**虚拟化类型**

**全虚拟化（Full Virtualization)**

全虚拟化也成为原始虚拟化技术，该模型使用虚拟机协调guest操作系统和原始硬件，VMM在guest操作系统和裸硬件之间用于工作协调，一些受保护指令必须由Hypervisor（虚拟机管理程序）来捕获处理。

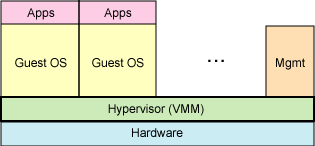
[](https://huangwei.me/wiki/image/22242080_1.gif)

图1 全虚拟化模型

全虚拟化的运行速度要快于硬件模拟，但是性能方面不如裸机，因为Hypervisor需要占用一些资源。

**半虚拟化（Para Virtualization）**

半虚拟化是另一种类似于全虚拟化的技术，它使用Hypervisor分享存取底层的硬件，但是它的guest操作系统集成了虚拟化方面的代码。该方法无需重新编译或引起陷阱，因为操作系统自身能够与虚拟进程进行很好的协作。

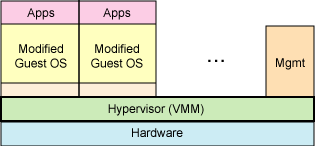
[](https://huangwei.me/wiki/image/22242080_2.gif)

图2 半虚拟化模型

半虚拟化需要guest操作系统做一些修改，使guest操作系统意识到自己是处于虚拟化环境的，但是半虚拟化提供了与原操作系统相近的性能。

**虚拟化技术**

**KVM(Kernel-based Virtual Machine)基于内核的虚拟机**

KVM是集成到Linux内核的Hypervisor，是X86架构且硬件支持虚拟化技术（Intel VT或AMD-V）的Linux的全虚拟化解决方案。它是Linux的一个很小的模块，利用Linux做大量的事，如任务调度、内存管理与硬件设备交互等。

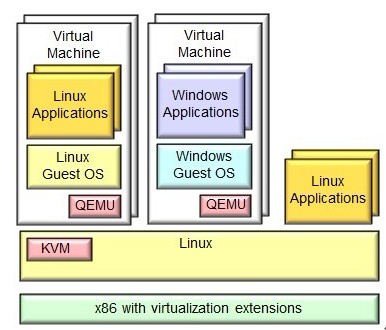
[](https://huangwei.me/wiki/image/kvm%E8%99%9A%E6%8B%9F%E5%8C%96%E5%B9%B3%E5%8F%B0%E6%9E%B6%E6%9E%84.jpg)

图3 KVM虚拟化平台架构

KVM是linux内核的模块，它需要CPU的支持，采用硬件辅助虚拟化技术Intel-VT，AMD-V，内存的相关如Intel的EPT和AMD的RVI技术，Guest OS的CPU指令不用再经过Qemu转译，直接运行，大大提高了速度，KVM通过/dev/kvm暴露接口，用户态程序可以通过ioctl函数来访问这个接口。见如下伪代码：

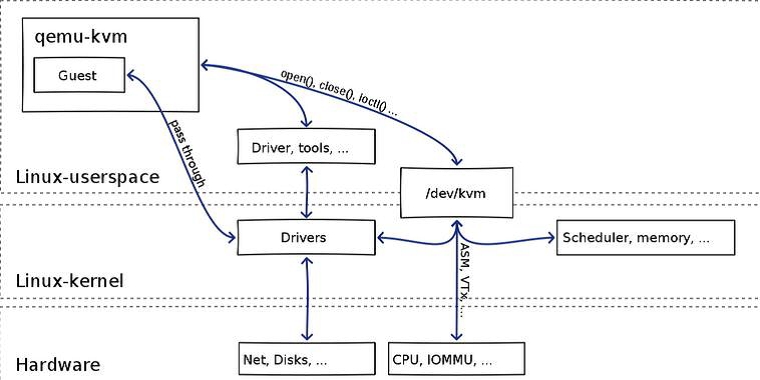
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | open("/dev/kvm")  ioctl(KVM\_CREATE\_VM)  ioctl(KVM\_CREATE\_VCPU)  for (;;) {      ioctl(KVM\_RUN)          switch (exit\_reason) {          case KVM\_EXIT\_IO:          case KVM\_EXIT\_HLT:      }  } |

