1.3-1虚拟化知识（虚拟机、虚拟化原理）

2021年5月18日 10:38

虚拟技术

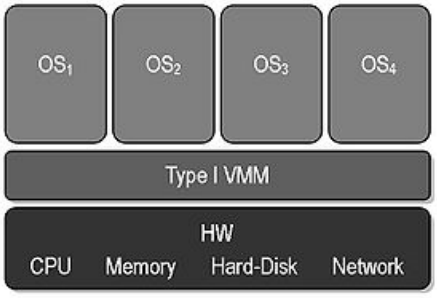
虚拟技术指的是一种资源管理技术，将计算机的各种实体资源予以抽象、转换后分割、组合成为一个或多个电脑配置环境，从而使得这些实体资源可以更好的被利用。

虚拟技术按照虚拟的对象可以分为以下几类：
硬件虚拟化、**虚拟机**（详见下节）、虚拟内存、存储虚拟化、网络虚拟化（VPN）、桌面虚拟化、数据库虚拟化、数据虚拟化等等。

虚拟机的概念、分类

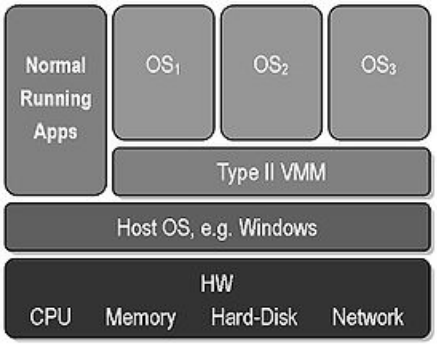
虚拟机是一种特殊的**软件**，通过软件模拟具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统，提供物理计算机的功能。

VMM（virtual machine monitor）虚拟机监控器，又称**Hypervisor**，指用来创建与运行、维护虚拟机的软件、固件或硬件。
VMM主要有两种类型，
类型 I：裸机 VMM：虚拟机管理程序直接运行在宿主机的硬件上来控制硬件和管理虚拟机操作系统。其特点是需要硬件支持，虚拟机监视器就是主操作系统，运行效率较高



例：VMware ESX服务器版本、Virtual PC 2005、KVM

类型 II：托管VMM，虚拟机管理程序运行在操作系统上，就像其他计算机程序那样运行。其特点是虚拟机监视器作为应用程序运行在主操作系统环境内，运行效率一般较类型 I 低



例：VMware workstation、Xen 3.0 以前版本、Virtual PC 2004

虚拟机的分类

按照用途分类：
1.系统虚拟机（平台虚拟化），提供一个可以运行完整操作系统的完整系统平台（virtualBox、VMware Workstation）

2.程序虚拟机（应用程序虚拟化），用来在与平台无关的环境中执行计算机程序。（JVM、Wine）

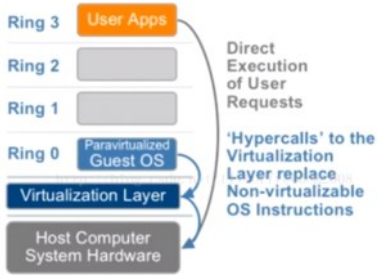
按照实现方式分类：

1.**全虚拟化**：

- VMM完全模拟所有硬件设备，虚拟机的操作系统好像独自占用了所有的计算机资源，虚拟机操作系统无需修改就能运行（操作系统“不知道”自己运行在虚拟机中）。这是目前最成熟和常见的虚拟机类型，此类型下的虚拟机既有“裸机”型的，又有“托管”型的。
- 全虚拟化的虚拟机操作系统在执行特权指令时会触发异常，这个异常由VMM捕获并翻译、模拟，再返回到虚拟机操作系统中，虚拟机操作系统认为自己的特权指令工作正常，继续执行。

2.**半虚拟化**：又叫做操作系统辅助虚拟化、超虚拟化。

- VMM需要在操作系统的协助下才能对特权指令进行虚拟化。需要对虚拟机的操作系统内核进行修改，（操作系统“知道”自己运行在虚拟机中。）
- 在操作系统内核中替换掉了不能虚拟化的指令，通过超级调用（Hypercall）直接和VMM通讯，省去了全虚拟化中的捕获和模拟过程，提高运行效率。（因此无法虚拟化像windows这样的非开源系统）



