硬件虚拟化在不同平台的实现

x86，ARM和RISC-V

王利明

泰晓科技 | Tinylab.org | 2023/03/04

大纲

- 硬件虚拟化介绍
- CPU虚拟化
- 内存虚拟化
- 中断虚拟化
- 设备虚拟化

硬件虚拟化介绍

- 什么是硬件虚拟化
- 各个平台的支持

硬件虚拟化介绍 – 什么是硬件虚拟化

- 体系结构支持：规范定义，包括
  - 模式切换：Host CPU <-> Guest CPU 切换。CPU资源隔离
  - 二阶段地址转换：GVA -> GPA 和 GPA -> HPA。内存资源隔离
  - 中断控制器(Interrupt Controller)支持：中断注入和透传。中断资源隔离
  - IOMMU：Input-Output Memory Management Unit，DMA Remapping。DMA和设备访问内存隔离
- 有相应的硬件实现
- 相关软件支持：包括
  - Firmware：OpenSBI，BIOS
  - Hypervisor：KVM，XEN
  - I/O用户态: qemu
  - OS：Linux

硬件虚拟化介绍 - x86架构

- 规范
  - **Intel:**
    - **VT-X**
    - **VT-D**
    - **APIC： x2APIC**
  - **AMD:**
    - **AMD-V**
    - **AMD IOMMU Spec**
    - **AVIC**
- 硬件实现
  - **Intel CPU**
  - **AMD CPU**
  - **Hygon CPU**

硬件虚拟化介绍 – ARM架构

- 规范
  - **ARMv8:**
    - **ARMv8.2 VHE**
    - **GICv3 & GICv4**
    - **SMMU**
  - **ARMv9**
- 硬件实现
  - **ARMv8.2：Kunpeng 920，Ampere Altra(Arm Neoverse N1)，AWS Graviton2(Arm Neoverse N1)**
  - **ARMv8.4：AWS Graviton3(Arm Neoverse V1)**
  - **ARMv9：阿里/平头哥的倚天710**

硬件虚拟化介绍 – RISC-V架构

- 规范
  - **Hypervisor Extension, Version 1.0**
  - **RISC-V Advanced Interrupt Architecture (AIA)**
  - **RISC-V IOMMU specification**
- 硬件实现(2023.3)
  - **SiFive P600：实现了Hypervisor Extension**

