 (2)
2016年9月 (1)
2016年6月 (1)
2016年5月 (3)
2015年12月 (4)
2015年10月 (5)
2015年9月 (2)
2015年6月 (1)
2015年4月 (23)

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



硬盘直接分配：

- 一般 SATA 或者 SAS 等类型的硬盘的控制器都是直接接入到 PCI 或者 PCI-E 总线的，所以也可以将硬盘作为普通的PCI设备直接分配个客户机。需要注意的是，当分配硬盘时，实际上将其控制器作为一个整体分配到客户机中，因此需要在硬件平台上至少有另两个或者多个SATA或者 SAS控制器。

1.2 在 RedHat Linux 6 上使用 virt-manger 分配一个光纤卡给虚机

准备工作：

（1）在 BIOS 中打开 Intel VT-d

（2）在 Linux 内核中启用 PCI Pass-through

添加 intel_iommu=on 到 /boot/grub/grub.conf 文件中。（在我的 RedHat Linux 6上，该文件是 /boot/grub.conf）

（3）重启系统，使得配置生效

实际分配：

（1）使用 lspci -nn 命令找到待分配的 PCI 设备。这里以一个 FC 卡为例：

```
[root@rh65 ~]# lspci -nn | grep Fibre
1f:00.0 Fibre Channel [0c04]: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PC
I Express HBA [1077:2532] (rev 02)
1f:00.1 Fibre Channel [0c04]: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PC
I Express HBA [1077:2532] (rev 02)
```

使用 lspci 命令得到的 PCI 数字的含义，以后使用 libvirt API 分配设备时会用到：

```
02:01.0 CardBus bridge: Ricoh Co Ltd RL5c476 II (rev b4)
02:01.1 FireWire (IEEE 1394): Ricoh Co Ltd R5C552 IEEE 1394 Controller
02:01.2 SD Host controller: Ricoh Co Ltd R5C822 SD/SDIO/MMC/MS/MSPro
```

Function number
PCI device number
PCI bus number

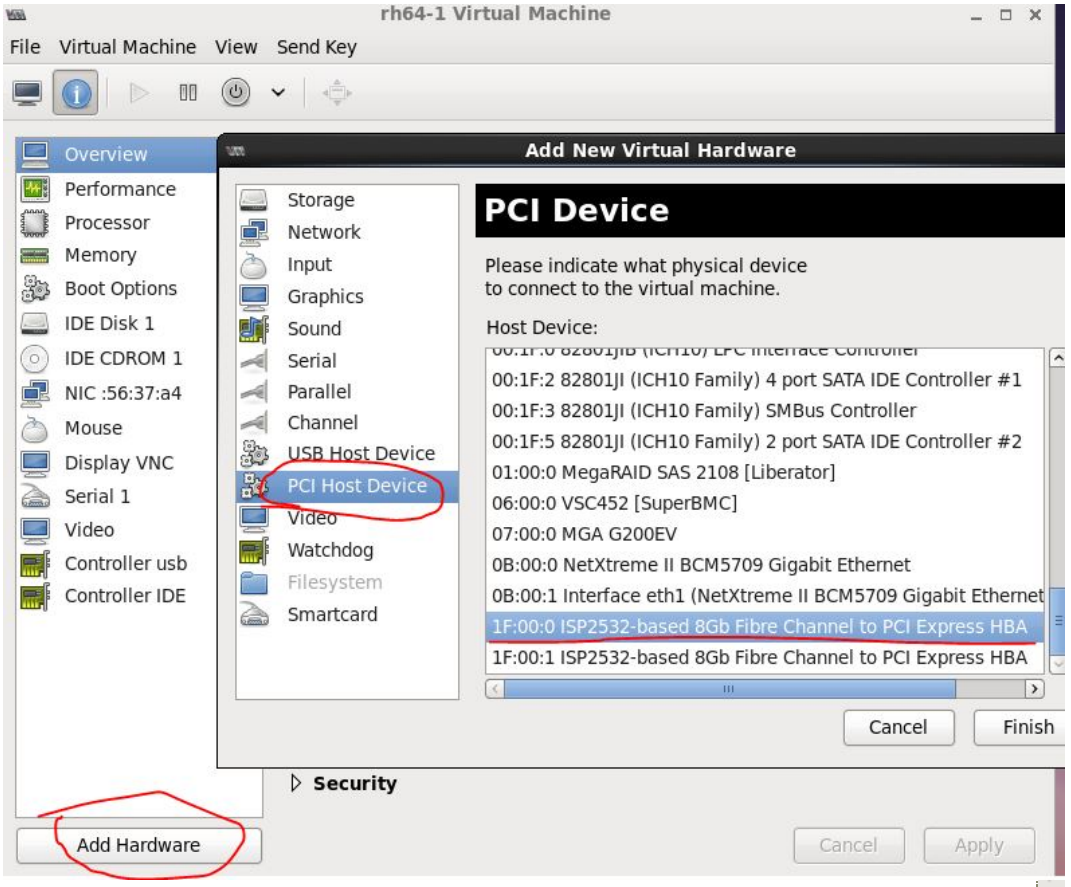
（2）使用 virsh nodedev-list 命令找到该设备的 PCI 编号

```
[root@rh65 ~]# virsh nodedev-list --tree | grep 1f
  |   +- pci_0000_1f_00_0
  |   +- pci_0000_1f_00_1
  +- pci_0000_00_1f_0
  +- pci_0000_00_1f_2
  +- pci_0000_00_1f_3
  +- pci_0000_00_1f_5
```

