北京航空航天大學
BEIHANG UNIVERSITY

云计算与大数据平台

王宝会　北航软件学院

2020年9月

VMM功能与组成

- 管理物理资源
  - CPU管理
  - 内存管理
  - 中断管理
  - 系统时间维护
  - 设备管理
- 管理虚拟环境
  - 虚拟环境（CPU、内存、设备虚拟化模块）
  - 虚拟环境调度：虚拟机处理上下文调度
  - 虚拟机间通信（特权域与虚拟机之间通信、普通虚拟机之间通讯）
  - 虚拟环境管理接口：为用户提供管理界面
- 其他模块
  - 软件定时器
  - 电源管理
  - 安全机制
  - 多处理器同步原语
  - 性能采集和分析工具
  - 调试工具

虚拟化技术

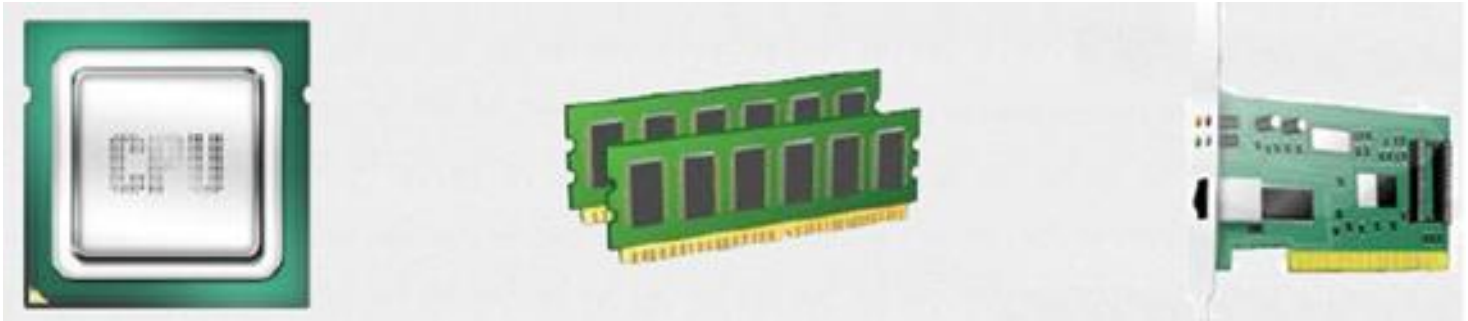
- **服务器虚拟化：**主流。将服务器物理资源抽象成逻辑资源，让一台服务器变成几台甚至上百台相互隔离的虚拟服务器，我们不再受限于物理上的界限，而是让CPU、内存、磁盘、I/O等硬件变成可以动态管理的“资源池”，从而提高资源的利用率，简化系统管理，实现服务器整合，让IT对业务的变化更具适应力
- **桌面虚拟化：**非主流到主流。一种基于服务器的计算模型，并且借用了传统的瘦客户端的模型，但是让管理员与用户能够同时获得两种方式的优点：将所有桌面虚拟机在数据中心进行托管并统一管理；同时用户能够获得完整PC的使用体验。桌面虚拟化是操作系统虚拟化技术+远程访问技术的结合，使得基于服务器的计算模型可以实现分布的计算模型的好处，同时管理效率大大提高。
- **应用虚拟化：**非主流。跨平台，MVC分层。应用虚拟化是将应用程序与操作系统解耦合，为应用程序提供了一个虚拟的运行环境。在这个环境中，不仅包括应用程序的可执行文件，还包括它所需要的运行时环境。从本质上说，应用虚拟化是把应用对低层的系统和硬件的依赖抽象出来，可以解决版本不兼容的问题。
- 网络虚拟化
- 存储虚拟化

计算、存储、网络

Openstack|GlusterFS

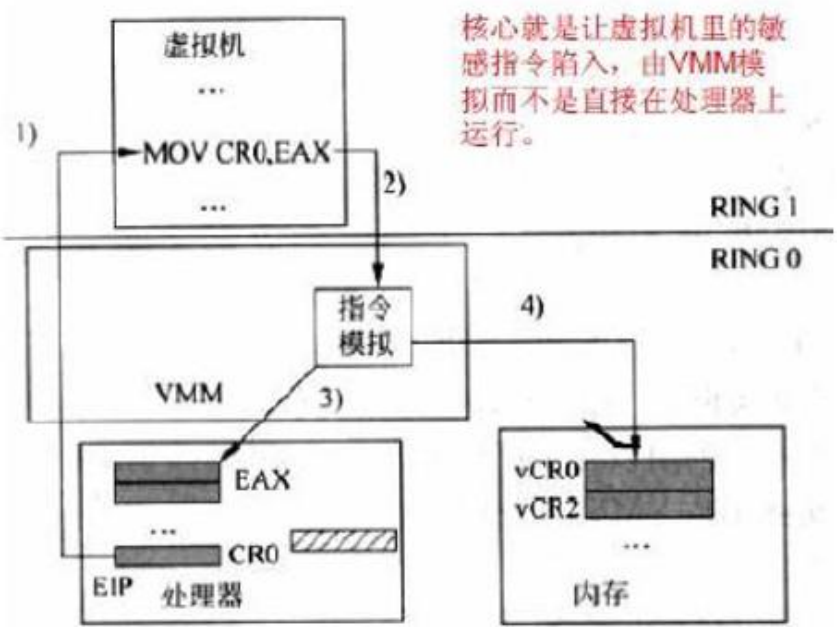
虚拟化原理

从现在开始，虚拟化就限定在服务器系统虚拟化的范畴了!
虚拟化软件对物理资源的虚拟可以归结为三个主要任务（要想虚拟一台计算机必须具备三要素）



CPU虚拟化　　内存虚拟化　　I/O虚拟化

虚拟化原理



- 处理器虚拟化

指令的模拟

传统方式:

1.操作系统内核运行在RING0;

2.对CPU有完全控制权;

3.直接读写寄存器，执行指令流;

4.直接切换上下文;

虚拟化以后:

1.客户操作系统内核运行在RING1;

2.VMM运行在RING0;

3.VMM为每个虚拟机开辟内存作为虚拟寄存器;

4.VMM切换整个虚拟处理器上下文（多用户多进程操作系统）;

5.客户操作系统在虚拟处理器中切换进程上下文;

虚拟化原理

●处理器虚拟化

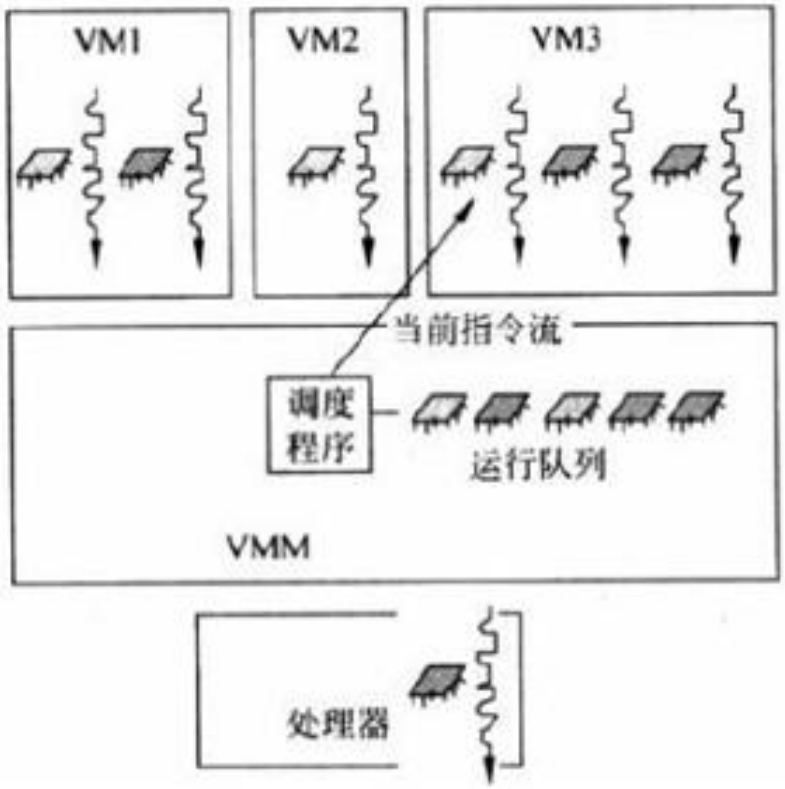
中断和异常（VMM主要手段）的模拟及注入

中断包含软中断和硬中断

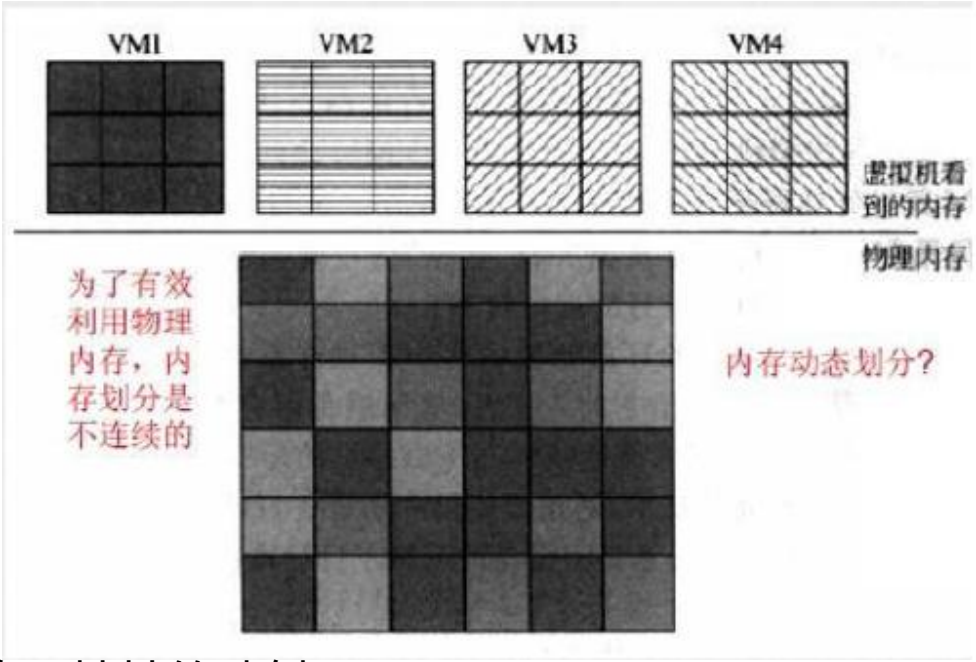
上下文切换

虚拟化原理

●处理器虚拟化
SMP的模拟：对称多处理器
多个处理器协同处理



虚拟化原理



●内存虚拟化

操作系统对内存的要求:
1.从物理地址0开始;
2.内存是连续的;

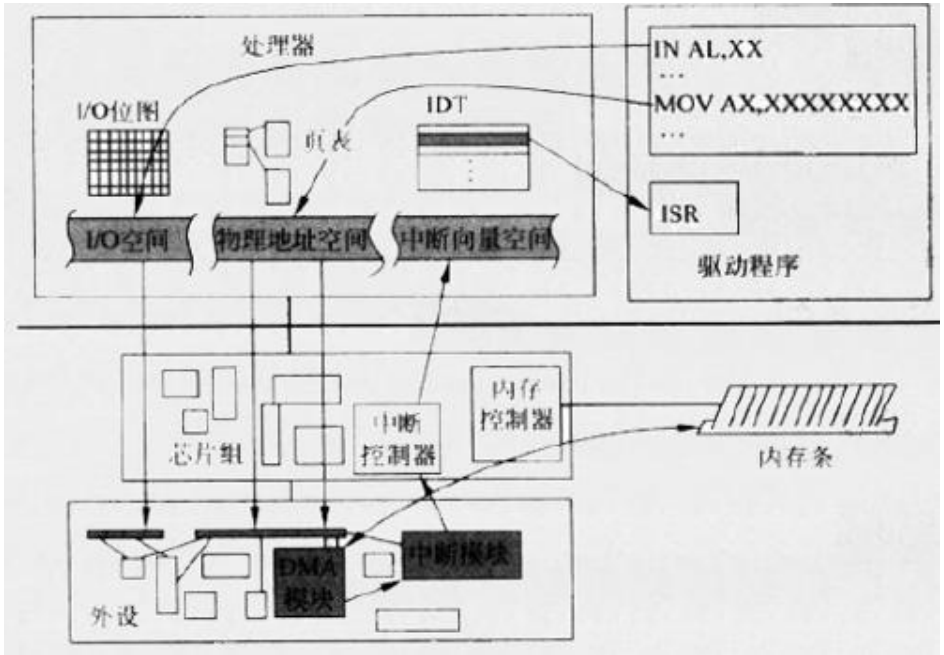
VMM要做的是:
1.维护客户机物理地址到宿主机物理地址的映射;
2.截获虚拟机对客户机物理地址的访问并转换为宿主机物理地址

