为什么有虚拟化？

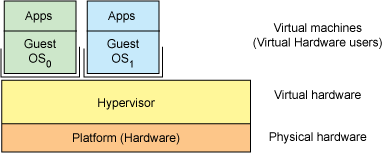
1、随着CPU计算能力的提高，单独的OS已不能充分利用CPU的计算能力

2、很多应用的执行需要单独占用一个OS环境

hypervisor和虚拟化的关系

虚拟化就是通过某种方式隐藏底层物理硬件的过程，从而让多个操作系统可以透明地使用和共享它。这种架构的另一个更常见的名称是平台虚拟化。在典型的分层架构中，提供平台虚拟化的层称为 hypervisor （有时称为虚拟机管理程序 或 VMM）。来宾操作系统称为虚拟机（VM），因为对这些 VM 而言，硬件是专门针对它们虚拟化的

**常用硬件虚拟化的简单分层架构**



hypervisor 分类

type-1：hypervisor 是直接运行在物理硬件之上的（硬件 --> hypervisor --> VM）

type-2：hypervisor 运行在另一个操作系统中的（硬件 --> os --> hypervisor --> VM）

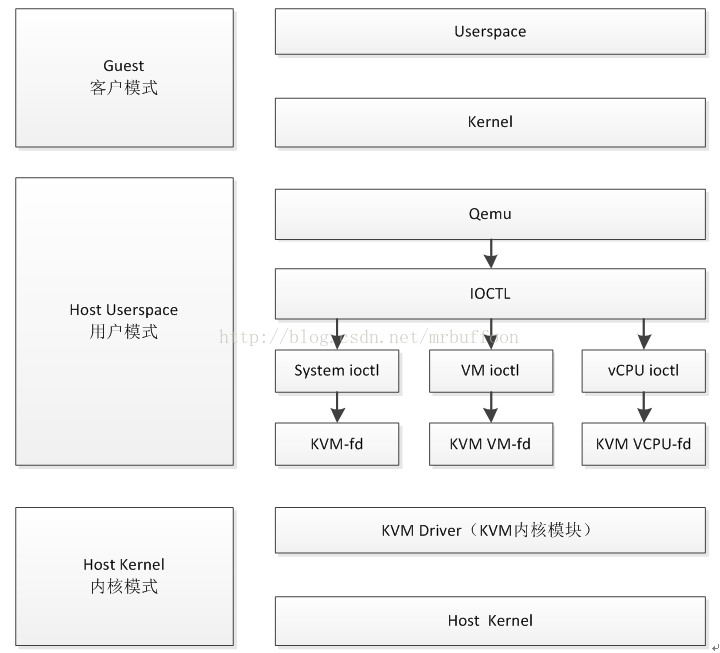
KVM载入系统的运行模式：KVM 是内核模块实现的，因此Linux只要加载该模块就会成为一个hypervisor

客户模式：我们可以简单理解成客户机操作系统运行的模式，它本身又分为自己的内核模式和用户模式

用户模式：为用户提供虚拟机管理的用户空间工具以及代表用户执行I/O，Qemu运行在这个模式之下

内核模式：模拟CPU以及内存，实现客户模式的切换，处理从客户模式的退出。KVM内核模块运行在这个模式下

模式之间的关系



当新的操作系统在 KVM 上启动时，它就成为宿主操作系统的一个进程，因此就可以像其他进程一样调度它。但与传统的 Linux 进程不一样，来宾操作系统被 hypervisor 标识为处于 “来宾” 模式独立于内核和用户模式

每个来宾操作系统都是通过 /dev/kvm 设备映射的，它们拥有自己的虚拟地址空间，该空间映射到主机内核的物理地址空间。KVM 使用底层硬件的虚拟化支持来提供原生虚拟化。I/O 请求通过主机内核映射到在主机上执行的QEMU进程

KVM 在 Linux 环境中以主机的方式运行，不过只要底层硬件虚拟化支持，它就能够支持大量的来宾操作系统

qemu：

qemu是面向各种架构的平台仿真器，负责模拟全系统，利用其他VMM(Xen, KVM)来使用硬件提供的虚拟化支持，创建接近于主机性能的虚拟机

KVM工作流程

用户模式的Qemu利用接口libkvm通过“io调用”进入内核模式。KVM驱动为虚拟机创建虚拟CPU和虚拟内存，然后执行VMLAUNCH指令进入客户模式，装载Guest OS并运行

全虚拟化技术：全虚拟化技术又叫硬件辅助虚拟化技术，它在虚拟机（VM）和硬件之间加了一个软件层(Hypervisor)

hypervisor可以划分为两大类：

type-1：hypervisor 是直接运行在物理硬件之上的（硬件 --> hypervisor --> VM）

type-2：hypervisor 运行在另一个操作系统中的（硬件 --> os --> hypervisor --> VM）

因为运行在虚拟机上的操作系统通过Hypervisor来最终分享硬件，所以虚拟机发出的指令需经过Hypervisor处理。为此每个客户操作系统所发出的指令都要被翻译成CPU能识别的指令格式，这里的客户操作系统即是运行的虚拟机，所以Hypervisor的工作负荷会很大，因此会占用一定的资源，所以在性能方面不如裸机和半虚拟化技术。

半虚拟化技术：在全虚拟化的基础上，把客户操作系统进行了修改，增加了一个专门的API，这个API可以将客户操作系统发出的指令进行最优化，即不需要Hypervisor耗费一定的资源进行翻译操作，因此Hypervisor的工作负担变得非常的小，因此整体的性能也有很大的提高。

KVM常见三种网络基础模型：

隔离模型：所有虚拟机通过一个vswitch将连接到一个vswitch的虚拟机组织成一个隔离网络，在这个网络中虚拟机与虚拟机之间能够正常通行，但不能访问外部网络

NAT模型：将虚拟机的网卡连接到br上，br同物理机网卡连接，通过NAT地址转换实现虚拟机可以访问外网的功能

桥接模型：客户机与宿主机处于同一网络环境，将外网地址配置在br上，将br当做物理机网卡使用，br关联到物理网卡通过物理网卡发送报文，直接访问网络资源(安全性较低)