虚拟化与云计算

——服务器虚拟化技术

主讲教师：常晓东

提 纲

- 上节内容回顾
- 软件辅助的虚拟化技术
  - CPU虚拟化
  - 内存虚拟化
  - I/O虚拟化
- 硬件辅助的虚拟化技术
  - CPU虚拟化的硬件支持
  - 内存虚拟化
  - I/O虚拟化的硬件支持
  - 时间问题
- 类虚拟化技术
  - 类虚拟化概述
  - CPU的虚拟化
  - 内存虚拟化
  - IO虚拟化
  - –时间与时钟管理
- Xen安全分析
- 实验课安排
  - CloudStack

上节内容回顾

- 云计算相关
  - 云计算的概念、特征、模式
  - 一些开源云计算平台介绍
- 虚拟化的概念与基本原理
  - CPU、内存、IO的虚拟化
- 虚拟化技术的分类
  - 完全虚拟化
    - 软件辅助的完全虚拟化
    - 硬件辅助的完全虚拟化
  - 类虚拟化

软件辅助的虚拟化技术

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
- 内存的虚拟化
  - 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的虚拟化

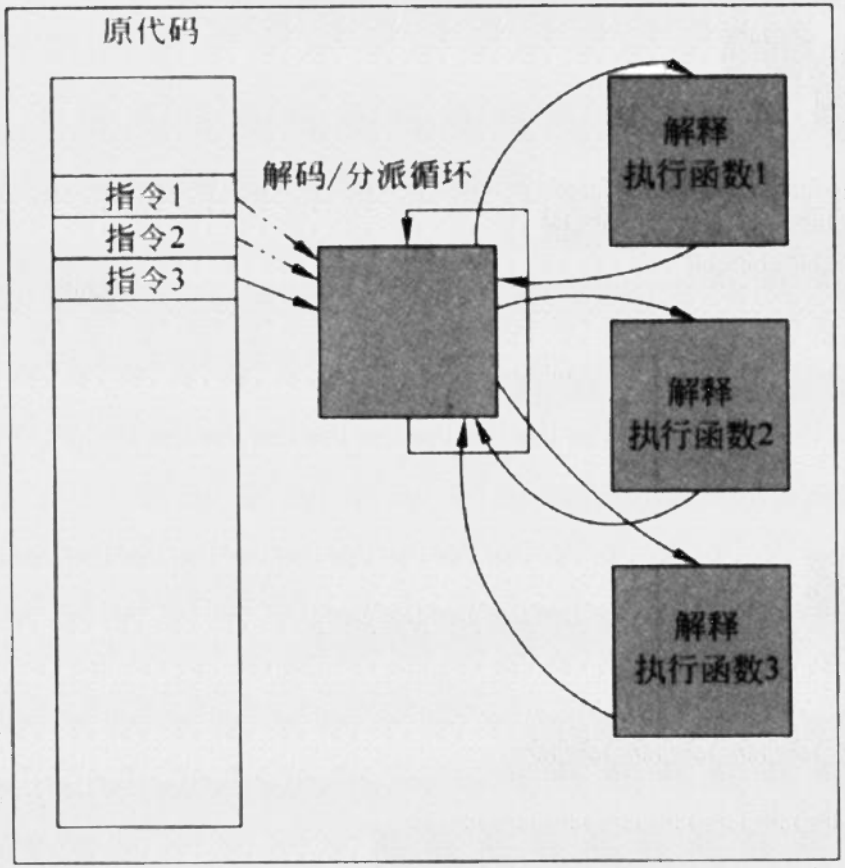
- CPU的虚拟化
  - 基于软件的CPU完全虚拟化
  - 其本质就是软件模拟
- 几种模拟实现方式
  - 解释执行
  - 扫描与修补
  - 二进制代码翻译

