libvirt的常用api详使用方法

[1相关知识 3](#_Toc28492)

[1.1 虚拟化的方式 3](#_Toc31383)

[1.2 libvirt主要类的介绍 5](#_Toc7070)

[2 API介绍 6](#_Toc707)

[2.1管理程序(hypervisor)相关 6](#_Toc30171)

[2.2虚拟机(domain)相关 7](#_Toc6241)

[2.3 网络(network)相关 10](#_Toc7258)

[2.4错误消息相关 12](#_Toc19880)

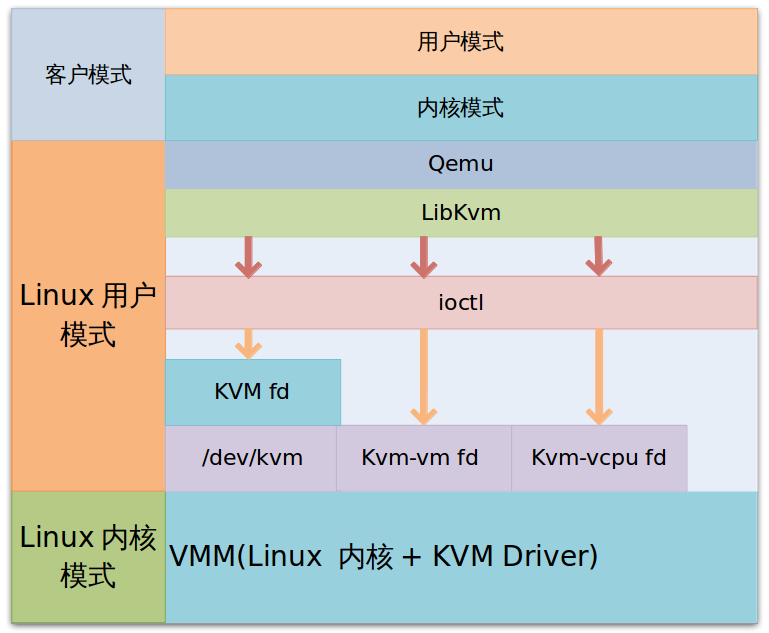
# 1相关知识

## 1.1虚拟化的方式

**1.全虚拟化**

Hypervisor(即vmm虚拟机管理程序）会抽象出虚拟机所用的全部硬件资源（如CPU、磁盘、网卡）。

kvm和qemu介绍

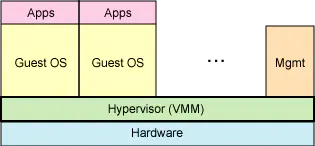


kvm是内核的一部分（内核代码的一部分），它使得Linux在用户模式和内核模式的基础上增加了客户模式，虚拟机就是运行在客户模式下的。它同样具有内核模式和用户模式。kvm负责虚拟机的创建，虚拟内存的分配，VCPU寄存器的读写和VCPU的运行（模拟内存和CPU).

qemu是一套独立的模拟计算机的软件，它是用户模式下的一个进程。通过qemu，用户可以和虚拟机交互了。它通过特定的接口调用kvm模块提供的功能。QEMU通过KVM模块提供的系统调用接口进行内核设置，由KVM模块负责将虚拟机置于处理器的特殊模式运行(也就是说客户模式里的内核模式是通过kvm来实现的）QEMU使用了KVM模块的虚拟化功能，为自己的虚拟机提供硬件虚拟化加速以提高虚拟机的性能。

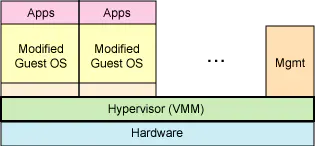
qemu是用户模式下的一个计算机模拟软件，kvm为这个软件提供加速功能。

使用vmm一个中间层来管理客户机



**2.半虚拟化**

半虚拟化是一部分使用虚拟化设备（除了CPU和内存）它使用Hypervisor(虚拟机管理程序)分享存取底层的硬件。CPU和内存是直接使用硬件，因此速度非常快。但是因为直接使用硬件，所以客户机的操作系统的内核代码需要进行修改（不然可能会造成对物理机的误操作等情况）



