

49 | 虚拟机：如何成立子公司，让公司变集团？

2019-07-19 刘超

趣谈Linux操作系统

[进入课程 >](#)



讲述：刘超

时长 22:01 大小 20.17M



我们前面所有章节涉及的 Linux 操作系统原理，都是在一台 Linux 服务器上工作的。在前面的原理阐述中，我们一直把 Linux 当作一家外包公司的老板来看待。想要管理这么复杂、这么大的一个公司，需要配备咱们前面讲过的所有机制。

Linux 很强大，Linux 服务器也随之变得越来越强大了。无论是计算、网络、存储，都越来越牛。例如，内存动不动就是百 G 内存，网络设备一个端口的带宽就能有几十 G 甚至上百 G，存储在数据中心至少是 PB 级别的（一个 P 是 1024 个 T，一个 T 是 1024 个 G）。

公司大有大了的好处，自然也有大的毛病，也就是咱们常见的“大公司病”——**不灵活**。这里面的不灵活，有下面这几种，我列一下，你看看你是不是都见过。

资源大小不灵活：有时候我们不需要这么大规模的机器，可能只想尝试一下某些新业务，申请个 4 核 8G 的服务器试一下，但是不可能采购这么小规格的机器。无论每个项目需要多大规格的机器，公司统一采购就限制几种，全部是上面那种大规格的。

资源申请不灵活：规格定死就定死吧，可是每次申请机器都要重新采购，周期很长。

资源复用不灵活：反正我需要的资源不多，和别人共享一台机器吧，这样不同的进程可能会产生冲突，例如 socket 的端口冲突。另外就是别人用过的机器，不知道上面做过哪些操作，有很多的历史包袱，如果重新安装则代价太大。

这些是不是和咱们在大公司里面遇到的问题很像？按说，大事情流程严禁没问题，很多小事情也要被拖累走整个流程，而且很容易出现资源冲突，每天跨部门的协调很累人，历史包袱严重，创新没有办法轻装上阵。

很多公司处理这种问题采取的策略是成立独立的子公司，独立决策，独立运营，往往用于创新型的项目。

Linux 也采取了这样的手段，就是在物理机上面创建虚拟机。每个虚拟机有自己单独的操作系统、灵活的规格，一个命令就能启动起来。每次创建都是新的操作系统，很好地解决了上面不灵活的问题。

但是要使用虚拟机，还有一些问题需要解决一下。

我们知道，操作系统上的程序分为两种，一种是用户态的程序，例如 Word、Excel 等，一种是内核态的程序，例如内核代码、驱动程序等。

为了区分内核态和用户态，CPU 专门设置四个特权等级 0、1、2、3 来做这个事情。

当时写 Linux 内核的时候，估计大牛们还不知道将来虚拟机会大放异彩。大牛们想，一共两级特权，一个内核态，一个用户态，却有四个等级，好奢侈、好富裕，于是就敞开了用。内核态运行在第 0 等级，用户态运行在第 3 等级，占了两头，中间的都不用，太不会过日子了。

大牛们在写 Linux 内核的时候，如果用户态程序做事情，就将扳手掰到第 3 等级，一旦要申请使用更多的资源，就需要申请将扳手掰到第 0 等级，内核才能在高权限访问这些资源，申请完资源，返回到用户态，扳手再掰回去。

这个程序一直非常顺利地运行着，直到虚拟机出现了。

三种虚拟化方式

如果你安装 VirtualBox 桌面版，你可以用这个虚拟化软件创建虚拟机，在虚拟机里面安装一个 Linux，外面的操作系统也可以是 Linux。VirtualBox 这个虚拟化软件，和你的 Excel 一样，都是在你的任务栏里面并排放着，是一个普通的应用。

当你进入虚拟机的时候，虚拟机里面的 Excel 也是一个普通的应用。

这个时候麻烦的事情出现了，当你设身处地地站在虚拟机的内核角度，去思考一下人生，你就会出现困惑了，会想，我到底是啥？

在硬件上的操作系统来看，我是一个普通的应用，只能运行在用户态。可是大牛们“生”我的时候，我的每一行代码都告诉我，我是个内核啊，应该运行在内核态。当虚拟机里面的 Excel 要访问网络的时候，向我请求，我的代码就要努力地去操作网卡。尽管我努力，但是我做不到啊，我没有权限！

我分裂了.....

怎么办呢？虚拟化层，也就是 Virtualbox 会帮你解决这个问题，它有三种虚拟化的方式。

我们先来看第一种方式，**完全虚拟化**（Full virtualization）。其实说白了，这是一种“骗人”的方式。虚拟化软件会模拟假的 CPU、内存、网络、硬盘给到我，让我自我感觉良好，感觉自己终于又像个内核了。

但是，真正的工作模式其实是下面这样的。

虚拟机内核说：我要在 CPU 上跑一个指令！

虚拟化软件说：没问题，你是内核嘛，可以跑！

虚拟化软件转过头去找物理机内核说：报告，我管理的虚拟机里面的一个要执行一个 CPU 指令，帮忙来一小段时间空闲的 CPU 时间，让我代它跑个指令。

物理机内核说：你等着，另一个跑着呢。（过了一会儿）它跑完了，该你了。

虚拟化软件说：我代它跑，终于跑完了，出来结果了。

虚拟化软件转头给虚拟机内核说：哥们儿，跑完了，结果是这个。我说你是内核吧，绝对有权限，没问题，下次跑指令找我啊！

虚拟机内核说：看来我真的是内核呢，可是，哥，好像这点儿指令跑得有点慢啊！

虚拟化软件说：这就不错啦，好几个排着队跑呢！

内存的申请模式是下面这样的。

虚拟机内核说：我启动需要 4G 内存，我好分给我上面的应用。

虚拟化软件说：没问题，才 4G，你是内核嘛，我马上申请好。

虚拟化软件转头给物理机内核说：报告，我启动了一个虚拟机，需要 4G 内存，给我 4 个房间呗。

物理机内核：怎么又一个虚拟机啊！好吧，给你 90、91、92、93 四个房间。

虚拟化软件转头给虚拟机内核说：哥们，内存有了，0、1、2、3 这个四个房间都是你的。你看，你是内核嘛，独占资源，从 0 编号的就是你的。

虚拟机内核说：看来我真的是内核啊，能从头开始用。那好，我就在房间 2 的第三个柜子里面放个东西吧！

虚拟化软件说：要放东西啊，没问题。但是，它心里想：我查查看，这个虚拟机是 90 号房间开头的，它要在房间 2 放东西，那就相当于在房间 92 放东西。

虚拟化软件转头给物理机内核说：报告，我上面的虚拟机要在 92 号房间的第三个柜子里面放个东西。

好了，说完了 CPU 和内存的例子，网络和硬盘就不细说了，情况也是类似的，都是虚拟化软件模拟一个给虚拟机内核看的，其实啥事儿都需要虚拟化软件转一遍。

这种方式一个坏处就是，慢，而且往往慢到不能忍受。

于是，虚拟化软件想，我能不能不当传话筒，要让虚拟机内核正视自己的身份。别说你是内核，你还真喘上了。你不是物理机，你是虚拟机！

但是，怎么解决权限等级的问题呢？于是，Intel 的 VT-x 和 AMD 的 AMD-V 从硬件层面帮上了忙。当初谁让你们这些写内核的大牛用等级这么奢侈，用完了 0，就是 3，也不省着