解释执行

- 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

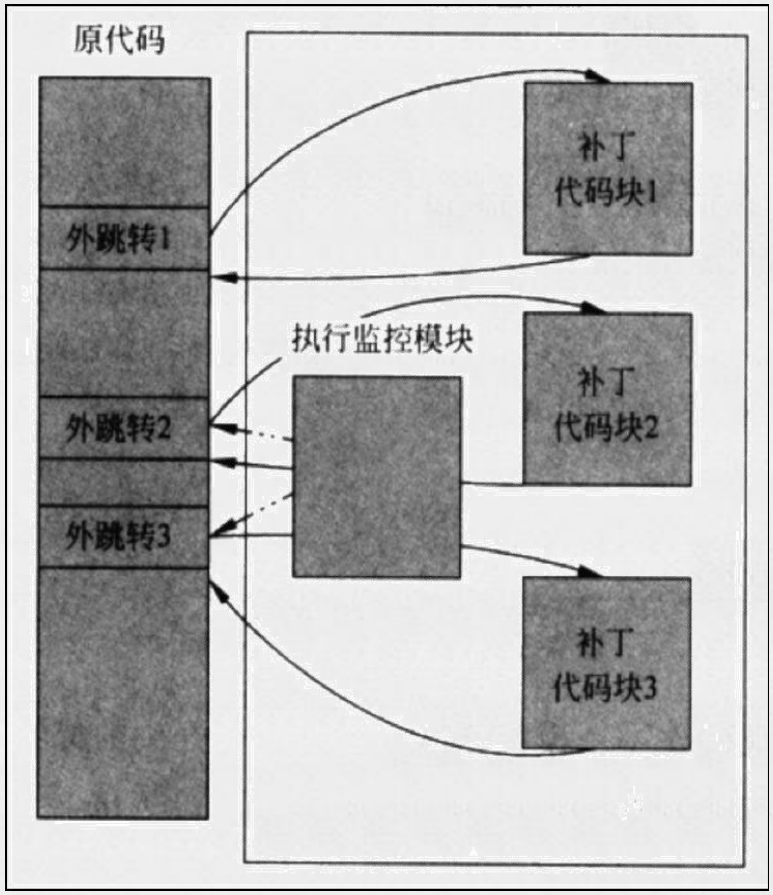


扫描与修补

- 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

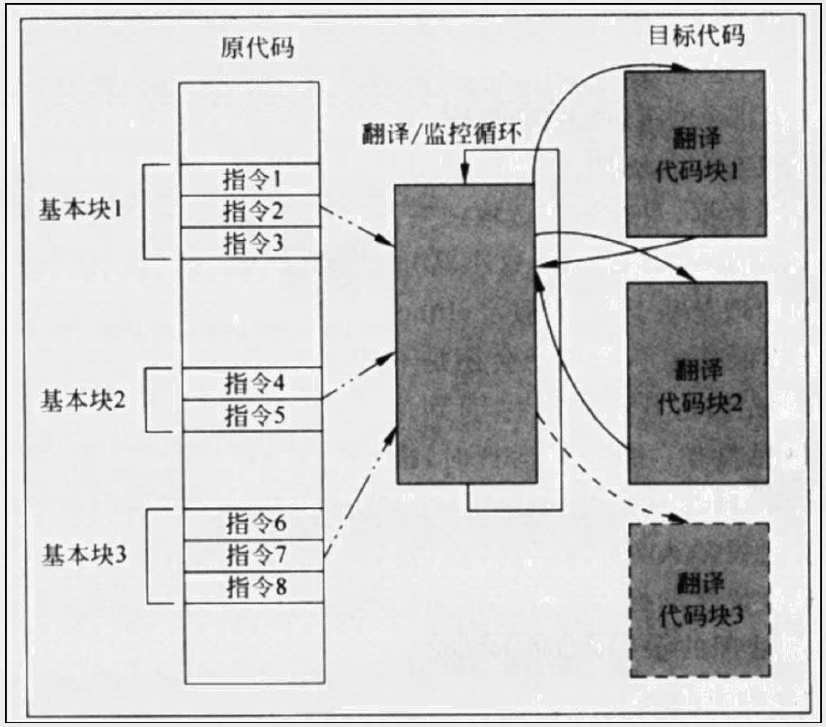


二进制代码翻译（BT）

- 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。

- 对基本块进行反汇编
- 通过翻译模板将其变成中间形式
- 进行代码优化
- 生成目标代码

- 性能最好



二进制代码翻译（BT）

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

内存虚拟化

- 目的
  - 提供给客户操作系统一个从零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

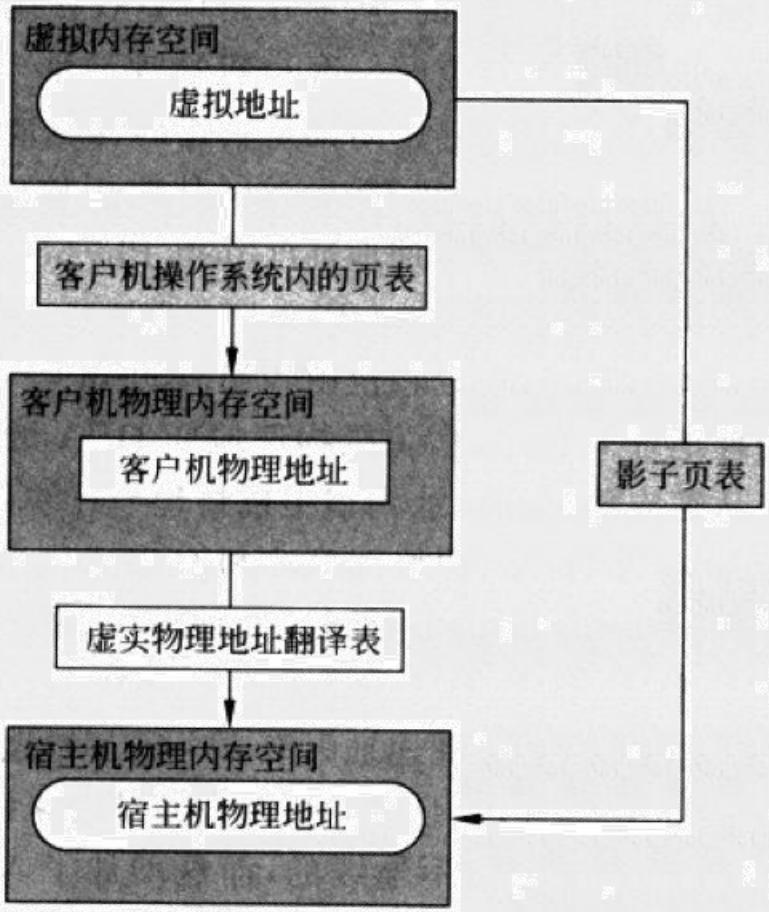
- 方式
  - GVA → GPA → HPA
  - GVA → HPA (影子页表)

影子页表

- 其目标就是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
- 客户机的页表被装载到虚拟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时通知“气球”
  - “气球”向客户机操作系统申请内存
  - “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？

内存虚拟化的优化

- 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，删除其它页。
  - 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）

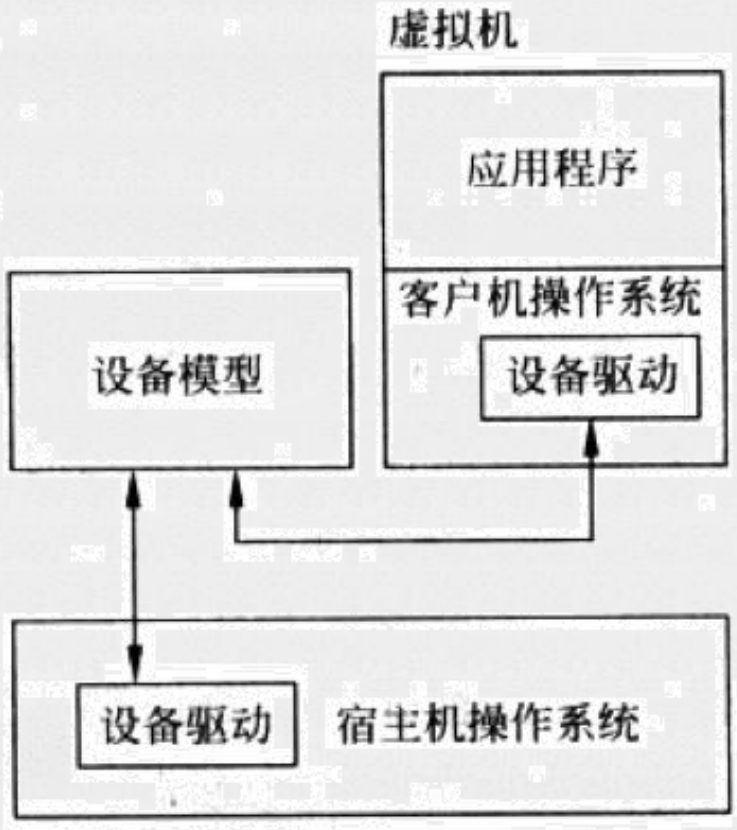
怎样搜索相同页？
计算并记录每个页内数据的hash值，如果Hash值相同说明很有可能两个页的内容相同，然后对两个页的内容进行比较以确定是否相同。