大纲

- 硬件虚拟化介绍
- **CPU虚拟化**
- 内存虚拟化
- 中断虚拟化
- 设备虚拟化

CPU虚拟化

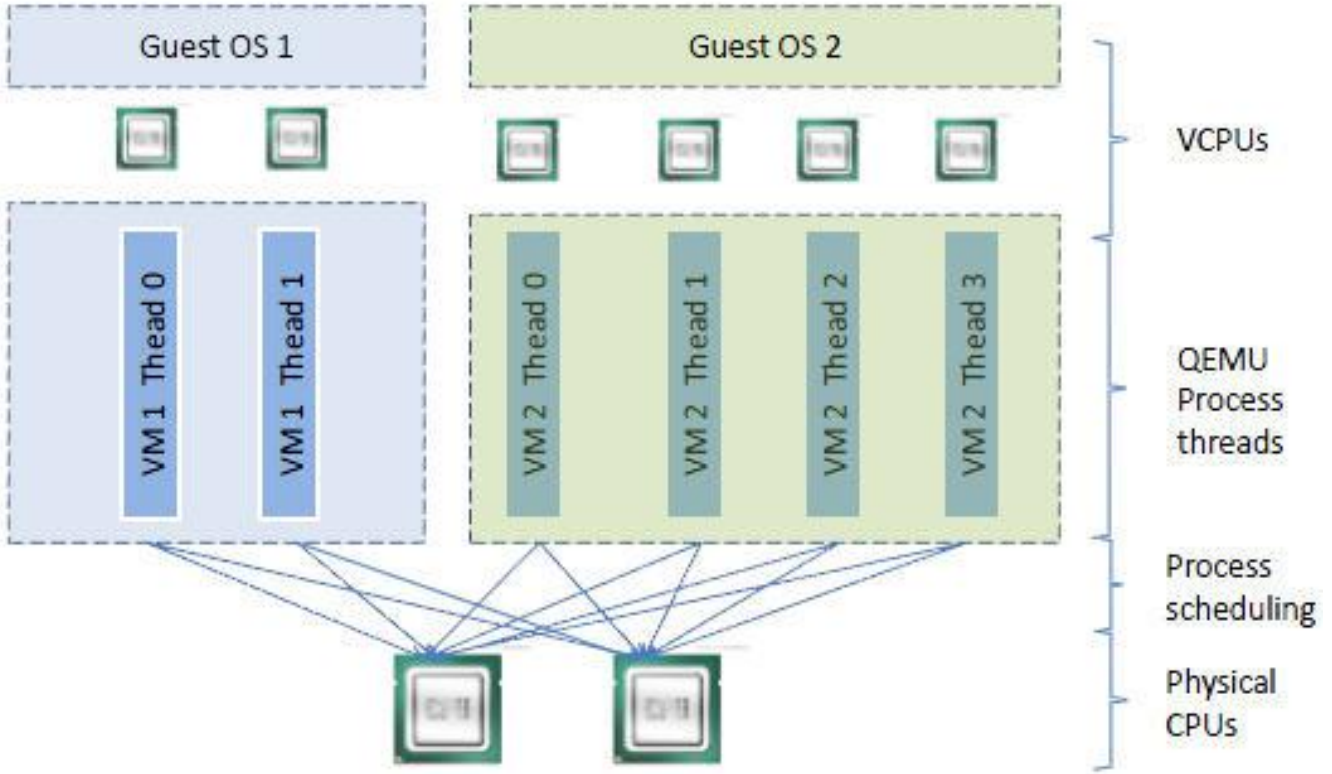
- CPU虚拟化介绍
- vCPU介绍
- 各个平台的支持

CPU虚拟化 – CPU虚拟化介绍

- **CPU支持模式切换：硬件支持**
  - **Host CPU <-> Guest CPU：CPU运行环境的隔离，CPU虚拟化的基础**
- **模式切换过程：**
  - **vm entry：从Host CPU进入到Guest CPU**
  - **vm exit：从Guest CPU退出到Host CPU**
- **切换过程的上下文的保存和恢复：注意，都是在Host CPU模式下操作。**
  - **vm entry前：保存Host CPU State，恢复Guest CPU state**
  - **vm exit后：保存Guest CPU state，恢复Host CPU State**
- **模式切换方法：**
  - **vm entry：特殊指令实现**
  - **vm exit：执行特定指令；中断；异常**

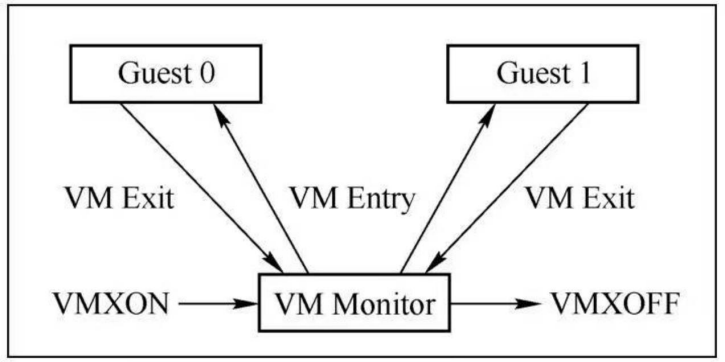
CPU虚拟化 – vCPU介绍

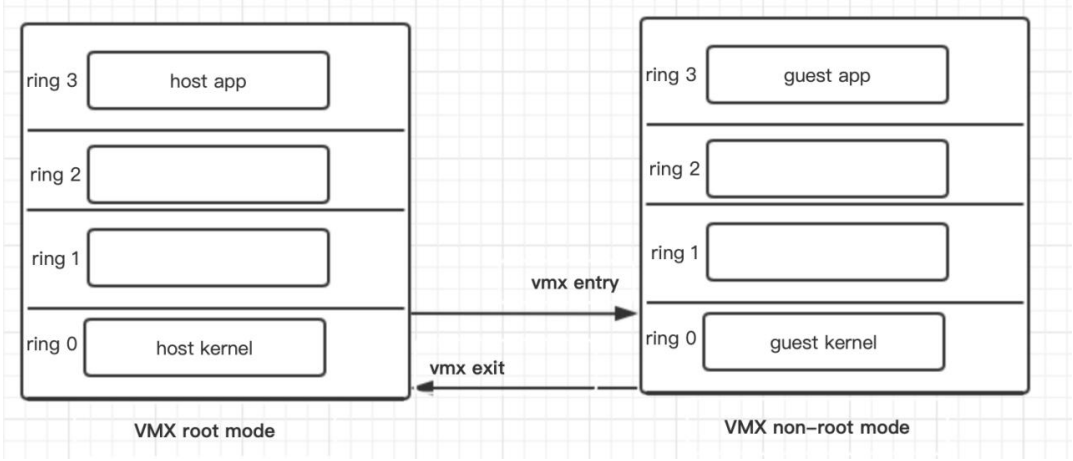
- vCPU是什么?
  - qemu的一个Linux线程
- vCPU
  - 主要工作就是在Host CPU和Guest CPU之间不断切换和运行
- pCPU和vCPU的关系
  - pCPU：物理CPU
  - pCPU 和 vCPU的关系
    - 1:1
    - m:n
- 进程调度器的作用
  - vCPU线程调度
  - vCPU线程迁移



CPU虚拟化 - x86架构(1)