虚拟机弹性配置有时需要重启

虚拟化原理

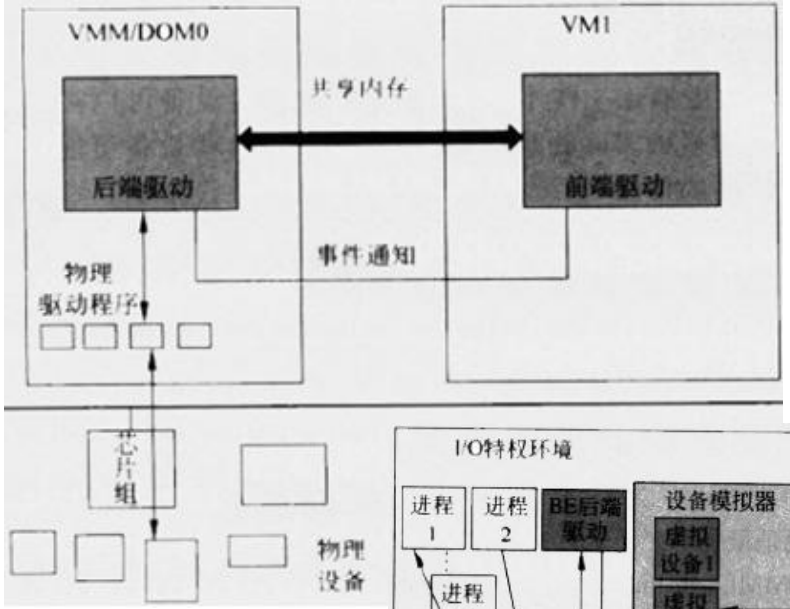
●计算机I/O原理

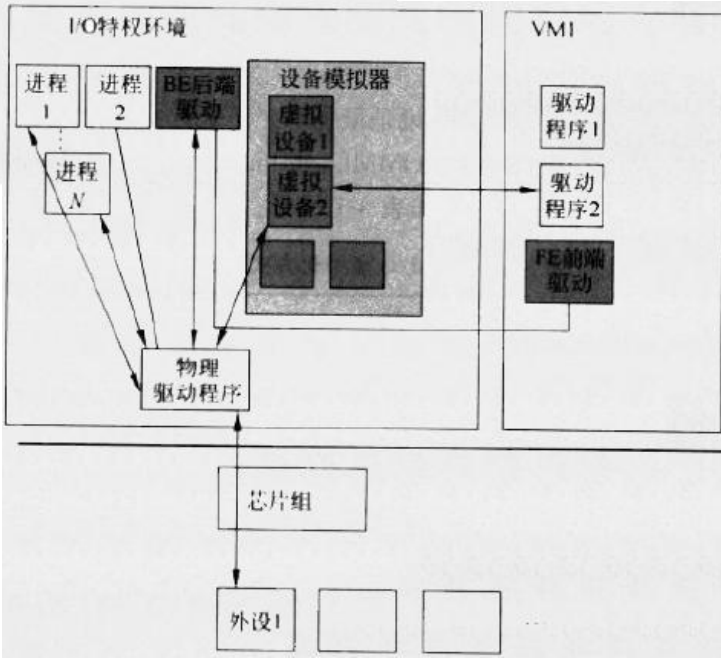
1.I/O端口寄存器被映射到I/O地址空间
2.中断模块向中断控制器发出中断信号
3.处理器中断当前指令流，通过IDT查找对应的中断服务程序
4.通过I/O访问端口执行I/O操作
5.在DMA情况下，MMIO寄存器会映射到物理地址空间，通过页表的方式来进行访问



DMA直接读取内存，不通过CPU

虚拟化原理





I/O虚拟化

1.设备发现: VMM提供一种方式，让客户操作系统发现虚拟设备，加载相关驱动

2.访问截获: VMM截获客户机操作系统对虚拟设备的访问，并进行模拟

3.设备模拟

4.设备共享

软件辅助的虚拟化技术

- CPU的虚拟化
-解释执行
-扫描与修补
-二进制代码翻译

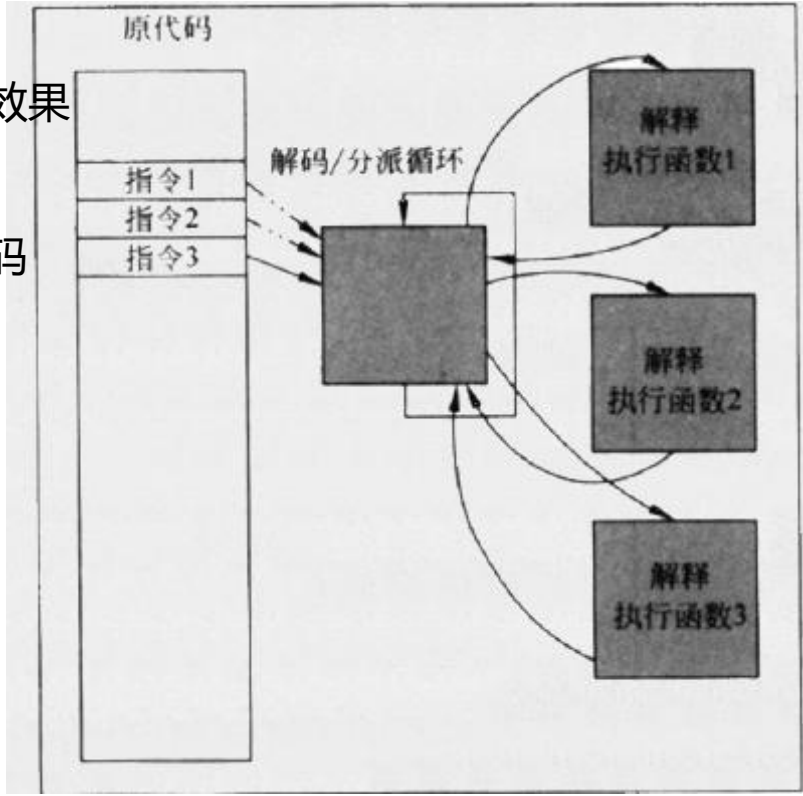
- 内存虚拟化
-影子页表
-内存虚拟化的优化技术

- I/O虚拟化
-I/O设备模型
-拦截和模拟

CPU虚拟化

- CPU的虚拟化
-基于软件的CPU完全虚拟化
-其本质就是软件模拟
- 几种模拟实现方式
-解释执行
-扫描与修补
-二进制代码翻译

- 取出一条指令
- 模拟出这条指令执行的效果
- 再继续取下一条指令
- 周而复始
- 每一条指令由解释器解码
- 调用解释执行函数
- 性能太差



CPU虚拟化

- CPU的虚拟化

-基于软件的CPU完全虚拟化

-其本质就是软件模拟

- 几种模拟实现方式

-解释执行

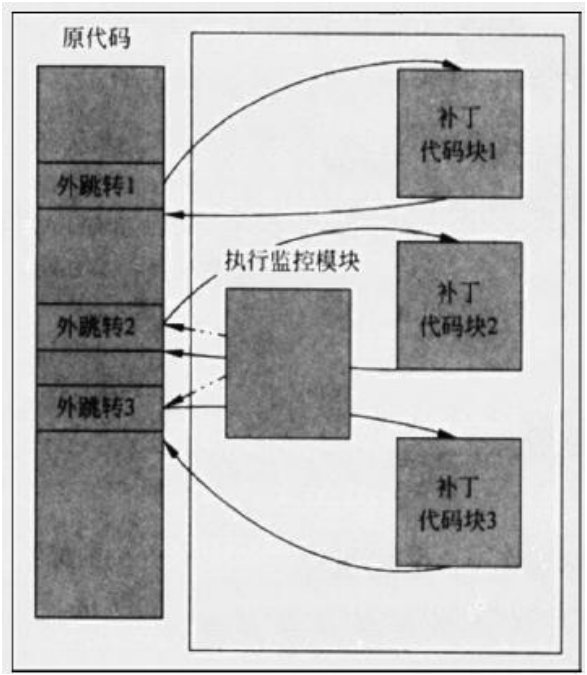
-扫描与修补

-二进制代码翻译

- 大部分非敏感指令直接在CPU上执行
- 执行每段代码之前进行扫描
- 将敏感指令替换为跳转指令或可陷入的指令

- 补丁代码是动态生成的
- 补丁代码存放在VMM内存空间
- 存放空间不足时，补丁代码可能会被逐出缓存

- 性能损失小



CPU虚拟化

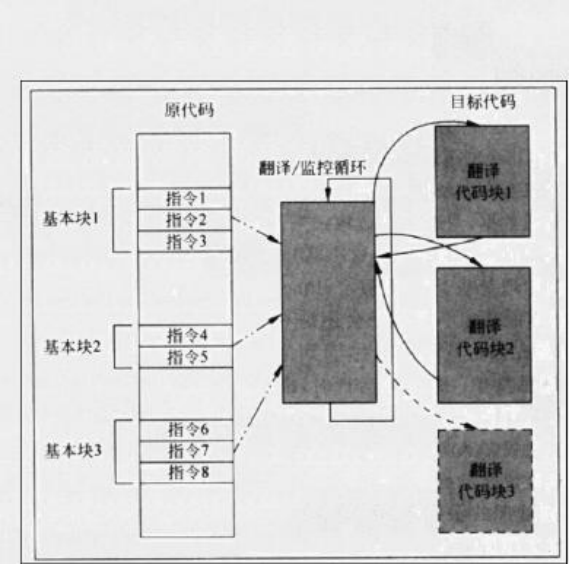
- CPU的虚拟化
-基于软件的CPU完全虚拟化
-其本质就是软件模拟
- 几种模拟实现方式
-解释执行
-扫描与修补
-二进制代码翻译

- 二进制代码翻译的难点
-自修改代码?
-资参考代码?
-异常的精确定位?
-实时代码时间精确度损失?

- 原始的客户机操作系统代码并不会被直接在物理CPU上执行，模拟器会以基本块为单位将其翻译成目标代码，然后执行

- 对基本快进行反汇编
- 通过翻译模块将其变成中间形式
- 进行代码优化
- 生成目标代码

- 性能最好



内存虚拟化

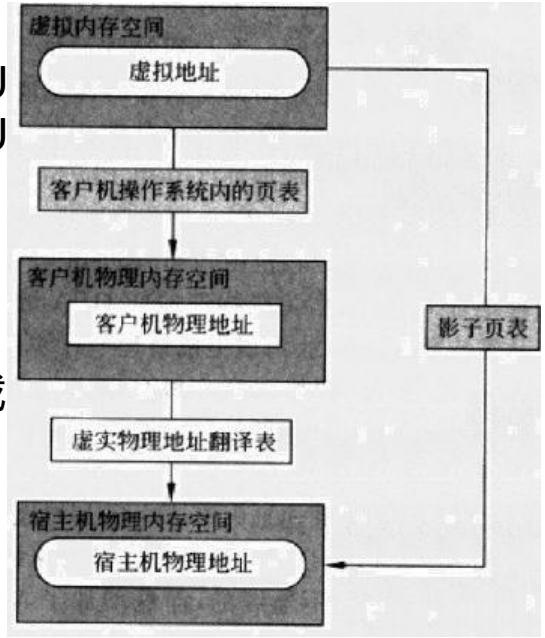
- 目的

-提供给客户操作系统一个从0地址开始的连续内存空间
-虚拟机之间有效隔离，调度，共享内存资源

- 方式

-GVA→GPA → HPA
-GVA→ HPA（影子列表）

- 其目的就是GVA直接转换到HPA
- 客户操作系统不能直接操作MMU
- 客户操作系统操作的是虚拟MMU
- 客户机的页表被载到虚拟MMU
- 物理MMU装载影子页表
- 每个客户机会有一套自己的影子页表
- 客户机更新页表时，VMM需要截获此行为并同步更新对应的影子页表



内存虚拟化的优化

- 自伸缩内存调节技术

-在客户机内安装一个伪设备驱动或一个内核模块：“气球”模块
-“气球”模块只负责和VMM交互
-当VMM要回收分配给客户机的内存是通知“气球”
-“气球”向客户及操作系统申请内存
-“气球”申请内存后并不会使用它，而是由VMM回收
-这部分内存就可以分配给其他需要的虚拟机
-**想想内存的归还过程?**

- 页共享技术 （copy on write）

-一个宿主机上运行很多客户机，会造成很多物理页中的内容是相同（运行相同的操作系统，执行相同的程序等等）
-VMM控制共享这些内容相同的页，删除其他页。
-此时读取时没问题的
-当客户操作系统对某个页进行修改时，发生缺页，VMM为其分配新页，并拷贝内容到新页。（写时备份）

怎样搜索相同页?