点儿用，没办法，只好另起炉灶弄一个新的标志位，表示当前是在虚拟机状态下，还是在真正的物理机内核下。

对于虚拟机内核来讲，只要将标志位设为虚拟机状态，我们就可以直接在 CPU 上执行大部分的指令，不需要虚拟化软件在中间转述，除非遇到特别敏感的指令，才需要将标志位设为物理机内核态运行，这样大大提高了效率。

所以，安装虚拟机的时候，我们务必要将物理 CPU 的这个标志位打开。想知道是否打开，对于 Intel，你可以查看 `grep "vmx" /proc/cpuinfo`；对于 AMD，你可以查看 `grep "svm" /proc/cpuinfo`

这叫作**硬件辅助虚拟化**（Hardware-Assisted Virtualization）。

另外就是访问网络或者硬盘的时候，为了取得更高的性能，也需要让虚拟机内核加载特殊的驱动，也是让虚拟机内核从代码层面就重新定位自己的身份，不能像访问物理机一样访问网络或者硬盘，而是用一种特殊的方式。

我知道我不是物理机内核，我知道我是虚拟机，我没那么高的权限，我很可能和很多虚拟机共享物理资源，所以我要学会排队，我写硬盘其实写的是一个物理机上的文件，那我的写文件的缓存方式是不是可以变一下。我发送网络包，根本就不是发给真正的网络设备，而是给虚拟的设备，我可不可以直接在内存里面拷贝给它，等等等等。

一旦我知道我不是物理机内核，痛定思痛，只好重新认识自己，反而能找出很多方式来优化我的资源访问。

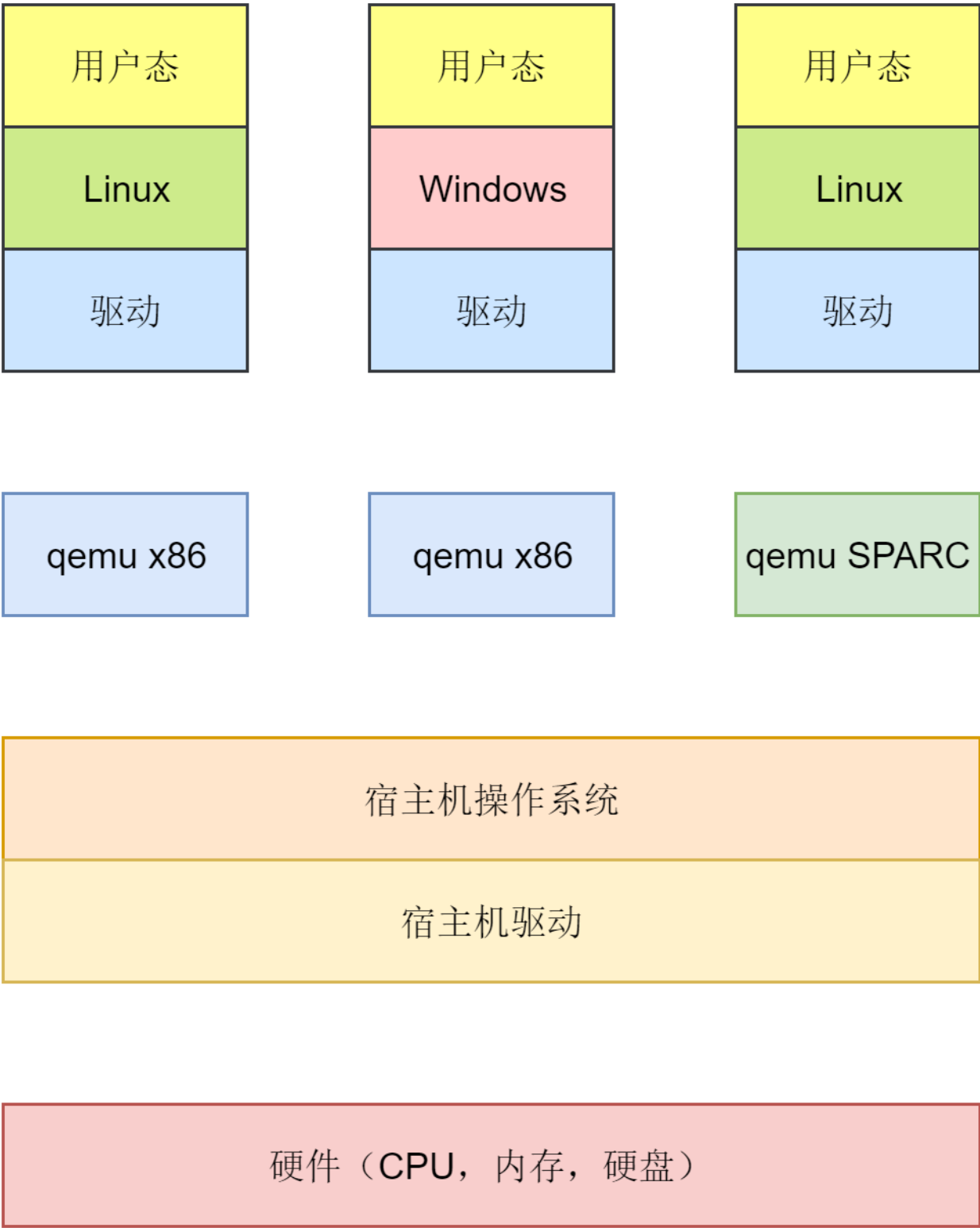
这叫作**半虚拟化**（Paravirtualization）。

对于桌面虚拟化软件，我们多采用 VirtualBox，如果使用服务器的虚拟化软件，则有另外的选型。

服务器上的虚拟化软件，多使用 qemu，其中关键字 emu，全称是 emulator，模拟器。所以，单纯使用 qemu，采用的是完全虚拟化的模式。

qemu 向 Guest OS 模拟 CPU，也模拟其他的硬件，GuestOS 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 会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，因而性能就会比较差。