图源：51CTO学院

3.硬件辅助虚拟化：

VMM在硬件的支持下完成对硬件资源的虚拟化。随着一系列cpu厂商开始支持虚拟化技术，新的虚拟化技术如Intel-VT和AMD-V被提出，引入了这种技术的CPU带有特别的指令集来控制虚拟过程，VMM和Guest OS分别运行在不同的模式下，由硬件支持模式切换。在硬件辅助下的全虚拟化技术性能越来越逼近半虚拟化。

	全虚拟化	**硬件辅助虚拟化**	**半虚拟化**
实现技术	直接执行，特权指令由VMM捕获并模拟	遇到特权指令转到root模式执行	Hypercall
客户操作系统修改/兼容性	无需修改客户操作系统，最佳兼容性	无需修改客户操作系统，最佳兼容性	客户操作系统需要修改来支持hypercall，因此它不能运行在物理硬件本身或其他的hypervisor上，兼容性差，不支持Windows
性能	差	全虚拟化下，CPU需要在两种模式之间切换，带来性能开销；但是，其性能在逐渐逼近半虚拟化。	好。半虚拟化下CPU性能开销几乎为0，虚机的性能接近于物理机。
应用厂商	VMware Workstation/QEMU/Virtual PC	VMware ESXi/Microsoft Hyper-V/Xen 3.0/KVM	Xen

按照VMM的类型分类：

1.虚拟机直接运行在系统硬件上，创建硬件全仿真实例（“裸机”型）

2.虚拟机运行在操作系统上，创建硬件全仿真实例，（“托管”型）

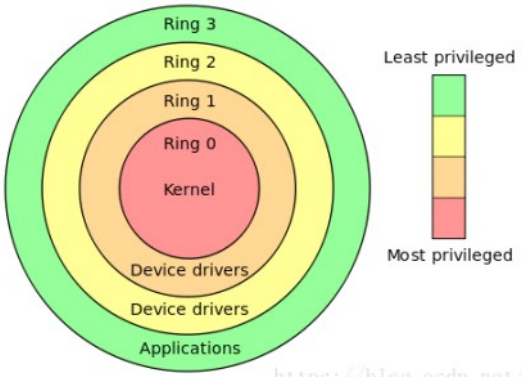
参考链接

https://zh.wikipedia.org/wiki/%E8%99%9B%E6%93%AC%E6%A9%9F%E5%99%A8
https://hurray0.com/menu/50/
https://edu.51cto.com/center/course/lesson/index?id=153444
https://wenku.baidu.com/view/cdf1c7d7240c844769eaee9c.html

虚拟化原理简介

一、CPU指令环

以intel的cpu为例，cpu有4个特权级别，编号分别从RING0~3。RING0代表最高的特权级别，RING0代表最低的特权级别。操作系统运行在RING0中，而用户应用程序运行在RING3中。



内存、I/O端口的控制以及一系列特殊及其指令执行的能力受到这一系列特权级别的限制，有一些机器指令只能在RING0执行（保留给内核使用），如果在RING0之外执行这些指令就会产生错误信息。

如果应用程序要执行特权指令，需要通过系统调用，让CPU运行级别从RING3切换到RING0，并跳转到系统调用对应的内核代码位置执行，执行完成之后再切换回原来的状态。

在早期虚拟化技术实现的时候，因为RING指令环遇到了麻烦，因为RING0不允许多个OS同时运行在上面，最早的解决办法是把虚拟机OS当做一个用户程序来运行。

二、BT技术

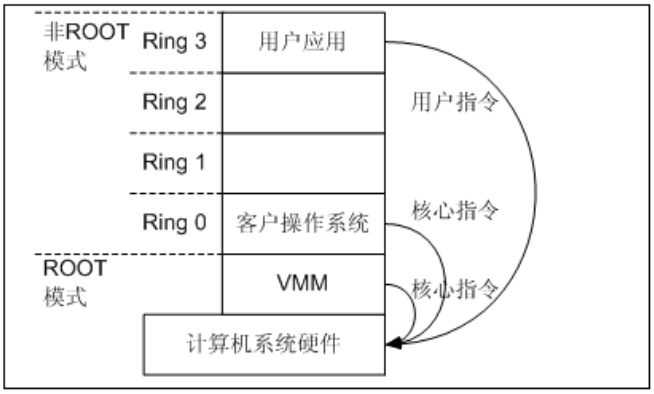
前面提到过，虚拟机实现方式有三种，全虚拟化，半虚拟化和硬件辅助虚拟化。其中的全虚拟化暴扣基于软件（或者说基于BT，二进制翻译技术）的全虚拟化，和硬件辅助实现的全虚拟化。

