从虚拟化kvm-到openstack

8点半开始

讲师：wendao（夏中云老师）

虚拟化

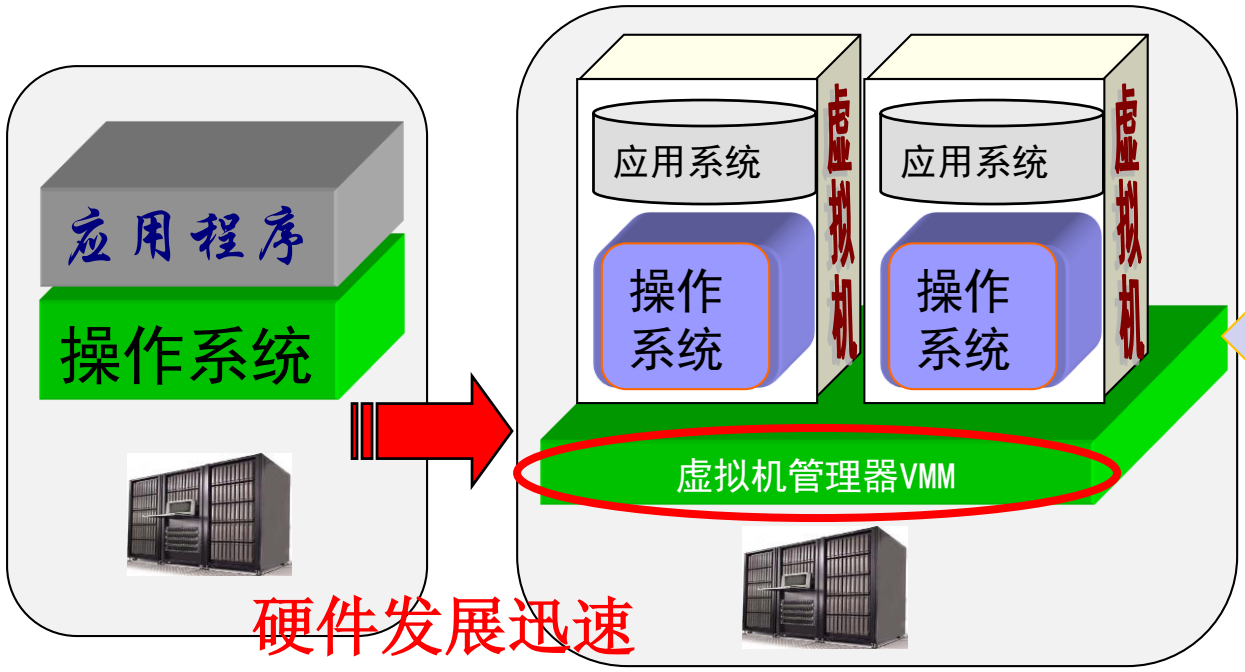
- 明白Kvm 干什么
- 掌握kvm搭建，c语言，python,java调用kvm
- 理解kvm网络
- Openstack
  - **帮公司构建私有云平台**
  - 明白openstack各个模块的作用

大纲

- 什么是虚拟化
- 为什么需要虚拟化
- Kvm大纲
  - Kvm介绍
  - Kvm实战
  - Kvm网络
- Kvm虚拟化总结

什么是系统虚拟化

- **系统虚拟化**是将底层物理设备与上层操作系统、软件分离的一种**去耦合**技术，在一台物理机器上逻辑的划分出多台机器。
- **虚拟化的目标**是实现IT资源**利用效率和灵活性**的最大化
- vmware vsphere exsi



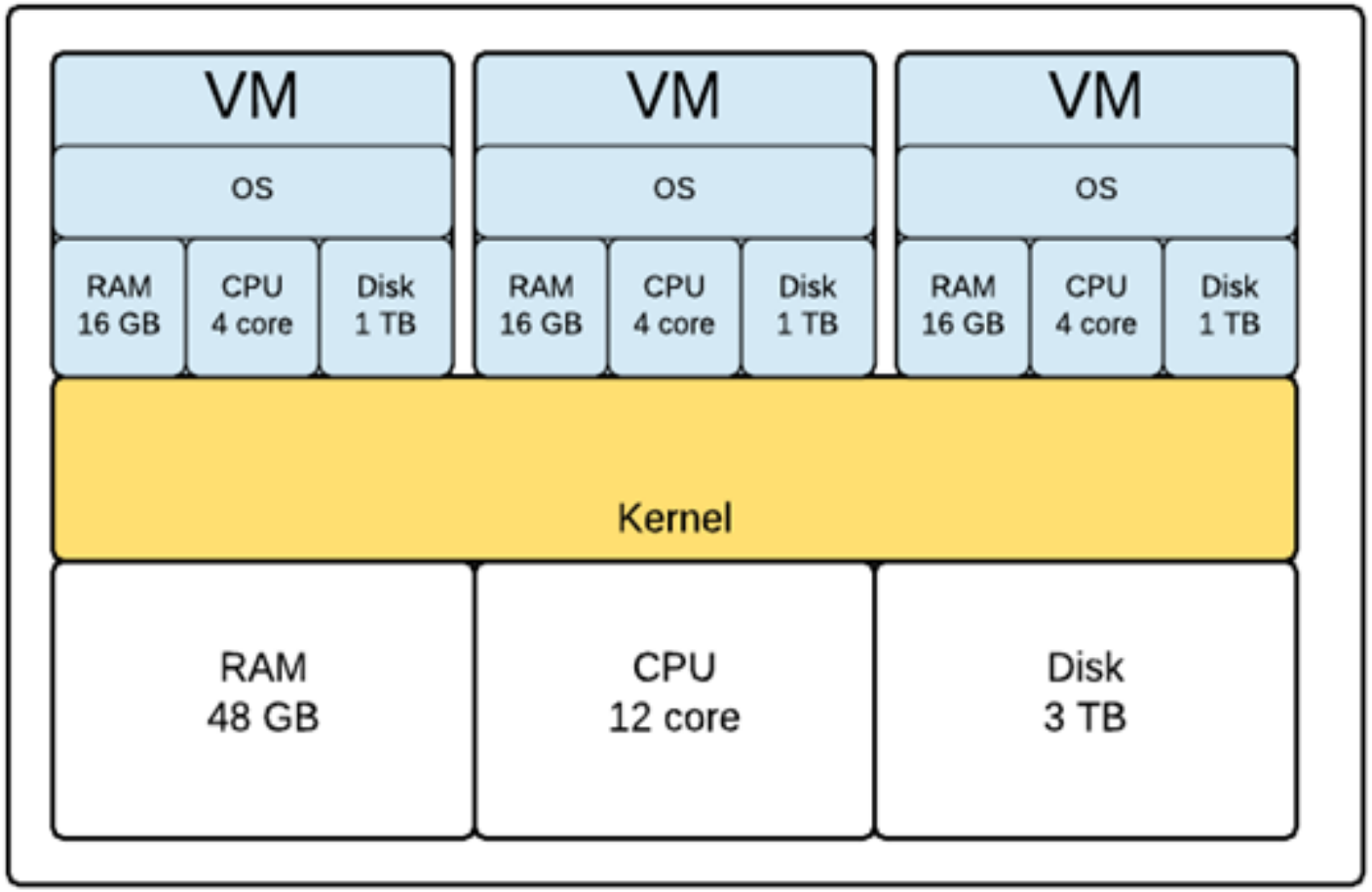
多个系统融合在一台服务器上

——资源利用率高

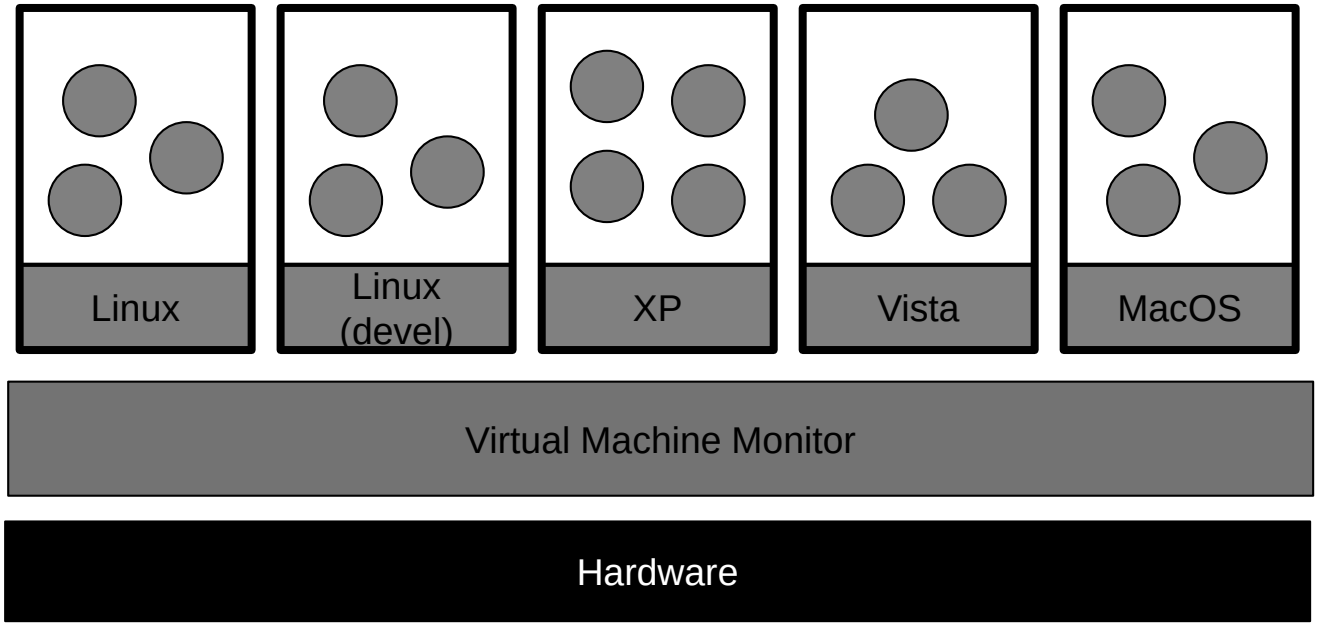
应用系统不再依赖特定的硬件

——系统维护灵活

虚拟化基础架构


VM
OS
RAM
16 GB
CPU
4 core
Disk
1 TB
VM
OS
RAM
16 GB
CPU
4 core
Disk
1 TB
VM
OS
RAM
16 GB
CPU
4 core
Disk
1 TB
Kernel
RAM
48 GB
CPU
12 core
Disk
3 TB