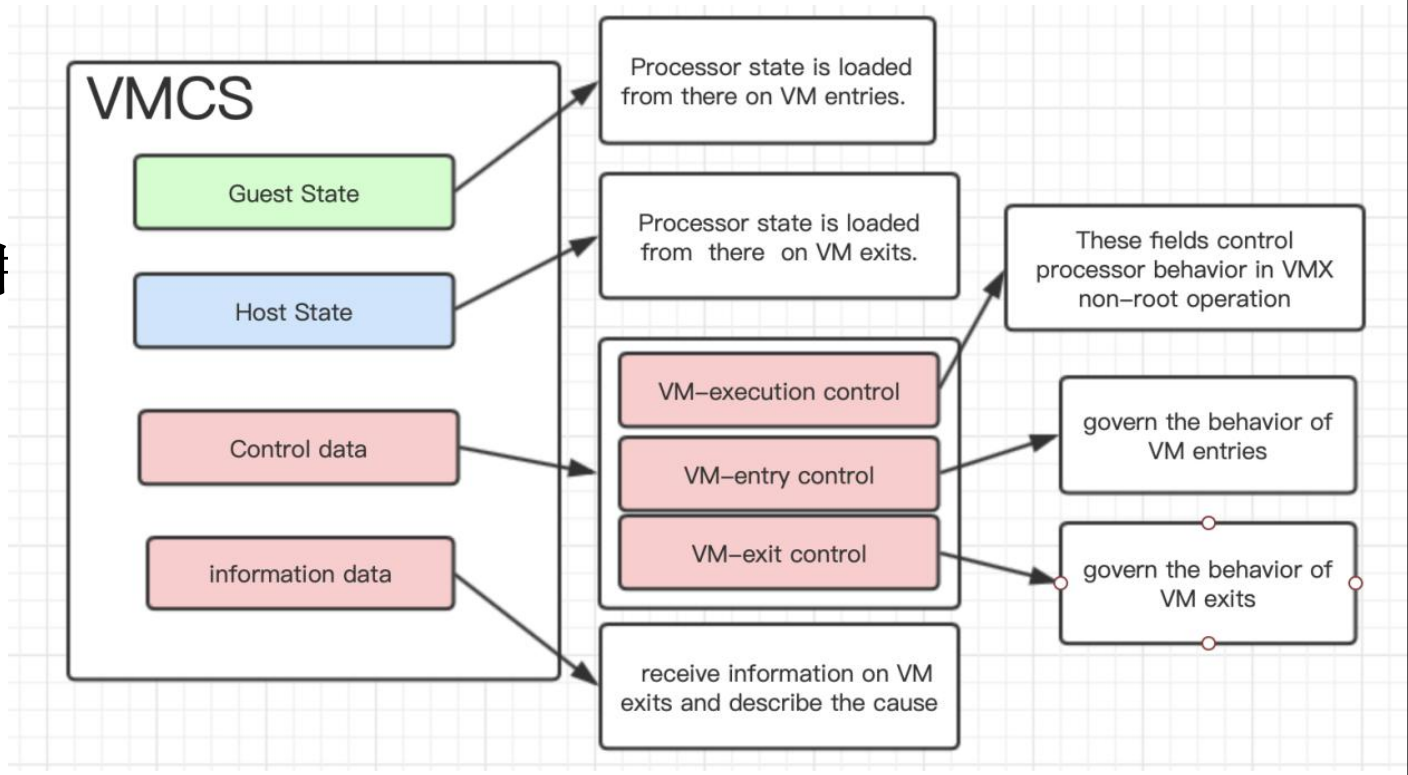
- **VMX Operation Mode**
  - **VMX root mode：Host CPU**
  - **VMX non-root mode：Guest CPU**
- **VMX Operation Mode的开启和关闭**
  - **VMXON**
  - **VMXOFF**
- **模式切换方法：**
  - **vm entry：vmlaunch/vmresume指令**
  - **vm exit：执行指令；发生异常或中断**