 KVM内核模块本身只能提供CPU和内存的虚拟化，所以它必须结合QEMU才能构成一个完成的虚拟化技术，这就是下面要说的qemu-kvm。

**qemu-kvm**

Qemu将KVM整合进来，通过ioctl调用/dev/kvm接口，将有关CPU指令的部分交由内核模块来做。kvm负责cpu虚拟化+内存虚拟化，实现了cpu和内存的虚拟化，但kvm不能模拟其他设备。qemu模拟IO设备（网卡，磁盘等），kvm加上qemu之后就能实现真正意义上服务器虚拟化。因为用到了上面两个东西，所以称之为qemu-kvm。

Qemu模拟其他的硬件，如Network, Disk，同样会影响这些设备的性能，于是又产生了pass through半虚拟化设备virtio\_blk, virtio\_net，提高设备性能。

[](http://s3.51cto.com/wyfs02/M01/6F/77/wKiom1WdDc2CEwy6AAGPf4VzQao172.jpg)

**Xen**

Xen是第一类运行在裸机上的虚拟化管理程序(Hypervisor)。它支持全虚拟化和半虚拟化,Xen支持hypervisor和虚拟机互相通讯，而且提供在所有Linux版本上的免费产品，包括Red Hat Enterprise Linux和SUSE Linux Enterprise Server。Xen最重要的优势在于半虚拟化，此外未经修改的操作系统也可以直接在xen上运行(如Windows)，能让虚拟机有效运行而不需要仿真，因此虚拟机能感知到hypervisor，而不需要模拟虚拟硬件，从而能实现高性能。

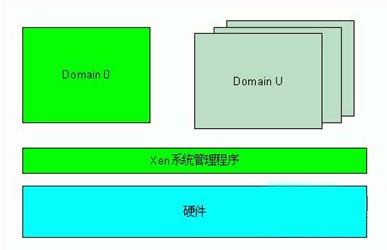
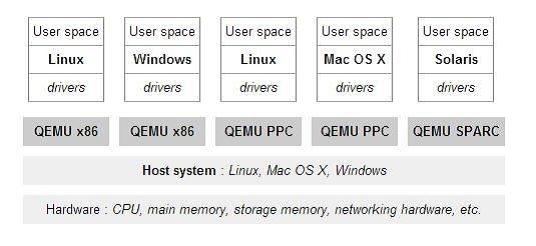
[](https://huangwei.me/wiki/image/Xen%E8%99%9A%E6%8B%9F%E5%8C%96%E5%B9%B3%E5%8F%B0%E6%9E%B6%E6%9E%84.jpg)

图4 Xen虚拟化平台架构

**QEMU**

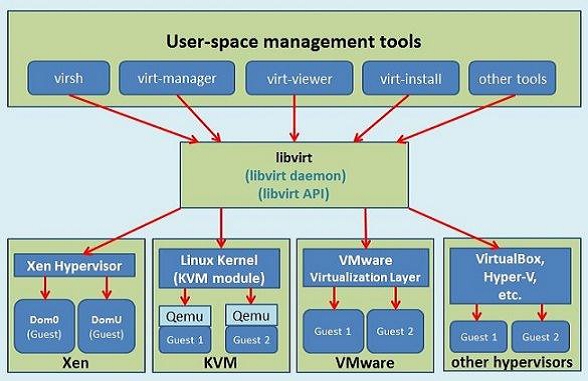
QEMU是一套由Fabrice Bellard所编写的模拟处理器的自由软件。它与Bochs，PearPC近似，但其具有某些后两者所不具备的特性，如高速度及跨平台的特性。经由kqemu这个开源的加速器，QEMU能模拟至接近真实电脑的速度。