- 在一台物理主机上虚拟出多个**虚拟计算机（虚拟机，Virtual Machine，VM）**，其上能同时运行多个独立的操作系统，这些**客户操作系统（Guest OS）**通过**虚拟机管理器（Virtual Machine Monitor，VMM，也称作Hypervisor）**访问实际的物理资源



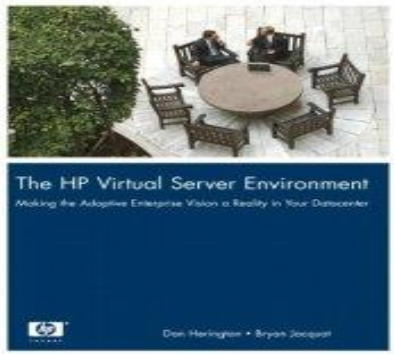
为什么需要使用虚拟化

- 公司服务器越来越多
  - 充分利用是个问题
  - 统一运维管理是个问题
- 浪费时间
- 操作繁琐
- 机器闲置时间较多

系统虚拟化的产生

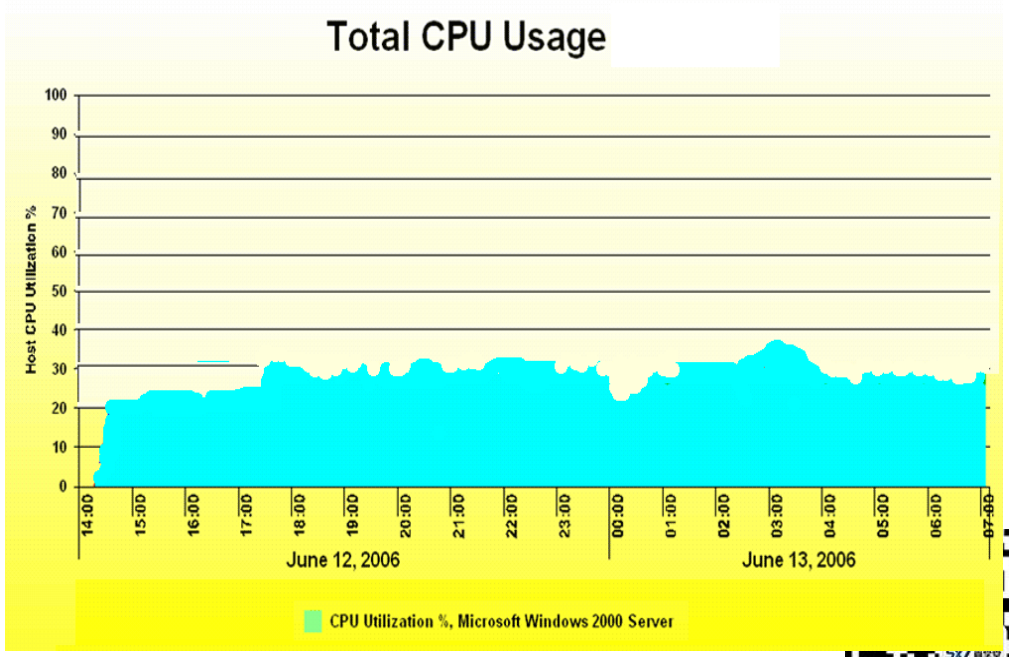
- 计算系统利用率不高！

  - “多数用户承认，计算系统平均利用率只有25%～30%”



Dan Herington
HP虚拟化技术首席科学家

  - 性能测试报告，来自权威性能测试机构Metron's Athene
  - 对一个计算系统进行两天监测的数据

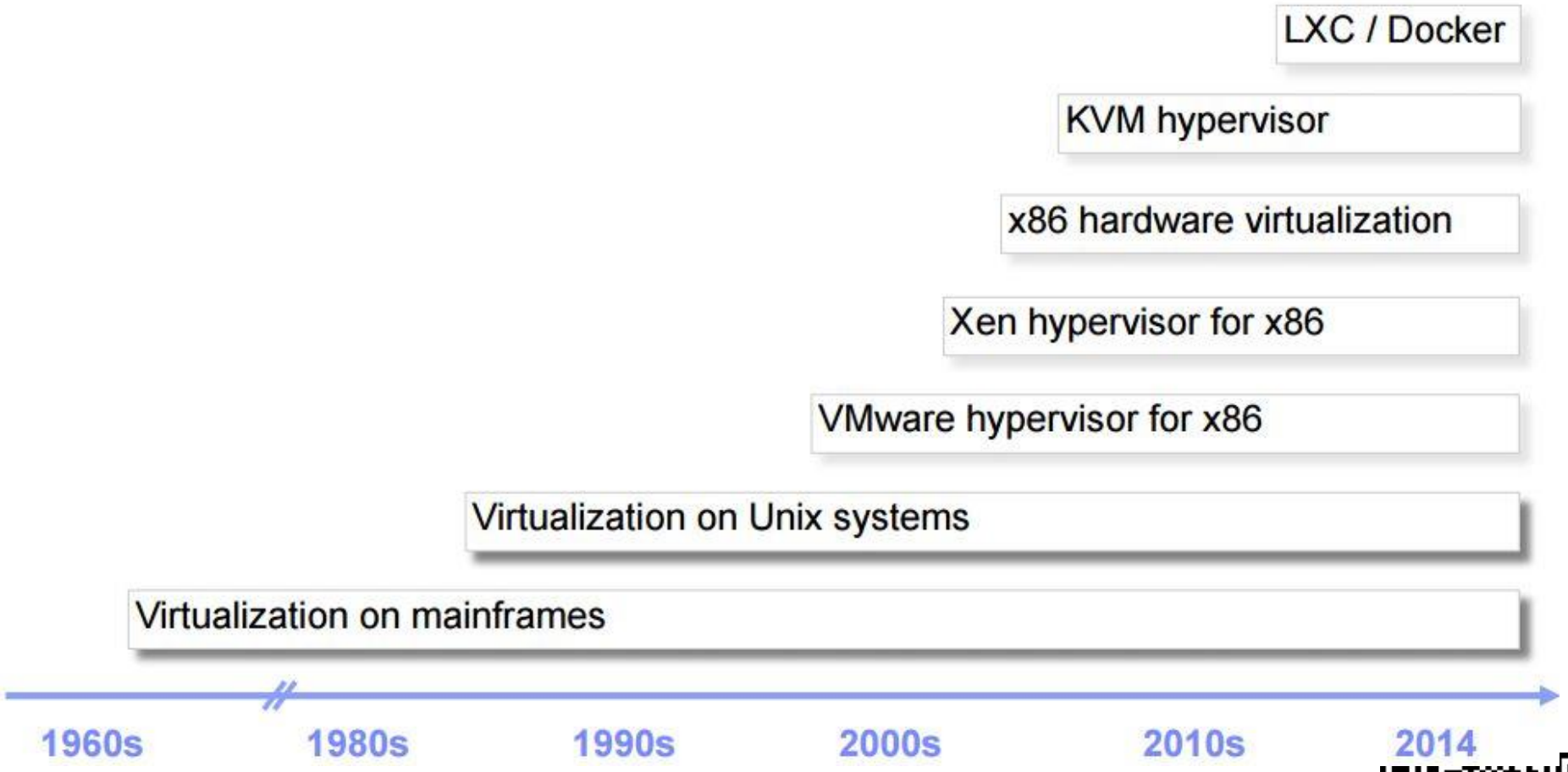


为什么客户要选择服务器虚拟化？

1. 提高硬件资源利用率
2. 减少数据中心的建设成本
3. 提升业务连续性
4. 实现数据中心灵活运营

1. 打破“一台服务器对应一套应用”的模式，将物理服务器进行整合，提升利用率
2. 服务器和相关IT硬件更少，节省了机房空间，也减少了散热和电力需求
3. 具备灵活数据备份和应用迁移机制，保障服务永不中断
4. 资源动态调配和模板化部署，应用系统快速上线，及时响应业务变化

虚拟化发展历史


LXC / Docker
KVM hypervisor
x86 hardware virtualization
Xen hypervisor for x86
VMware hypervisor for x86
Virtualization on Unix systems
Virtualization on mainframes
1960s
1980s
1990s
2000s
2010s
2014

你知道那些虚似化技术？

VMWare

Citrix Xen

Sun
VirtualBox

Microsoft
Hyper-V

HP
vPars

Parallels
OpenVZ

IBM
LPARS

Parallels(SWSoft)
Virtuozzo

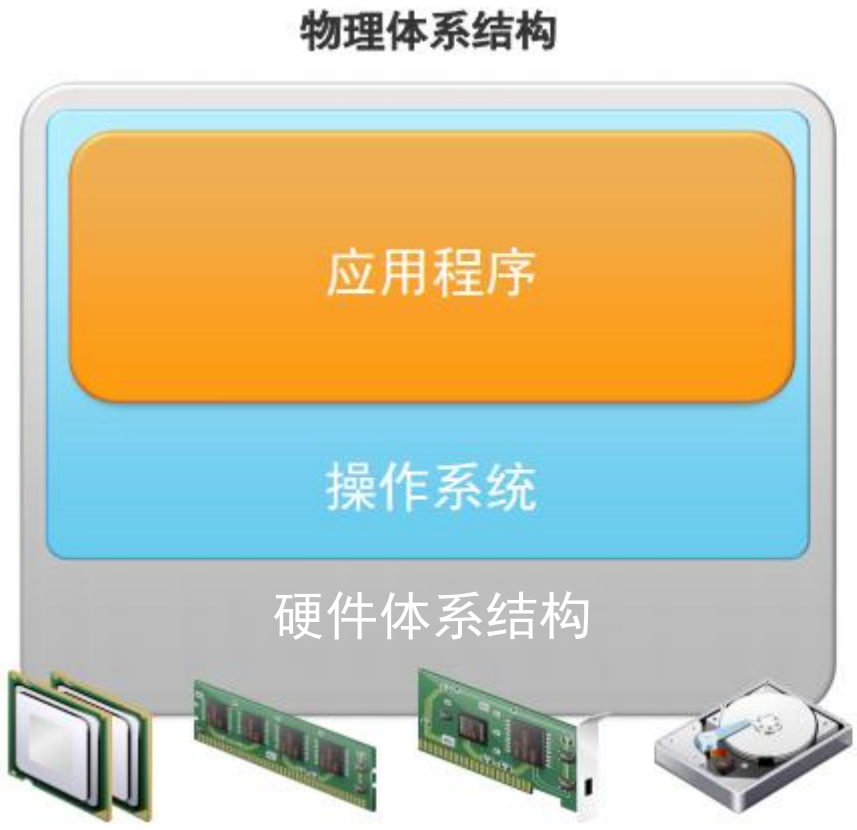
Sun
LDOMs

IBM
PowerVM

什么是桌面虚拟化？

- **桌面虚拟化**（MS：Remote Desktop、Citrix：XenDesktop、Vmware：View）
  - **将原本在本地电脑安装的桌面系统统一在后端数据中心进行部署和管理；**
  - 用户可以通过任何设备，在任何地点，任何时间访问属于自己的桌面系统环境。