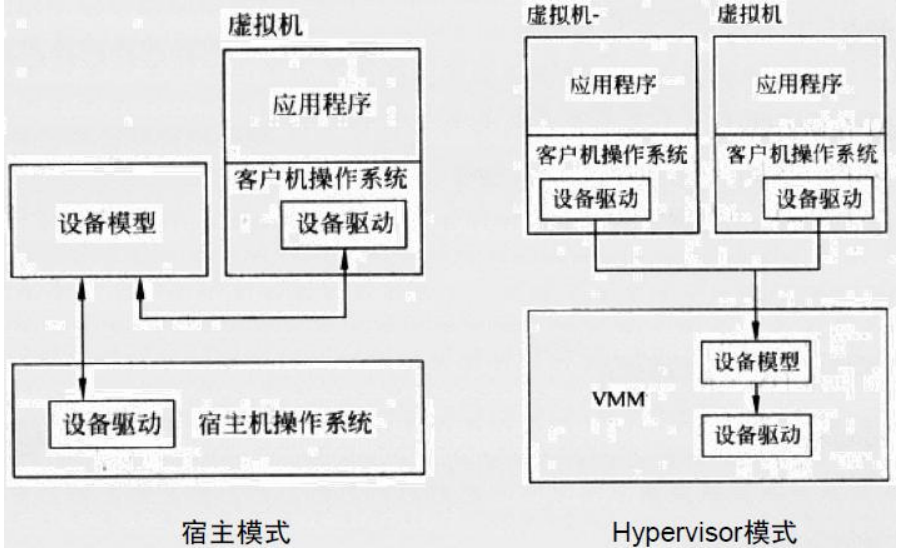
计算并记录每个页内数据的hash值，如果Hash值相同说明很有可能两个页的内容相同，然后对两个页的内容进行比较以确定是否相同。

I/O虚拟化

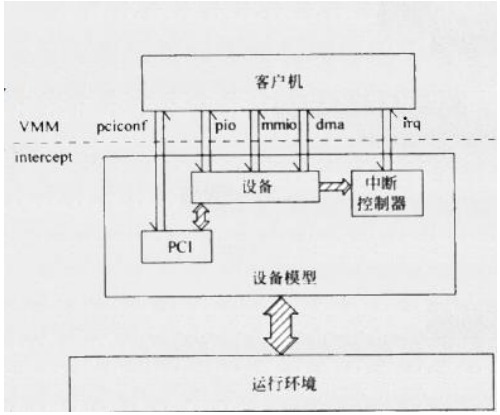
- 一个重要的概念：设备模型。
- 所谓设备模型是指VMM中进行设备模拟，并处理所有设备请求和响应的逻辑模块

- 作用

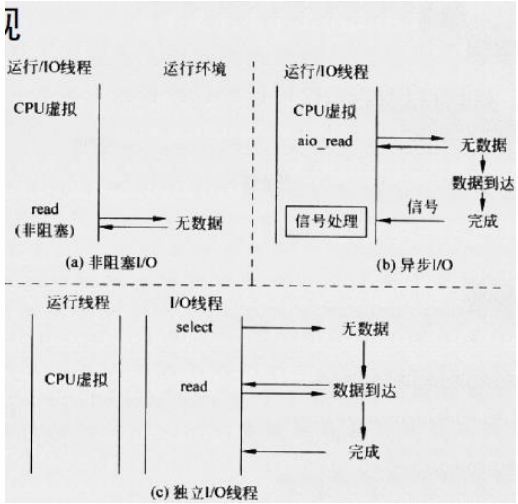
-模拟目标设备的软件接口
-实现目标设备的功能



设备模型的软件接口
-PCI配置空间：发现和识别设备
-端口IO：in,out,ins,outs
-MMIO:设备寄存器访问
-DMA：直接内存访问
-中断：消息通知



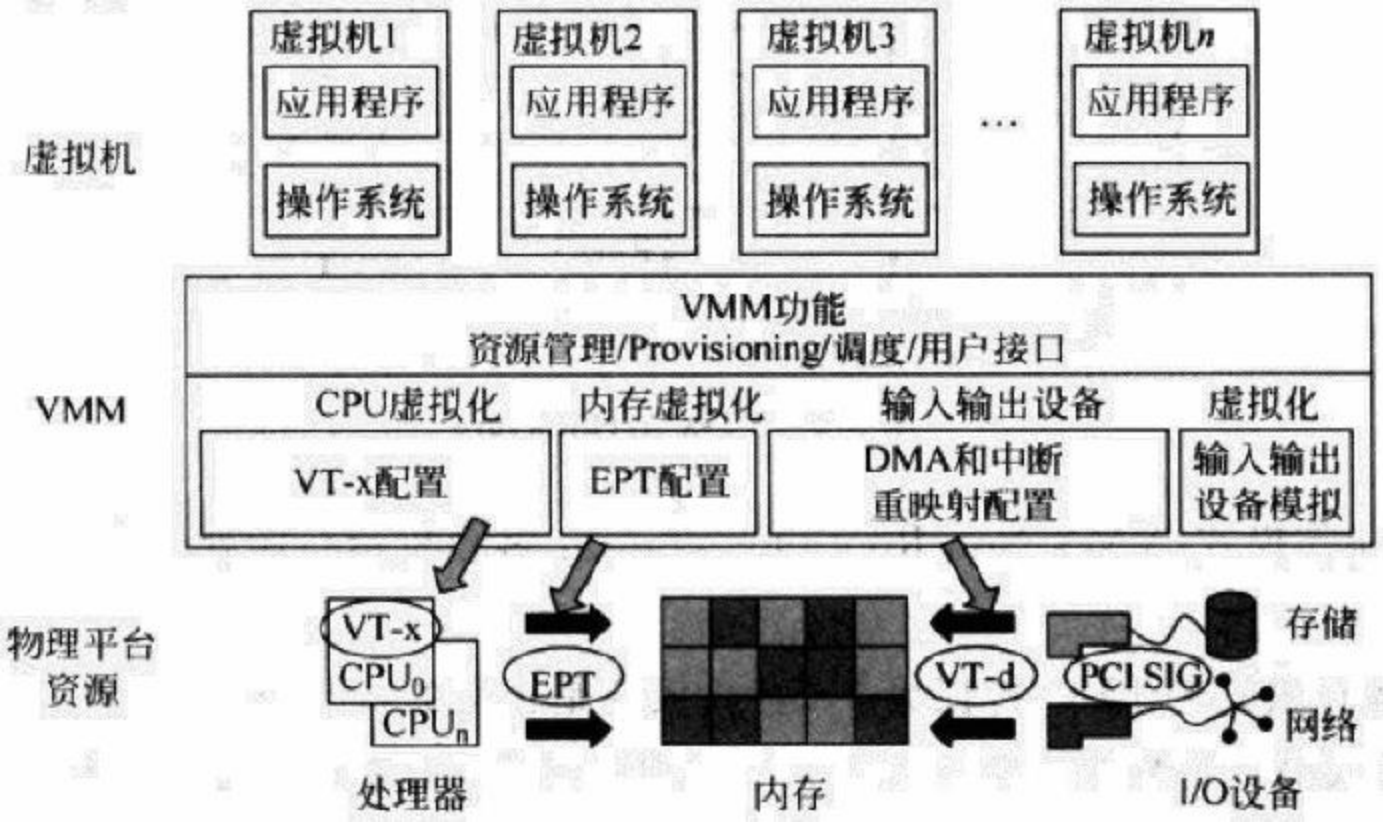
设备模型的功能实现
-非阻塞IO（性能差）
-异步IO（性能好）
-独立IO线程（性能最好）



硬件辅助的虚拟化技术

IntelVT的总体结构
CPU虚拟化的硬件支持（VT-x）
内存虚拟化的硬件支持（EPT）
IO虚拟化的硬件支持（VT-d）
时间虚拟化

Intel VT的总体结构

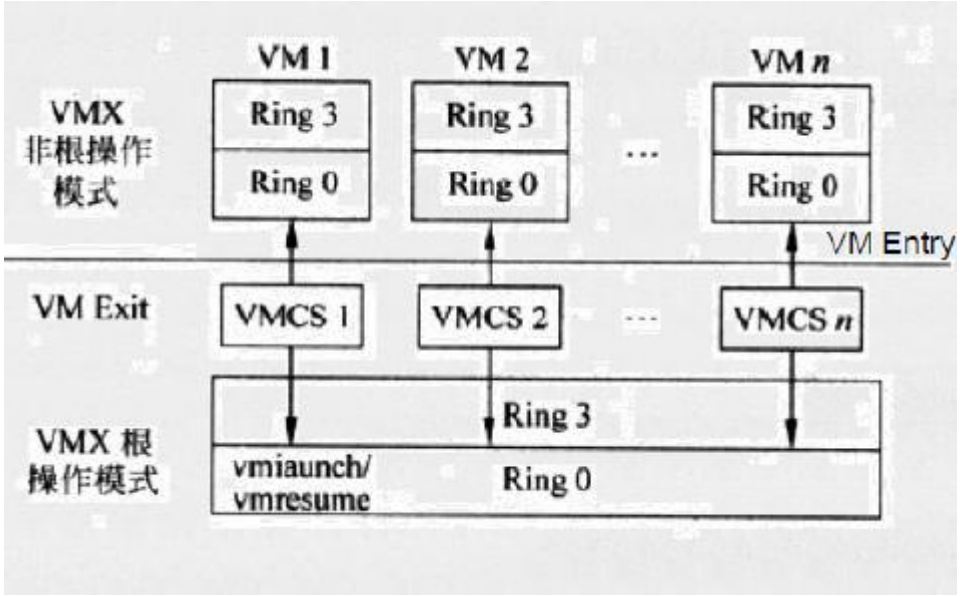


CPU虚拟化的硬件支持

VT-x引入了两种操作模式，统称VMX模式

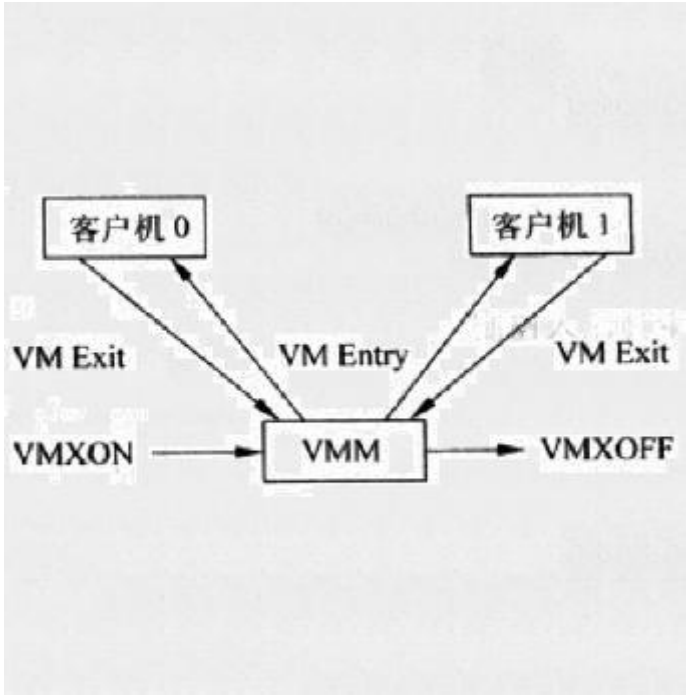
- 根模式
-VMM运行的模式
-传统模式
- 非根模式
-客户操作系统运行的模式
-所有敏感指令都将陷入
-陷入处理：VM-Exit

- 每种模式中都有0~3特权级



VMX的操作过程

- 系统启动后，VMM执行VMXON，进入到VMX模式；
- VMM执行VMLUNGH或VMRESUME指令产生VM-Entry，进入非根模式，客户机开始执行；
- 当客户机执行特权指令，或发生中断、异常，触发VM-Exit;
- 如果VMM决定退出，执行VMXOFF，关闭VMX模式