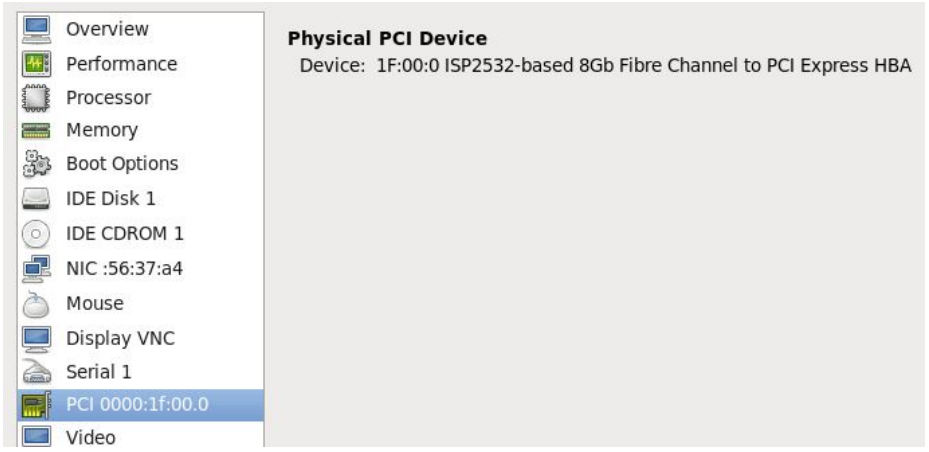
（3）将设备从主机上解除

```
[root@rh65 ~]# virsh nodedev-dettach pci_0000_1f_00_0
Device pci_0000_1f_00_0 detached
```