I/O虚拟化

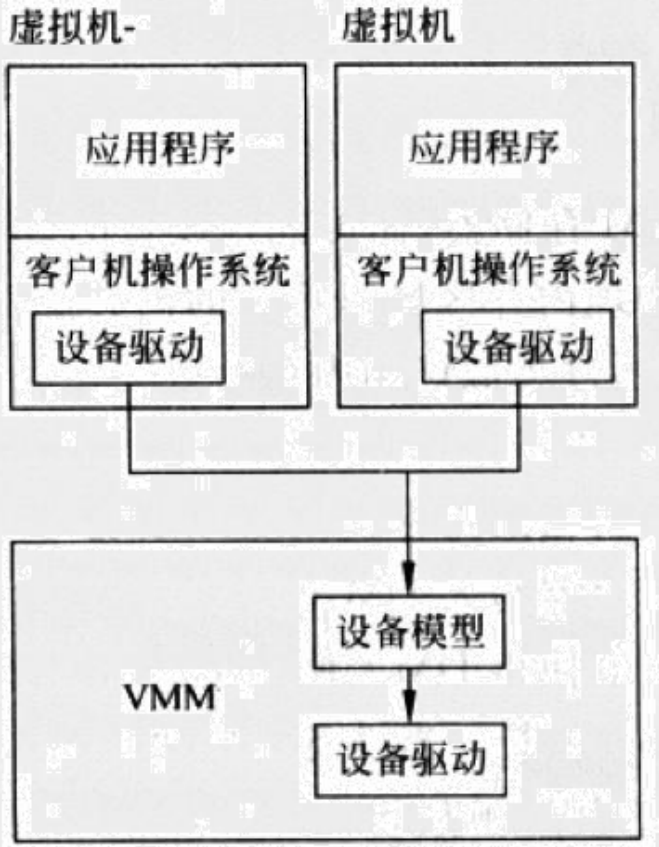
- 一个重要的概念：设备模型。
- 所谓设备模型是指VMM中进行设备模拟，并处理所有设备请求和响应的逻辑模块。