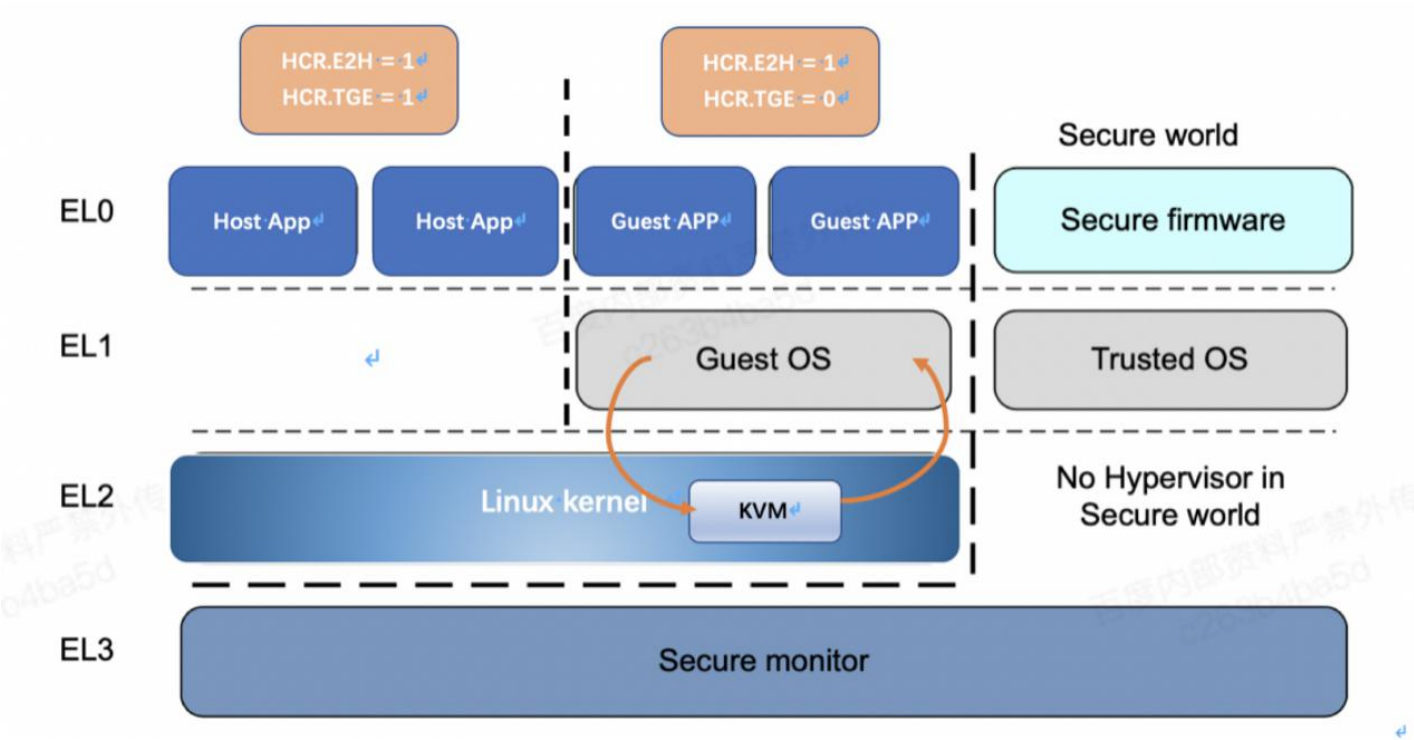
CPU虚拟化 – x86架构(2)

- 上下文状态：**VMCS，Virtual Machine Control Structure**
- **一个vCPU对应一个VMCS 状态**
- **vm entry和vm exit的时候，VMCS由硬件进行保存和恢复，只有x86支持**