服务器虚拟化


物理体系结构
应用程序
操作系统
硬件体系结构


虚拟体系结构
APP
OS
APP
OS
APP
OS
（hypervisor）虚拟化层
硬件体系结构

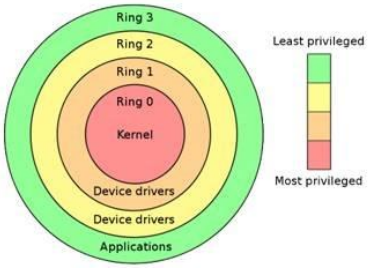
Kvm介绍

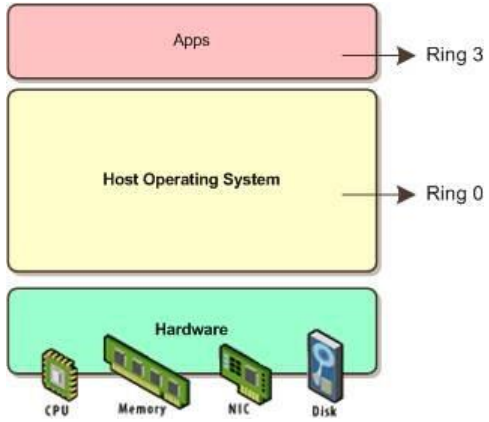
- 其中，KVM 全称是 基于内核的虚拟机（Kernel-based Virtual Machine），它是一个 Linux 的一个内核模块，该内核模块使得 Linux 变成了一个 Hypervisor：
- 它由 Quramnet 开发，该公司于 2008年被 Red Hat 收购。
- 它支持 x86 (32 and 64 位), s390, Powerpc 等 CPU。
- 它从 Linux 2.6.20 起就作为一模块被包含在 Linux 内核中。
- 它需要支持虚拟化扩展的 CPU。
- 它是完全开源的。官网。

KVM特性

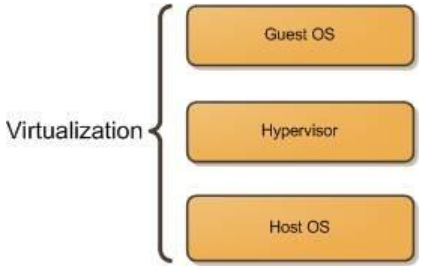
- 嵌入到Linux正式Kernel（提高兼容性）
- 代码级资源调用（提高性能）
- 虚拟机就是一个进程（内存易于管理）
- ---------- RedHat收购KVM ----------
- 保持开源发展模式
- 更好的商业支持及服务保障

- 无虚拟化
  - CPU一般设为四个Ring
  - Kernel Mode一般跑在Ring 0上
  - User Mode一般跑在Ring 3上
  - 对于一个普通的传统的Linux系统没有问题





- 虚拟化
  - 在Guest机器和Host机器中间加一层Hypervisor
  - Host机器看它像跑在自己上面的程序
  - Guest机器看它像自己所运行的硬件
  - 如果Host机器和Guest机器都跑相同的Linux，它们的Kernel都想运行在Ring 0，可怎么办？



虚拟化分类

1. 基于二进制翻译的全虚拟化（Full Virtualization with Binary Translation）
2. 超虚拟化（或者半虚拟化/操作系统辅助虚拟化 Paravirtualization）
3. 硬件辅助的全虚拟化

传统cpu工作模式

- X86 操作系统是设计在直接运行在裸硬件设备上的，因此它们自动认为它们完全占有计算机硬件。x86 架构提供四个特权级别给操作系统和应用程序来访问硬件。 Ring 是指 CPU 的运行级别，Ring 0是最高级别，Ring1次之，Ring2更次之……就 Linux+x86 来说，
- 操作系统（内核）需要直接访问硬件和内存，因此它的代码需要运行在最高运行级别 Ring0上，这样它可以使用特权指令，控制中断、修改页表、访问设备等等。
- 应用程序的代码运行在最低运行级别上ring3上，不能做受控操作。如果要做，比如要访问磁盘，写文件，那就要通过执行系统调用（函数），执行系统调用的时候，CPU的运行级别会发生从ring3到ring0的切换，并跳转到系统调用对应的内核代码位置执行，这样内核就为你完成了设备访问，完成之后再从ring0返回ring3。这个过程也称作用户态和内核态的切换。

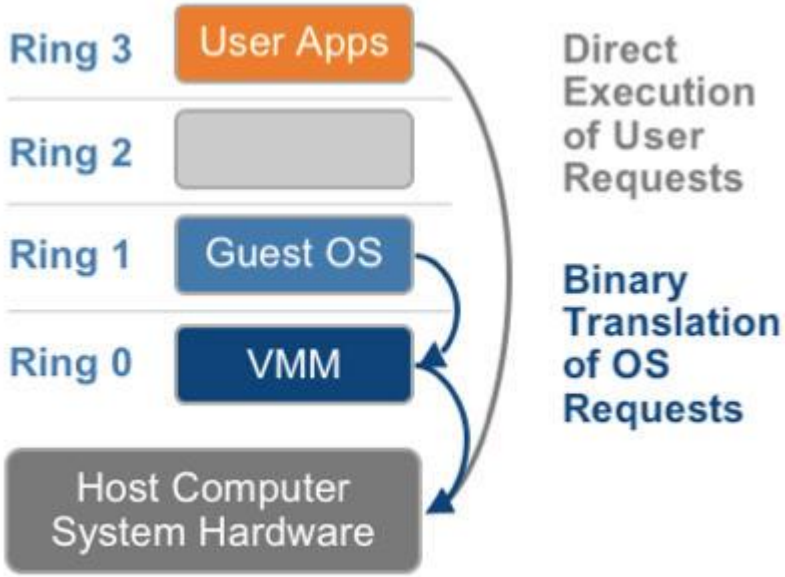
为什么需要 CPU 虚拟化

虚拟化分类

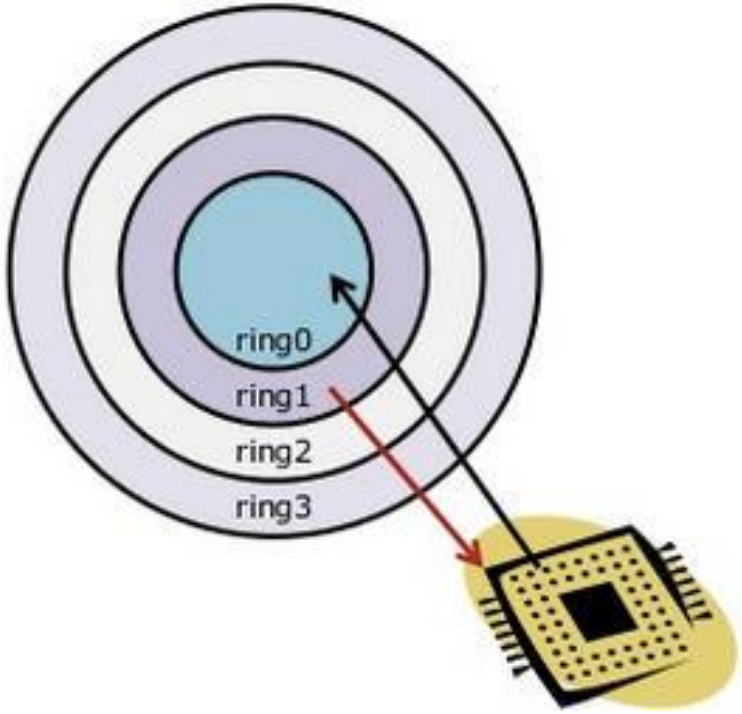
- 那么，虚拟化在这里就遇到了一个难题，因为宿主操作系统是工作在 ring0 的，客户操作系统就不能也在 ring0 了，但是它不知道这一点，以前执行什么指令，现在还是执行什么指令，但是没有执行权限是会出错的。所以这时候虚拟机管理程序（VMM）需要避免这件事情发生。 虚机怎么通过 VMM 实现 Guest CPU 对硬件的访问，根据其原理不同有三种实现技术：
- 1. 全虚拟化
- 2. 半虚拟化
- 3. 硬件辅助的虚拟化

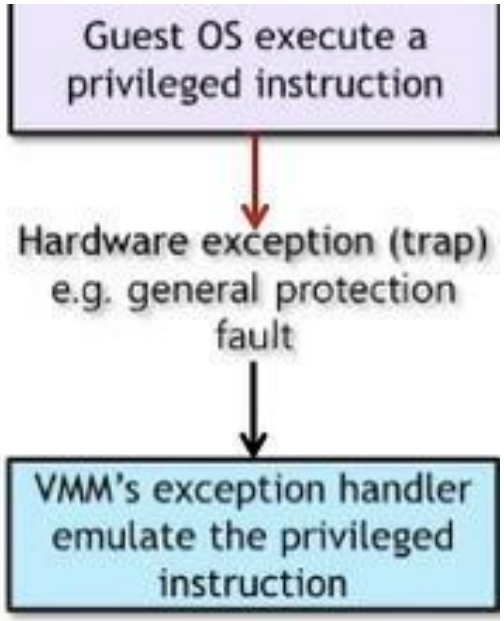
基于二进制翻译的全虚拟化（Full Virtualization with Binary Translation）

- 客户操作系统运行在 Ring 1，它在执行特权指令时，会触发异常（CPU的机制，没权限的指令会触发异常），然后 VMM 捕获这个异常，在异常里面做翻译，模拟，最后返回到客户操作系统内，客户操作系统认为自己的特权指令工作正常，继续运行。但是这个性能损耗，就非常的大， 简单的一条指令，执行完，了事，现在却要通过复杂的异常处理过程


Ring 3
User Apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host Computer
System Hardware
Direct
Execution
of User
Requests
Binary
Translation
of OS
Requests