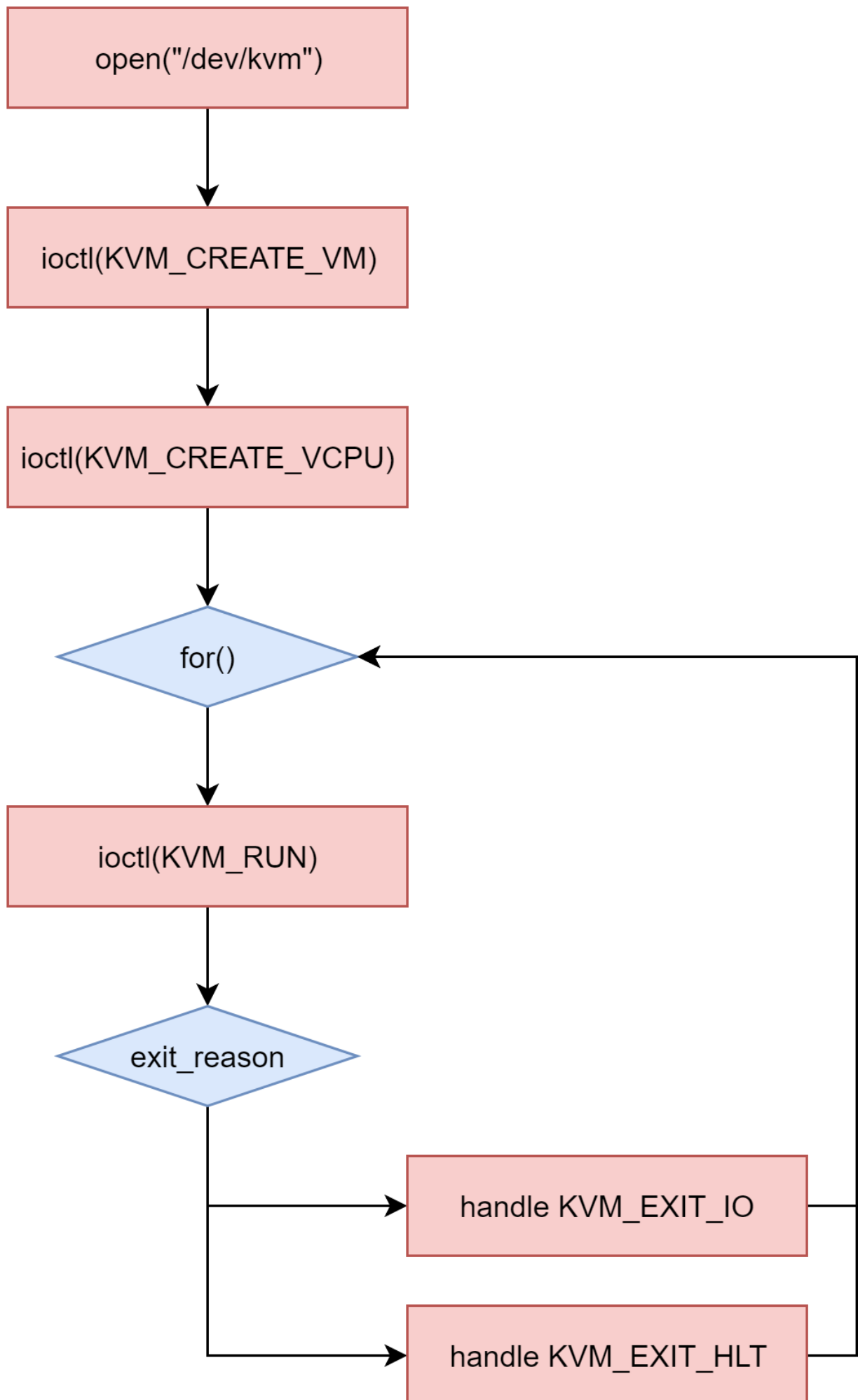
按照上面的介绍，完全虚拟化是非常慢的，所以要使用硬件辅助虚拟化技术 Intel-VT，AMD-V，所以需要 CPU 硬件开启这个标志位，一般在 BIOS 里面设置。

当确认开始了标志位之后，通过 KVM，GuestOS 的 CPU 指令不用经过 Qemu 转译，直接运行，大大提高了速度。

所以，KVM 在内核里面需要有一个模块，来设置当前 CPU 是 Guest OS 在用，还是 Host OS 在用。

下面，我们来查看内核模块中是否含有 kvm, `lsmod | grep kvm`。

KVM 内核模块通过 `/dev/kvm` 暴露接口，用户态程序可以通过 `ioctl` 来访问这个接口。例如，你可以通过下面的流程编写程序。



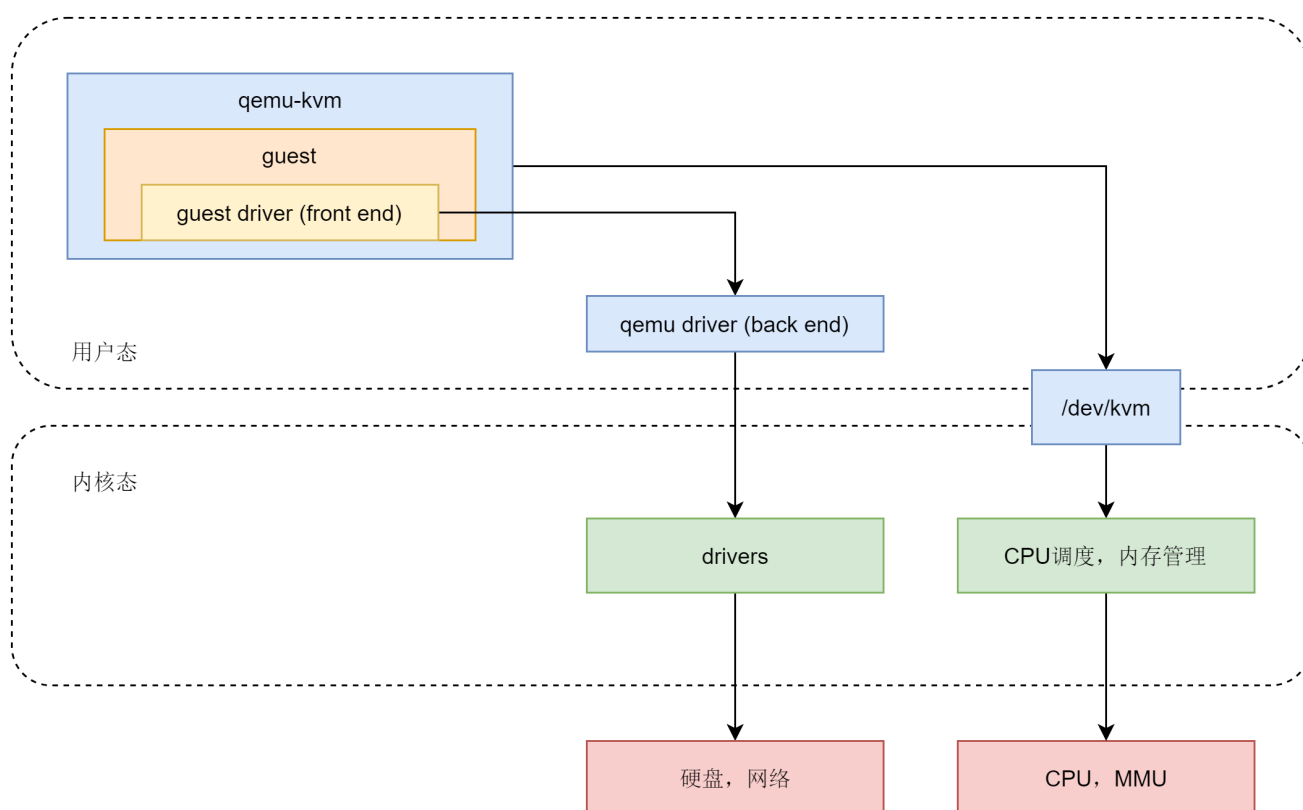
Qemu 将 KVM 整合进来，将有关 CPU 指令的部分交由内核模块来做，就是 qemu-kvm (qemu-system-XXX)。

qemu 和 kvm 整合之后，CPU 的性能问题解决了。另外 Qemu 还会模拟其他的硬件，如网络 and 硬盘。同样，全虚拟化的方式也会影响这些设备的性能。

于是，qemu 采取半虚拟化的方式，让 Guest OS 加载特殊的驱动来做这件事情。

例如，网络需要加载 virtio_net，存储需要加载 virtio_blk，Guest 需要安装这些半虚拟化驱动，GuestOS 知道自己是虚拟机，所以数据会直接发送给半虚拟化设备，经过特殊处理（例如排队、缓存、批量处理等性能优化方式），最终发送给真正的硬件。这在一定程度上提高了性能。