BT（二进制翻译）技术，是一种直接翻译可执行二进制程序的技术，该技术可以将各个虚拟机对特权指令的调用直接翻译成对主机的特权指令的调用。在此模式下，虚拟机的os内核运行在ring1，VMM运行在ring0，虚拟机os会在ring1上执行特权指令，从而会产生异常，该异常会被VMM捕获。VMM发现捕获到的是特权指令产生的异常之后就会通过BT技术对该指令进行模拟，从而实现特权指令的执行。虚拟机os中的非特权指令执行在ring3，不会被VMM拦截。

三、硬件辅助虚拟化

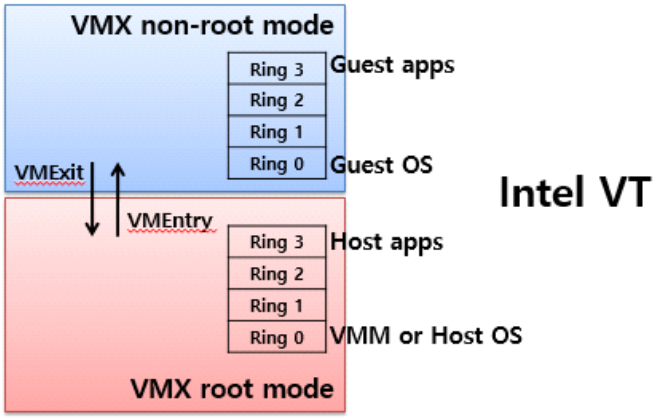
基于BT技术的虚拟化的一大问题是，频繁的对指令进行捕获和模拟带来大量的开销，虚拟机运行效率低下。为了解决这个问题，CPU厂商修改了CPU的硬件结构，从而帮助了虚拟化的实现。目前硬件辅助虚拟化技术主要有INTEL的Intel VT-X和AMD的AMD-V.

这种技术通过引入新的指令和运行模式，让VMM和虚拟机OS都运行在ring0之下，但是运行在不同模式之中。该模式下虚拟机OS的核心指令可以直接下达到硬件执行，不需要通过VMM，只有执行到特殊指令时，系统才会切换到VMM，让VMM处理特殊指令。



以intel VT-x技术为例，VT-x提供了2 个运行环境：根（Root）环境和非根（Non-root）环境。根环境专门为VMM准备，

非根环境做为一个受限环境用来运行多个虚拟机。，从根环境到非根环境叫**VMEntry**；从非根环境到根环境叫**VMExit**。VT-x定义了VMEntry操做，使CPU由根模式切换到非根模式，运行客户机操做系统指令。若在非根模式执行了敏感指令或发生了中断等，会执行VMExit操做，切换回根模式运行VMM。



AMD-V 在 AMD 传统的x86-64 基础上引入了“guest”操作模式。“guest”操作模式就是 CPU 在进入客操作系统运行时所处的模式。 “guest”操作模式为客操作系统设定了一个不一样于 VMM 的运行环境而不须要改变客操作系统已有的 4 个特权级机制，也就是说在“guest”模式下，客操作系统的内核仍然运行在 Ring 0， 用户程序仍然在 Ring 3。 裸机上的操作系统和 VMM 所在的操作模式依然和传统的 x86 中同样，且称之为“host”操做模式。 VMM 经过执行 VMRUN 指令使CPU 进入“guest”操作模式而执行客操作系统的代码； 客操做系统在运行时，遇到敏感指令或事件，硬件就执行 VMEXIT 行为，使 CPU 回到“host”模式而执行 VMM 的代码。
“guest”模式的意义在于其让客操做系统处于彻底不一样的运行环境，而不须要改变客操做系统的代码。“guest”模式的设立在系统中创建了一个比 Ring 0 更强的特权控制，即客操做系统的 Ring 0 特权必须让位于 VMM 的 Ring 0 特权。

关于虚拟化技术，这里了解的仍然很浅，进一步了解其中的原理可以阅读VMWARE等官方的技术白皮书。

参考链接：
https://blogs.oracle.com/ravello/nested-virtualization-with-binary-translation
http://www.javashuo.com/article/p-xnueqizf-dv.html
https://www.cnblogs.com/wangwangfei/p/13942235.html
https://www.cnblogs.com/tcicy/p/10185347.html