Qemu是一个模拟器，它向Guest OS模拟CPU和其他硬件，Guest OS认为自己和硬件直接打交道，其实是同Qemu模拟出来的硬件打交道，Qemu将这些指令转译给真正的硬件。

由于所有的指令都要从Qemu里面过一手，因而性能较差[。[](http://s3.51cto.com/wyfs02/M00/6F/77/wKiom1WdDYyjiVZiAAECBtAEQ5E590.jpg)](http://s3.51cto.com/wyfs02/M00/6F/77/wKiom1WdDYyjiVZiAAECBtAEQ5E590.jpg" \t "_blank)

libvirt

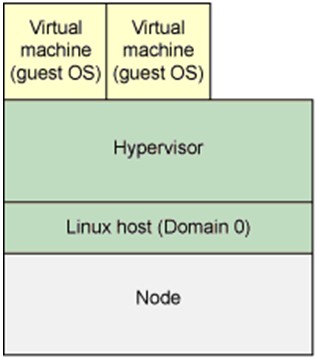
libvirt是目前使用最为广泛的对KVM虚拟机进行管理的工具和API。Libvirtd是一个daemon进程，可以被本地的virsh调用，也可以被远程的virsh调用，Libvirtd调用qemu-kvm操作虚拟机。

[](http://s3.51cto.com/wyfs02/M02/6F/74/wKioL1WdD72RRy8mAAIuDm6sVAY591.jpg)

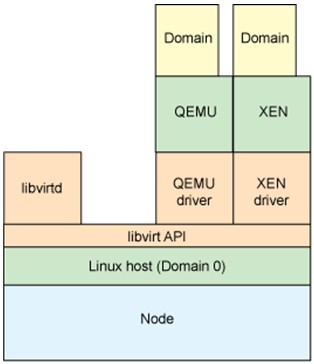
参考：http://blog.csdn.net/gaoxingnengjisuan/article/details/9674315

**Libvirt体系结构**

    没有使用libvirt的虚拟机管理方式如下图所示：

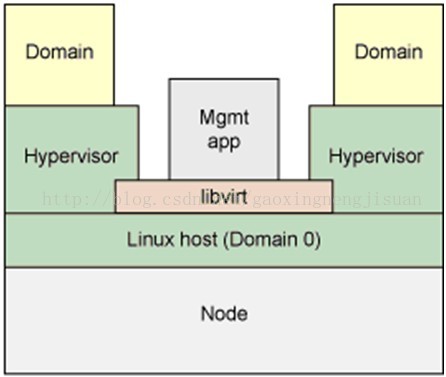


    为支持各种虚拟机监控程序的可扩展性，libvirt实施一种基于驱动程序的架构，该架构允许一种通用的API以通用方式为大量潜在的虚拟机监控程序提供服务。下图展示了libvirt API与相关驱动程序的层次结构。这里也需要注意，libvirtd提供从远程应用程序访问本地域的方式。

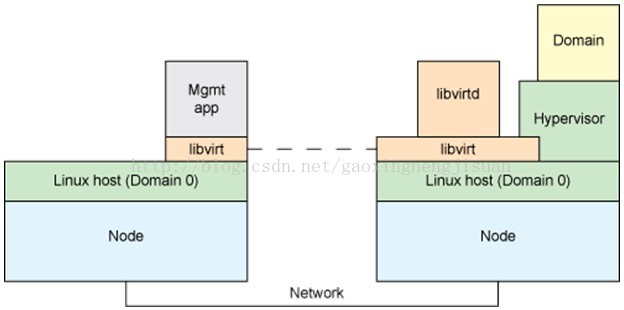


    libvirt的控制方式有两种：

    1）管理应用程序和域位于同一节点上。管理应用程序通过libvirt工作，以控制本地域。



    2）管理应用程序和域位于不同节点上。该模式使用一种运行于远程节点上、名为libvirtd的特殊守护进程。当在新节点上安装libvirt时该程序会自动启动，且可自动确定本地虚拟机监控程序并为其安装驱动程序。该管理应用程序通过一种通用协议从本地libvirt连接到远程libvirtd。