至此，整个关系如下图所示。



创建虚拟机

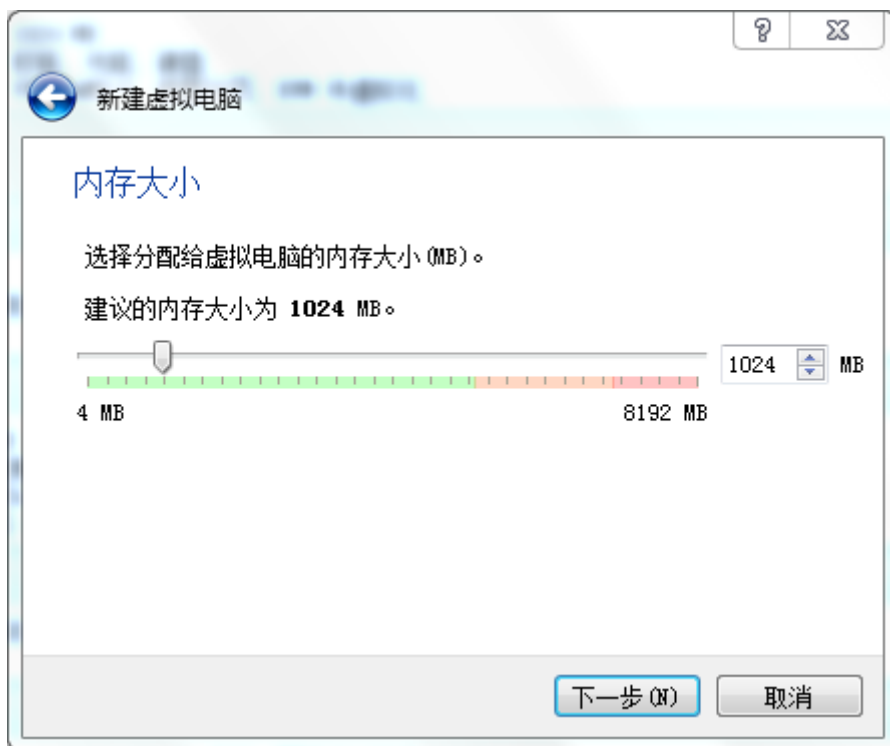
了解了 qemu-kvm 的工作原理之后，下面我们来看一下，如何使用 qemu-kvm 创建一个能够上网的虚拟机。

如果使用 VirtualBox 创建过虚拟机，通过界面点点就能创建一个能够上网的虚拟机。如果使用 qemu-kvm，就没有这么简单了。一切都得自己来做，不过这个过程可以了解 KVM 虚拟机的创建原理。

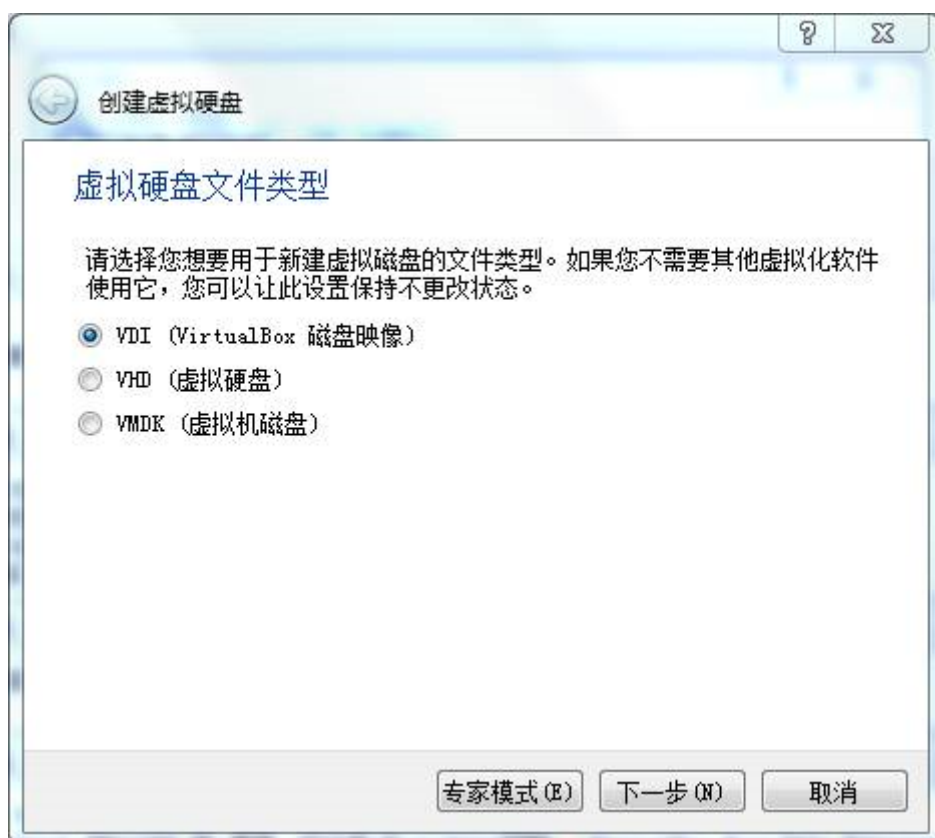
首先，我们要给虚拟机起一个名字，在 KVM 里面就是 -name ubuntutest。