异常“捕获（trap）-翻译（handle）-模拟（emulate）”过程：

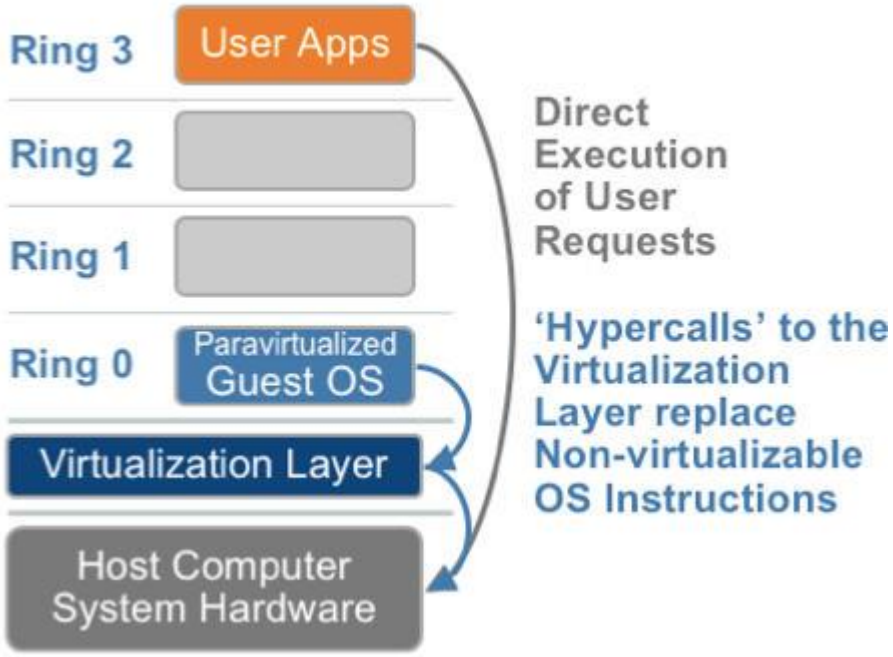




半虚拟化

- 半虚拟化的思想就是，修改操作系统内核，替换掉不能虚拟化的指令，通过超级调用（hypercall）直接和底层的虚拟化层hypervisor来通讯，hypervisor 同时也提供了超级调用接口来满足其他关键内核操作，比如内存管理、中断和时间保持。
- 这种做法省去了全虚拟化中的捕获和模拟，大大提高了效率。所以像XEN这种半虚拟化技术，客户机操作系统都是有一个专门的定制内核版本，和x86、mips、arm这些内核版本等价。这样以来，就不会有捕获异常、翻译、模拟的过程了，性能损耗非常低。这就是XEN这种半虚拟化架构的优势。这也是为什么XEN只支持虚拟化Linux，无法虚拟化windows原因，微软不改代码啊。

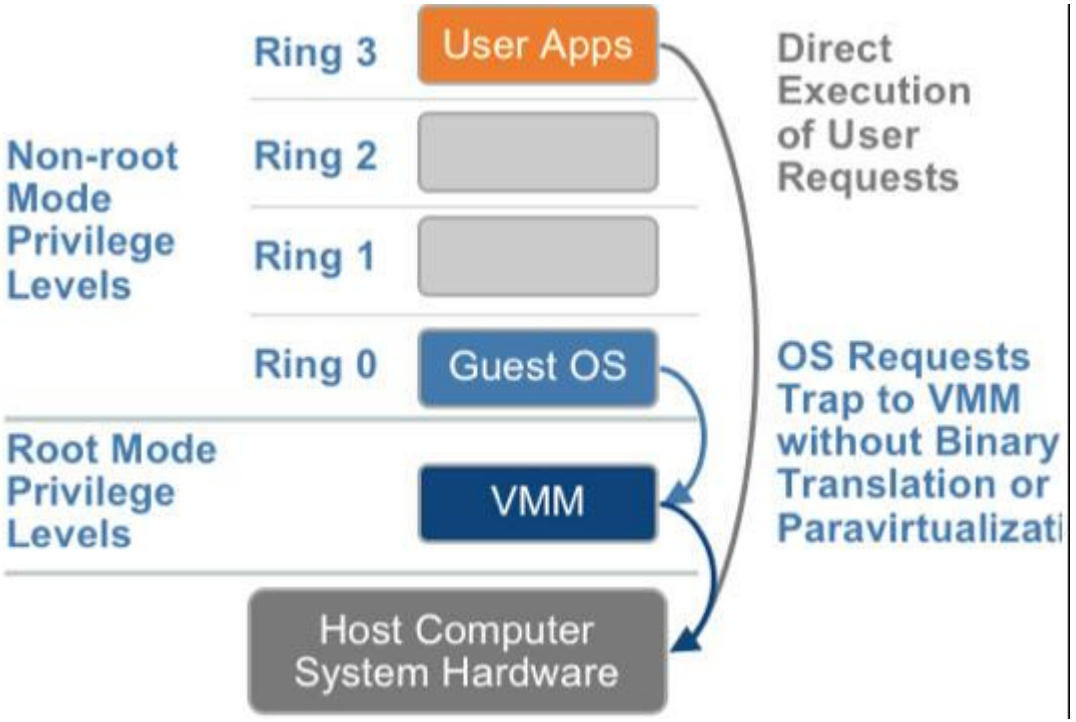
超虚拟化（或者半虚拟化/操作系统辅助虚拟化 Paravirtualization）



硬件辅助的全虚拟化

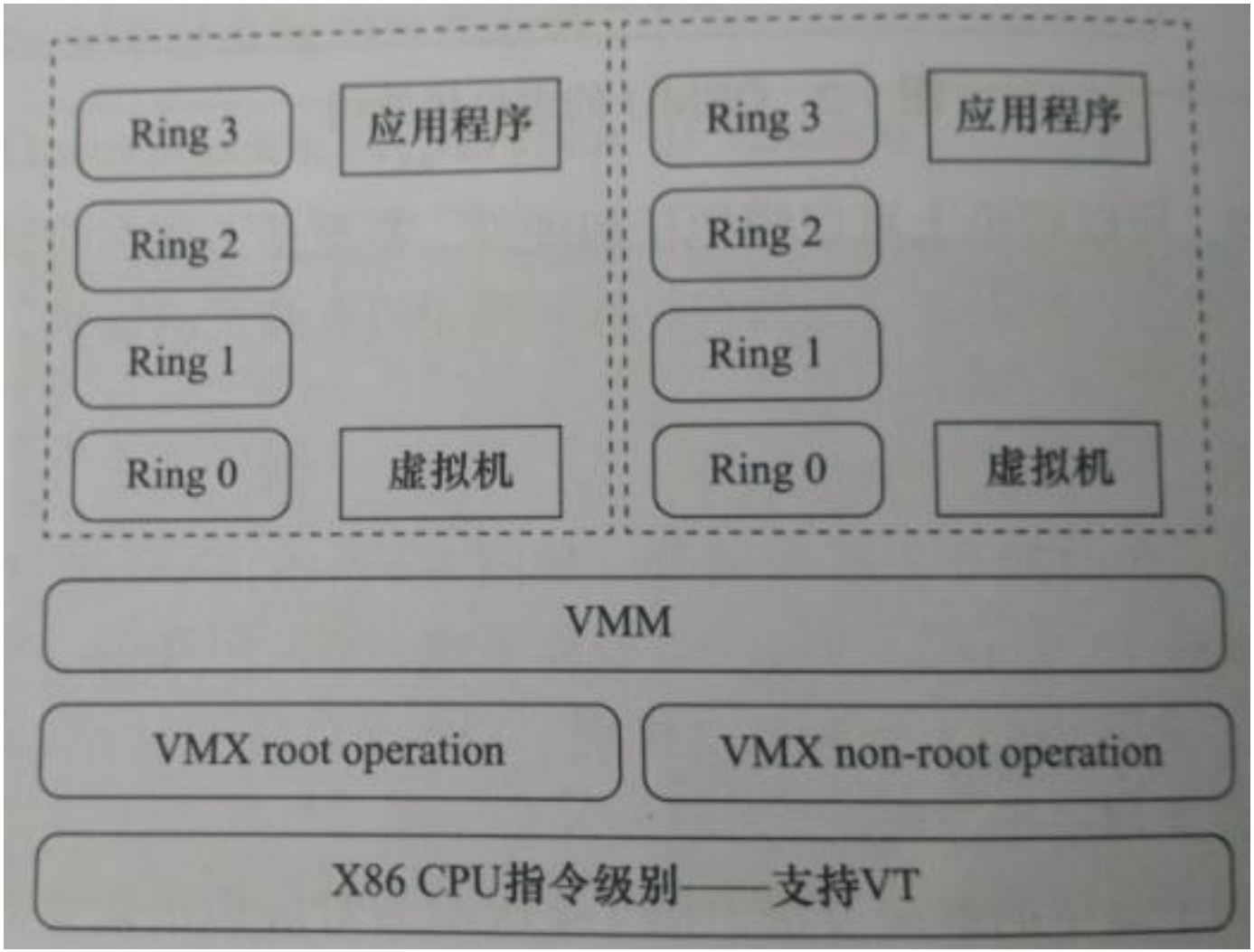
- 2005年后，CPU厂商Intel 和 AMD 开始支持虚拟化了。Intel 引入了 Intel-VT （Virtualization Technology）技术。这种 CPU，有 VMX root operation 和 VMX non-root operation两种模式，两种模式都支持Ring 0 ~ Ring 3 共 4 个运行级别。这样，VMM 可以运行在 VMX root operation 模式下，客户 OS 运行在VMX non-root operation模式下。

硬件辅助的全虚拟化


Ring 3
User Apps
Ring 2
Ring 1
Ring 0
Guest OS
VMM
Host Computer System Hardware
Non-root Mode Privilege Levels
Root Mode Privilege Levels
Direct Execution of User Requests
OS Requests Trap to VMM without Binary Translation or Paravirtualizati

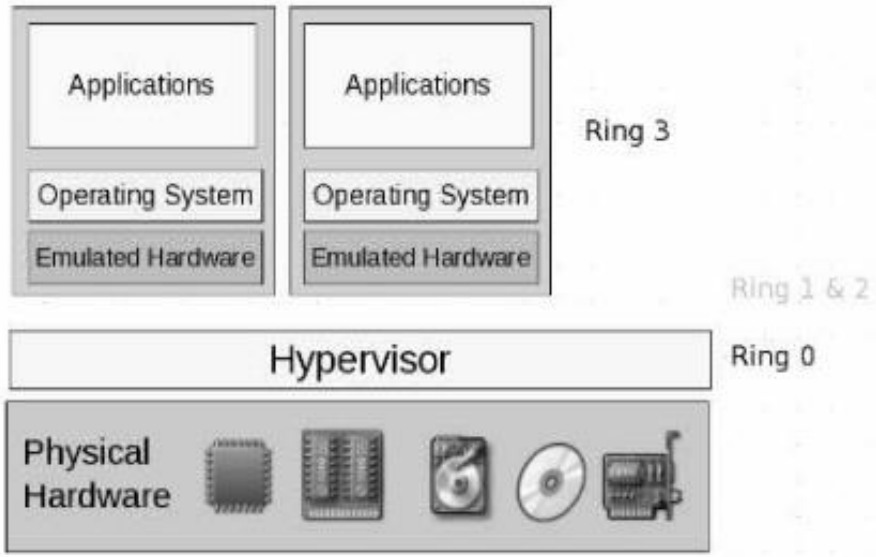
- 也就说，硬件这层就做了些区分，这样全虚拟化下，那些靠“捕获异常-翻译-模拟”的实现就不需要了。而且CPU厂商，支持虚拟化的力度越来越大，靠硬件辅助的全虚拟化技术的性能逐渐逼近半虚拟化，再加上全虚拟化不需要修改客户操作系统这一优势，全虚拟化技术应该是未来的发展趋势