（4）使用 virt-manager 将设备直接分配给一个启动了的虚拟机



（5）添加好了后的效果



（6）在虚机中查看该PCI设备

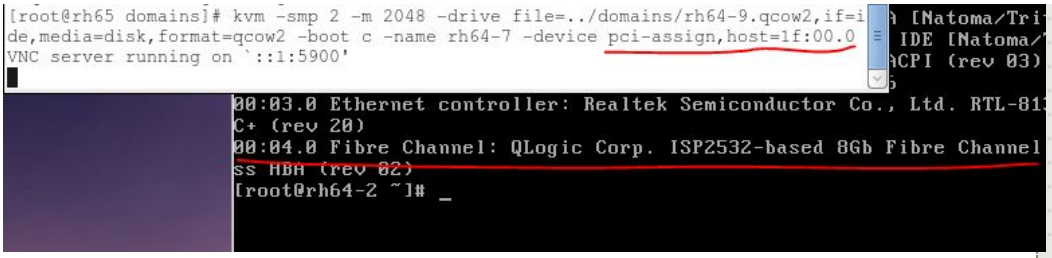
```
[root@rh64-1 ~]# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.2 USB controller: Intel Corporation 82371SB PIIX3 USB [Natoma/Triton II] (
rev 01)
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: Realtek Semiconductor Co., Ltd. RTL-8139/8139C/8139
C+ (rev 20)
00:04.0 RAM memory: Red Hat, Inc Virtio memory balloon
00:05.0 Fibre Channel: QLogic Corp. ISP2532-based 8Gb Fibre Channel to PCI Expre
ss HBA (rev 02)
```

（7）不再使用的话，需要在 virt-manager 中首先将该设备移除，然后在主机上重新挂载该设备

```
[root@rh65 s1]# virsh nodedev-reattach pci_0000_0b_00_0
Device pci_0000_0b_00_0 re-attached
```

1.3 在 RedHat Linux 6 上使用 qemu-kvm 分配一个光纤卡给虚机

除了步骤（4），其他步骤同上面。



1.4 设备直接分配让客户机的优势和不足

- 好处：在执行 I/O 操作时大量减少甚至避免 VM-Exit 陷入到 Hypervisor 中，极大地提高了性能，可以达到几乎和原生系统一样的性能。VT-d 克服了 virtio 兼容性不好和 CPU 使用频率较高的问题。
- 不足：（1）一台服务器主板上的空间比较有限，因此允许添加的 PCI 和 PCI-E 设备是有限的。大量使用 VT-d 独立分配设备给客户机，让硬件设备数量增加，这会增加硬件投资成本。（2）对于使用 VT-d 直接分配了设备的客户机，其动态迁移功能将受限，不过也可以使用热插拔或者libvirt 工具等方式来缓解这个问题。
- 不足的解决方案：（1）在一台物理宿主机上，仅少数 I/O 如网络性能要求较高的客户机使用 VT-d直接分配设备，其他的使用纯模拟或者 virtio 已达到多个客户机共享同一个设备的目的（2）对于网络I/O的解决办法，可以选择 SR-IOV 是一个网卡产生多个独立的虚拟网卡，将每个虚拟网卡分配个一个客户机使用。

2. SR-IOV 设备分配

2.1 原理

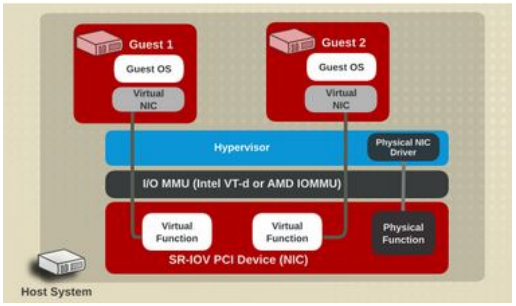
VT-d 的性能非常好，但是它的物理设备只能分配给一个客户机使用。为了实现多个虚机共享一个物理设备，并且达到直接分配的目的，PCI-SIG 组织发布了 SR-IOV（Single Root I/O Virtualization and sharing）规范，它定义了一个标准化的机制用以原生地支持实现多个客户机共享一个设备。不过，目前 SR-IOV（单根 I/O 虚拟化）最广泛地应用还是网卡上。

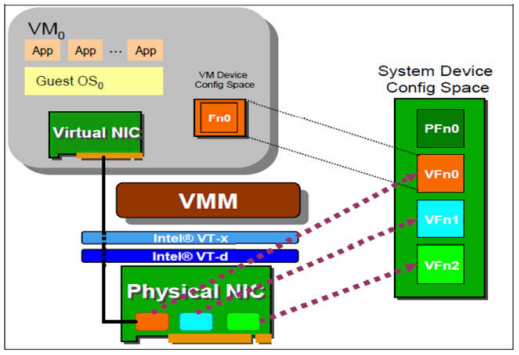
SR-IOV 使得一个单一的功能单元（比如，一个以太网端口）能看起来像多个独立的物理设备。一个带有 SR-IOV 功能的物理设备能被配置为多个功能单元。SR-IOV 使用两种功能（function）：