**3.总结**

简单来说，全虚拟化就是所有虚拟化内容都走中间层（HyPervisor)，半虚拟化就是有些部分走中间层，有些部分（比如CPU和内存）它直接使用硬件，所以就是一部分的虚拟化，即半虚拟化。

**4.说明**

ComputeService所用的模拟方式是kvm/qemu虚拟化方式，libvirt用作管理平台。

## 1.2 libvirt中主要类的介绍

virConnectPtr

它是和hypervisor的连接，它通过virConnectOpen来建立连接。

virDomainPtr

这就代表某一个虚拟机（或者客户机Guest OS)，因为虚拟机是运行在hypervisor的基础上的。

virNetWorkPtr

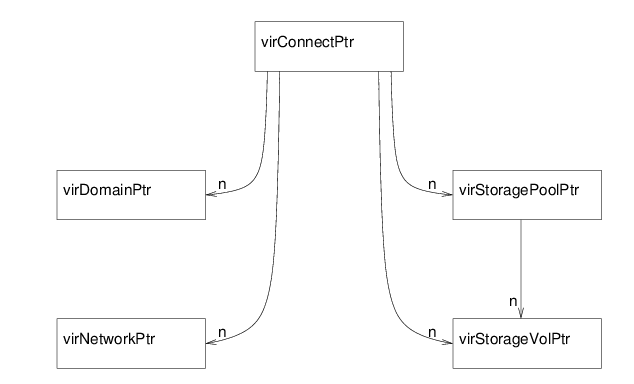
代表一个网络。

virStorageVolPtr

代表一个虚拟机的一个存储卷（分区），这个存储卷通常是一个块设备（磁盘就是块设备）。

virStoragePoolPtr

代表一个存储池，是一个逻辑的区域，用来分配和存储存储卷。



# 2.API介绍

## 2.1 管理程序(hypervisor)相关

**1、virConnectPtr virConnectOpen (const char \* name)**

这个name是一个URI(不是URL),用来表示哪个一个hypervisor。

如果name为空

1.查看环境变量LIBVIRT\_DEFAULT\_URI是否有值。

2.查看客户端的配置文件中的uri\_default参数。

3.搜索所有的hypervisor，直到有一个连接成功。

URI可以是

1. xen本地

xen:///system : 代表连接一个本地的Xen虚拟机管理程序。（hypevisor)

2.qemu本地

不能直接连接qemu，需要连接到它的守护进程（daemon)—名字是libvirtd的进程。libvirtd进程需要root权限，并且需要开机自启动，它用来管理qemu的实例。

qemu:///system 连接本地的系统模式守护进程。

qemu:///session 连接本地的会话模式守护进程。

**2、int virConnectClose (virConnectPtr conn)**

参数是一个指向虚拟机管理程序连接的指针。

如果还存在指向虚拟机管理程序连接的指针的引用，那么会返回一个正值（但返回值并不是总的引用计数），如果返回0代表连接关闭，内存释放。返回-1代表关闭失败。

**3、int virConnectRegisterCloseCallback ( virConnectPtr conn, virConnectCloseFunc cb, void \* opaque, virFreeCallback freecb)**

这个函数是用在连接被关闭的回调函数，只要连接关闭了，这个回调函数就会被调用。

conn:指向连接的指针

cb:连接断开的回调函数

原型如下：typedef void (\*virConnectCloseFunc ) (virConnectPtr conn, int reason, void\* opaque)

conn:连接。

reson:连接被关闭的原因（是枚举virConnectCloseReason中的一个，具体哪些原因不展开了）。

opaque:用户传给回调函数的数据。

opaque:用户传给回调函数的数据。

freecb:用来释放opaque的回调函数，它执行在主回调(cb)注销的时候

原型如下：typedef void (\*virFreeCallback) (void \* opaque)

opague:用户传给回调函数的数据。

返回值：0代表成功，-1代表失败。

## 2.2 虚拟机(domain)相关

**1、virDomainPtr virDomainDefineXML (virConnectPtr conn, const char \* xml)**

定义一个虚拟机，但不打开它。直到调用virDomainUndefine()方法，这个虚拟机的定义就不存在了 ，否则它会一直存在。另外只有有相同uuid和名称的定义会被这次覆盖掉。这是创建虚拟机的第一步。