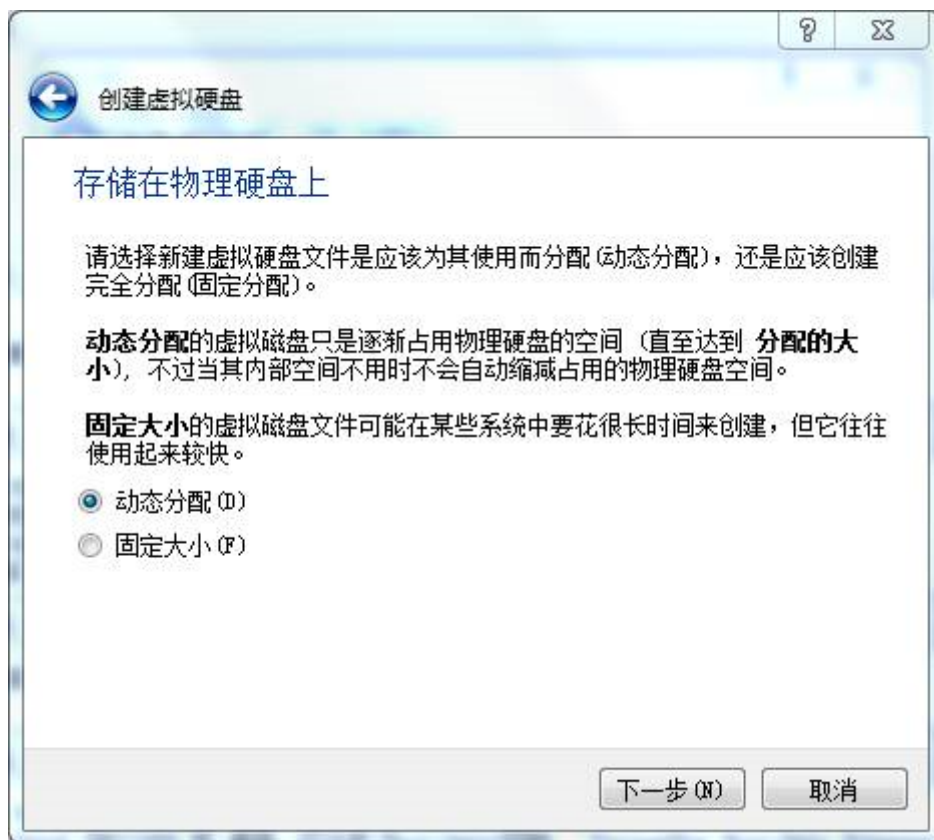
设置一个内存大小，在 KVM 里面就是 -m 1024。



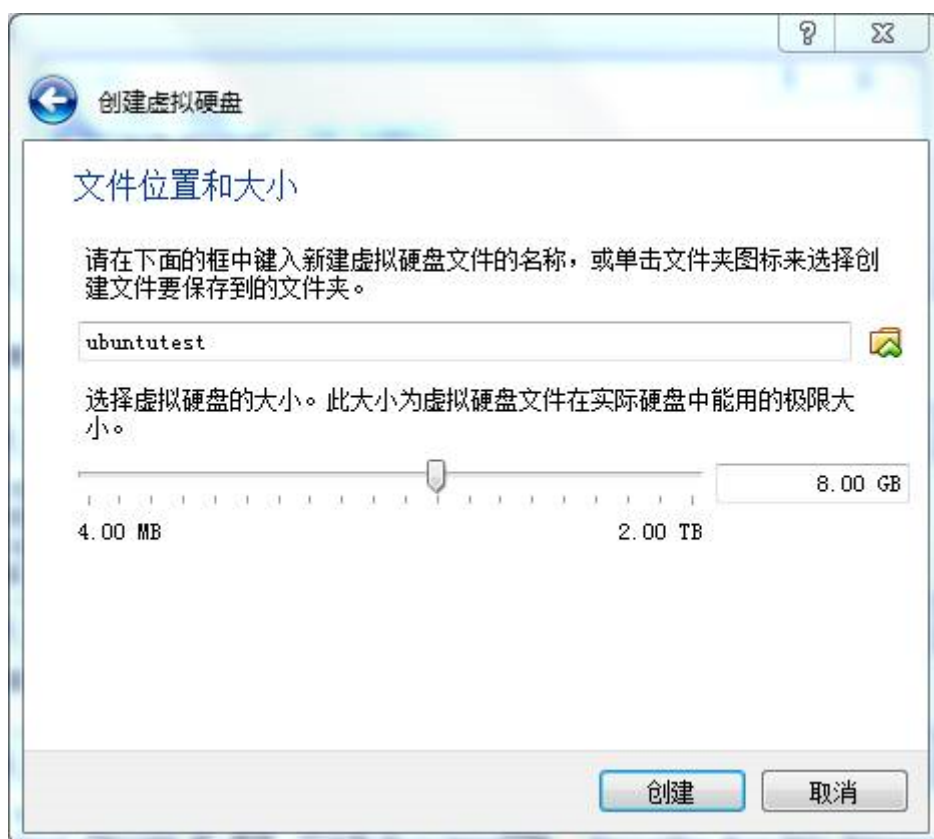
创建一个虚拟硬盘，对于 VirtualBox 是 VDI 格式，对于 KVM 则不同。



硬盘有两种格式，一个是动态分配，也即开始创建的时候，看起来很大，其实占用的空间很少，真实有多少数据，才真的占用多少空间。一个是固定大小，一开始就占用指定的大小。



比如，我这台电脑，硬盘的大小为 8G。



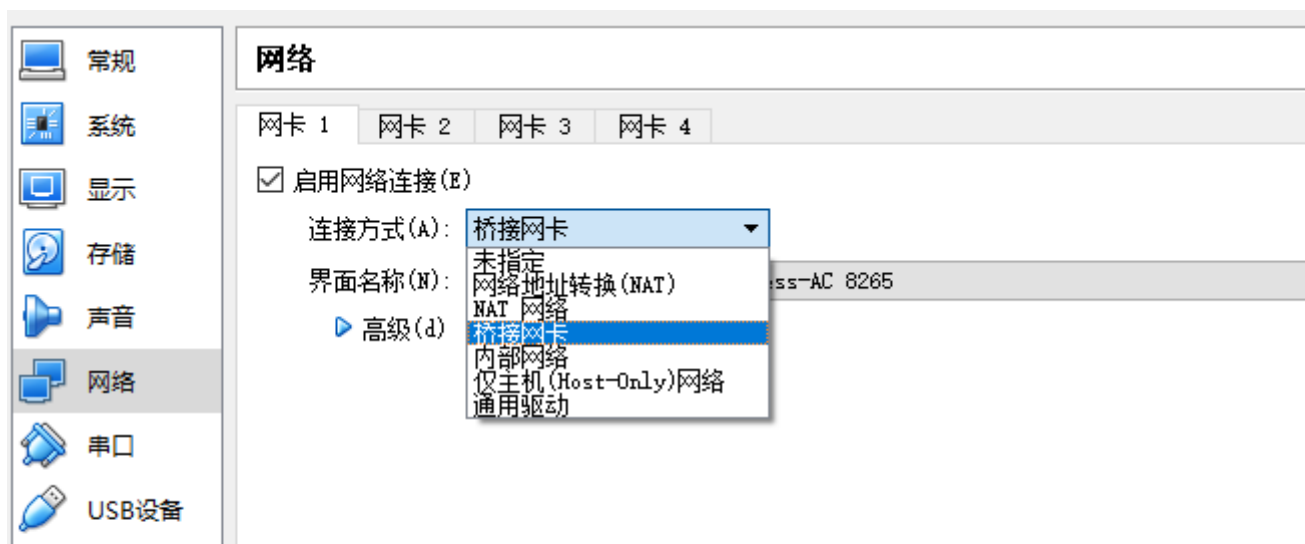
在 KVM 中，创建一个虚拟机镜像，大小为 8G，其中 qcow2 格式为动态分配，raw 格式为固定大小。

```
1 qemu-img create -f qcow2 ubuntu-test.img 8G
```

我们将 Ubuntu 的 ISO 挂载为光盘，在 KVM 里面 `-cdrom ubuntu-xxx-server-amd64.iso`。



创建一个网络，有时候会选择桥接网络，有时候会选择 NAT 网络，这个在 KVM 里面只有自己配置了。




☒ 启用网络连接(E)