- 物理功能（Physical Functions，PF）：这是完整的带有 SR-IOV 能力的PCIe 设备。PF 能像普通 PCI 设备那样被发现、管理和配置。
- 虚拟功能（Virtual Functions，VF）：简单的 PCIe 功能，它只能处理I/O。每个 VF 都是从 PF 中分离出来的。每个物理硬件都有一个 VF 数目的限制。一个 PF，能被虚拟成多个 VF 用于分配给多个虚拟机。

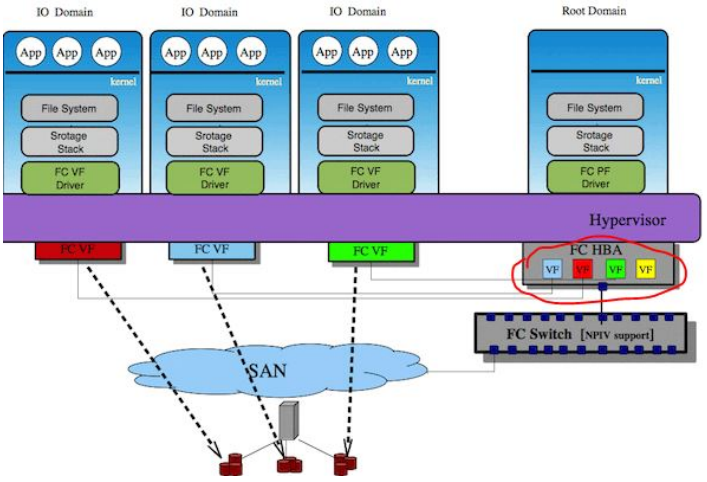
Hypervisor 能将一个或者多个 VF 分配给一个虚机。在某一时刻，一个 VF 只能被分配给一个虚机。一个虚机可以拥有多个 VF。在虚机的操作系统看来，一个 VF 网卡看起来和一个普通网卡没有区别。SR-IOV 驱动是在内核中实现的。

网卡 SR-IOV 的例子：





光纤卡 SR-IOV 的例子：



2.2 SR-IOV 的条件

1. 需要 CPU 支持 Intel VT-x 和 VT-D（或者 AMD 的 SVM 和 IOMMU）
2. 需要有支持 SR-IOV 规范的设备：目前这种设备较多，比如Intel的很多中高端网卡等。
3. 需要 QEMU/KAM 的支持。

RedHat Linux 6.0 官方只完整测试了下面的几款 SR-IOV 网卡：

- Intel® 82576NS Gigabit Ethernet Controller (**igb** 驱动)
- Intel® 82576EB Gigabit Ethernet Controller (**igb** 驱动)
- Intel® 82599ES 10 Gigabit Ethernet Controller (**ixgbe** 驱动)
- Intel® 82599EB 10 Gigabit Ethernet Controller (**ixgbe** 驱动)