- 作用
  - 模拟目标设备的软件接口
  - 实现目标设备的功能

I/O虚拟化



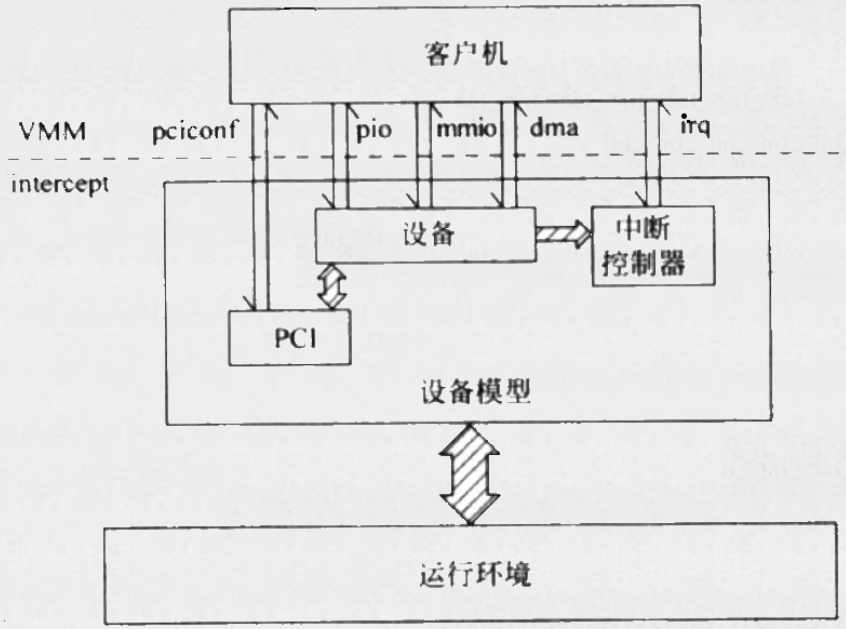
宿主模式



Hypervisor模式

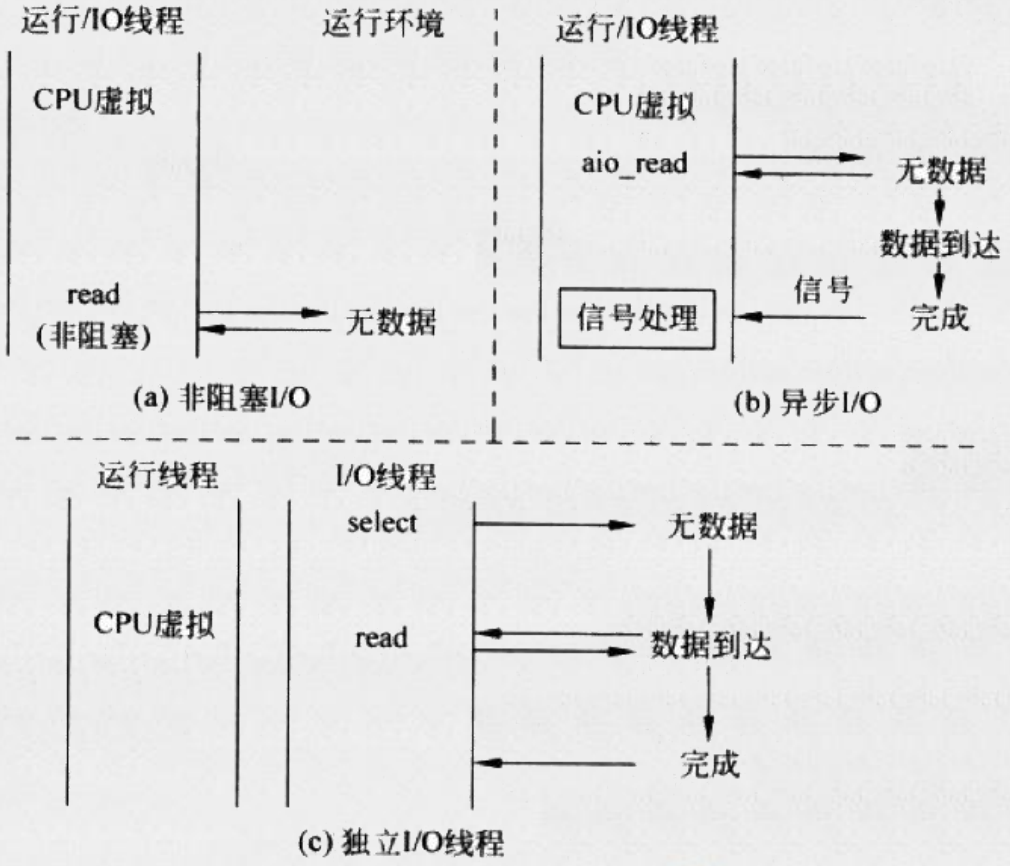
I/O虚拟化

- 设备模型的软件接口
  - PCI配路空间：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



I/O虚拟化

- 设备模型的功能实现
  - 非阻塞IO
  - 异步IO
  - 独立IO线程

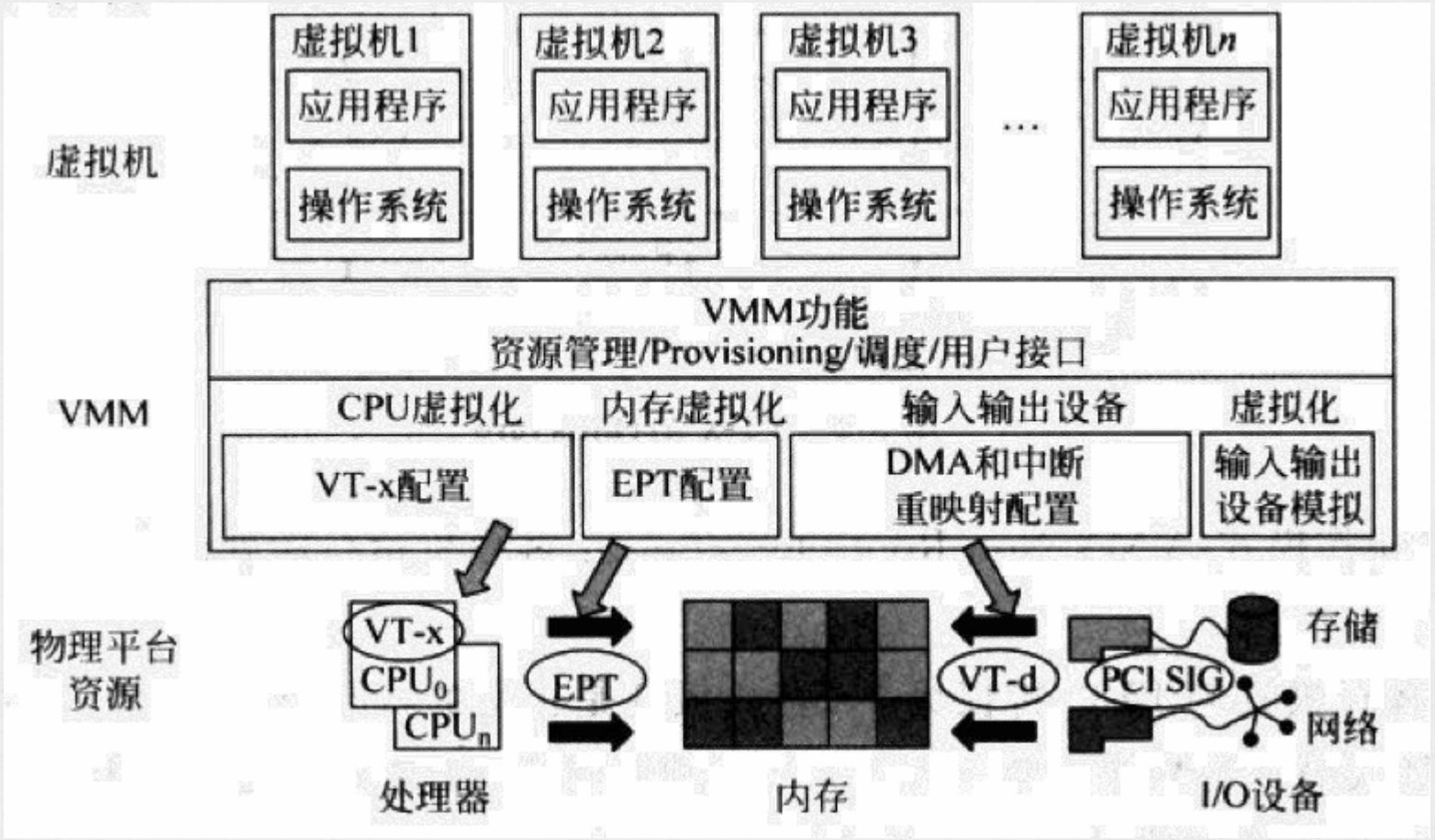


硬件辅助的虚拟化技术

硬件辅助的虚拟化技术

- Intel VT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

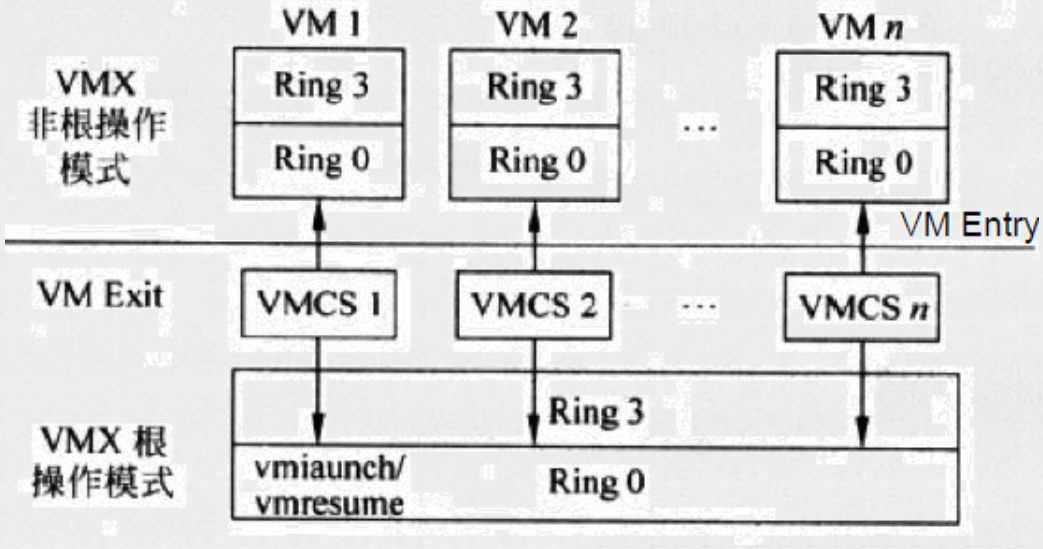
Intel VT的总体结构



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



CPU虚拟化的硬件支持

- VMCS (Virtual-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

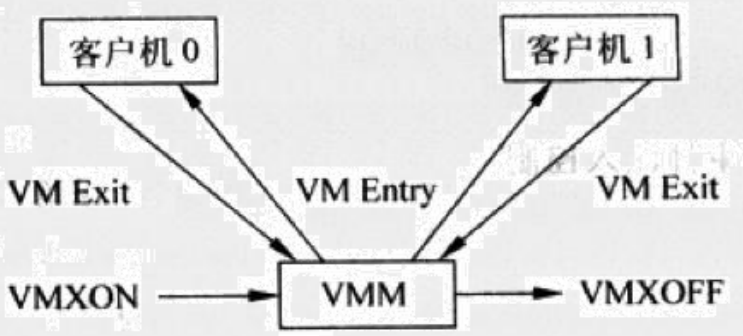
字节偏移	描　述
0	VMCS revision identifier
4	VMX-abort indicator
8	VMCS data(implementation-specific format)