Kvm虚拟化架构

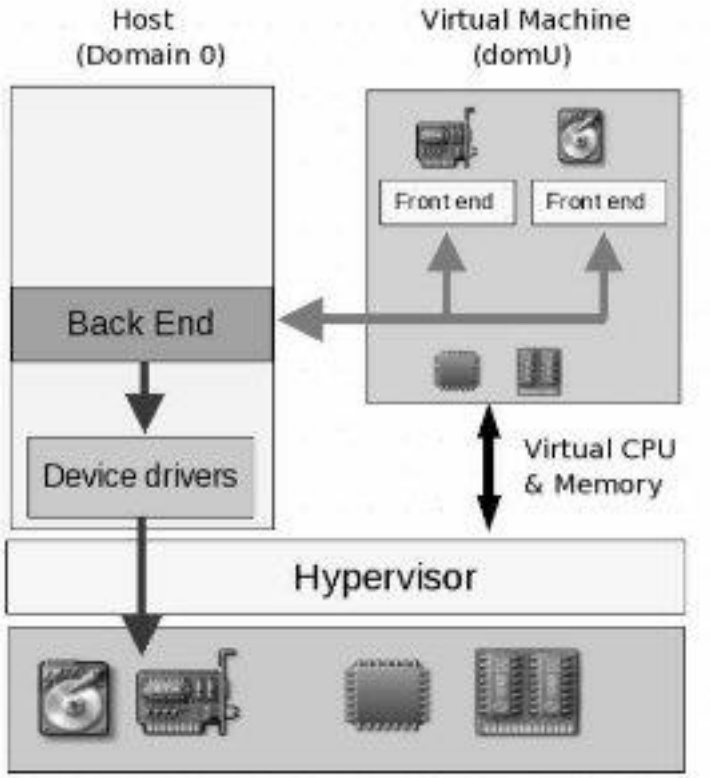

Ring 3
应用程序
Ring 2
Ring 1
Ring 0
虚拟机
Ring 3
应用程序
Ring 2
Ring 1
Ring 0
虚拟机
VMM
VMX root operation
VMX non-root operation
X86 CPU指令级别——支持VT

	利用二进制翻译的全虚拟化	硬件辅助虚拟化	操作系统协助/半虚拟化
实现技术	BT和直接执行	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 box	VMware ESXi/Microsoft Hyper-V/Xen 3.0/KVM	Xen

全虚拟化 .vs. 半虚拟化

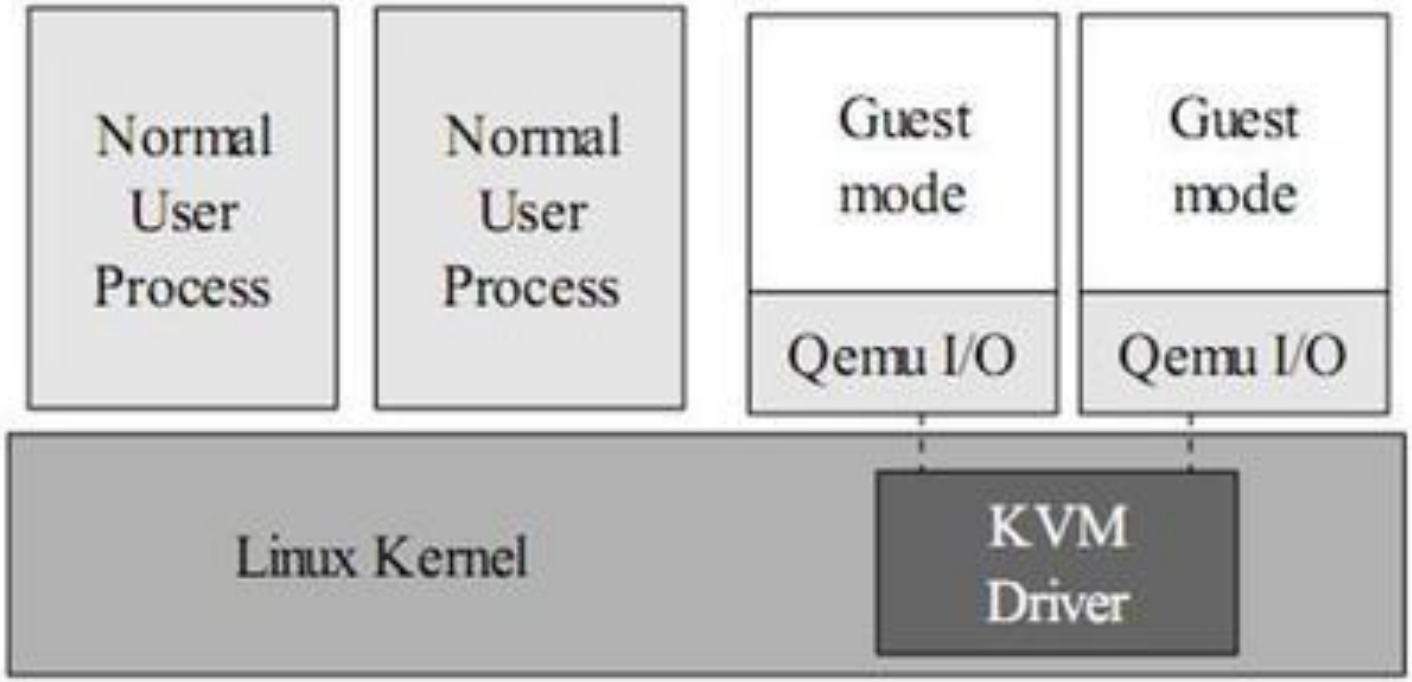


全虚拟化

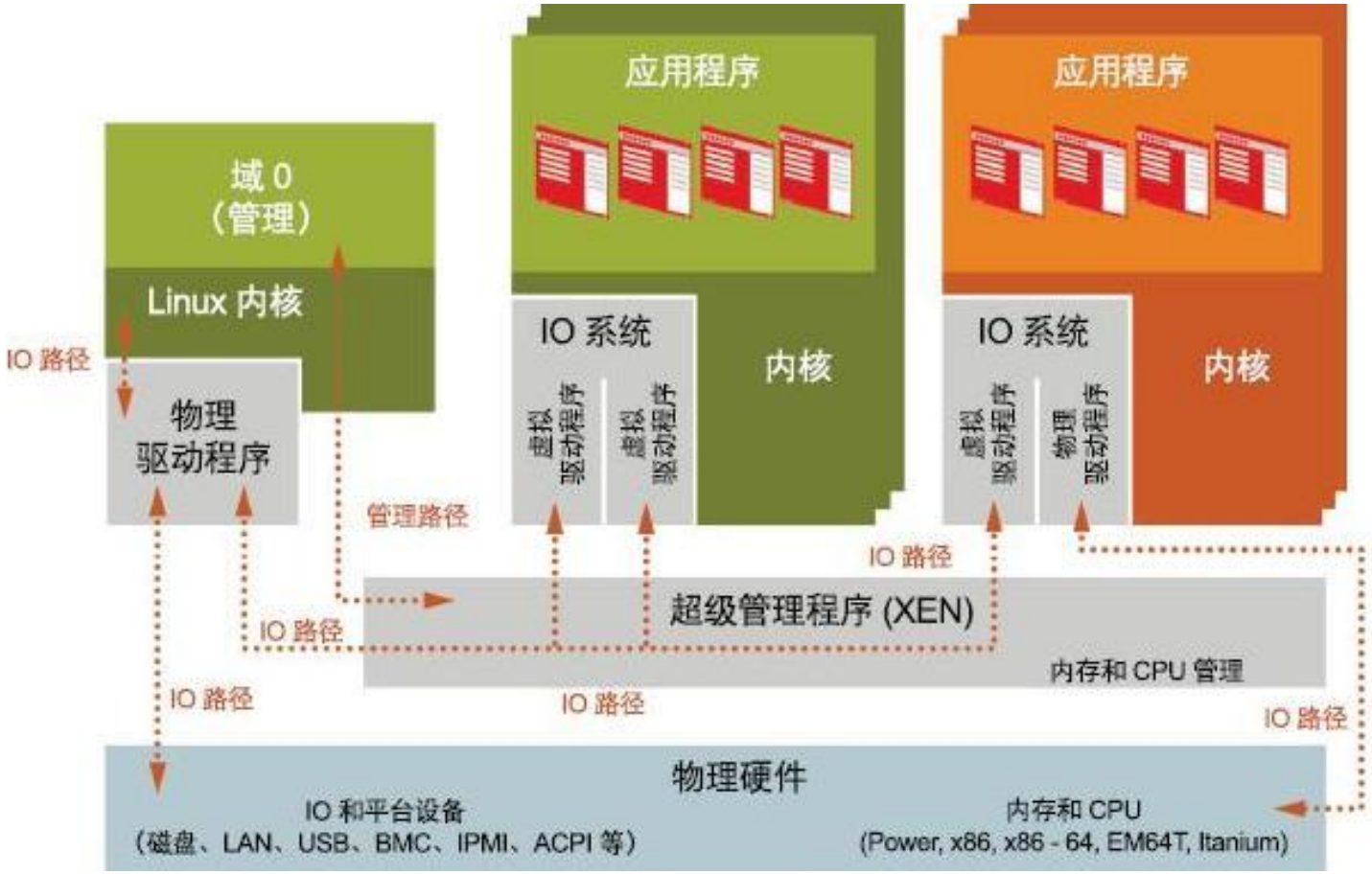


半虚拟化

Kvm架构图


Normal User Process
Normal User Process
Guest mode
Guest mode
Qemu I/O
Qemu I/O
Linux Kernel
KVM Driver