2.3 分配 SR-IOV 设备的步骤

手头没有支持SR-IOV的设备。这是 RedHat 上 SR-IOV 的配置步骤： Using SR-IOV。

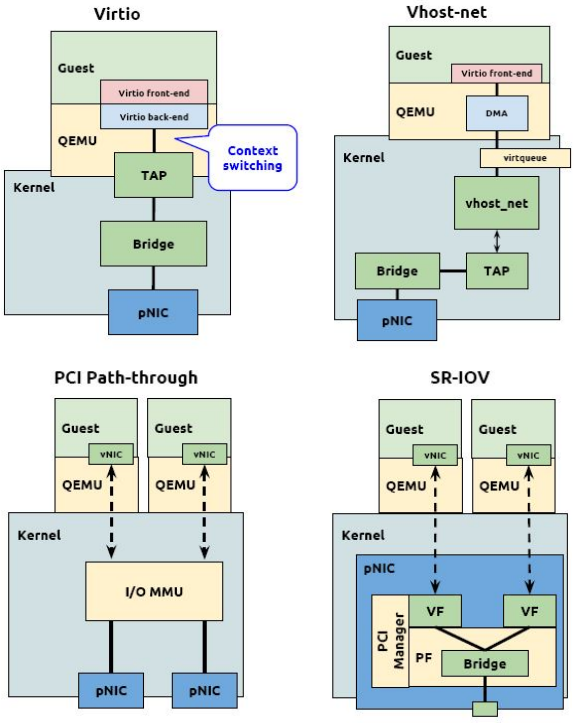
简单来说，SR-IOV 分配步骤和设备直接分配相比基本类似，除了要使 PF 虚拟化成多个 VF 以外。

2.4 优势和不足

优势	不足
1. 真正实现设备共享（多个客户机共享一个 SR-IOV 设备的物理端口） 2. 接近原生性能 3. 相比 VT-d， SR-IOV 可以使用更少的设备来支持更多的客户机，可以提高数据中心的空间利用率。	1. 对设备有依赖，目前只有部分设备支持 SR-IOV。RedHat Linux 只是测试了 Intel 的几款高端网卡。 2. 使用 SR-IOV 时不方便动态迁移客户机。 这是因为这时候虚机直接使用主机上的物理设备，因此虚机的迁移（migiration）和保存（save）目前都不支持。这个在将来有可能被改变。

3. 各种设备虚拟化方式的比较

3.1 架构上的比较（以网卡为例）



3.2 性能上的比较（以网卡为例）

纯模拟网卡和物理网卡的比较：

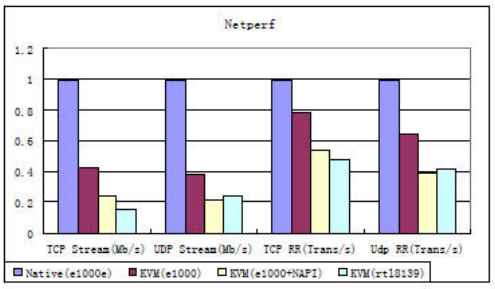


Fig. 4. Netperf benchmark test, Guest OS used different NIC configurations - virtual e1000 NIC (e1000 driver, e1000 driver with NAPI support), virtual rtl8139 NIC

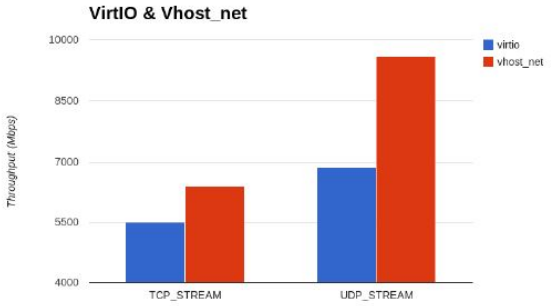
（来源：Evaluating and Optimizing I/O Virtualization in Kernel-based Virtual Machine (KVM), Binbin Zhang, Xiaolin Wang, Rongfeng Lai, Liang Yang, Zhenlin Wang,Yingwei Luo, Xiaoming Li）

（测试环境：两台物理服务器 HostA 和 HostB，都使用GB以太网。HostA 使用 82566DC 网卡，HostB 使用 82567LM-2 网卡，一台虚机运行在 HostB 上，使用 KVM-76.）