CPU虚拟化的硬件支持

- VMCS（Vritual-Machine Control Structure）

-虚拟寄存器概念在硬件上的应用
-VMCS主要由CPU直接操作
-4KB
-VT-x提供两个指令VMRead,VMWrite来访问VMCS

字节偏移	描　　述
0	VMCS revision identifier
4	VMX-abort indicator
8	VMCS data(implementation-specific format)

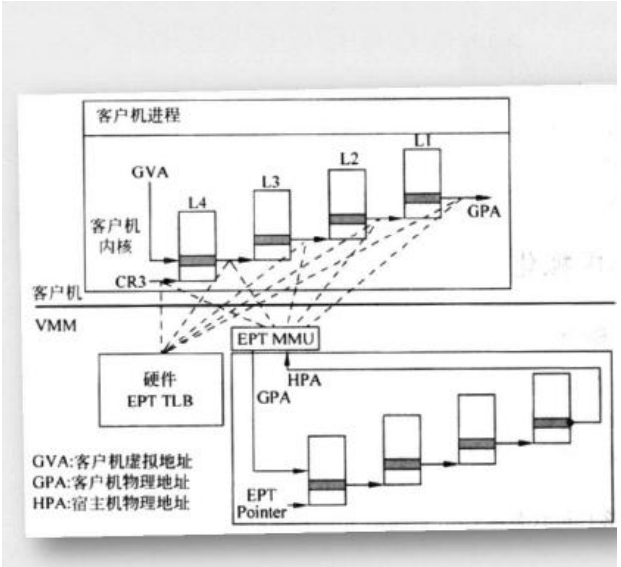
内存虚拟化

- VT-x在硬件级提供了EPT技术实现“影子页表”的功能
- EPT=Extended Page Table
- VPID=Virtual Processor ID（EPT的TLB）

- EPT原理

-CR3中存放的是L4的GPA
-通过EPT页表找到L4的HPA
-结合GNA找到L3的GPA
-通过EPT页表找到L3的HPA
-… … …
-直到找到VGA的HPA

- VM-Execution中的Enable EPT字段（是否启用）
- Extended page table pointer字段（EPT基地址）



内存虚拟化

- VT-x在硬件级提供了EPT技术实现“影子页表”的功能
- EPT=Extended Page Table
- VPID=Virtual Processor ID(EPT的TLB)

- VPID原理

-TLB是EPT页表的缓存
-每次VCPU上下文切换，都需要使TLB无效，造成浪费
-在TLB的每一项上增加一个标志，来识别这个TLB项属于哪个VCPU，避免每次都要切换
-VMCS中Enable VPID字段（是否启用）
-VMCS中VPID字段（标示VMCS对应的TLB）

IO虚拟化的硬件支持

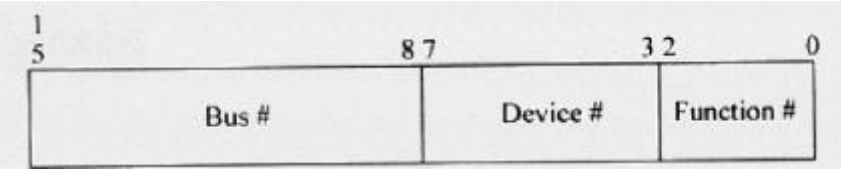
- VT-d=Intel VT for Directed IO
- VT-d通过在北桥引入DMA重映射硬件，实现设备重映射和直连的功能
- 整个映射过程对上层软件是透明的
- 让虚拟机直接使用物理设备的DMA方式要解决两个问题
  - 让虚拟机中的客户操作系统能直接访问设备的IO接口
  - 物理设备能直接对客户操作系统中的内存进行读写
- 为了进行DMA重映射，VT-d引入两个数据结构
  - 根条目：用来描述PCI总线，每条总线对应一个根条目
  - 上下文条目：用于描述一条总线上的某个具体设备

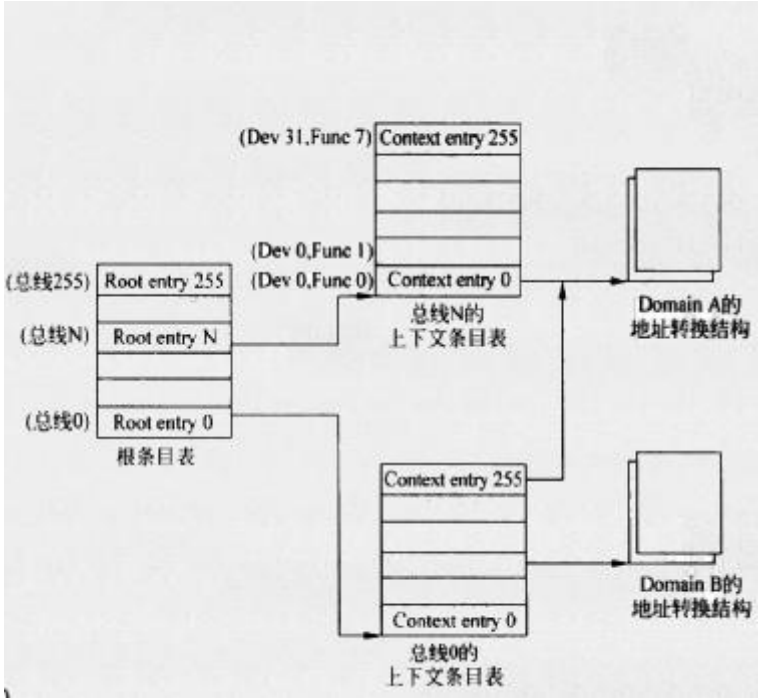
- 根条目的两个关键字段
  - 第0位，存在位，该位为0时表示该总线无效，所有DMA信号都将被屏蔽；该位为1时表示总线有效
  - 12~62位为CTP，上下文条目表指针
  - 其余位为保留位

- 上下文条目的关键字段
  - 第0位，存在位，该位为0时表示该设备无效，所有DMA信号都将被屏蔽；该位为1时表示该设备有效
  - 12~62为，ASR，指向IO页表的指针
  - 72~87位，DID，客户机ID
  - 其余位。。。

IO虚拟化的硬件支持

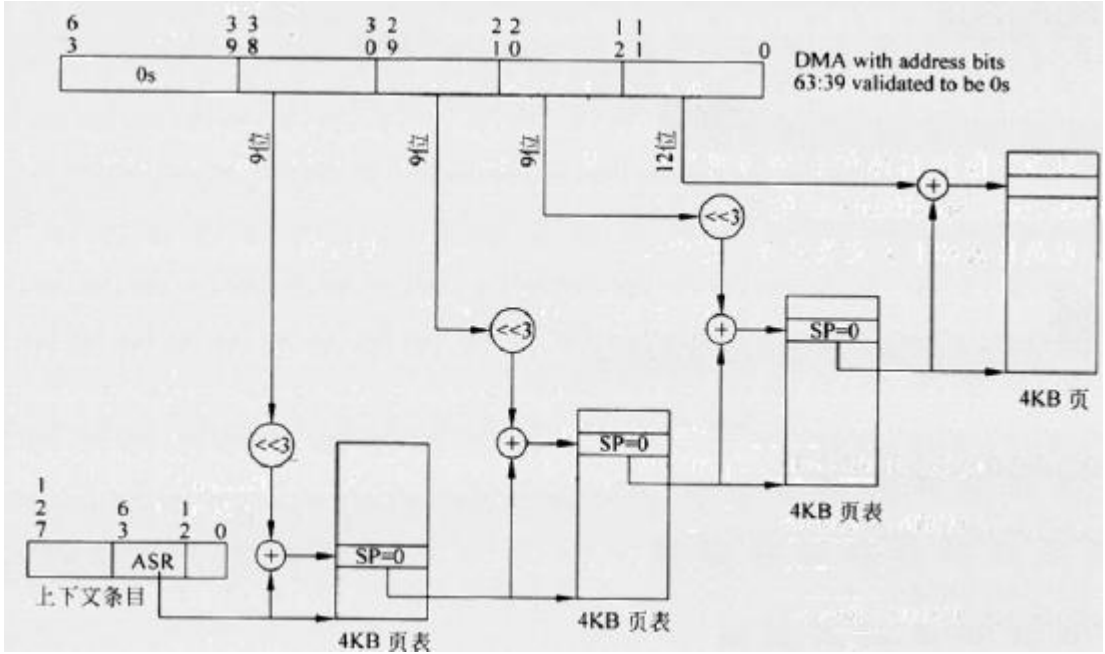
- 根据BDF的bus字段索引根条目表，得到根条目
- 根据根条目中的CTP得到上下文条目表地址
- 根据BDF的dev、func字段可以从上下文条目表中获取发起该DMA的设备的上下文条目
- 通过上下文条目中的ASR可获取IO页表
- 此时就可以做地址转换了





IO虚拟化的硬件支持

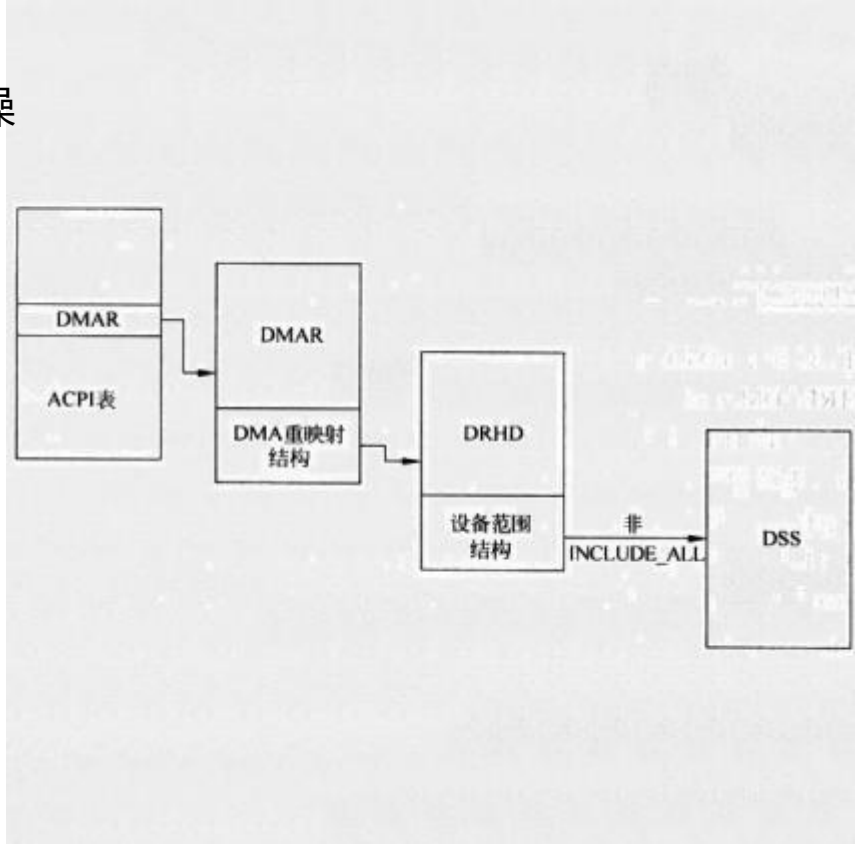
- 为了进行DMA重映射，VT-d技术还提供了IO页表
- 提供了IOTLB，以加快映射速度
- 原理同页表
- 4级4KB



IO虚拟化的硬件支持

- VT-d设备的发现
- 同所有设备一样，通过BIOS的ACPI表向操作系统汇报设备重映射状态

- 由三个数据结构描述
  - DMAR：汇报平台DMA设备的总体状况，总表；
  - DRHD：描述DMA映射硬件设备，每个设备对应一个DRHD；
  - DSS：描述DRHD所管辖的具体设备