连接方式(A): 桥接网卡

界面名称(N): Intel(R) Dual Band Wireless-AC 8265

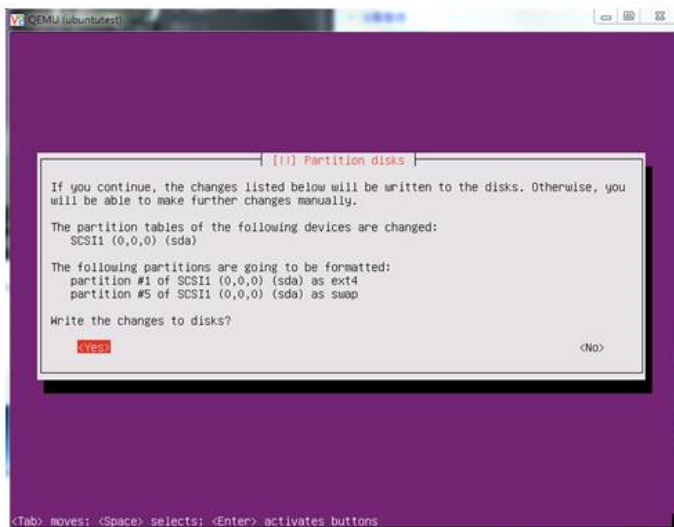
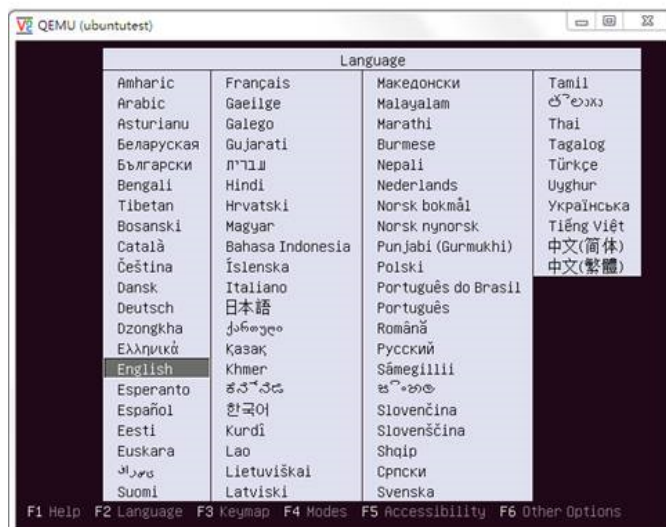
接下来 Virtualbox 就会有一个界面，可以看到安装的整个过程，在 KVM 里面，我们用 VNC 来做。参数为 `-vnc :19`

于是，我们也可以创建 KVM 虚拟机了，可以用下面的命令：

 复制代码

```
1 qemu-system-x86_64 -enable-kvm-name ubuntutest -m 2048 -hda ubuntutest.img -cdromubuntu
```