**KVM和QEMU的关系**

准确来说，KVM是Linux kernel的一个模块。可以用命令modprobe去加载KVM模块。加载了模块后，才能进一步通过其他工具创建虚拟机。但仅有KVM模块是 远远不够的，因为用户无法直接控制内核模块去作事情,你还必须有一个运行在用户空间的工具才行。这个用户空间的工具，kvm开发者选择了已经成型的开源虚拟化软件 QEMU。说起来QEMU也是一个虚拟化软件。它的特点是可虚拟不同的CPU。比如说在x86的CPU上可虚拟一个Power的CPU，并可利用它编译出可运行在Power上的程序。KVM使用了QEMU的一部分，并稍加改造，就成了可控制KVM的用户空间工具了。所以你会看到，官方提供的KVM下载有两大部分(qemu和kvm)三个文件(KVM模块、QEMU工具以及二者的合集)。也就是说，你可以只升级KVM模块，也可以只升级QEMU工具。这就是KVM和QEMU 的关系。

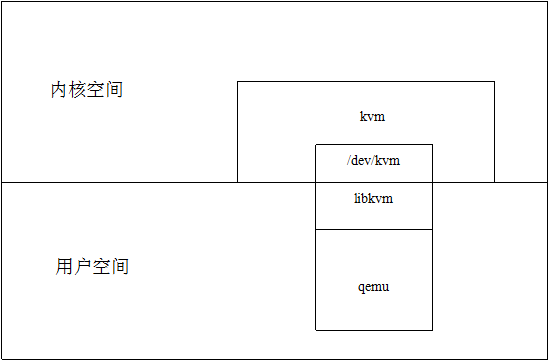
[](https://huangwei.me/wiki/image/KVM%E5%92%8CQEMU%E5%85%B3%E7%B3%BB.png)

图5 KVM和QEMU关系

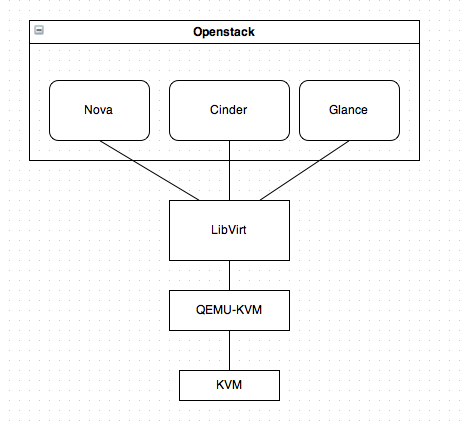
**openstack, kvm, qemu-kvm以及libvirt之间的关系**

KVM是最底层的hypervisor，它是用来模拟CPU的运行，它缺少了对network和周边I/O的支持，所以我们是没法直接用它的。

QEMU-KVM就是一个完整的模拟器，它是构建基于KVM上面的，它提供了完整的网络和I/O支持。

Openstack不会直接控制qemu-kvm，它会用一个叫libvirt的库去间接控制qemu-kvm。libvirt提供了跨VM平台的功能，它可以控制除了QEMU之外的模拟器，包括vmware, virtualbox， xen等等。

所以为了openstack的跨VM性，所以openstack只会用libvirt而不直接用qemu-kvm。libvirt还提供了一些高级的功能，例如pool/vol管理。

[](https://huangwei.me/wiki/image/openstack_libvirt_qemu_kvm.png)

原文：https://huangwei.me/wiki/tech\_cloud\_kvm\_qemu\_libvirt\_openstack.html