结论：

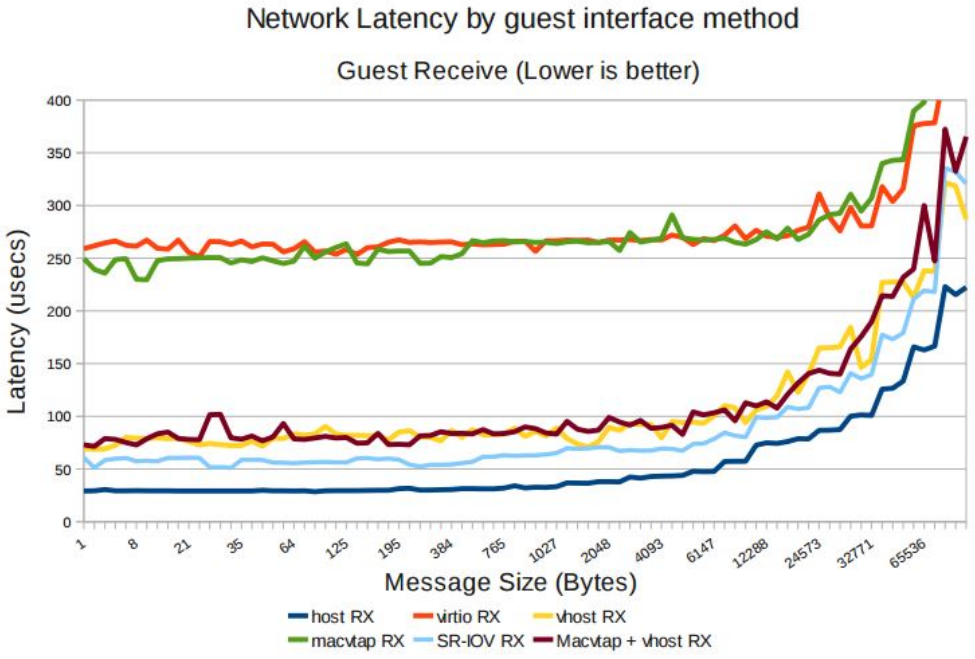
- 纯模拟网卡的性能只有物理网卡的四成到六成
- 纯模拟网卡的 UDP 性能比 TCP 性能高 50% 到 100%
- 在虚拟网卡上使用 NAPI，不但不会提高性能，反而会是性能下降
- e1000 的性能比 rt18139 的性能高不少（为什么 RedHat Linux KVM 上默认的网卡是 rt18139 呢？）

Virtio 和 vhost_net 的吞吐量比较：



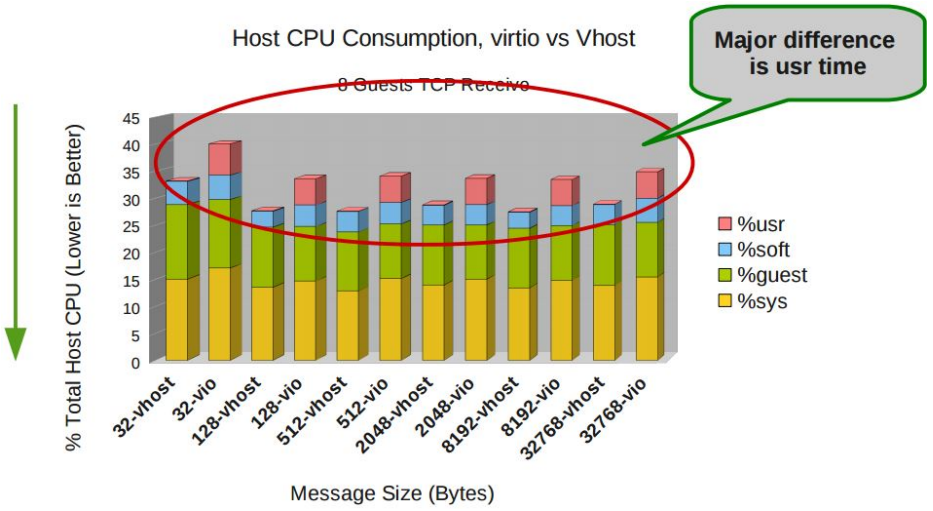
- 来源：CANONICAL, KVM Performance Optimization, Paul Sim,Cloud Consultant, paul.sim@canonical.com
- 结论： vhost_net 比 virtio 的 UDP 和 TCP 性能高 20% 左右。