时间虚拟化

操作系统中的时间概念

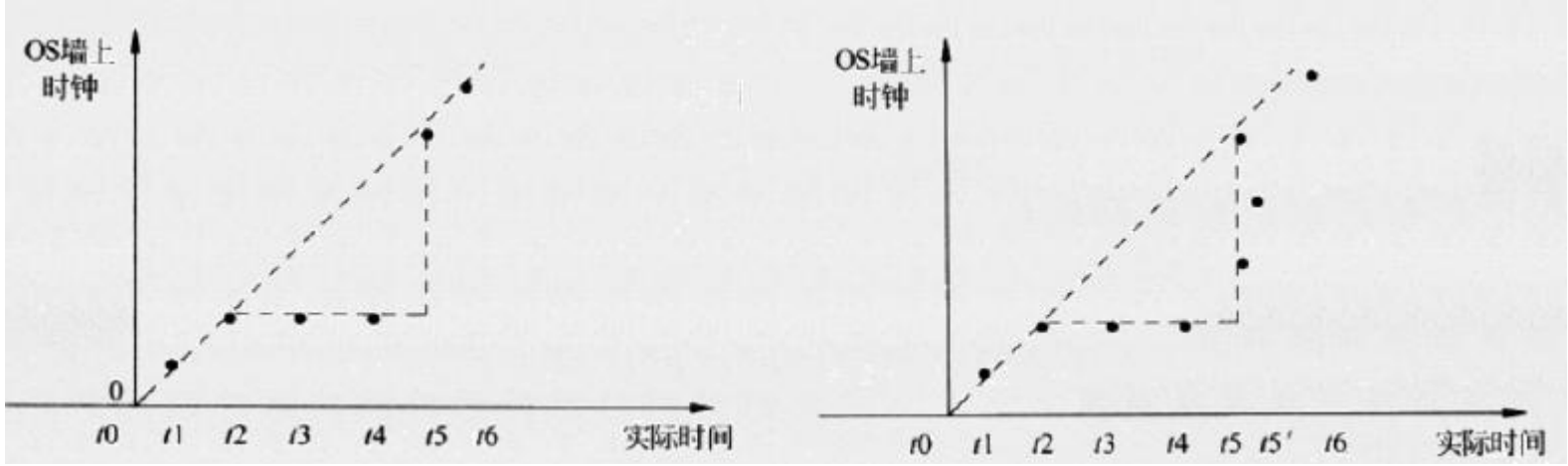
- 绝对时间：（Wall Time）操作系统启动后，到目前为止的总运行时间。
- 相对时间：两个时间之间的间隔，如两次时钟中断之间的间隔
- 当前的绝对时间 = 系统启动时的时间 + 系统运行时间
- 系统的时间是由硬件定时器定时发送时钟中断来维护的。

客户操作系统中的时间概念

- 虚拟环境下，客户操作系统不能得到全部处理器时间，它会被调度进入休眠状态
- 休眠后就不能处理时钟中断，造成时间的停滞
- 如何维护客户机中的时间?

时间虚拟化

- 实现客户操作系统中虚拟时间的一种方法
  - 假设客户机在t2时刻被调度出去，t3，t4的时候需要插入时钟中断，但此时客户机没有运行。
  - 在t5时刻被调度进来，把丢掉的两个中断补上。



类虚拟化

类虚拟化是修改虚拟机的硬件抽象以及客户操作系统，使得客户操作系统和VMM协同工作。

类虚拟化的优势

- 降低虚拟化带来的性能开销；
- 消除了虚拟层和客户操作系统的语义鸿沟；
- 为虚拟化的进一步研究带来空间

类虚拟化的缺点

- 需要修改客户操作系统

CPU的虚拟化

内存虚拟化

IO虚拟化

时间与时钟管理

类虚拟化CPU的虚拟化

指令集
- 提供的是实际CPU指令的一个子集；
- 特权指令和敏感指令将不被支持；
- 通过VMM提供的超调用来实现特权指令和敏感指令的功能；

超调用
- 从客户操作系统到VMM的系统调用
- 以用130号中断向量
- 超调用页被划分为128个块，被映射到客户操作系统的固定虚拟地址上。

中断
- 类虚拟化环境中，虚拟机将不能直接接受来自硬件的中断；
- 所有中断必须由VMM注入；

类虚拟化中的4种中断
- 来自物理外部中断；
- 来自VMM的中断；
- 来自同一个虚拟机的其他VCPU的中断；
- 来自其他虚拟机的中断；

类虚拟化内存的虚拟化

如何防止客户操作系统访问VMM页内存

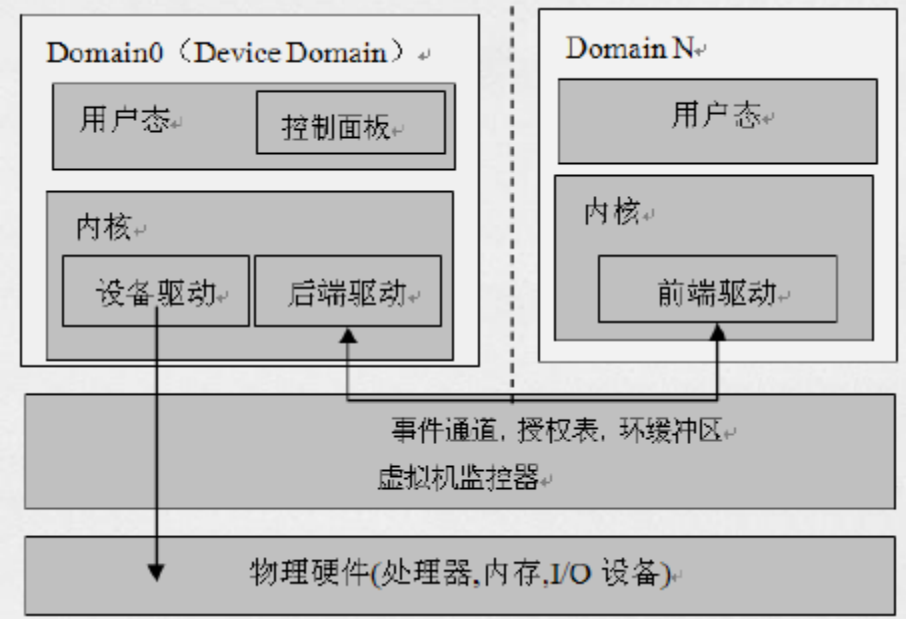
- VMM运行在特权级0；
- 客户机运行在特权级1；
- 通过修改段表述符，使得只用特权级0才能访问VMM的虚拟地址空间；

如何防止客户操作系统访问其他虚拟机内存页

- 使页表页只读；
- 客户机更新页表项只能利用超调用；
- 超调用检查地址的合法性，防止访问其他虚拟机的内存页；

Xen的类虚拟化采用前后端驱动

- 客户操作系统内的一端称为前端设备驱动；
- Dom0中的一端成为后端设备驱动；
- Driver Domain



物理内存空间

- Xen将物理地址到机器地址的映射表暴露给客户操作系统；
- 这样客户操作系统就可以直接进行地址转换了

虚拟内存空间

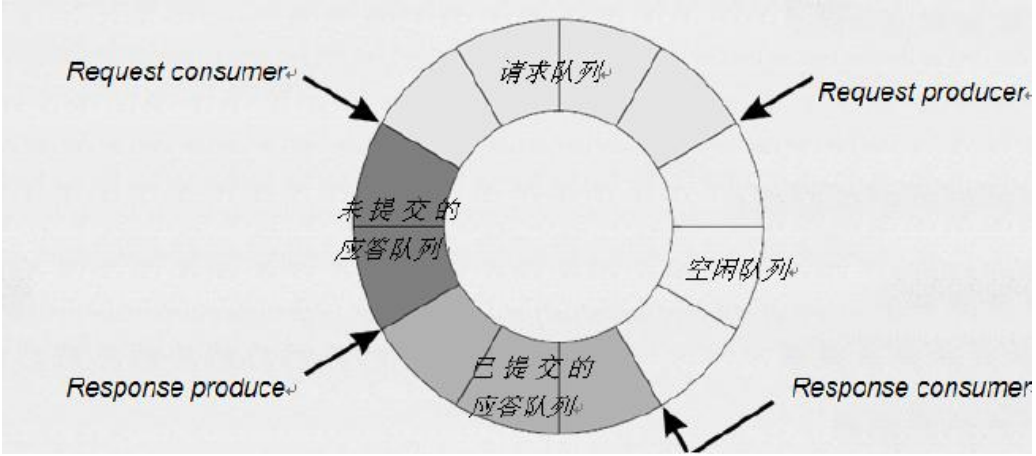
- X86架构中TLB是由硬件直接管理的；
- 虚拟机的调度会引起TLB的刷新，带来性能问题；
- Xen将VMM和虚拟机落于同一个虚拟内存空间；
- 将虚拟内存空间的前64MB划分给VMM使用；

类虚拟化IO的虚拟化

XenDriver Domain

- 默认情况下，只有Dom0直接与硬件交互，一旦驱动程序出错，会导致Dom0整体崩溃，所以，专门设置一个虚拟机作为Driver Domain，来驱动特定的设备。这样就算驱动程序出问题，也不会影响dom0，最多重启Driver Domain
- 性能、容错

Xen 采用一种成为环形缓冲区的数据结构实现IO请求发起机制，这个缓冲区是客户机和VMM共享的内存页



类虚拟化IO的虚拟化

前后端设备驱动间通过事件通道机制进行异步通信

- 前端发送一个IO请求，并通过事件通道通知后端；
- 当IO响应到达前端，前端设备驱动会收到一个时间告知其IO响应到达；
- 前端驱动再到环形缓冲区读取IO响应描述符；

数据传输

- 前端设备驱动分配共享页
- 通过授权表允许后端驱动映射和访问这个共享页
- 后端设备驱动被允许通过DMA直接读写这个共享页

大量降低系统开销

类虚拟化时间和时钟的管理

类虚拟化如何维护客户机内的时间概念?

Xen维护了三种时间

- 实际时间：物理机开机后的时间累计；
- 虚拟时间：虚拟机占用CPU的时间；
- 墙钟时间：真实时间

虚拟时间

- 当客户机被调度运行时，它接受来自VMM的时间中断，相应更新其虚拟时间
- 客户操作系统依赖虚拟时间调度其内部进程，以确保不管虚拟机被VMM如何调度，内部进程可获得同等的虚拟机运行时间

墙钟时间

- 当客户机被调度运行时，VMM计算出当前真实时间，写入客户机的共享页，供客户操作系统读取

XEN安全

目标是增强Xen虚拟化环境的安全性
首先要增强Dom0系统的安全性

- Dom0中提供尽量少的服务
- 配置有效的防火墙
- 不允许用户对dom0的访问

- Driver Domain中DMA的安全
  - 驱动程序运行在内核态；
  - 在没有IOMMU的体系结构中（大部分的X86平台都是这样），一个硬件驱动程序可以通过DMA的方式直接访问它控制域范围之外的内存。所以在选择硬件平台的时候最好还是选择由IOMMU的。
- 共享数据总线的安全
  - 共享的数据传输总线也存在被嗅探和欺骗的安全隐患。
  - 假定设备A被绑定到虚拟机A
  - 假定设备B被绑定到虚拟机B
  - 设备A可以窃取共享数据总线上的数据，然后发送给虚拟机A（嗅探）
  - 设备A甚至可以向数据总线发送欺骗数据
- Driver Domain中的中断安全
  - 共享中断信号线的平台中，一个设备可以发起一个中断，并永不清除它，这样就可以有效的阻止中断级别相同或低的其他设备发出中断，来通知对应Driver Domain中的驱动程序为其服务。
- Driver Domain中的IO地址空间粒度安全
  - Xen只能限制页表一级的设备IO地址空间。
  - 而中断和I/O port地址空间的粒度要比页表小的多。
  - 如果两个设备的IO地址空间不幸被分配到同一个页表，更不幸的是这两个设备被分配到了两个不同的虚拟机，那么… …
  - 允许每个设备有自己的中断信号线的系统架构，可以有效避免这种拒绝服务的问题。

虚拟化技术的王者VMWare

vmware®

1999

2003年，开源Xen通过最新的半虚拟化（Para-virtualization）技术在数据中心用户群体中流行开来。Xen成为开源虚拟化领域的一件大事。它免费，还开源，业界对其给予厚望，希望能与VMware抗衡，分得一杯羹。