VMCS数据域

- 客户机状态域
  - 记录客户机CPU状态
- 宿主机状态域
  - 记录VMM的CPU状态
- VM-Entry控制域
  - 控制VM-Entry过程
- VM-Execution控制域
  - 控制处理器在非根模式下的行为
- VM-Exit控制域
  - –控制VM-Exit过程
- VM-Exit信息域
  - 提供VM-Exit原因和其他信息

VMX的操作过程

- 系统启动后，VMM执行VMXON，进入到VMX模式；
- VMM执行VMLUNCH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。

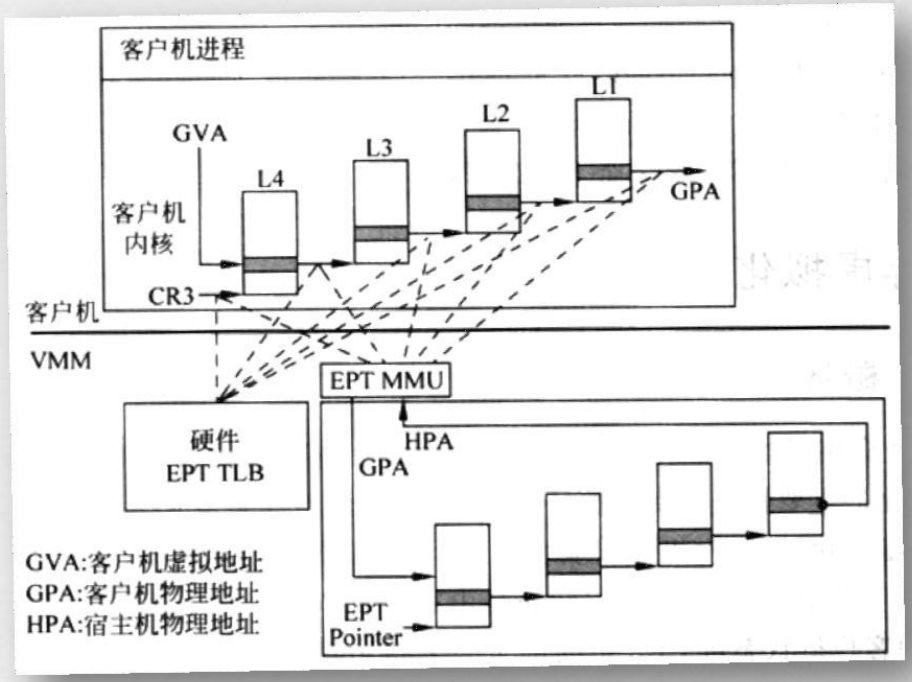


内存的虚拟化

- 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）

内存的虚拟化

- EPT原理
  - CR3中存放的是L4的GPA
  - 通过EPT页表找到L4的HPA
  - 结合GVA找到L3的GPA
  - 通过EPT页表找到L3的HPA
  - …… …
  - 直到找到GVA的HPA
- VM-Execution中的Enable EPT字段（是否启用）
- Extended page table pointer字段（EPT基地址）



内存的虚拟化

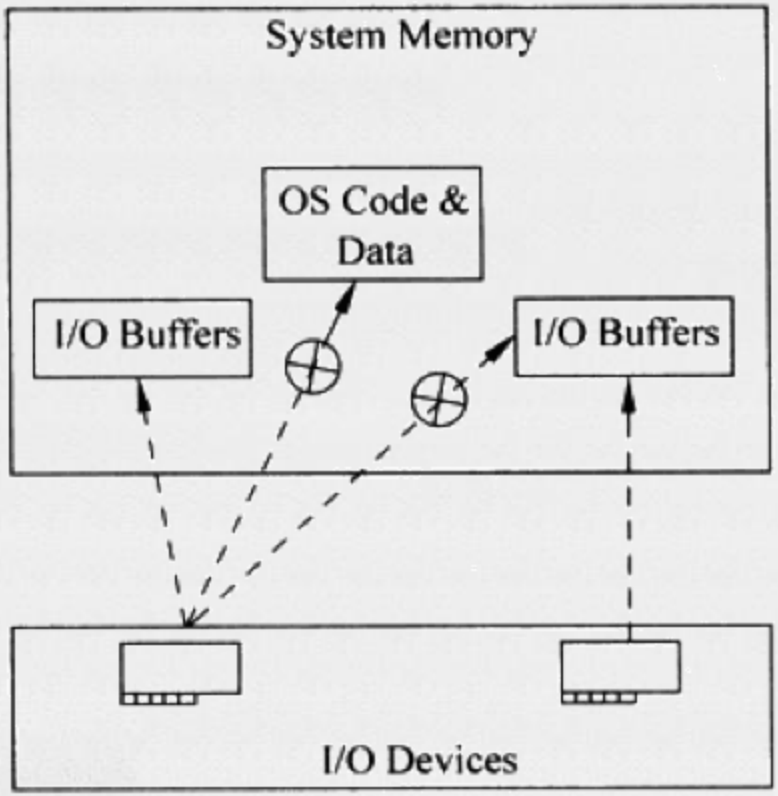
- VPID原理
  - –TLB是EPT页表的缓存
  - –每一次VCPU上下文切换，都需要使TLB无效
  - –浪费
  - –当在TLB的每一项上增加一个标志，来识别这个TLB项属于哪个VCPU
  - –这样就可以避免每次都要切换
  - –VMCS中Enable VPID字段（是否启用）
  - –VMCS中的VPID字段（标示VMCS对应的TLB）