RedHat Linux 6 上 virtio，vhost_net，SR-IOV 和物理设备网络延迟的比较：



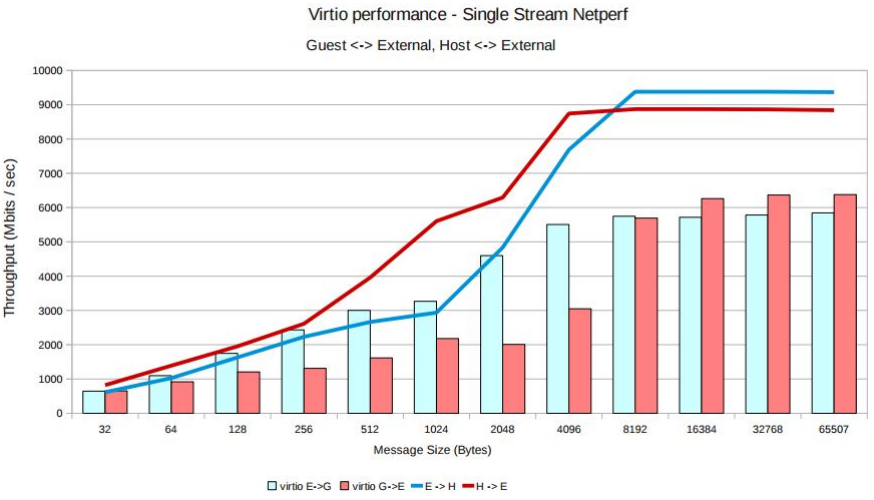
（来源：RedHat 官网）

RedHat Linux 6 上 virtio 和 vhost_net 所消耗的主机CPU资源的比较：



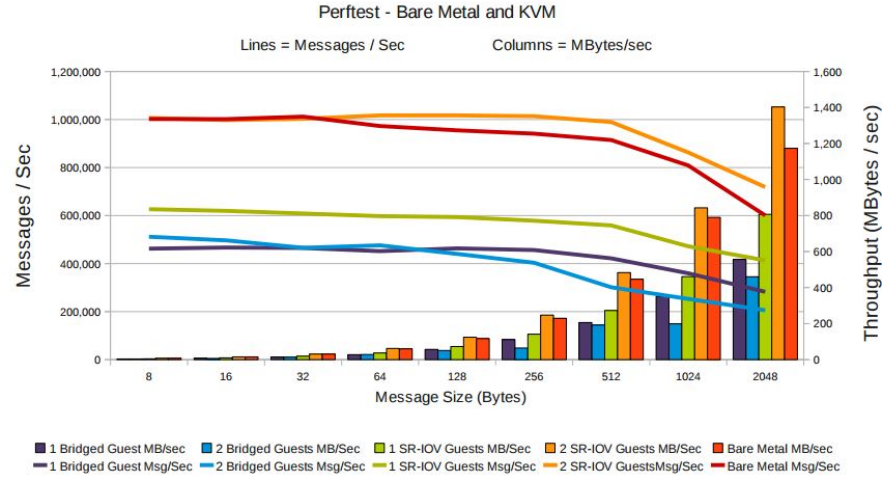
（来源同上）

使用 virtio 的 KVM 与物理机的 TCP 吞吐量对比：



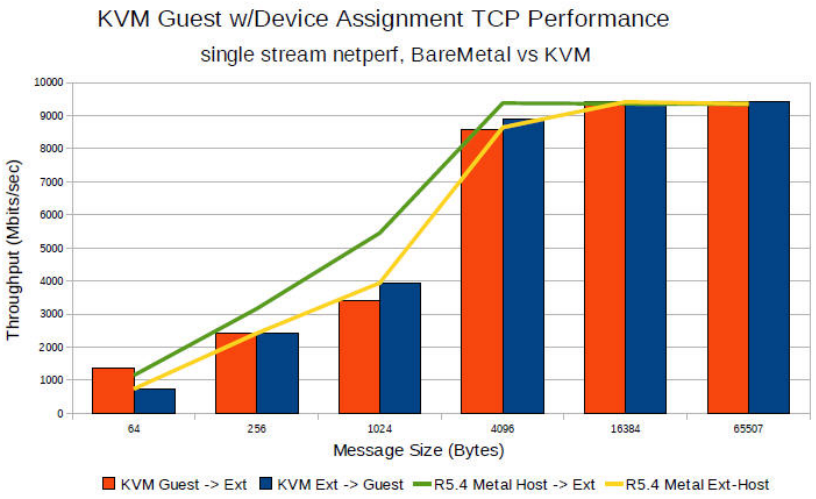
（数据来源：RedHat 官网）

物理机与使用 SR-IOV 的 KVM 的网络性能对比：



（来源：同上）

物理机与使用 Pass-through 的KVM 的 TCP 性能对比：



（资料来源：Open Source Virtualization: KVM and Linux, Chris Wright, Principal Software Engineer, Red Hat,September 4, 2009）

3.3 Virtio 和 Pass-Through 的详细比较

Basic Operation	- Backend/Guest direct access to shared Vring buffers - PIO - Switching at software level - Management Flexibility – internal SDN support ovs-vsctl add-port br0 <phys-intfc> - vSwitch ovs-ofctl – control flows - IRQ bottleneck – QEMU – call into kvm inject Kernel – inject directly	- Direct access to hw memory regions - DMA Support - Switching at hw level – SR-IOV depends on #of Queues - Management Flexibility – external SDN capable - IRQ bottleneck – hw enhancements, posted interrupts, exitless EOI improve things – closer to native
Migration	- Virtio lockless - Saves device state, tracks dirty pages	- QEMU sets 'unmigratable', or installs migration blocker - Guest can be holding a lock – deadlock, hw state, ...