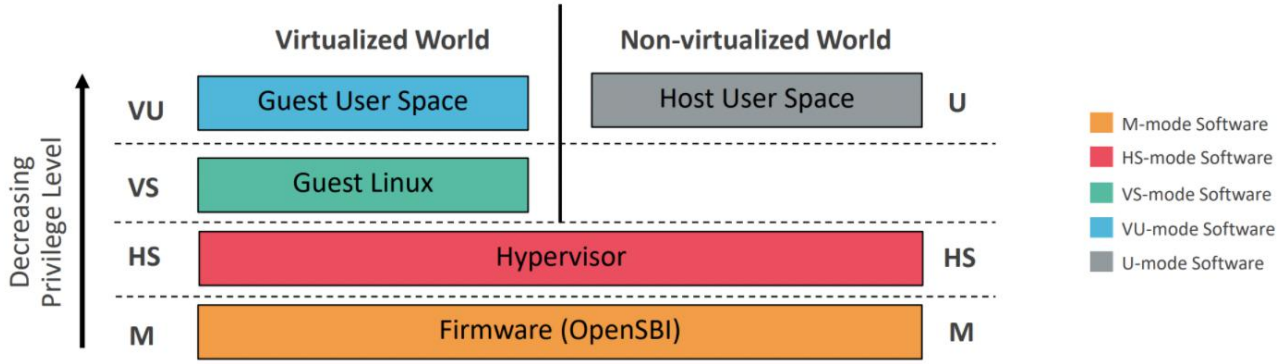
CPU虚拟化 – ARM架构

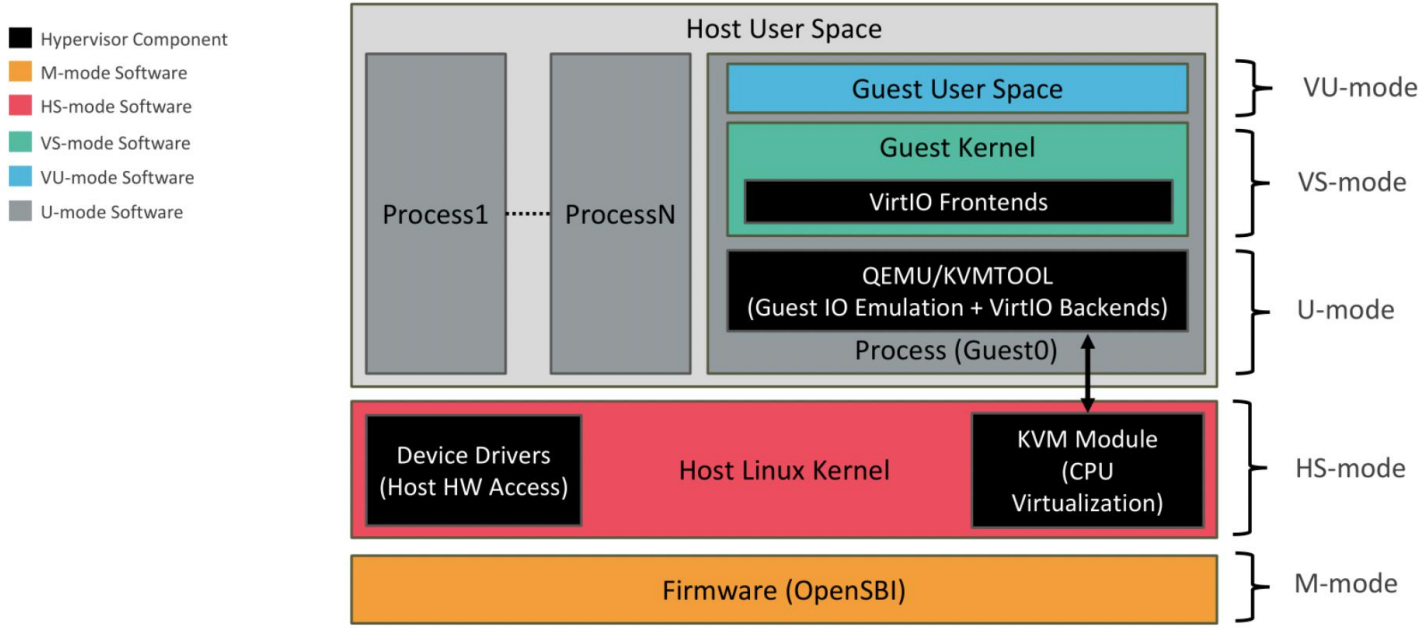
- **Exception Level**
  - **EL0**
    - **Host App：Host CPU**
    - **Guest App：Guest CPU**
  - **EL1：Guest CPU**
  - **EL2：Host CPU**
  - **EL3**
- **模式切换方法**
  - **vm entry：eret指令**
  - **vm exit：指令或者异常**
- **上下文状态自定义，由软件来实现保存和恢复**



CPU虚拟化 – RISC-V架构

- **Privilege Level**
  - **U：Host CPU**
  - **VU：Guest CPU**
  - **VS：Guest CPU**
  - **HS：Host CPU**
  - **M**
- **模式切换方法**
  - **vm entry：sret指令**
  - **vm exit：指令或者异常**
- **上下文状态自定义，由软件来实现保存和恢复**





大纲

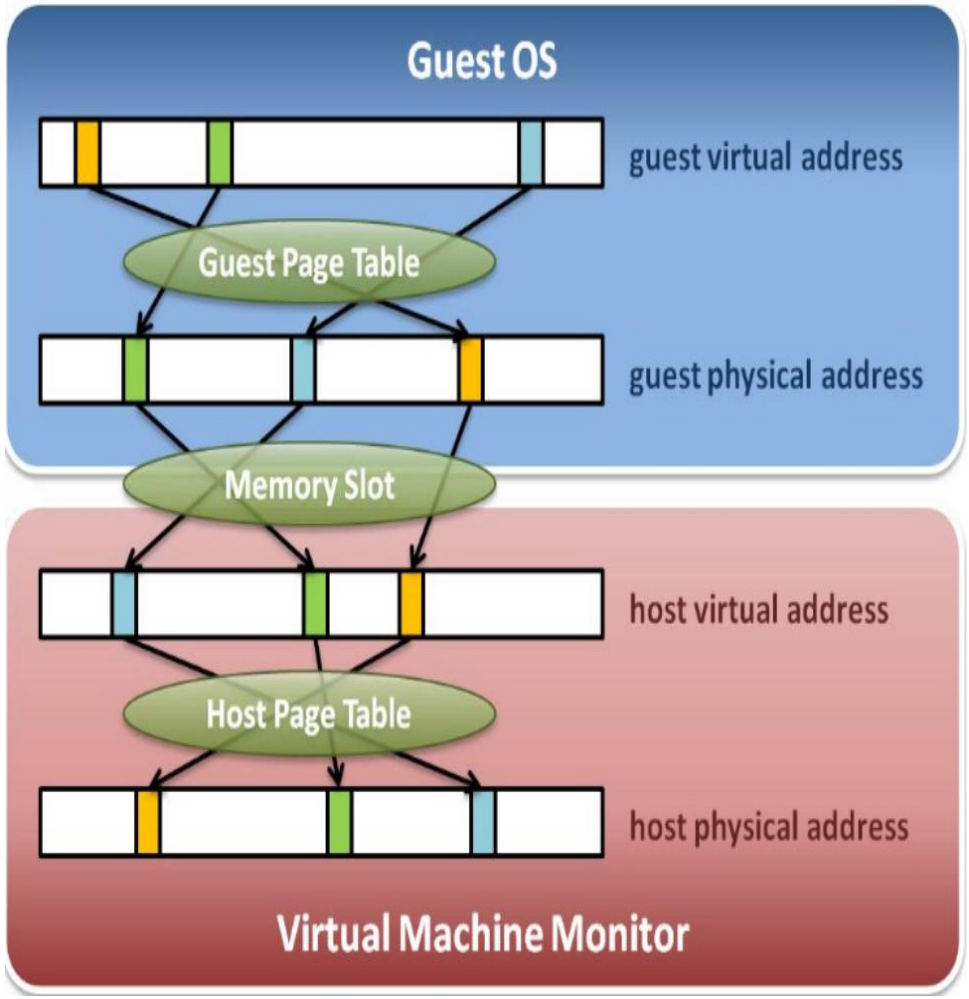
- 硬件虚拟化介绍
- CPU虚拟化
- 内存虚拟化
- 中断虚拟化
- 设备虚拟化