IO虚拟化的硬件支持

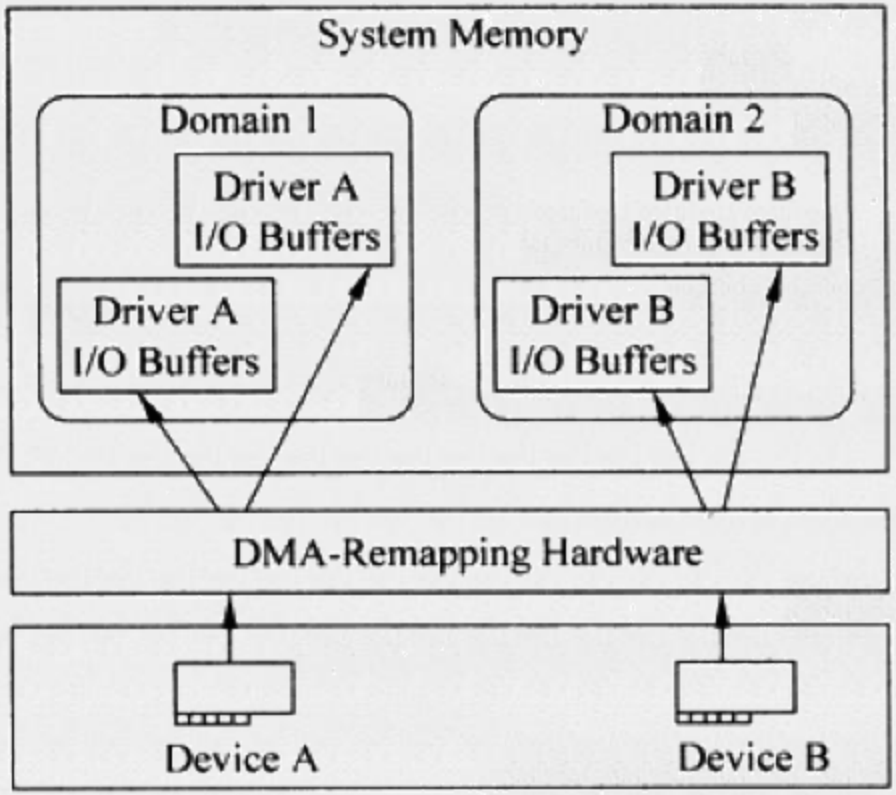
- VT-d ＝Intel VT for Directed IO
- VT-d技术通过在北桥引入DMA重映射硬件，实现设备重映射和直连的功能
- 整个映射过程对上层软件是透明的

- 让虚拟机直接使用物理设备的DMA方式要解决两个问题
  - 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

IO虚拟化的硬件支持



传统方式的DMA



VT-d方式的DMA

IO虚拟化的硬件支持

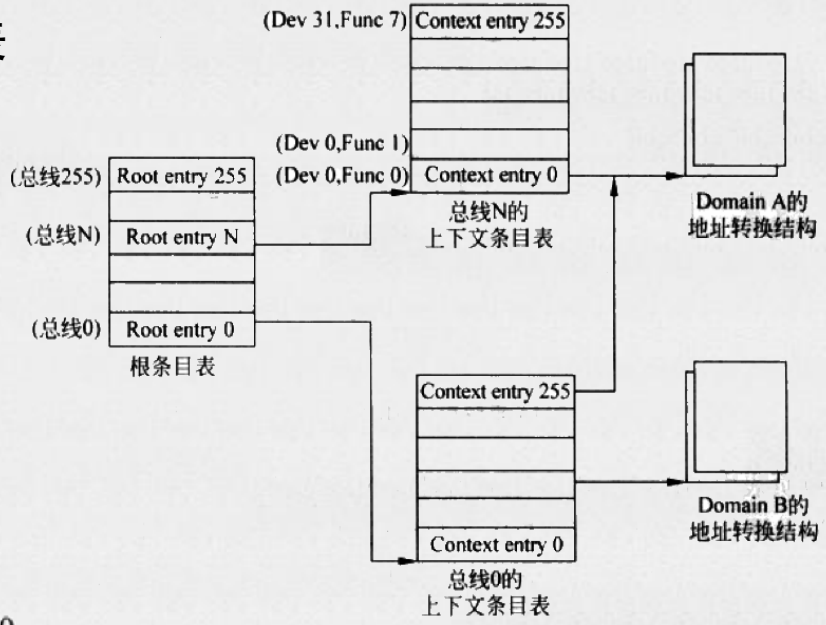
- 为了进行DMA重映射，VT-d引入两个数据结构
  - 根条目：用来描述PCI总线，每条总线对应一个根条目
  - 上下文条目：用于描述一条总线上的某个具体设备

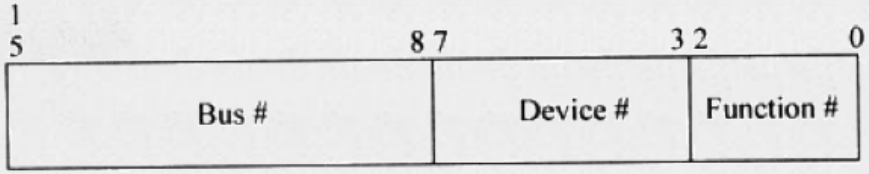
- 根条目的两个关键字段
  - 第0位，存在位，该位为0时表示该总线无效，所有DMA信号都将被屏蔽；该位为1时表示总线有效
  - 12～62位为CTP，上下文条目表指针
  - 其余位为保留位

- 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位，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