总结来说，就是根据xml配置文件得到一个虚拟机指针。

conn : 和虚拟机管理程序(hypervisor)连接的指针。

xml : 一个虚拟机的xml文件的字符流形式。

返回值：如果错误返回NULL, 正确返回一个指针。

**2、int virDomainUndefine (virDomainPtr domain)**

取消一个虚拟机。如果虚拟机正在运行，那么它会转变成瞬时态（短暂态），如果它未运行，直接删除配置项。

domain : 定义好的虚拟机的指针。

返回值：0代表成功，-1代表失败。

**3、 int virDomainCreateWithFlags (virDomainPtr domain, unsigned int flags)**

用于启动一个定义好的虚拟机。如果调用成功，这个虚拟机会从定义态变到正在运行态（加入运行池）

如果flags设置为VIR\_DOMAIN\_NONE，表示按默认方式启动。

如果flags设置为VIR\_DOMAIN\_START\_PAUSED，或者虚拟机有一个请求暂停状态的托管保存映像（managed save image，它由virDomainManagedSave()创建），这两种情况，虚拟机会开始运行，但是CPU还是处于暂停状态，所以仍然不可用(可通过virDomainResume来启动CPU).如果不是这两种情况，虚拟机都会开始运行。

如果flags设置为VIR\_DOMAIN\_START\_AUTODESTROY，那么当virConnectPtr释放、客户程序崩溃，虚拟机断开和libvirted守护进程的连接的时候，即连接断开的时候，这个虚拟机就自动关机了。

如果flags设置了VIR\_DOMAIN\_START\_FORCE\_BOOT，重头开始启动。不管任何的代理保持映像。

domain: 代表一个虚拟机对象的指针。

flags：关闭的方式，详见virDomainCreateFlagss枚举，可以用|来连接多个启动方式。

返回值：0代表成功，-1代表失败

**4、int virDomainReboot (virDomainPtr domain, unsigned int flags)**

用于重启一个虚拟机，这个domain接下来还是可用的。但是hypevisor会检查xml里on\_reboot的相关设置，所以它可能会关机而不是重启。并且虚拟机可能会忽略这个请求。

domain: 代表一个虚拟机对象的指针。

flags:重启的方式，详见virDomainRebootFlagValues枚举。

返回值：0代表成功，-1代表失败。

**5、int virDomainShutdownFlags (virDomainPtr domain, unsigned int flags)**

用于关闭一个虚拟机，这个domain接下来还是可用的。但是hypevisor会检查xml里on\_reboot的相关设置，所以它可能会重启而不是关机。并且虚拟机可能会忽略这个请求。

domain: 代表一个虚拟机对象的指针

flags：关闭的方式，详见virDomainRebootFlagValues枚举

返回值：0代表成功，-1代表失败。

**6、int virDomainSuspend (virDomainPtr domain)**

让一个虚拟机处于挂起状态。这个进程已经不能使用CPU和I/O设备了，但在Hypevisror层面上的虚拟机的内存仍然是分配的。用virDomainResume()来解除挂起状态。如果虚拟机的状态是VIR\_DOMAIN\_PMSUSPENDED（即被虚拟机自身的电源管理挂起），这个函数无效。