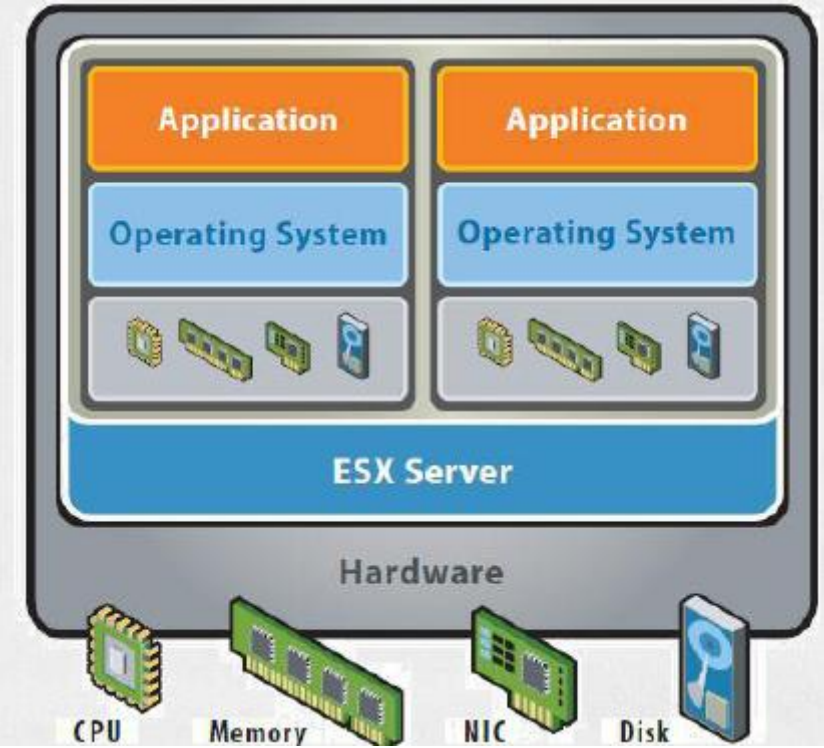
2006年**wmware被EMC收购，2007**年8月，EMC在纽约证券交易所发行了VMware 10%的股份，每股29美元，当天报收51美元，上涨了22美元。虚拟化技术成为IT领域众星捧月般的“明星”技术。

2003

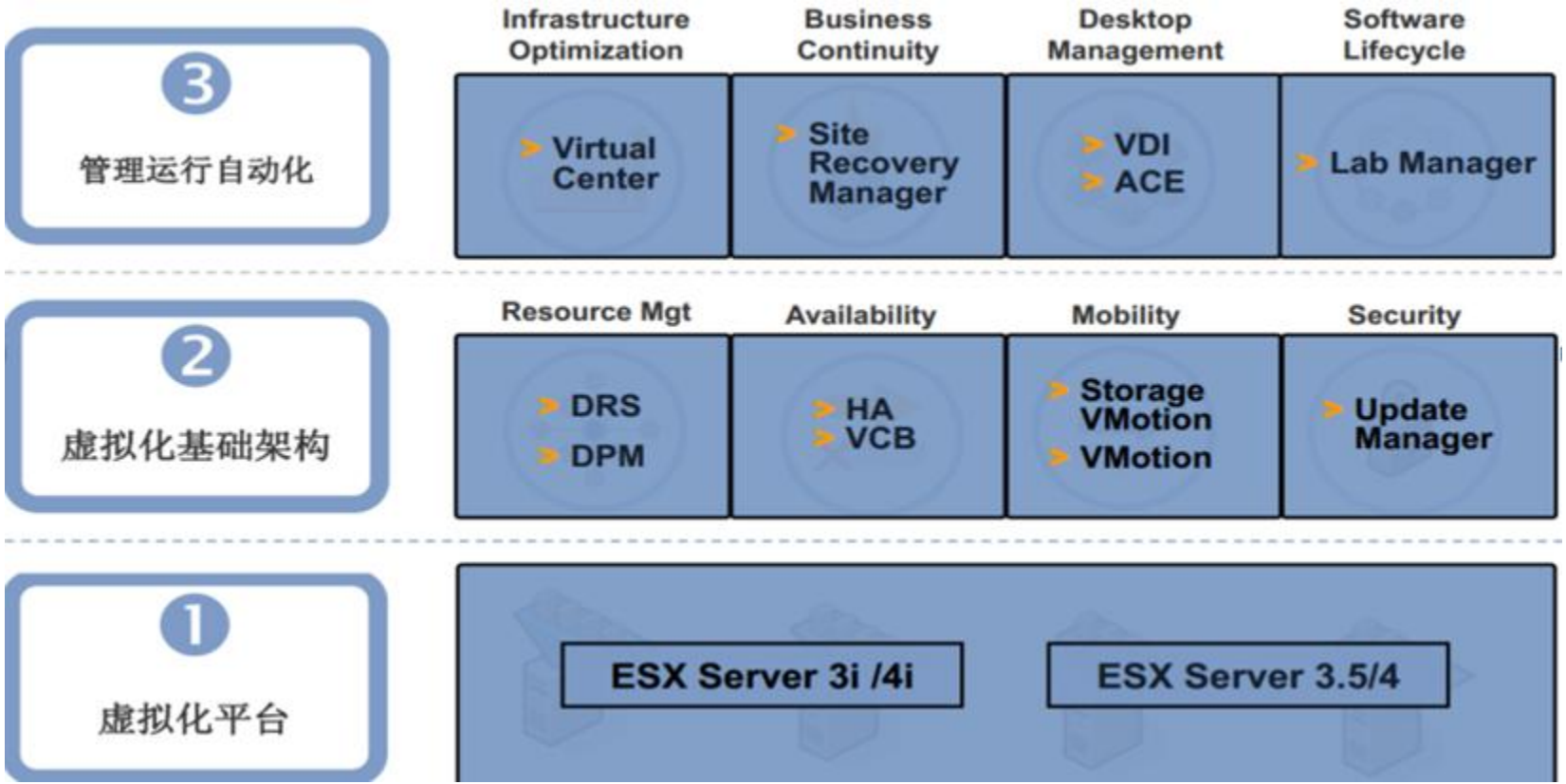
Citrix

虚拟化技术的王者VMWare

-Hypervisor模型
-面向企业级的产品
-支持完全虚拟化和半虚拟化
-支持Intel VT和AMD-V技术

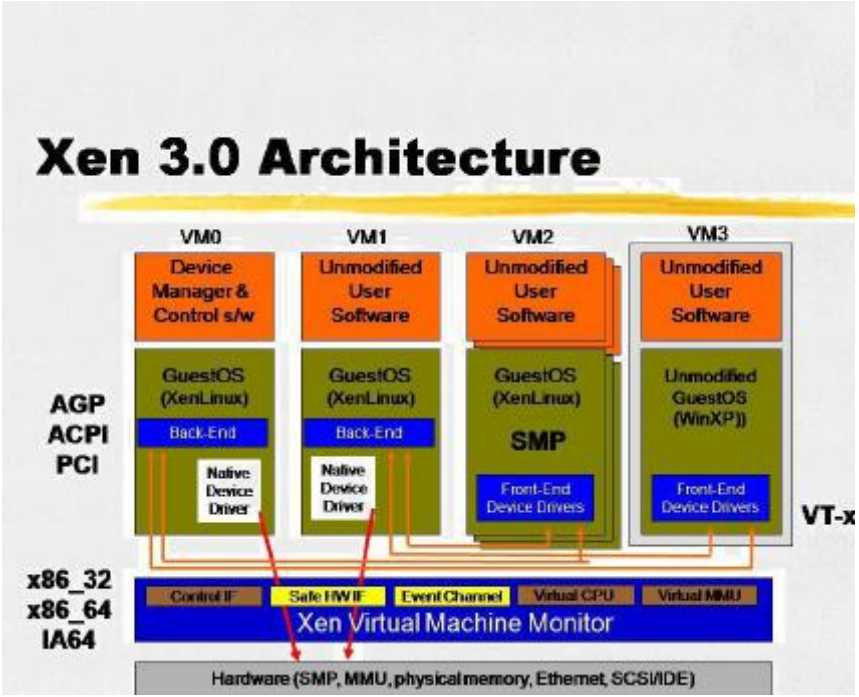


虚拟化技术的王者VMWare



XEN虚拟化技术

-开源(GPL)
-混合模型
-支持Intel VT和AMD-V技术
-支持完全虚拟化和半虚拟化
-I/O虚拟化借用了QEMU



XEN虚拟化技术

- Xen采用超虚拟化技术和完全虚拟化技术。Xen虚拟化环境由以下部件构成：1）XenHypervisor 2）Domain0（包括Domain管理和控制工具） 3）DomainU（ DomainUPV客户系统和DomainUHVM客户系统)
- Xen的VMM(XenHyperviso)位于操作系统和硬件之间,负责为上层运行的操作系统内核提供虚拟化的硬件资源，负责管理和分配这些资源，并确保上层虚拟机（称为域）之间的相互隔离。Xen采用混合模式，因而设定了一个特权域用以辅助Xen管理其他的域，并提供虚拟的资源服务，该特权域称为Domain0，而其余的域则称为DomainU。

XEN的失落与微软的崛起

- 开源Xen不敌商业化的wmare
- 开源的Xen Source被Citrix高价收购、以及citrix与微软密切的关系，让业界其他大佬对Xen保持距离
- 难与Linux内核集成的这个缺陷导致后来者KVM乘虚而入
- 2011年初开源Xen终于获得了Linux的完全支持，但是来得太晚已经错过了提高市场占有率的机会。

IBM的选择与KVM的崛起

2008年9月，红帽收购了一家名叫Qumranet的以色列小公司，由此入手了一个叫做KVM的虚拟化技术

Qumranet Kernel-based Virtual Machine

5.4

Xen/kvm

6.0

kvm

2008/9 2009/9 2010/11 2011/5

OVA已经拥有超过250名成员公司

KVM背后的天才

Avi Kivity
KVM之父

Avi Kivity对于计算机体系架构有很深的理解、深入了解Linux内核，精通汇编和C

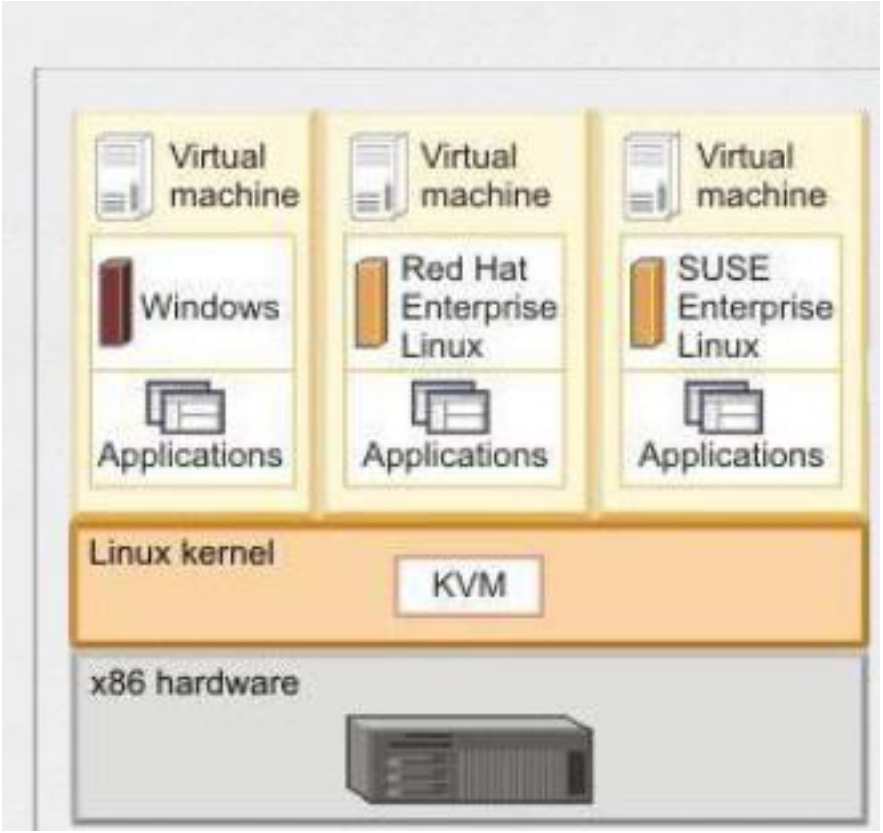
Key features

- Superior Performance
- Rapid VM build and deploy
- Zero OS Management
- DevOps/PaaS like deployment
- Common Java framework integration
- Optimize your Native apps

KVM之后，他大胆提出并设计了专为云计算设计的操作系统OSv，当然，其依然是建立在Linux基础上，但是其无需Context switch，真正的Zero Copy以及Lock Free等特性使得其Less One second boot相对于众多宣称秒级响应的厂商来说更加具有说服力。