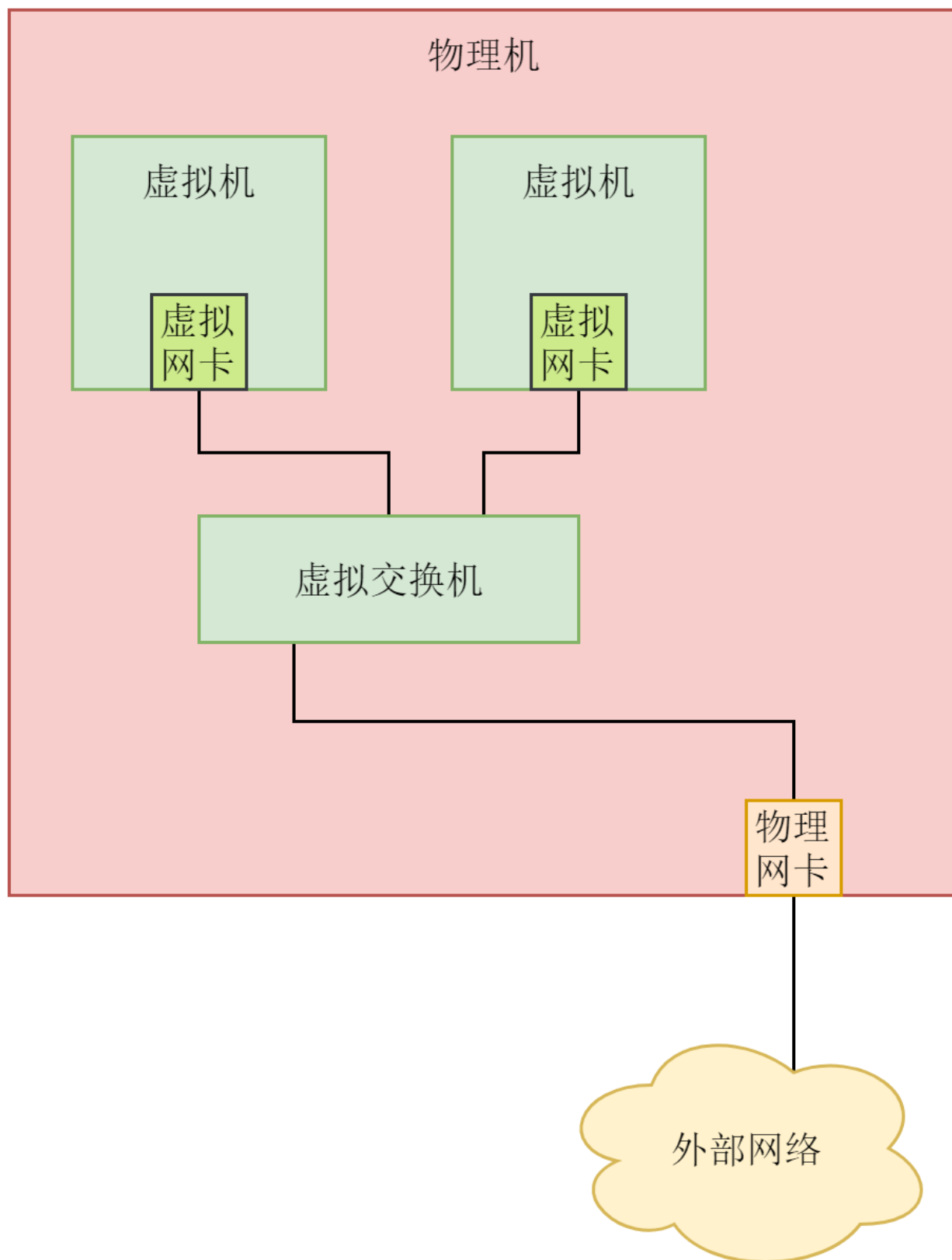
启动了虚拟机后，连接 VNC，我们也能看到安装的过程。



按照普通安装 Ubuntu 的流程安装好 Ubuntu , 然后 `shutdown -h now` , 关闭虚拟机。

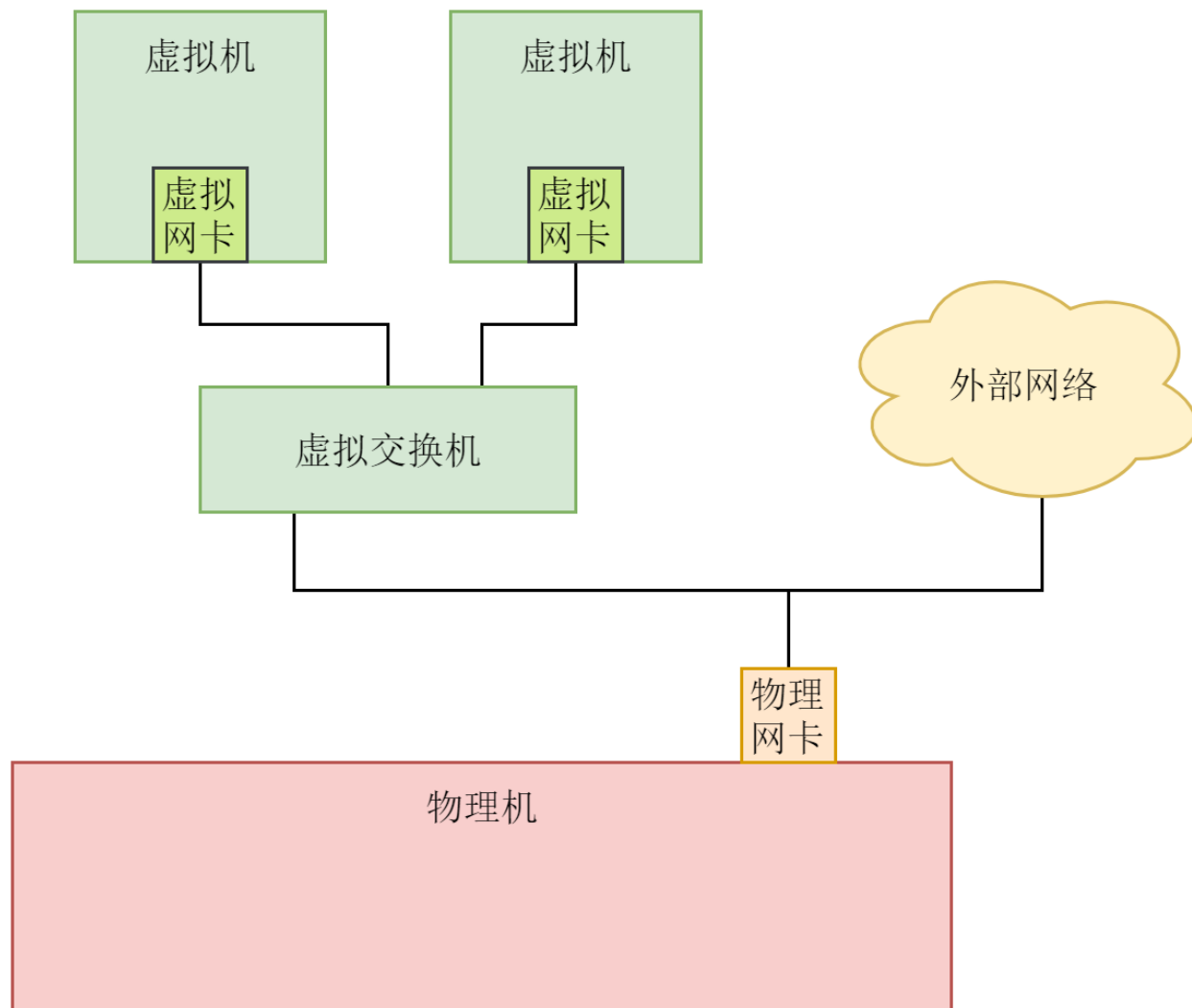
接下来 , 我们可以对 KVM 创建桥接网络了。这个要模拟 virtualbox 的桥接网络模式。

如果在桌面虚拟化软件上选择桥接网络 , 在你的笔记本电脑上 , 就会形成下面的结构。

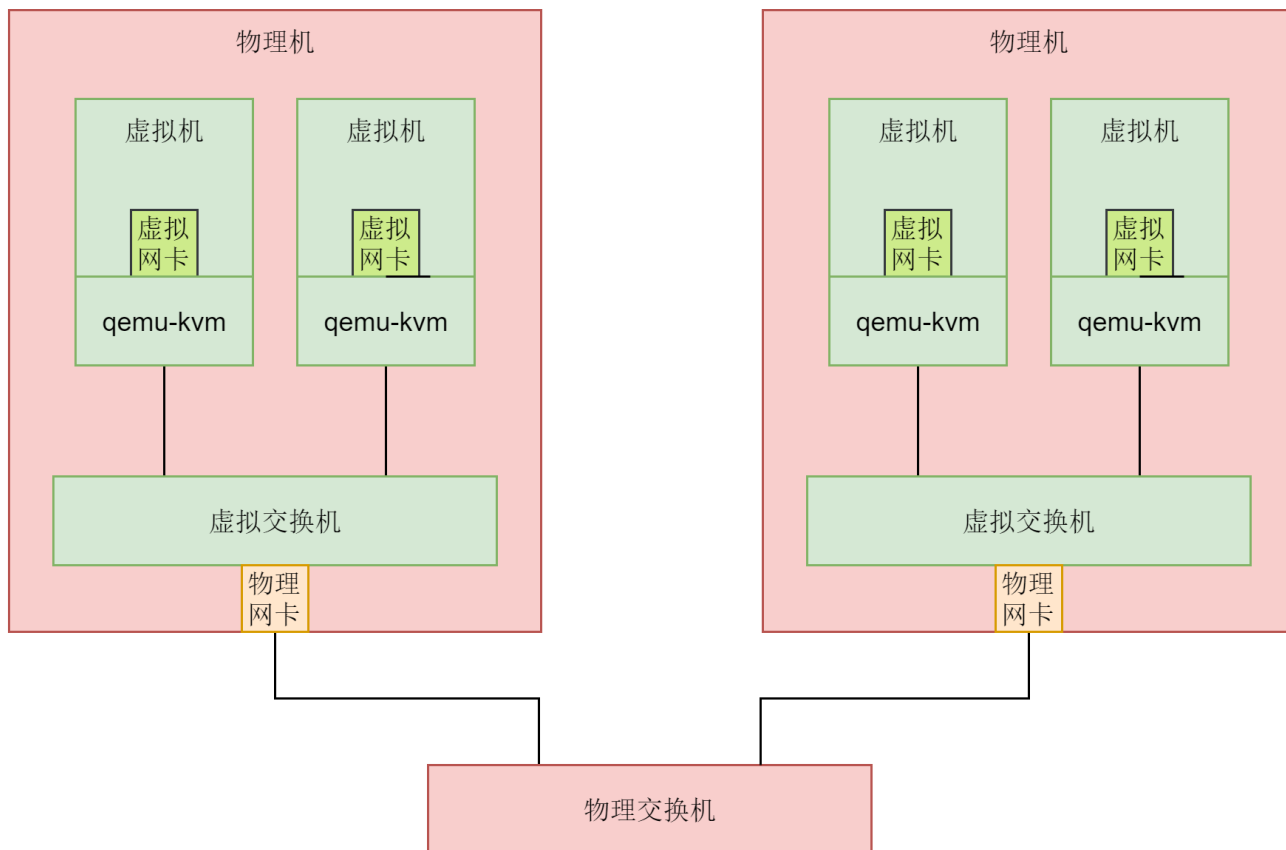


每个虚拟机都会有虚拟网卡，在你的笔记本电脑上，会发现多了几个网卡，其实是虚拟交换机。这个虚拟交换机将虚拟机连接在一起。在桥接模式下，物理网卡也连接到这个虚拟交换机上。物理网卡在桌面虚拟化软件的“界面名称”那里选定。

