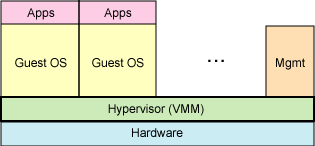
1. **全虚拟化**

全虚拟化也称原始虚拟化技术，该模型使用虚拟机协调guest操作系统和原始硬件，VMM在guest操作系统和硬件之间用于协调工作。现有的guest操作系统不需要进行修改就可以运行在这样的虚拟平台上。

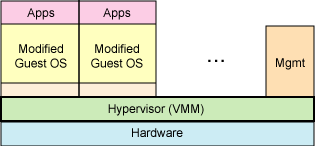


图一 全虚拟化模型

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

1. **半虚拟化**

半虚拟化是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。半虚拟化需要guest操作系统做一些修改，使操作系统意识到自己处于虚拟化环境之中，它提供了与原操作系统相近的性能。



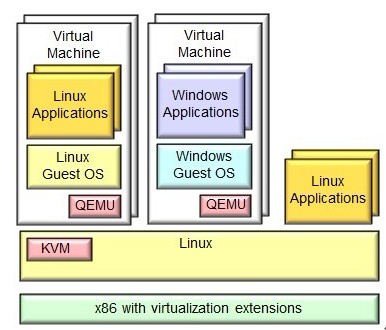
图二 半虚拟化模型

1. **KVM**

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

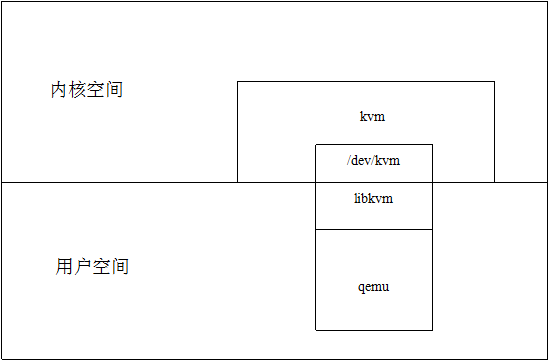
KVM是一个独特的管理程序，通过将KVM作为一个内核模块实现，在虚拟环境下Linux内核集成管理程序将其作为一个可加载的模块，可以简化程序和提升性能。在这种模式下，每个虚拟机都是一个常规的Linux调度程序进行调度。

需要注意的是，KVM本身不执行任何模拟，需要用户空间程序QEMU通过/dev/kvm接口设置一个虚拟机的地址空间，向它提供模拟的I/O设备，并将它的视频显示映射回宿主机的显示屏去显示。



图三 KVM模型

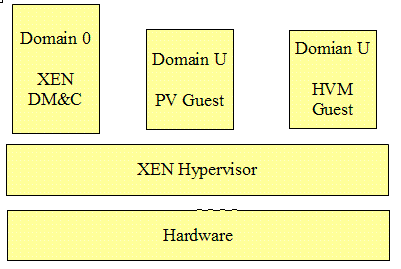
准确来说，KVM是Linux内核的一个模块，可以用命令modprobe去加载KVM模块。加载了该模块后才能进一步通过其他工具创建虚拟机。由于用户不可能直接通过控制内核模块去做事情，还需要一个运行在用户空间的工具才行，这个工具就是之前说的QEMU。两者的关系如图所示：



1. **XEN**

Xen是一个直接在系统硬件上运行的虚拟机管理程序。它支持全虚拟化和半虚拟化，支持Hypervisor和虚拟机之间相互通信，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。其优势在于半虚拟化，同时未经修改的操作系统比如Windows也可以直接在Xen上运行，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，从而实现高性能。

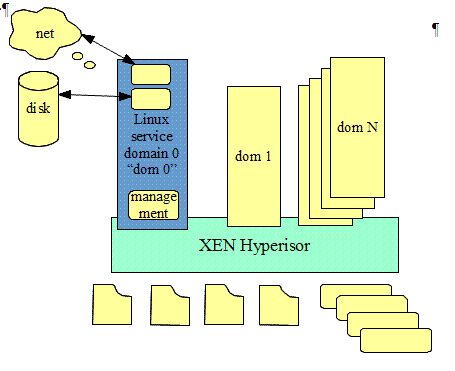
Xen虚拟机环境主要由以下几部分组成：Xen Hypervisor、Domain 0-Domain Management and Control（Xen DM&C）、Domain U Guest（Dom U）（A.PV Guest B.HVM Guest）。下图显示了三部分之间的关系：



Xen Hypervisor：介于操作系统和硬件之间的一个软件描述层。它负责在各个虚拟机之间进行CPU调度和内存分配。它不会处理网络、存储设备、视频以及其他I/O。

Domain 0：是一个修改后的Linux内核，是唯一运行在Xen Hypervisor之上的虚拟机，它拥有访问物理I/O资源的权限，同时和系统上运行的其它虚拟机进行交互。Domain 0需要在其它Domain启动之前启动。

Domain U：运行在Xen Hypervisor上的所有半虚拟化虚拟机被称为“Domain U PV Guests”，其上运行着被修改过内核的操作系统，如 Linux、Solaris、FreeBSD 等其它 UNIX 操作系统。所有的全虚拟化虚拟机被称为“Domain U HVM Guests”，其上运行着不用修改内核的操作系统，如 Windows 等。