内存虚拟化

- 内存虚拟化介绍
- 各个平台的支持

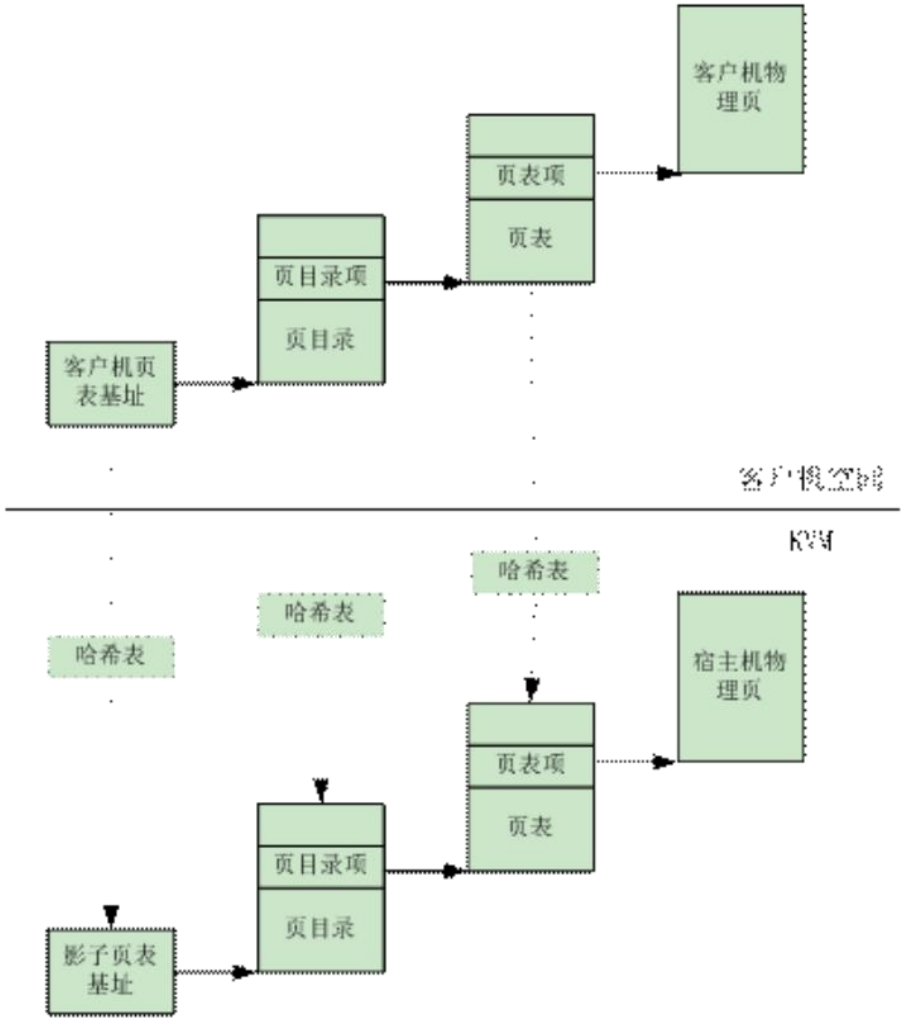
内存虚拟化 – 介绍

- 基本概念
  - GVA： Guest Virual Address
  - GPA： Guest Physical Address
  - HVA： Host Virual Address
  - HPA： Host Physical Address
- 转换关系和维护
  - GVA ----> GPA : guest OS维护
  - HVA ----> HPA : host OS维护
  - GPA <----> HVA : qemu/KVM
  - GPA <----> HPA： 硬件支持 EPT/NPT
- 内存虚拟化目标:
  - 如何让Guest的访问地址(GVA或者GPA)最终转换成HPA