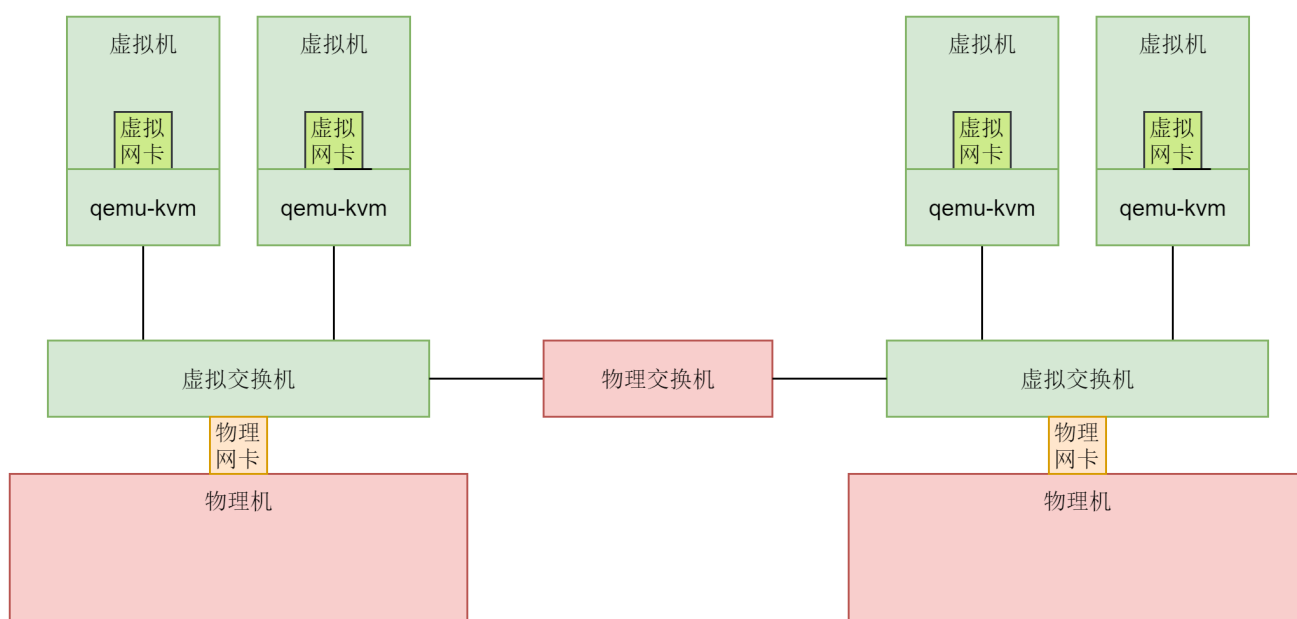
如果使用桥接网络，当你登录虚拟机里看 IP 地址时会发现，你的虚拟机的地址和你的笔记本电脑的地址，以及你旁边的同事的电脑的网段是一个网段。这是为什么呢？这其实相当于将物理机和虚拟机放在同一个网桥上，相当于这个网桥上有三台机器，是一个网段的，全部打平了。



在数据中心里面，采取的也是类似的技术，连接方式如下图所示，只不过是 Linux 在每台机器上都创建网桥 br0，虚拟机的网卡都连到 br0 上，物理网卡也连到 br0 上，所有的 br0 都通过物理网卡连接到物理交换机上。




同样我们换一个角度看待这个拓扑图。同样是将网络打平，虚拟机会和物理网络具有相同的网段，就相当于两个虚拟交换机、一个物理交换机，一共三个交换机连在一起。两组四个虚拟机和两台物理机都是在一个二层网络里面的。




qemu-kvm 如何才能创建一个这样的桥接网络呢？

1. 在 Host 机器上创建 bridge br0。

 复制代码


```
1 brctl addbr br0
```

2. 将 br0 设为 up。

 复制代码


```
1 ip link set br0 up
```

3. 创建 tap device。

 复制代码


```
1 tuncctl -b
```

4. 将 tap0 设为 up。

 复制代码

```
1 ip link set tap0 up
```

5. 将 tap0 加入到 br0 上。

 复制代码


```
1 brctl addif br0 tap0
```

6. 启动虚拟机, 虚拟机连接 tap0、tap0 连接 br0。

 复制代码

```
1 qemu-system-x86_64 -enable-kvm -name ubuntutest -m 2048 -hda ubuntutest.qcow2 -vnc :19
```

7. 虚拟机启动后，网卡没有配置，所以无法连接外网，先给 br0 设置一个 ip。

 复制代码

```
1 ifconfig br0 192.168.57.1/24
```

8.VNC 连上虚拟机，给网卡设置地址，重启虚拟机，可 ping 通 br0。

9. 要想访问外网，在 Host 上设置 NAT，并且 enable ip forwarding，可以 ping 通外网网关。

 复制代码

```
1 # sysctl -p
2 net.ipv4.ip_forward = 1
3
4 sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
5
```

10. 如果 DNS 没配错，可以进行 apt-get update。

在这里，请记住 qemu-system-x86_64 的启动命令，这里面有 CPU 虚拟化 KVM，有内存虚拟化、硬盘虚拟化、网络虚拟化。接下来的章节，我们会看内核是如何进行虚拟化的。

总结时刻

今天我们讲了虚拟化的基本原理，并且手动创建一个可以上网的虚拟机。请记住下面这一点，非常重要，理解虚拟机启动的参数就是理解虚拟化技术的入口。学会创建虚拟机，在后面做内核相关实验的时候就会非常方便。

具体到知识点上，这一节你需要记住下面的这些知识点：

虚拟化的本质是用 qemu 的软件模拟硬件，但是模拟方式比较慢，需要加速；

虚拟化主要模拟 CPU、内存、网络、存储，分别有不同的加速办法；

CPU 和内存主要使用硬件辅助虚拟化进行加速，需要配备特殊的硬件才能工作；

网络和存储主要使用特殊的半虚拟化驱动加速，需要加载特殊的驱动程序。

课堂练习

请你务必自己使用 qemu，按照上面我写的步骤创建一台虚拟机。

欢迎留言和我分享你的疑惑和见解，也欢迎可以收藏本节内容，反复研读。你也可以把今天的内容分享给你的朋友，和他一起学习和进步。

 极客时间

趣谈 Linux 操作系统

像故事一样的操作系统入门课

刘超

网易杭州研究院
云计算技术部首席架构师



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 48 | 接收网络包（下）：如何搞明白合作伙伴让我们做什么？

精选留言 (2)

写留言



ty

2019-07-21

不是有cpu四个等级么，操作系统用了0和3，虚拟化软件为什么不直接用0或者2，而要cpu提供新机制呢



小龙的城堡

2019-07-19



展开 ∨