1.3-2虚拟机的使用（虚拟网络配置、镜像制作）

2021年5月20日　17:37

虚拟机的下载和安装

前面已经学到，虚拟机的实现方式有三种，全虚拟化、半虚拟化、硬件辅助虚拟化，在这里首先尝试配置和使用这三种实现方式中各自有代表性的虚拟机。

1. KVM

KVM是运行在linux内核上的“托管式”架构虚拟机，KVM的虚拟化是硬件辅助下的完全虚拟化，需要硬件支持。

首先通过CPU-V软件检测当前CPU是否支持VT技术。

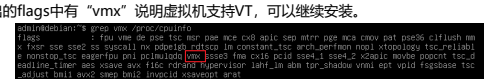
显示虚拟化技术支持并且已经启用。

因为主机是windows系统，所以在有虚拟化引擎的debian虚拟机上安装kvm虚拟机。

命令行执行

```
grep vmx /proc/cpuinfo
```

看到输出的flags中有“vmx”说明虚拟机支持VT，可以继续安装。



安装kvm：

```
root@debian:~# sudo apt install qemu qemu-kvm qemu-system qemu-utils
```

安装virt管理软件：

```
apt install libguestfs-tools
apt install virt-viewer
apt install virt-manager
```

可以看到安装成功了，但是目前还没有安装虚拟机操作系统。

```
root@debian:~# virsh -c qemu:///system list
 Id    Name                           State
----------------------------------------------------
```

打开虚拟系统管理器

点击创建新的虚拟机

提前将主机中下载好的系统映像文件加载到虚拟机的CD驱动器中。

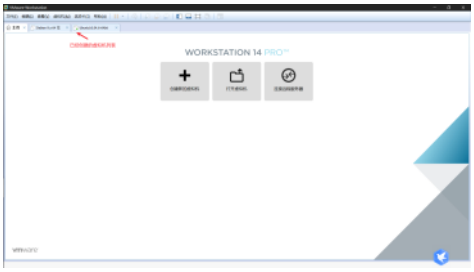
选择使用已经下载好的映像文件。

安装UBUNTU12.4，之后就可以启动并使用虚拟机了。

2 VMware workstaion

这个是我第一个安装的虚拟机，已经使用了一段时间，在windows系统下安装按照安装向导的提示步步操作即可。

虚拟机主界面：



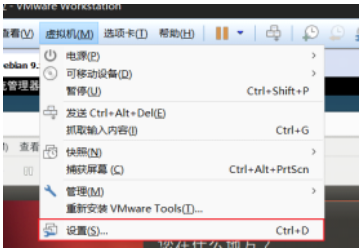
虚拟机设置界面：

控制虚拟网络适配器：

VMware tools是vmware提供的更加方便的操作和控制虚拟机的工具包，安装之后可以直接在主机和虚拟机之间复制文本、传递文件。

虚拟网络配置

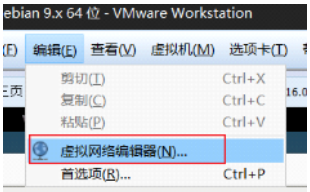
以vmware workstation虚拟机为例，



在“设置”中可以对虚拟机网络进行配置。

“网络适配器”中共有三种网络连接模式，分别是桥接模式、NAT模式和仅主机模式。
如果想要连接到外网，需要通过桥接模式或者NAT模式。

使用**NAT模式**进行网络配置是最简单的，虚拟机选择通过NAT模式连接网络之后，勾选设备状态中的“已连接”和“启动时连接”选项，随后打开虚拟网络编辑器。



在这里可以看到VMWARE的虚拟网络适配器列表。点击更改设置可以进行更改，

可以通过DHCP将IP自动分配给虚拟机，也可以手动配置IP地址。
如果手动配置IP地址，需要确保VMware workstation中的配置与主机网络适配器中的一致。
首先打开虚拟网络编辑器，选择“NAT设置”

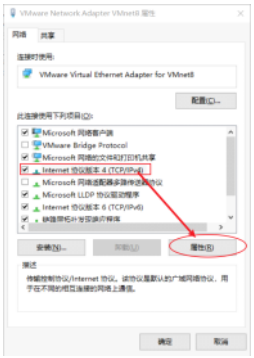
记录此处的子网ip和网关ip

随后在主机网络设置中

选择“更改适配器选项”