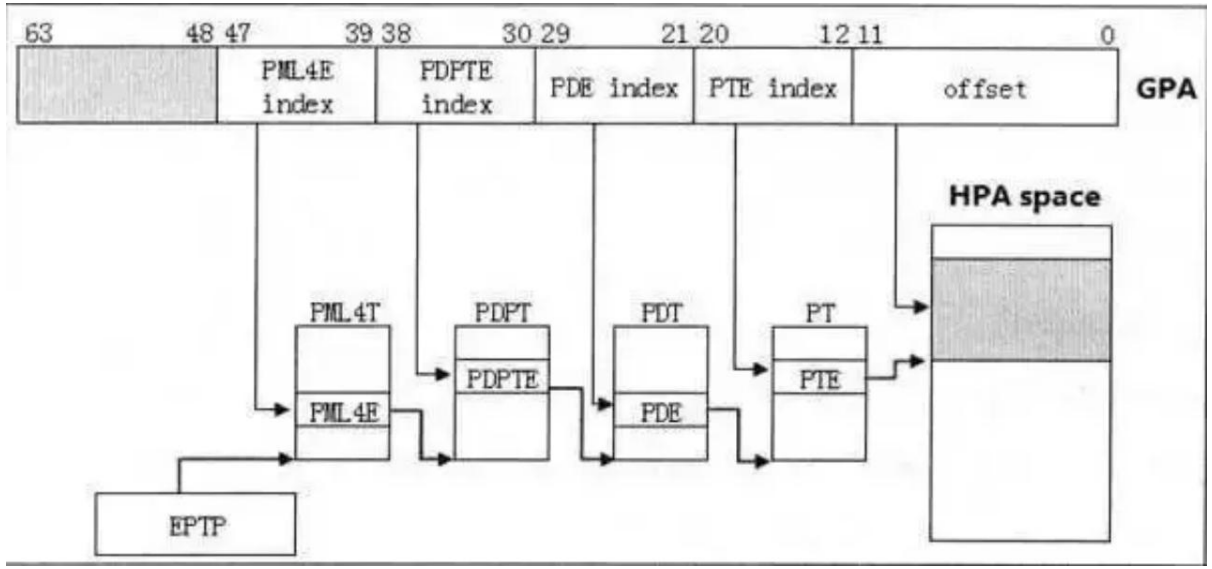
内存虚拟化 – 软件实现之影子页表

- 影子页表
  - GVA <----> HPA：直接完成GVA到HPA的装换
  - 被hypervisor载入到物理MMU中的页表是影子页表
- 缺陷
  - 需要为Guest OS的每个进程维护一个影子页表，资源消耗大
  - Guest OS在读写CR3、执行INVLPG指令或客户页表不完整等情况下均会导致vm exit，内存虚拟化效率很低
  - Guest OS的页表和和影子页表的同步也比较复杂。