Xen架构图


应用程序
应用程序
域 0
（管理）
Linux 内核
IO 系统
内核
IO 系统
内核
IO 路径
物理
驱动程序
虚拟
驱动程序
虚拟
驱动程序
虚拟
驱动程序
物理
驱动程序
管理路径
IO 路径
超级管理程序 (XEN)
IO 路径
内存和 CPU 管理
IO 路径
IO 路径
IO 路径
物理硬件
IO 和平台设备
（磁盘、LAN、USB、BMC、IPMI、ACPI 等）
内存和 CPU
(Power, x86, x86 - 64, EM64T, Itanium)

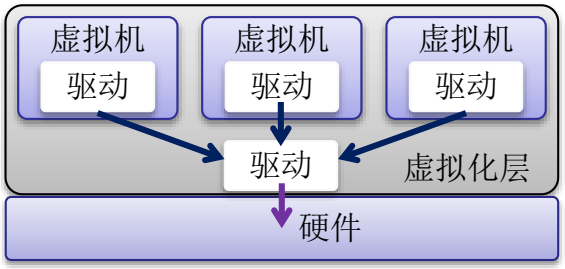
Xen包含三大部分

- **Xen Hypervisor**：直接运行于硬件之上是Xen客户操作系统与硬件资源之间的访问接口(如：)。通过将客户操作系统与硬件进行分类，Xen管理系统可以允许客户操作系统安全，独立的运行在相同硬件环境之上。
- **Domain 0**：运行在Xen管理程序之上，具有直接访问硬件和管理其他客户操作系统的特权的客户操作系统。
- **DomainU**：运行在Xen管理程序之上的普通客户操作系统或业务操作系统，不能直接访问硬件资源（如：内存，硬盘等），但可以独立并行的存在多个。

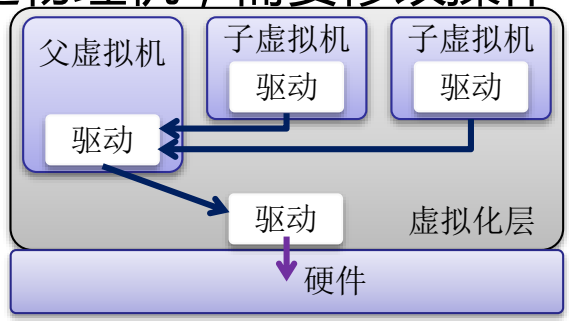
- Xen 半虚拟化
  - 可以运行特权指令直接在硬件上运行
  - Vm系统内核xen对齐进行了修改

服务器虚拟化方法

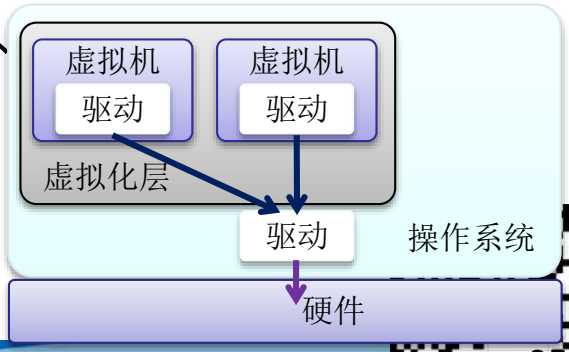
- **全虚拟化**(Full-Virtulization)：
  无需修改操作系统，VM ESXi、Linux KVM



- **半虚拟化**(Para-Virtulization)：
  集成半虚拟化代码，直接运行特权指令，性能接近物理机，需要修改操作系统，
  MS Hyper-V、 Ctrix Xen、IBM PowerVM



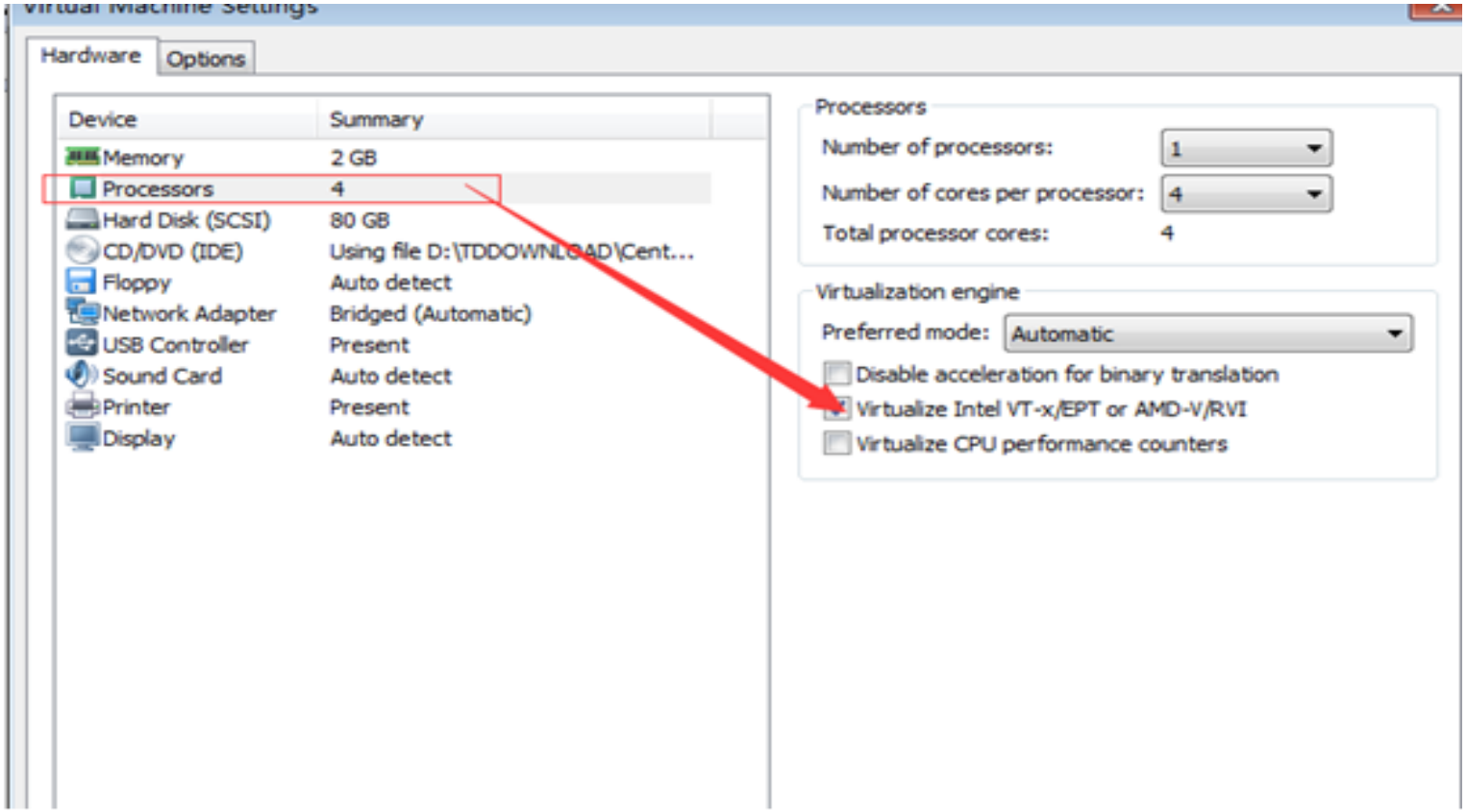
- **操作系统层虚拟化**
  开发、测试环境，VM Workstation、VM Server、



Cpu检测

- 支持vt技术
- **egrep -c '(vmx|svm)' /proc/cpuinfo**
- **命令结果大于0表示cpu支持虚拟化**

```
[root@localhost ~]# egrep -c '(vmx|svm)' /proc/cpuinfo
4
[root@localhost ~]#
```


Virtual Machine Settings
Hardware
Options
Device
Summary
Memory
2 GB
Processors
4
Hard Disk (SCSI)
80 GB
CD/DVD (IDE)
Using file D:\TDDOWNLOAD\Cent...
Floppy
Auto detect
Network Adapter
Bridged (Automatic)
USB Controller
Present
Sound Card
Auto detect
Printer
Present
Display
Auto detect
Processors
Number of processors:
1
Number of cores per processor:
4
Total processor cores:
4
Virtualization engine
Preferred mode:
Automatic
Disable acceleration for binary translation
Virtualize Intel VT-x/EPT or AMD-V/RVI
Virtualize CPU performance counters

Kvm安装

- 安装命令kvm：

  yum install qemu-kvm

- 安装虚拟化管理工具:

  yum install virt-manager libvirt libvirt-python python-virtinst bridge-utils

- libvirt：操作和管理KVM虚机的虚拟化 API，使用 C 语言编写，可以由 Python,Ruby, Perl, PHP, Java 等语言调用。可以操作包括 KVM，vmware，XEN，Hyper-v, LXC 等 Hypervisor。
- Virsh：基于 libvirt 的 命令行工具（CLI）
- Virt-Manager：基于 libvirt 的 GUI 工具

kvm 内核模块

lsmod | grep kvm

```
[root@localhost ~]# lsmod | grep kvm
kvm_intel              55496  0
kvm                   337900  1 kvm_intel
[root@localhost ~]#
```

modprobe kvm

modprobe kvm-intel

```
[root@localhost ~]# modprobe kvm
[root@localhost ~]# modprobe kvm-intel
```

- **Service iptables stop**
- **Chkconfig iptables off**

- **service libvirtd restart**
- chkconfig libvirtd on

创建虚拟机硬盘

- 创建安装盘：

qemu-img create -f qcow2 /kvmtest/centos-6.6.qcow2 10G

默认网络配置文件

/etc/libvirt/qemu/networks

创建虚拟机

```
virt-install --virt-type kvm --name centos-6.6 --ram 1024 \
--vcpus 1 \
--cdrom=/kvmtest/CentOS-6.6-x86_64-minimal.iso \
--disk /kvmtest/centos-6.6.qcow2,format=qcow2 \
--network network=default \
--graphics vnc,listen=0.0.0.0 --noautoconsole \
--os-type=linux --os-variant=rhel6
```

virt-install 常用参数一览表

1. -n NAME
   指定虚拟机的名称
2. -r MEMORY
   指定虚拟机的内存用量
3. -u UUID
   指定虚拟机的唯一通用标识符
   （Universally Unique Identifier，UUID）。
   省略这个参数时，virt-install 将会自动产生
4. --vcpus=VCPUS
   指定虚拟机的虚拟 CPU （Virtual CPU，VCPU） 数量
   -f DISKFILE
   指定虚拟磁盘的文件路径名称