domain : 定义好的虚拟机的指针

返回值：0代表成功，-1代表失败

**7、int virDomainResume (virDomainPtr domain)**

用来解除挂起状态。如果虚拟机的状态是VIR\_DOMAIN\_PMSUSPENDED（即被虚拟机自身的电源管理挂起），这个函数无效。

domain : 定义好的虚拟机的指针。

返回值：0代表成功，-1代表失败。

**8、int virDomainGetInfo (virDomainPtr domain, virDomainInfoPtr info)**

用来提取虚拟机信息，info是一个指向virDomainInfo结构体的指针。如果用来获取虚拟机的连接受限的话，只能提取一部分的信息。

domain: 定义好的虚拟机的指针。

info: 一个指向virDomainInfo结构体的指针。

返回值: 0代表成功，-1代表失败。

**9、int virDomainDestroy (virDomainPtr domain)**

销毁一个虚拟机对象，但不销毁虚拟机关联的指针。如果运行的虚拟机没有关闭，则关闭，并且将所有资源返回给虚拟机管理程序（hypervisor)

它的策略是先发一个SIGTERM信号，然后如果过了超时时间虚拟机还在运行的话，就发一个SIGKILL信号，这就会强制关机。

domain : 定义好的虚拟机的指针。

返回值：0代表成功，-1代表失败。

**10、 int virDomainDestroyFlags (virDomainPtr domain, unsigned int flags)**

它的不同是带一个标志位flags，相较于Destroy的强制关机而言，它如果设置了VIR\_DOMAIN\_DESTROY\_GRACEFUL，如果超时没有关机的话，它会返回一个错误，而不是强制关机。

domain : 定义好的虚拟机的指针。

flags：标志位，详细见宏virDomainDestroyFlagsValues。

返回值：0代表成功，-1代表失败。

**11、int virDomainFree (virDomainPtr domain)**

释放虚拟机对象，运行的虚拟机将继续保持运行，但它的数据结构释放以后，不应再使用。

它和virDomainDestroy的区别是Destroy是把资源交还给Hypevisor，而Free是彻底释放虚拟机对象，数据结构也会删除。在计算结点中，这个函数用在Destroy之后，做进一步的释放。

domain : 定义好的虚拟机的指针。

返回值：0代表成功，-1代表失败。

## 2.3 网络(network)相关

**1、virNetworkPtr virNetworkDefineXML (virConnectPtr conn, const char \* xml)**

跟定义一个虚拟机一样，只是它是用来定义一个虚拟网络。或者是根据xml来修改现存的永久的虚拟网络。

用virNetworkFree来释放资源（不再使用了）。

conn : 和虚拟机管理程序(hypervisor)连接的指针。

xml : 一个虚拟机的xml文件的字符流形式。

返回值：如果错误返回NULL, 正确返回一个指针。

**2、int virNetworkUndefine (virNetworkPtr network)**

取消网络的定义，如果虚拟网络正在运行，它不会关闭它。

netwrok : 网络对象。

返回值：0代表成功，-1代表失败。

**3、int virNetworkCreate (virNetworkPtr network)**

创建并运行一个已经定义好的网络对象。让它从已经定义好的状态变为运行态（加到运行池去）

netwrok : 网络对象。

返回值：0代表成功，-1代表失败。

**4、int virNetworkSetAutostart (virNetworkPtr network, int autostart)**

设置虚拟网络为开机自启动（物理主机开机自启动）。

netwrok : 网络对象。

autostart: 0代表开机自启动，1代表开机不自启动。

返回值： 0代表成功，-1代表失败。

**5、const char \* virNetworkGetName ( virNetworkPtr network)**

获取网络的公共名称。

netwrok : 网络对象。

返回值：返回一个名称的指针，或者NULL,这个指针不需要释放，它的生命周期跟网络对象相同。

**6、 int virNetworkIsActive (virNetworkPtr net)**

判断网络对象是否正在运行

netwrok : 网络对象。

返回值：1代表运行，0代表未运行， -1代表错误。

**7、int virNetworkDestroy (virNetworkPtr network)**

销毁一个网络对象，如果正在运行就关闭它。它的作用是把所有使用的资源交还给Hypevisor。

netwrok: 网络对象。

返回值：0代表成功，-1代表失败。

## 2.4 错误消息相关

**1、 const char \* virGetLastErrorMessage (void)**

返回这个线程最近的一个错误消息（如果什么都没有设置的话，就是一个通用消息）