图五Xen模型

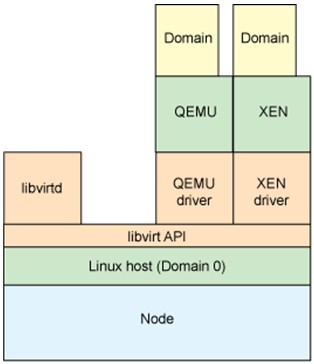
1. **Libvirt**

Libvirt是管理虚拟机和其他虚拟化功能，比如存储管理、网络管理的软件集合。它主要包括三部分：一个通用的API库、一个守护进程libvirtd和一个命令行工具virsh。其主要目标是为各种虚拟机化工具提供一套方便、可靠的编程接口，用单一的方式管理多种不同的虚拟化提供方式。

Libvirt支持多种虚拟化方案，既支持包括KVM、QEMU、Xen、VMware、VirtualBox等在内的平台虚拟化方案，又支持OpenVZ、LXC等Linux容器虚拟化系统，还支持用户态Linux（UML）的虚拟化。

Libvirt本身提供一套较为稳定的C语言应用程序接口，且包含对python的直接支持，目前还还支持大量语言绑定，包括ruby、java、perl、php以及OCaml等高级语言。Libvirt还提供了为基于AMQP的消息系统（如ApacheQpid）提供QMF代理，可以让云计算管理系统中宿主机和虚拟机、虚拟机和虚拟机之间的消息通信变得更易于实现。

Libvirt对多种不同的Hypervisor的支持是通过一种基于驱动程序的架构来实现的。其对不同的Hypervisor提供了不同的驱动：对Xen有Xen的驱动，对QEMU/KVM有QEMU驱动，对VMware有VMware驱动，其体系架构如图所示。



Libvirt体系架构图

现对其中的一些概念做出解释：

节点（node）：是一个物理机器，上面可能运行多个虚拟机。其中Hypervisor和Domain都运行在节点之上。

Hypervisor：也称为VMM，如KVM、Xen|VMware以及Hyper-v等，是虚拟化中的一个底层软件层，它可以虚拟化一个节点让其运行多个虚拟机。

域（Domain）：在Hypervisor上运行的一个客户机操作系统实例。域被称为实例、guest OS或者虚拟机。

Libvirt的管理功能既包括本地管理，也包括远程管理，其管理功能如下：

（1）域的管理：包括对节点上的域的各个生命周期的管理，如启动、停止、暂停、保存、回复和动态迁移。也包括对多种设备类型的热插拔操作，包括磁盘、网卡、内存和CPU，当然不同的Hypervisor对这些热插拔的支持程度也有所不同。

（2）远程节点管理：只要物理节点上运行了libvirtd这个守护进程，远程的管理程序就可以连接到该节点进程管理操作，经过认证和授权之后，所有的libvirt功能都可以被访问和使用。Libvirt支持多种分网络远程传输类型，如SSH、TCP套接字、Unix domain socket、支持TLS的加密传输等。

（3）存储管理：任何运行了libvirtd守护进程的主机，都可以通过libvirt来管理不同的存储，如创建不同格式的虚拟机镜像（qcow2、raw、去的、vmdk等）、挂载NFS共享存储系统、查看现有的LVM卷组、创建新的LVM卷组和逻辑卷、对磁盘设备分区、挂载iSCSI共享存储等等。当然同样支持对存储管理也是支持远程管理的。

（4）网络管理：任何运行了libvirtd守护进程的主机，都可以通过libvirt来管理物理和逻辑网络接口。包括列出现有的网络接口卡，配置网络接口，创建虚拟网络接口。网络接口的桥接，vlan管理，NAT网络设置，为虚拟机分配虚拟网络接口等。

（5）提供一个稳定、可靠、高效的应用程序接口，以便完成前面4种管理功能。

Libvirt中包含了virsh这一命令行工具，virsh用来管理虚拟机和Hypervisor的命令行工具，通过调用libvirt API来实现虚拟化的管理。Virsh是完全在命令行文本模式下运行的用户态工具，它是系统管理员通过脚本程序实现虚拟化自动部署和管理的理想工具之一。其有两种工作模式：交互模式和非交互模式。其中，交互模式是连接到相应的Hypervisor上，然后输入一个命令得到一个返回结构，直到用户使用quit命令退出连接。非交互模式是指直接在命令行中一个建立连接的URI之后添加需要执行的一个或者多个命令，执行完成后将命令的输出结果返回到当前终端上，然后自动断开连接。

Libvirt API 可划分为8个部分：连接Hypervisor相关的API、域管理的API、节点管理的API、网络管理的API、存储卷管理的API、存储池管理的API、事件管理的API以及数据流管理的API。

（1）连接Hypervisor相关的API：以virConnect开头的一系列函数。只有在与Hypervisor建立连接之后，才能进行虚拟机管理操作，故连接Hypervisor的API是其他API使用的前提条件。通过调用virConnectOpen函数可以建立个连接，其返回值是一个virConnectPtr对象，该对象就代表到Hypervisor的一个连接；如果连接出错则返回null。