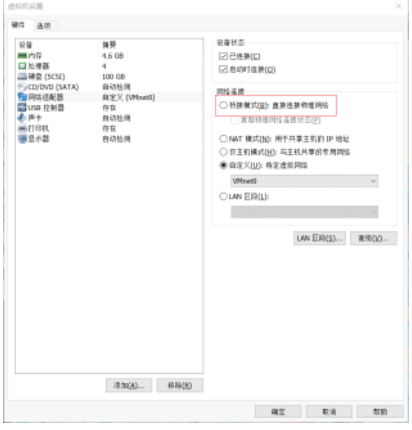
VMnet8即为配置NAT模式的网络适配器。

右键适配器，选择“属性”->选择IPv4-> ”属性”



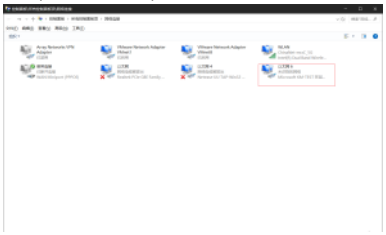
确保这里的默认网关与虚拟机中的设置一致，确保IP地址在子网范围内。

使用桥接模式连接网络时，比较简单的方式是自动配置桥接网络，只需要在虚拟机设置中选择桥接模式，剩余配置由虚拟机自动完成，虚拟机会自动桥接到目前有因特网连接的网络中。

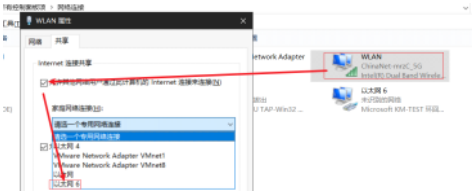


如果要添加一个新的**桥接模式**连接的网络，首先需要在主机中添加一个新的网络适配器。
打开计算机管理界面，选择设备管理器，选中我们的PC，点击“操作”->“添加过时硬件”

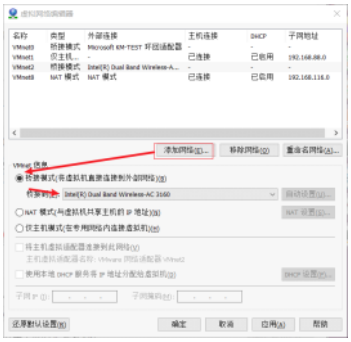
可以看到刚刚添加的网络适配器（我这里是以太网6）



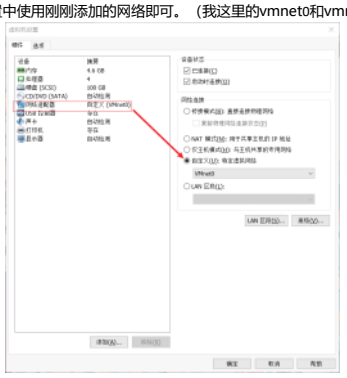
右键目前有网络的适配器，选择“属性” ->共享->允许其他网络用户通过此计算机的internet来连接->选择以太网6



随后在虚拟网络编辑器中添加一个新的网络，选择“桥接模式” ->桥接到环回适配器或者目前有网络连接的其他适配器。因为一个适配器只能桥接一个网络，所以如果需要添加多个桥接网络，则需要按照上文的方式添加多个网络适配器。



添加网络之后在虚拟机设置中使用刚刚添加的网络即可。（我这里的vmnet0和vmnet2都是已经配置好的桥接网络，一个直接桥接到无线网络适配器，一个桥接到刚才添加的以太网6）



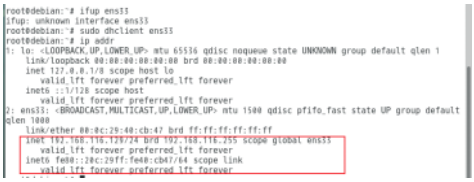
在某次重启debian虚拟机之后突然连不上网了，感觉可能是（未正常关机导致）网卡出了问题。

查看网卡名称以及状态：

```
root@debian:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 00:0c:29:40:cb:47 brd ff:ff:ff:ff:ff:ff
```

可以看到ens33网卡此时没有ip地址，使用ifup命令启动ens33无效，使用以下命令重启ens33网卡：

```
sudo dhclient ens33
```



```
root@debian:~# ifup ens33
ifup: unknown interface ens33
root@debian:~# sudo dhclient ens33
root@debian:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
qlen 1000
    link/ether 00:0c:29:40:cb:47 brd ff:ff:ff:ff:ff:ff
    inet 192.168.116.129/24 brd 192.168.116.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe40:cb47/64 scope link
       valid_lft forever preferred_lft forever
```

此时可以ping通网络了。