Scalability	- Practical limitations – primarily per formance	- Number of Devices limited, limits #VMs - SR-IOV - #of VF - # of queues
Network Performance	- Soft switching – bridge, vSwitch - Several IO HOPS - Can approach near native – 10Ge for few bridged Guest	- Switching done at HW level – hw queues - Performance scales with # of Guests - DMA support - IRQ Passthrough still a problem
Host Performance	- PIO – takes cpu cycles - Exits – few but still - Guest pages swapable	- Guest pinned – can't swap - Fewer exits - Less PIO
Cloud Environment	- Cloud friendly – migration, SDN, paging	- Not Cloud friendly, great for NFV/RT DPDK, run to completion

（来源：Reconnaissance of Virtio: What's new and how it's all connected? by Mario Smarduch）

4. 综合结论

KVM 依赖的Intel/AMD 处理器的各种虚拟化扩展：

处理器	CPU 虚拟化	内存虚拟化	PCI Pass-through
Intel	VT-x	VPID，EPT	VT-d
AMD	AMD-V	ASID，NPT	IOMMU

I/O 虚拟化方案的选择：

- I/O设备尽量使用准虚拟化（virtio 和 vhost_net）
- 如果需要实时迁移，不能使用 SR-IOV
- 对更高I/O要求又不需要实时迁移的，可以使用 SR-IOV
- 每种方案都有优势和不足，在特定环境下其性能有可能反而下降，因此在生产环境中使用各种虚拟化方式前需要经过完整测试

其它参考资料：

- RedHat Linux 6 官方文档
- KVM 官方文档
- KVM 虚拟化技术实战与解析 任永杰、单海涛 著
- KVM 虚拟化技术在 AMD 平台上的实现

分类: KVM,虚拟化

SammyLiu
关注 - 30
粉丝 - 470
荣誉：推荐博客
+加关注

« 上一篇：KVM 介绍（3）：I/O 全虚拟化和准虚拟化 [KVM I/O QEMU Full-Virtualizaiton Para-virtualization]
» 下一篇：KVM 介绍（5）：libvirt 介绍 [Libvrit for KVM/QEMU]

posted on 2015-06-05 08:06 SammyLiu 阅读(7732) 评论(0) 编辑 收藏