1）连接函数：libvirt库为连接到Hypervisor提供了三种不同的函数，三个函数之间的区别在于它们提供的验证方式以及由此产生的授权方法。

virConnectPtr virConnectOpen（const char \*name）：以实现完全的读写访问，不包含任何认证机制，而仅仅提供一种连接。

virConnectPtr virConnectOpenReadOnly(const char \*name)：打开一个只读访问连接。

virConnectPtr virConnectOpenAuth（const char \*name，virConnectAuthPtr auth，int flags）：提供一种基于认证机制的连接。

2）关闭连接：virConnectClose，当不需要连接的时候，必须调用virConnectClose来断开连接。

3）URI格式：libvirt使用统一资源标识符（URI）来识别管理连接。连接本地和远程的Hypervisor都是通过URI来进行处理。

本地URI：driver://system(session) driver+unix://system(session)

远程URI：driver[+transport]://[username@][hostname][:port]/[path][?extraparameters]

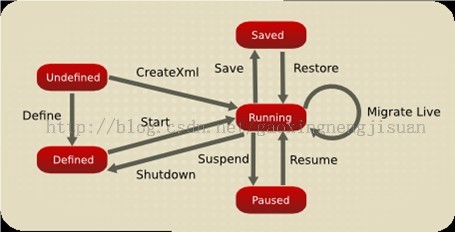
4）获取虚拟机功能信息的API： virConnectGetCapabilities的调用能够用于获取虚拟主机的功能信息。返回虚拟机监控程序和驱动程序的功能描述。它包含一个指针，如果调用成功，将返回一个字符串，其中包括了描述功能信息的XML文件。

5）获取虚拟机信息的API：包括主机最大支持的CPU个数等。virConnectGetHostname和virConnectGetMaVcpus等。

6）错误处理API：libvirt error API将会在一个正常的libvirt API调用返回错误标志的时候。提供关于引发错误的非常详细的信息。

（2）域管理API：以virDomain开头的一系列函数，要管理域首先要获取virDomainPtr这个域对象，然后才能对域进行操作。获取域对象的方式有三种：virDomainLookupByID, virDomainLookupByName 和virDomainLookupByUUID，三者都用一个连接对象作为第一个输入参数，以及域标识符作为其他的输入参数。其中域的标识符包括ID、name、UUID。

域生命周期的控制，libvirt能够控制域的整个生命周期。域可以在它的整个生命周期的几个状态之间进行转换：undefined-没有任何定义或者建立的域的状态；defined-域已经定义但没有运行的状态，这种状态可以描述为停止；running-已经定义并且运行在一个Hypervisor上面的域状态；paused是一个域系统从运行状态转换为暂停状态。它的内存镜像可以被暂时地存储，它可以恢复到运行状态。



（3）节点管理的API：以virNode开头的一系列函数。域运行在物理节点之上，libvirt也提供了对节点进行信息查询和控制的功能。节点管理的多数函数都需要使用一个连接Hypervisor的对象作为其中一个传入参数，以便可以查询或者修改到该连接上的节点的信息。virNodeGetInfo函数是获取节点的物理硬件信息，virNodeGetCPUStats函数可以获取节点上各个CPU的使用统计信息。

（4）网络管理的API：以virNetwork开头的一系列函数和部分以virInterface开头的函数。Libvirt首先需要创建virNetworkPtr对象，然后才能查询或控制虚拟网络。一些查询网络相关信息的函数，如virNetworkGetName函数可以获取网络的名称，virNetworkGetBridgeName函数可以获取该网络中网桥的名称。另外，virInterfaceCreate、virInterfaceFree、virInterfaceDestroy、virInterfaceGetName、virInterfaceIsActive等函数可以用于创建、释放和销毁网络接口以及查询网络接口的名称和激活状态。

（5）存储卷管理的API：以virStorageVol开头的一系列函数。Libvirt对存储卷的管理主要是针对域的镜像文件的管理，这些镜像文件的格式可能是raw、qcow2、vmdk、qed等各种格式。Libvirt对存储卷的管理，首先需要创建virStorageVolPtr这个存储卷对象，然后才能对其进行查询或控制操作。有三个函数可以通过不同的方式来获取存储卷对象，包括virStorageVolLookupBykey、virStorageVolLookupByName以及 virStorageVolLookupByPath。

（6）存储池管理的API：以virStoragePool开头的一系列函数。Libvirt对存储池的管理，包括对本地的基本文件系统、普通网络共享文件系统、iSCSI共享文件系统、LVM分区等的管理。Libvirt需要基于virStoragePoolPtr这一存储池对象才能进行查询和控制操作。获得对象的方法可以有virStoragePoolLookupByName和virStoragePoolLookupByVolume来获取。

（7）事件管理的API：以virEvent开头的一系列函数。在使用该机制注册之后，可以在发生特定的事件（如域的启动、暂停、回复、停止等）时得到自己定义的一些通知。

（8）数据流管理的API：以virStream开头的一系列函数。