   -s DISKSIZE
   用来指定虚拟磁盘的大小，这个参数需配合-f 使用。
   DISKSIZE为虚拟磁盘的大小，单位是GB

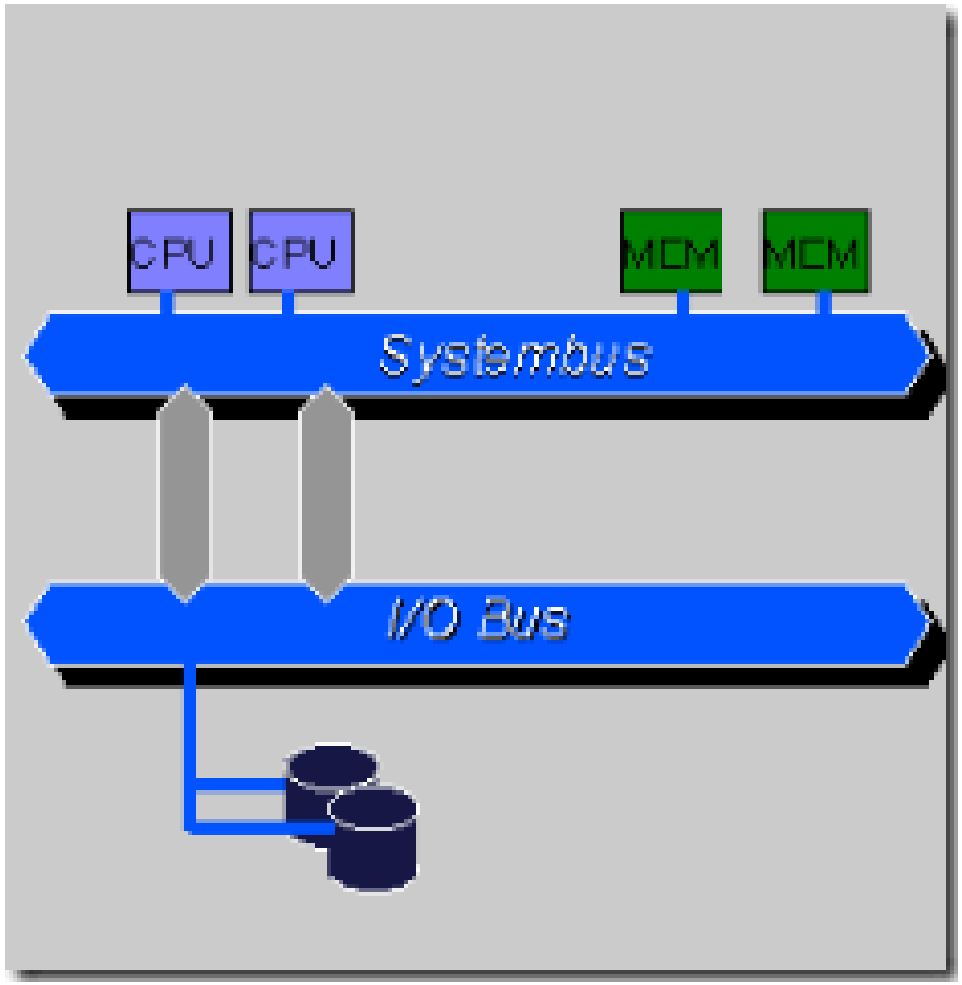
   -m MAC
   指定虚拟机的网络卡之硬件地址。
   这个参数可以省略，省略时virt-install 将自动产生

   -p 以半虚拟化的方式建立虚拟机

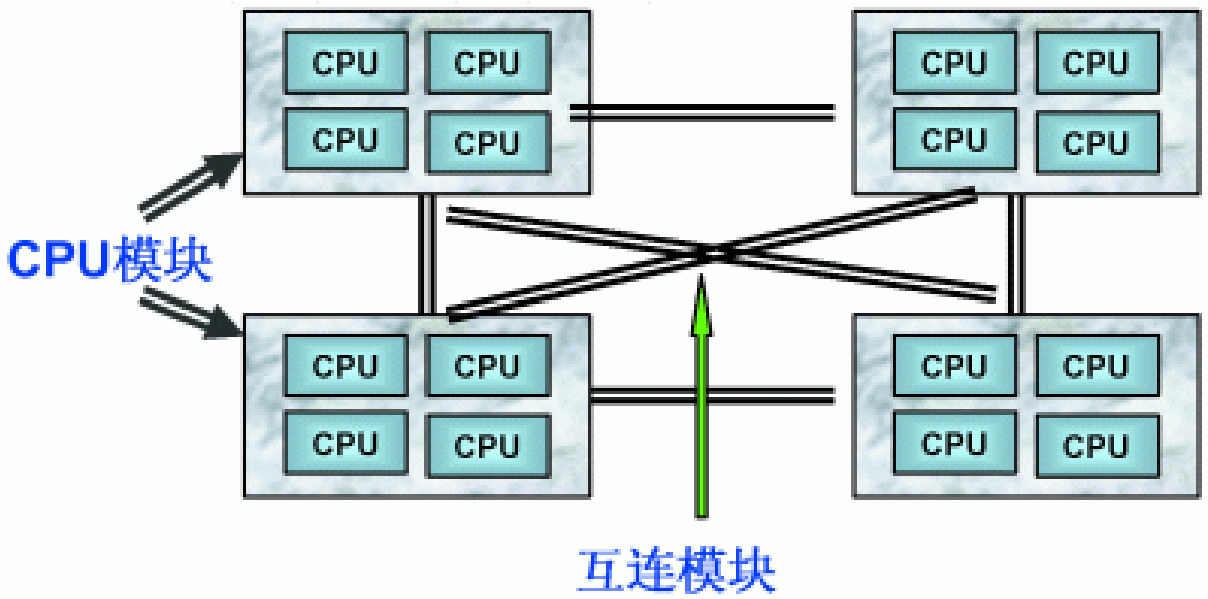
   -l LOCATION 指定安装来源

SMP(Symmetric Multi-Processor)

- SMP (Symmetric Multi Processing),对称多处理系统内有许多紧耦合多处理器，在这样的系统中，所有的CPU共享全部资源，如总线，内存和I/O系统等，操作系统或管理数据库的复本只有一个，这种系统有一个最大的特点就是共享所有资源。多个CPU之间没有区别，平等地访问内存、外设、一个操作系统。操作系统管理着一个队列，每个处理器依次处理队列中的进程。如果两个处理器同时请求访问一个资源（例如同一段内存地址），由硬件、软件的锁机制去解决资源争用问题。


CPU
CPU
MCM
MCM
Systembus
I/O Bus

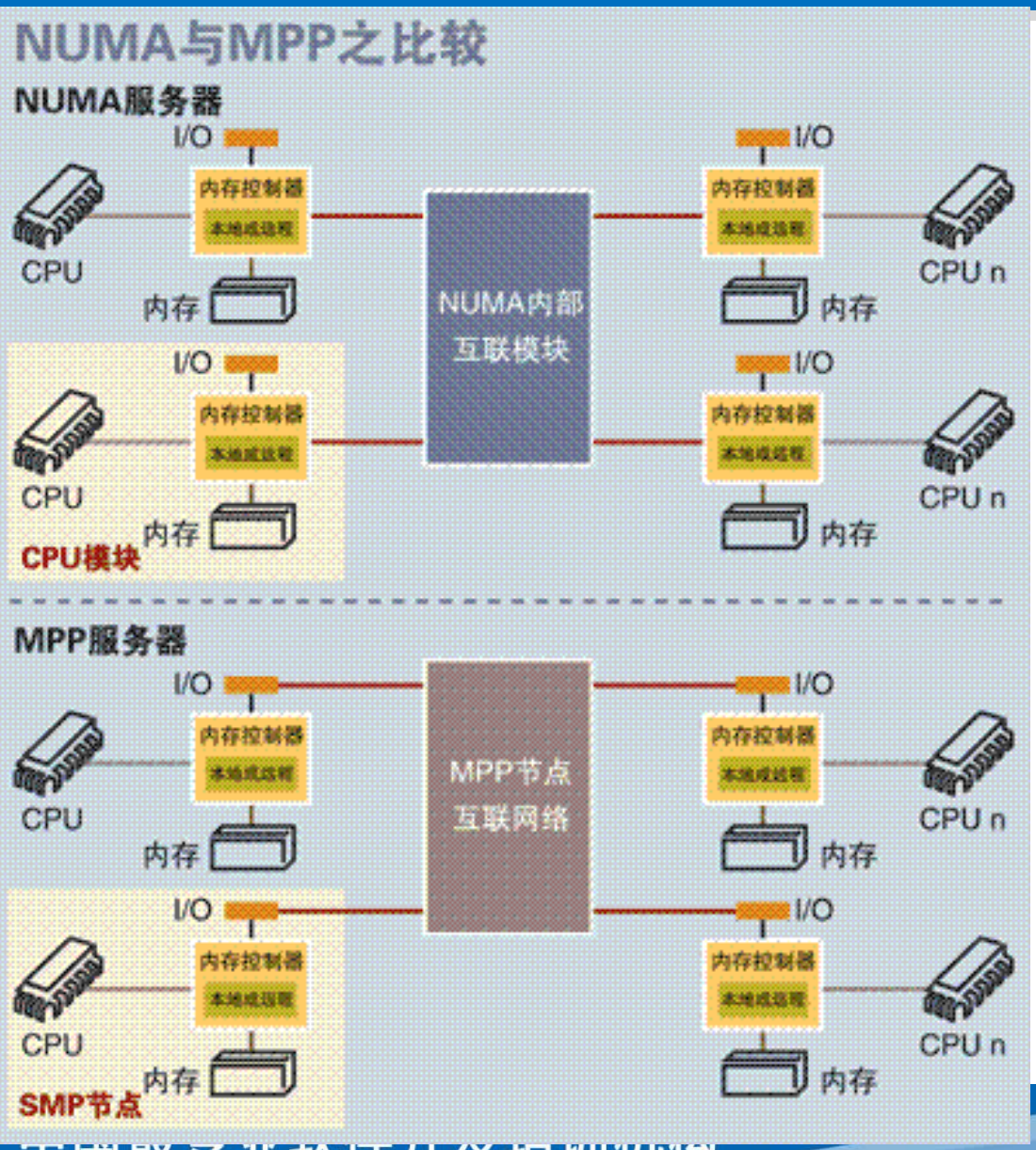
- NUMA 服务器的基本特征是具有多个 CPU 模块，每个 CPU 模块由多个 CPU(如 4 个) 组成，并且具有独立的本地内存、I/O 槽口等。由于其节点之间可以通过互联模块 (如称为 Crossbar Switch) 进行连接和信息交互，因此每个 CPU 可以访问整个系统的内存 (这是 NUMA 系统与 MPP 系统的重要差别)。显然，访问本地内存的速度将远远高于访问远地内存 (系统内其它节点的内存) 的速度，这也是非一致存储访问 NUMA 的由来。由于这个特点，为了更好地发挥系统性能，开发应用程序时需要尽量减少不同 CPU 模块之间的信息交互。


CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU模块
互连模块

MPP(Massive Parallel Processing)

- 和 NUMA 不同， MPP 提供了另外一种进行系统扩展的方式，它由多个 SMP 服务器通过一定的节点互联网络进行连接，协同工作，完成相同的任务，从用户的角度来看是一个服务器系统。其基本特征是由多个 SMP 服务器（每个 SMP 服务器称节点）通过节点互联网络连接而成，每个节点只访问自己的本地资源（内存、存储等），是一种完全无共享 (Share Nothing) 结构，因而扩展能力最好，理论上其扩展无限制，目前的技术可实现 512 个节点互联，数千个 CPU 。目前业界对节点互联网络暂无标准，如 NCR 的 Bynet ， IBM 的 SPSwitch ，它们都采用了不同的内部实现机制。但节点互联网仅供 MPP 服务器内部使用，对用户而言是透明的。

NUMA与MPP之比较



nat

- iptables -t nat -L -nv

Cpu硬件情况

```
[root@localhost ~]# numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3
node 0 size: 4095 MB
node 0 free: 3104 MB
node distances:
node   0
  0:  10
```

numastat

```
[root@localhost ~]# numastat
                           node0
numa_hit                  524959
numa_miss                      0
numa_foreign                   0
interleave_hit             18869
local_node                524959
other_node                     0
```

```
[root@localhost ~]# numastat -c qemu-kvm

Per-node process memory usage (in MBs)
PID               Node 0 Total
---------------  ------ -----
2167 (qemu-kvm)     267   267
2217 (qemu-kvm)     254   254
---------------  ------ -----
Total               521   521
```

Cpu


```
virsh # nodeinfo
CPU model:           x86_64
CPU(s):              4
CPU frequency:       2394 MHz
CPU socket(s):       1
Core(s) per socket:  4
Thread(s) per core:  1
NUMA cell(s):        1
Memory size:         3917768 KiB
```


VNC Viewer : Connection Details
Server:
192.168.1.201
Encryption:
Always Off
主机ip
About...
Options...
OK
Cancel

安装界面


QEMU (centos-6.4)
Welcome to CentOS 6.5!
Install or upgrade an existing system
Install system with basic video driver
Rescue installed system
Boot from local drive
Memory test
Press [Tab] to edit options
Automatic boot in 37 seconds...
CentOS 6
Community ENTerprise Operating System

Kvm桥接模式

- 创建桥接网卡
  - br0

常用命令

- Dumpxml vmid
- net-dumpxml default
  - 默认网络
- 列出虚拟机的所有网口：
- virsh domiflist domain
- 列出所有的块设备
- domblklist centos-6.6

- .启动，关闭和重启一个虚拟机
- virsh start domain_name
- virsh shutdown domain_name
- virsh reboot domain_name

虚拟机定义文件

- /etc/libvirt/qemu
  - 在这个目录下

虚拟机克隆

- 1.根据模板文件修改
  - 修改虚拟名称
  - 删除uuid
  - 指定disk
  - 删除mac

- 2.复制qcow2文件
- 3.virsh define 模板文件路径

桥接和nat访问的定义

使用网桥类型。确保每个kvm guest的mac地址唯一。将创建tun设备，名称为vnetx（x为0,1,2...）

```
<interface type='bridge'>

    <source bridge='br0'/>

    <mac address="3B:6E:01:69:3A:11"/>

</interface>
```

补充：使用默认的虚拟网络代替网桥，即guest为NAT模式。也可以省略mac地址元素，这样将自动生成mac地址。

```
<interface type='network'>

    <source network='default'/>

    <mac address="3B:6E:01:69:3A:11"/>

</interface>
```

默认网络

```
virsh # net-dumpxml default
<network connections='1'>
  <name>default</name>
  <uuid>792a1190-de22-446c-9a0f-b9dfdf8e5d90</uuid>
  <forward mode='nat'/>
  <bridge name='virbr0' stp='on' delay='0' />
  <mac address='52:54:00:91:E5:32'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254'
    </dhcp>
  </ip>
</network>

virsh #
```

修改桥接后nat不能上网问题？


```
virsh # net-dumpxml default
<network>
  <name>default</name>
  <uuid>8bd7dad9-8111-4296-a8ca-f4225921a844</uuid>
  <forward dev='br0' mode='nat'>
    <interface dev='br0'/>
  </forward>
  <bridge name='virbr0' stp='on' delay='0' />
  <mac address='52:54:00:F1:67:77'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254' />
    </dhcp>
  </ip>
</network>
```


- virsh # net-define /etc/libvirt/qemu/network/default.xml
- virsh # net-destroy default
- virsh # net-start default

C语言 调用kvm 测试

- 见老师实战演示

python调用kvm 测试

- 见老师实战演示

eclipse卡死

- -Dorg.eclipse.swt.internal.gtk.cairoGraphics=false

Openstack 国内使用状况

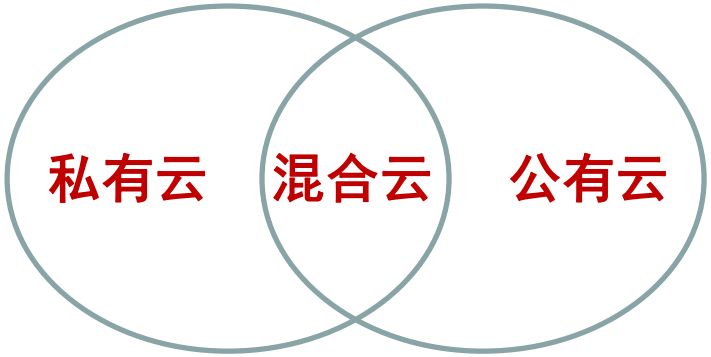
- http://www.csdn.net/article/2015-10-06/2825848
- http://www.csdn.net/article/2015-06-29/2825070
- http://www.csdn.net/article/2015-06-11/2824927
- http://www.csdn.net/article/2015-09-19/2825743
- http://www.csdn.net/article/2015-09-17/2825729
- http://www.csdn.net/article/2015-09-16/2825715-OpenStack
- http://www.csdn.net/article/2015-10-06/2825848
- http://www.csdn.net/article/2015-09-17/2825729
- http://www.csdn.net/article/2015-09-19/2825743

参考文章

- http://blog.chinaunix.net/uid-20917783-id-2560164.html
- http://blog.csdn.net/yuanchao99/article/details/18086743

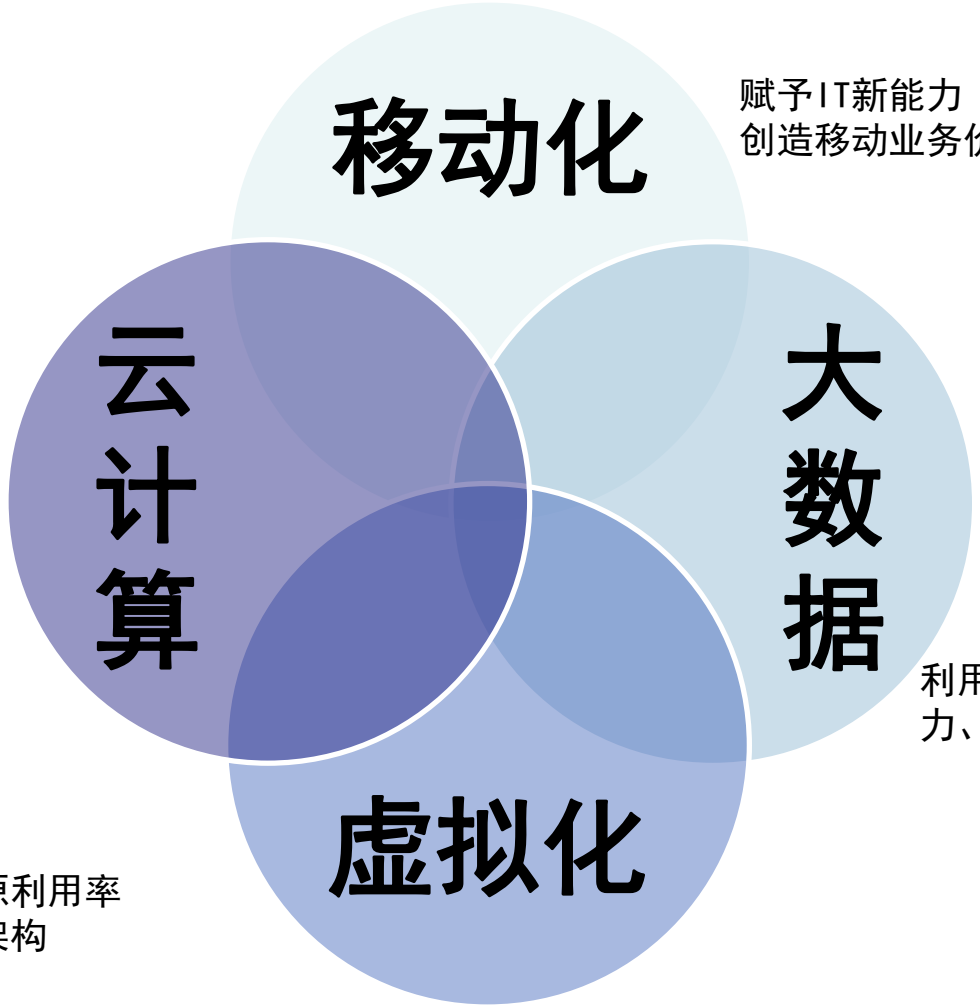
什么是云计算？

云计算将IT资源和功能以服务的方式通过网络交付给用户。**简单来说，就是把应用程序和数据都放在由大量服务器组成的云中，用户需要什么只要购买相应服务并使用即可。**



云计算模式

当今行业Top 4关键词



赋予IT新能力
创造移动业务价值

快速、按需定制云服务
降低成本、提升效率

利用海量数据提供更强的决策力、发现力和流程优化能力

资源整合，提升资源利用率
构建弹性数据中心架构

我们的课程为什么是这么架构的！

- Do you know!!!

联系方式

- 尚学堂大数据讨论群01 156927834
- 尚学堂大数据讨论群02 172599077
- 贾老师 1786418286
- 何老师 1926106490
- 詹老师 2805048645
- 张老师 3254755158

- 周末班开班时间：12月26日
- 脱产班开班时间：11月5日
- http://www.bjsxt.com/html/cloud/