KVM虚拟化技术

KVM
-开源(GPL)
-基于Intel-VT技术的硬件虚拟化
-I/O虚拟化借用了QEMU
-宿主模型

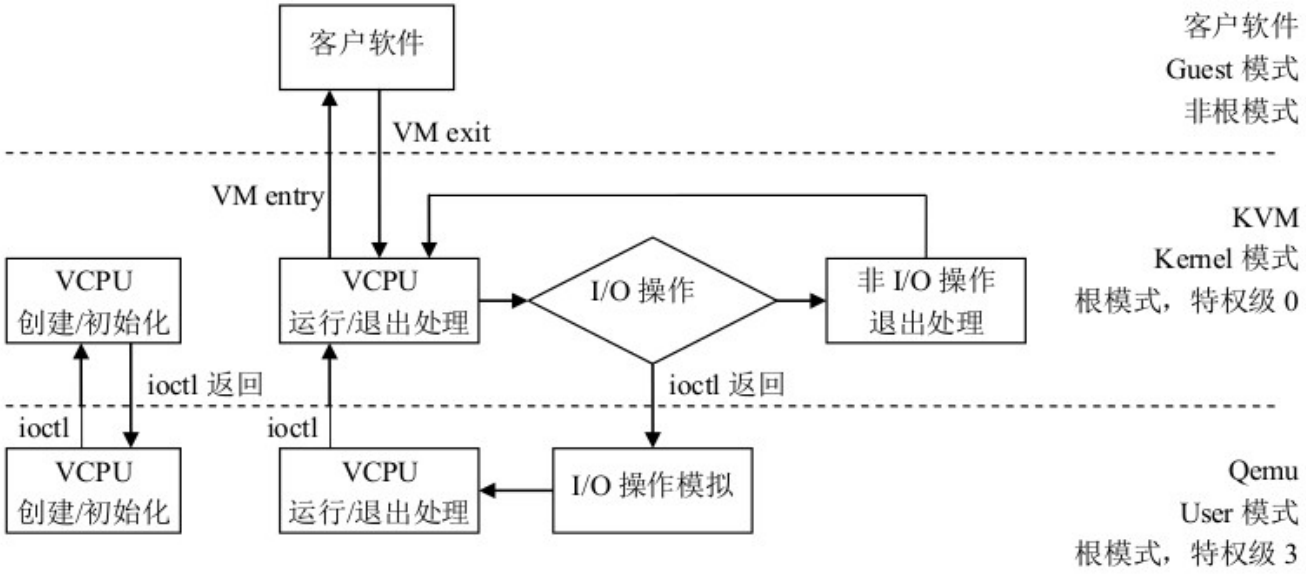


KVM虚拟化技术

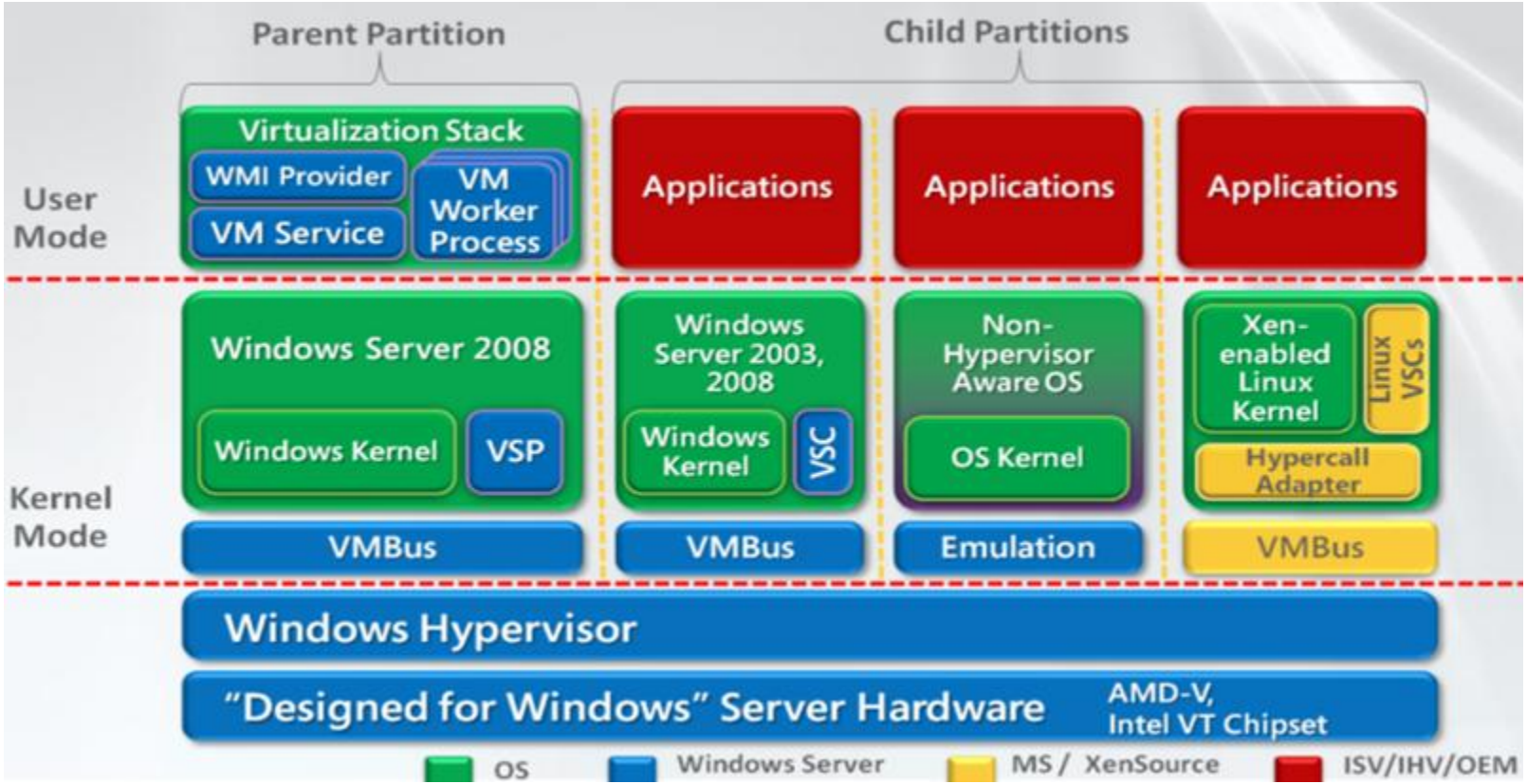
- KVM是嵌入在Linux操作系统标准内核中的一个虚拟化模块，它将一个Linux标准内核转换成为一个VMM，嵌有KVM模块的Linux标准内核支持通过kvm tools来进行加载的GuestOS。所以在这样的操作系统平台下，计算机物理硬件层上直接就是VMM虚拟化层，而没有独立出来的HostOS操作系统层。在这样的环境中HostOS就是一个VMM
- 每个由KVM创建的GuestOS都是HostOS(或VMM)上的一个单个进程。而在GuestOS上的User-space中运行的Applications可以理解为就是进程中的线程。
- KVM是Linux kernel的一个模块，使用命令modprobe去加载KVM模块。加载了该模块后，才能进一步通过工具创建虚拟机。但是仅有KVM模块是不够的。因为用户无法直接控制内核去做事情，还必须有一个运行在用户空间的工具才行。这个用户空间的工具，kvm开发者选择了已经成型的开源虚拟化软件QEMU。

KVM虚拟化技术

- KVM包含两部分，一部分是基于Linux内核支持的KVM内核模块，另一部分就是经过简化和修改Qemu。KVM负责cpu虚拟化+内存虚拟化，实现了CPU和内存的虚拟化，但KVM不能模拟其他设备。Qemu模拟IO设备（网卡，磁盘等），KVM加上Qemu之后就能实现真正意义上服务器虚拟化。



微软虚拟化技术


Parent Partition
Child Partitions
User Mode
Kernel Mode
Virtualization Stack
WMI Provider
VM Service
VM Worker Process
Applications
Applications
Applications
Windows Server 2008
Windows Kernel
VSP
VMBus
Windows Server 2003, 2008
Windows Kernel
VSC
VMBus
Non-Hypervisor Aware OS
OS Kernel
Emulation
Xen-enabled Linux Kernel
Linux VSCs
Hypercall Adapter
VMBus
Windows Hypervisor
"Designed for Windows" Server Hardware
AMD-V, Intel VT Chipset
OS
Windows Server
MS / XenSource
ISV/IHV/OEM

LCX

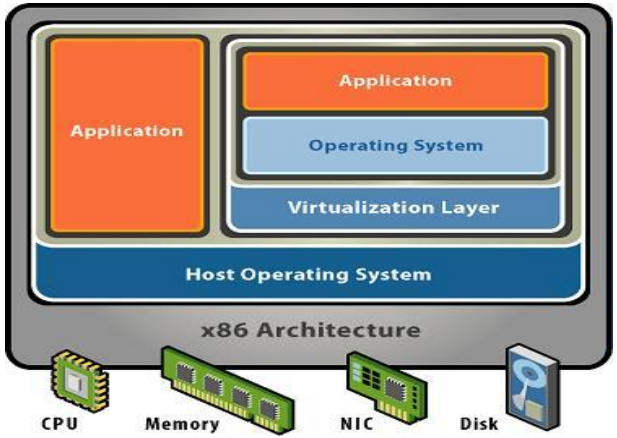
- Linux Container或称Linux容器系统，其本身仅提供最低程度虚拟化（硬件）的功能，并利用两个系统管理模块（cgroup与AppArmor），来有效的管理、控制，并隔离虚拟计算机与实体计算机的资源运用，另外，Linux Container并不需要再透过复杂的程序，来解译虚拟计算机与实体计算机之间CPU指令的运算或是外围装置的虚拟化。

三足鼎立

- wmare仍然是企业数据中心的首选方案
- 公共云平台上，80%以上的共同云平台都选择了**Xen，包括王者亚马逊（与Rackspace、SoftLayer并称Xen三巨头），此外阿里的云平台也是Xen技术**
- **IBM 在全球建立了八个数据中心来支撑其公共云服务，全部是KVM技术**
- **在私有云领域，HP走在前列，CloudSystem以及最近公布的HP Hellion都是以Openstack(KVM)为核心**
- **微软的hyper-v强势发展， Windows Server数据中心版本包括了无任何限制的hyper-v使用权，2013年5月22日，由世纪互联运营的微软Windows Azure公有云平台及服务正式在中国落地**

两种虚拟化架构

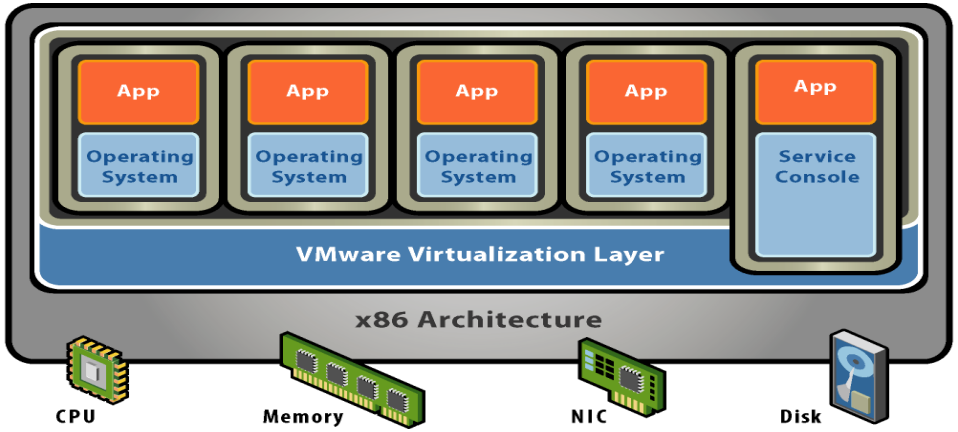
寄居架构（Hosted Architecture）



- 例如：**Server, Workstation**
- 安装和运行应用程序
- 依赖于主机操作系统对设备的支持和物理资源的管理

VMware Server, Workstation
VPC 2007, Virtual Server 2005 R2
Sun VirtualBox

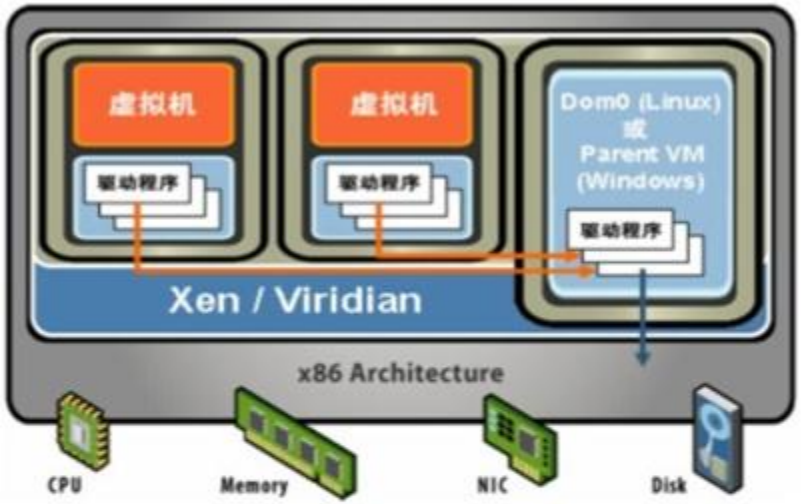
高级架构 (“Bare Metal” Architecture)