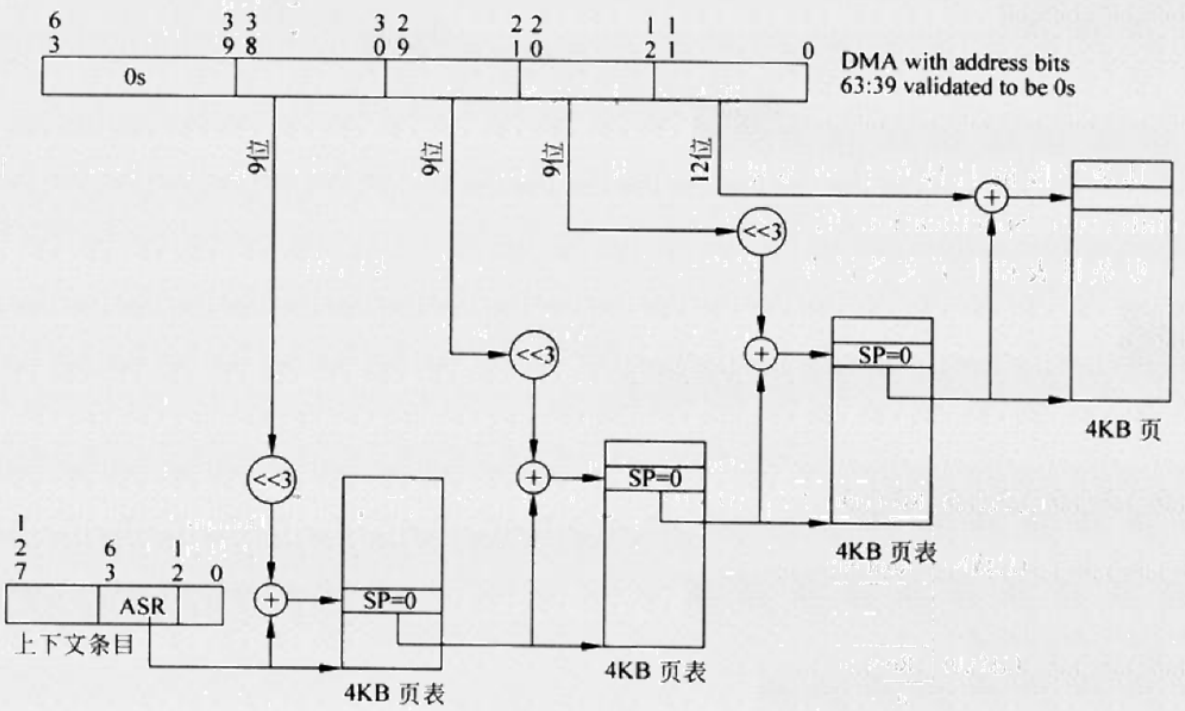
- 根据BDF的bus字段索引根条目表，得到根条目
- 根据根条目中的CTP得到上下文条目表地址
- 根据BDF的dev、func字段可以从上下文条目表中获取发起该DMA的设备的上下文条目
- 通过上下文条目中的ASR可获取IO页表
- 此时就可以做地址转换了





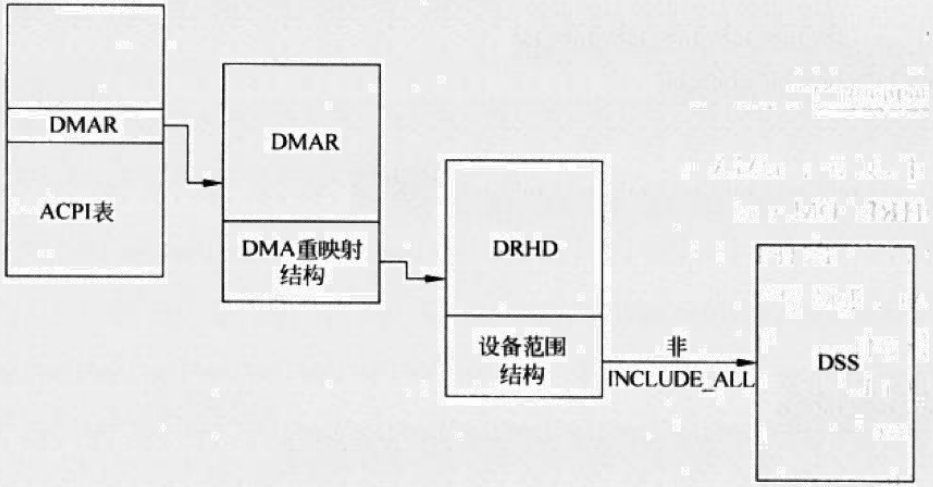
IO虚拟化的硬件支持

- 为了进行DMA重映射，VT-d技术还提供了IO页表
- 同时提供了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

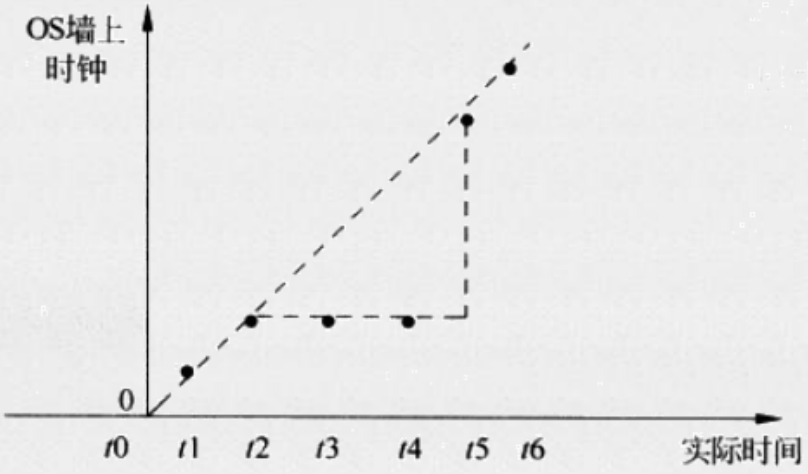
- 操作系统中的时间概念
  - 绝对时间：(Wall Time)操作系统启动后，到目前为止的总运行时间。
  - 相对时间：两个时间之间的间隔，如两次时钟中断之间的间隔
  - 当前的绝对时间= 系统启动时的时间+ 系统运行时间
  - 系统的时间是由硬件定时器定时发送时钟中断来维护的。

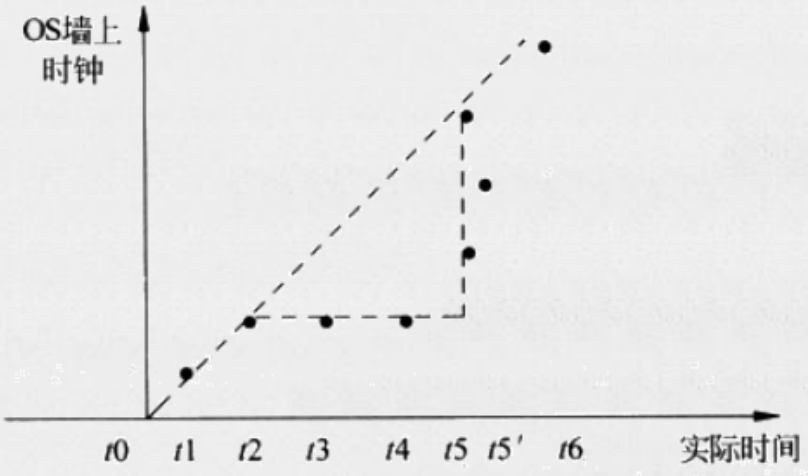
时间虚拟化

- 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。





类虚拟化技术

类虚拟化技术

- 类虚拟化概述
- CPU的虚拟化
- 内存虚拟化
- IO虚拟化
- 时间与时钟管理

类虚拟化概述

- 类虚拟化的核心思想是：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

- 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
  - 利用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。

CPU的虚拟化

- 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；

内存虚拟化

- 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

内存虚拟化

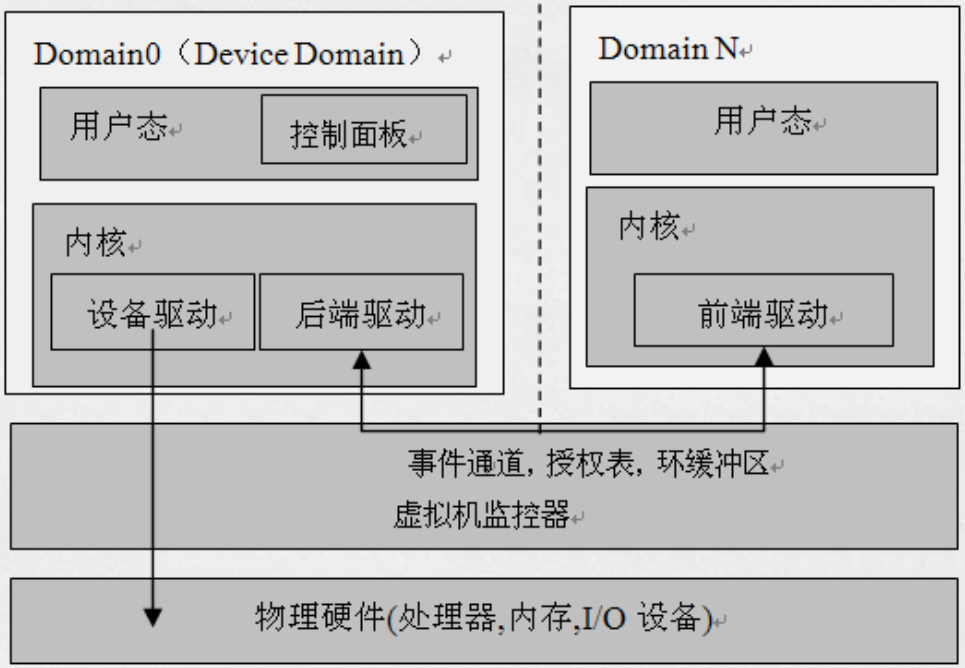
- 虚拟内存空间
  - X86架构中TLB是由硬件直接管理的；
  - 虚拟机的调度会引起TLB的刷新，带来性能问题；
  - Xen将VMM和虚拟机落于同一个虚拟内存空间；
  - 将虚拟内存空间的前64MB划分给VMM使用；

内存虚拟化

- 如何防止客户操作系统访问VMM内存页
  - 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；

内存虚拟化

- 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段设备驱动；
  - Dom0 中的一端称为后端设备驱动；
  - Driver Domain

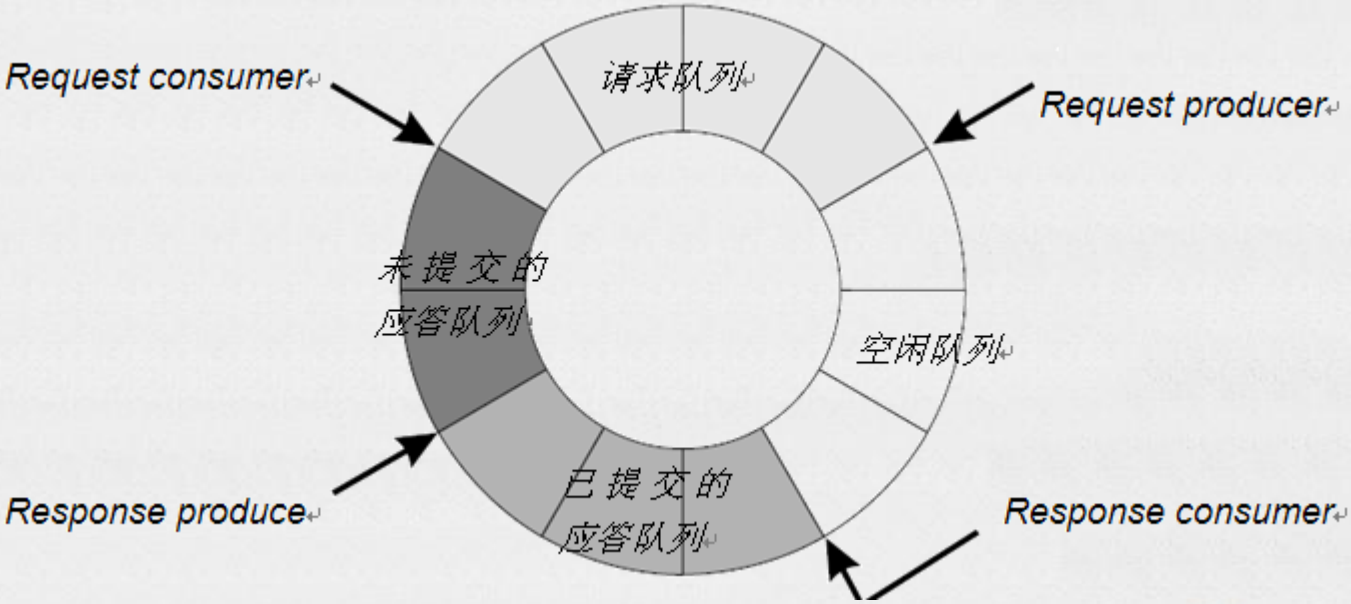


IO虚拟化

- 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，专门设路一个虚拟机作为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

IO虚拟化

- Xen采用一种称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



IO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；

IO虚拟化

- 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

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

时间和时钟的管理

- 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安全分析

Xen安全分析

- 目标是增强Xen虚拟化环境的安全性
- 首先要增强Dom 0 系统的安全性
  - Dom 0 中提供尽量少的服务
  - 配路有效的防火墙
  - 不允许用户对dom0 的访问
  - … …

Xen安全分析

- Driver Domain中DMA的安全
  - 驱动程序是运行在内核态的；
  - 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有IOMMU的。

Xen安全分析

- 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
  - 虚拟机B通过共享数据总线向设备B发送数据
  - 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

Xen安全分析

- 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（never clear it），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。
  - 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。

Xen安全分析

- 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……

实验课安排

实验课安排

- CloudStack+ XenServer
- 学习XenServer，及其安装方法
- 学习CloudStack，及其安装方法
- 学习配路CloudStack来管理XenServer的方法
- 学习CloudStack的管理

谢　谢！