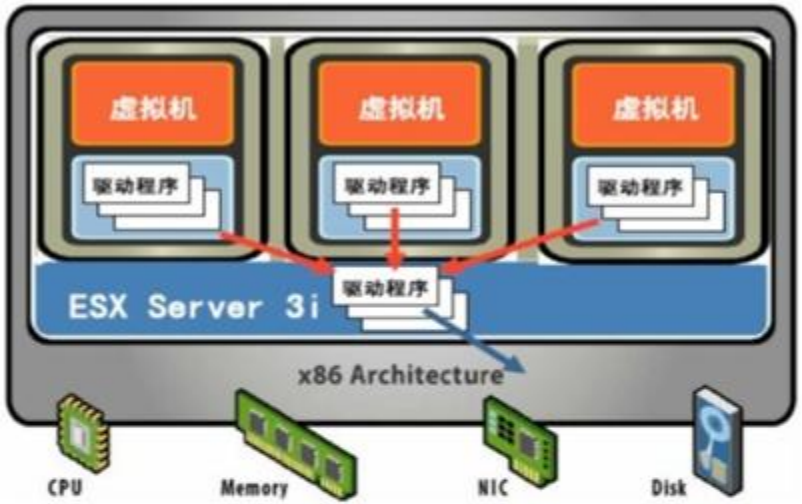


- 例如：**ESX/ESXi**
- 依赖虚拟层内核
- 代理和帮助应用的服务控制台

VMware ESX Server，ESXi
Citrix（思杰）Xen，Redhat KVM
Hyper-V Server 2008

管道程序驱动程序模型





- **Xen/Microsoft Viridian**
  - 间接驱动程序模型
  - 在管理分区中使用通用驱动程序
  - 负载下的性能差

- **VMware ESX Server**
  - 直接驱动程序模型
  - 驱动程序针对虚拟机而优化
  - 扩展性更好

服务器虚拟化的两个方向

一变多

将一台服务器虚拟化成更多的虚拟机

- 大机的虚拟化：IBM的LPAR
- UNIX服务器：
  - IBM的LPAR
  - HP的nPAR，vPAR
  - Sun的Domain、Container
- x86架构服务器：
  - VMware的VI&vSphere
  - Windows Server 2008 Hyper-V

多变一

将多台服务器虚拟化成一台虚拟机

- 分布式运算
- 网格计算（Net Grid）
- 高性能运算（HPC）
- 云计算（Cloud Computing）
  - VMware vCloud

服务器虚拟化

没有利用虚拟化软件之前是300台服务器

使用虚拟化软件后，整合成8台服务器、一个机架

服务器虚拟化

服务器虚拟化将硬件、操作系统和应用程序一同装入一组可迁移的文件之中。

虚拟化前

- 软件必须与硬件相结合
- 每台机器上只有单一的操作系统镜像
- 每个操作系统只有一个应用程序负载

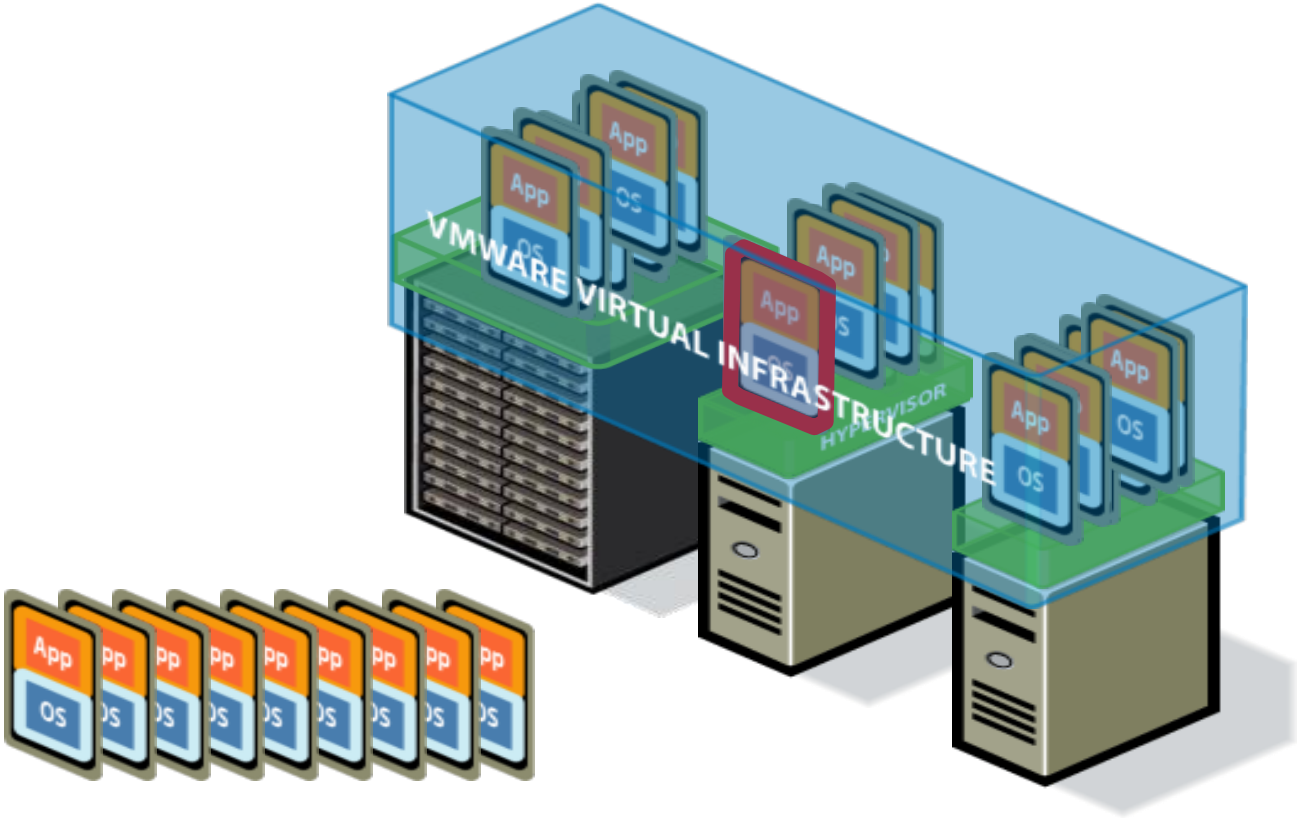
虚拟化后

- 每台机器上有多个负载
- 软件相对于硬件独立

服务器虚拟化

	VMware	XEN	KVM	Hyper-V
授权	商业付费	开源免费	开源免费，Linux内核	商业付费
主机操作系统	安装VMware ESX或ESXi	安装带XEN的Redhat Linux或SUSE Linux	安装带KVM的Redhat Linux或SUSE Linux	安装带Hyper-V的Windows 2008 64-bit edition
虚拟机操作系统	Windows Linux	Windows Linux	Windows Linux	Windows Linux兼容版本
许可费用	主机VMware许可 虚拟机Windows和Linux许可	主机Linux许可 虚拟机Windows许可	主机Linux许可 虚拟机Windows许可	主机Hyper-V许可 虚拟机Windows和Linux许可
管理工具	单独安装VMware vCenter集中管理	主机操作系统带命令行和GUI	主机操作系统带命令行和GUI	System Center
主机硬件	32位或64位	32位或64位	32位或64位 带VT或AMD-V	64位 带VT或AMD-V
虚拟化技术	全虚拟化 二进制码翻译(无VT或AMD-V时)	全虚拟化 半虚拟化（无VT或AMD-V时，虚拟机上支持修改过的Linux操作系统）	全虚拟化	全虚拟化
I/O	VMware Kernel	Dom-0	Linux	
虚拟机迁移	静态/动态迁移	静态/动态迁移	静态/动态迁移	静态/动态迁移

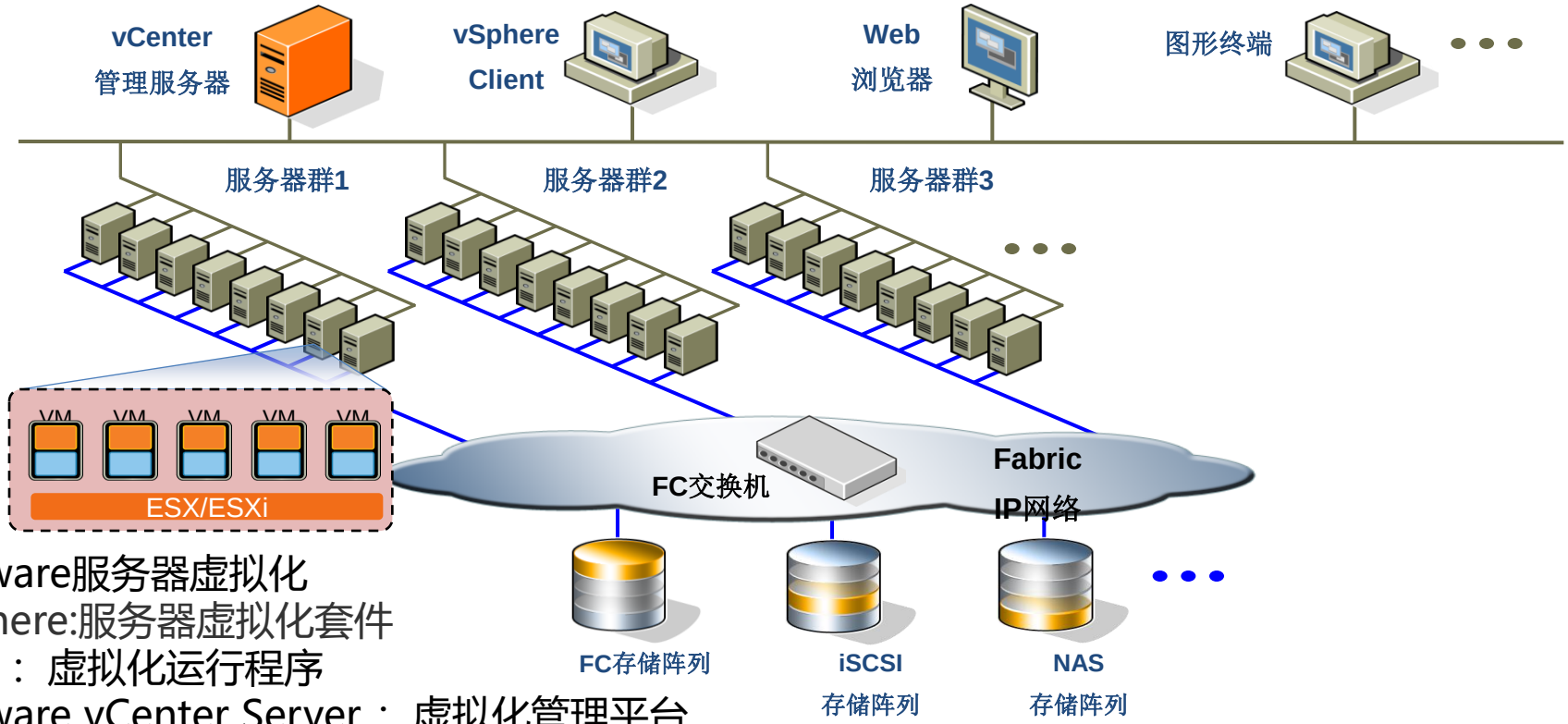
从主机虚拟化到云操作系统



按需部署

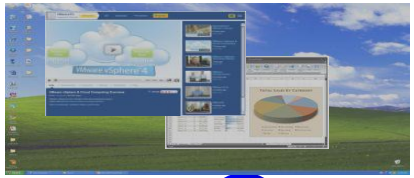
在线迁移

vSphere虚拟化平台的网络拓扑结构



VMware服务器虚拟化
vSphere:服务器虚拟化套件
ESXi： 虚拟化运行程序
VMware vCenter Server： 虚拟化管理平台
vSphere Client:管理控制台

桌面虚拟化—关注业务的本质和细节



屏幕变化传送到客户端设备

鼠标点击和键盘信息传递到
数据中心端

应用执行100%在服务器

- **任何设备、任何系统均可以按需接入高性能网络传输，允许您透过任何链路、在任何地方接入**

谢谢！