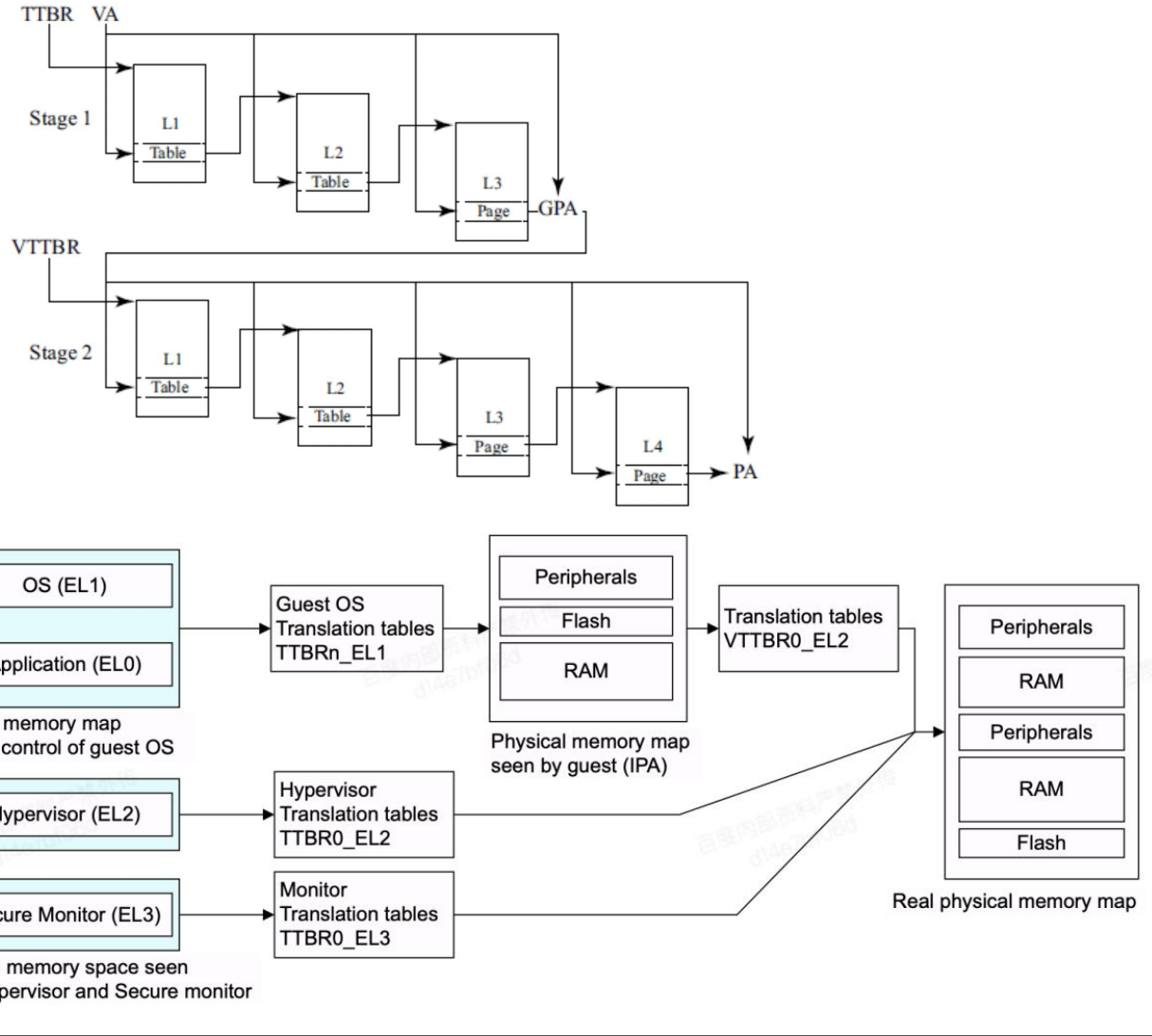
内存虚拟化 - x86架构

- **EPT/NPT**
  - **Intel：EPT，Extended Page Table**
  - **AMD：NPT**
- **二阶段地址转换，完成内存虚拟化:**
  - **GVA -> GPA：CR3指向Guest OS页表，完成第一阶段转换，由硬件MMU完成**
  - **GPA -> HPA：EPTP，指向EPT表，完成第二阶段转换，由硬件自动完成**
- **相对影子页表，只需要一张表即可完成Guest OS的内存虚拟化**



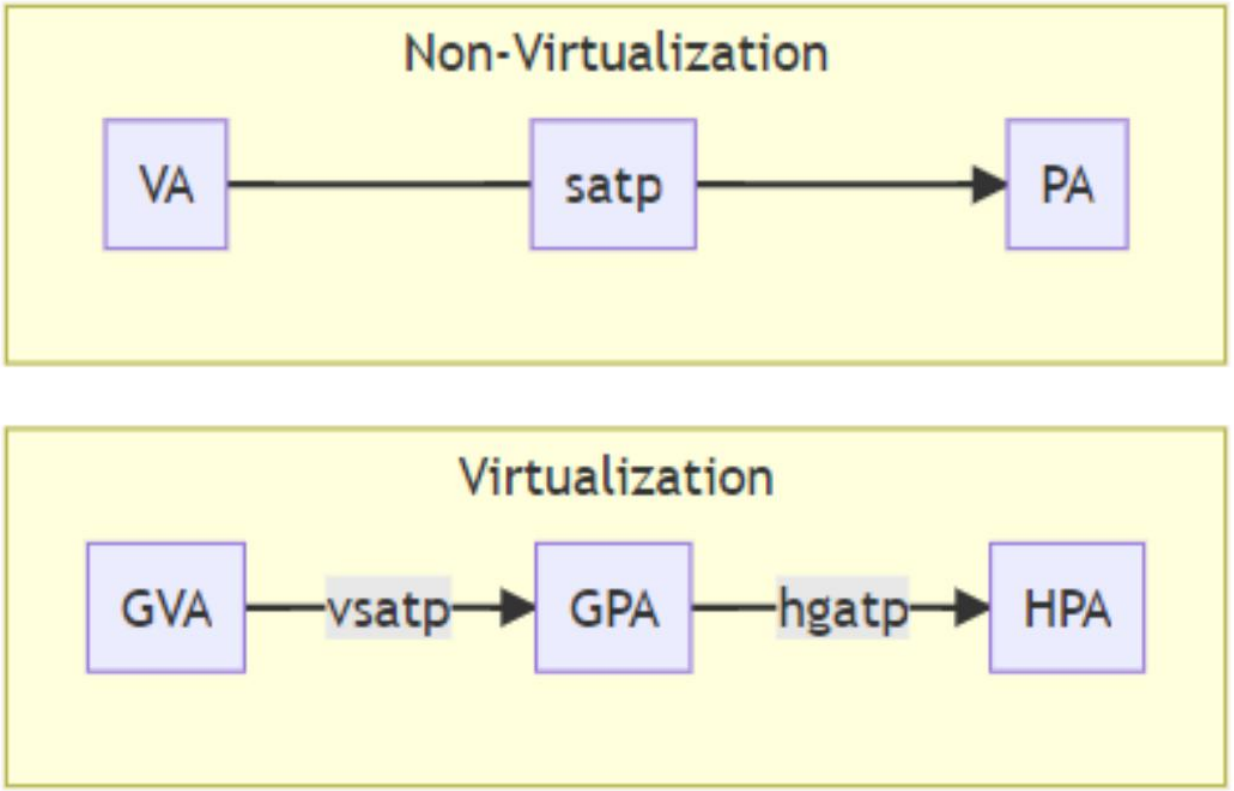
内存虚拟化 - ARMv8架构

- **VTTBR：Virtualization Translation Table Base Register**
- **ARMv8的二阶段地址转换：**
  - **GVA -> GPA(IPA)：TTBRn_EL1**
  - **GPA -> HPA：VTTBR0_EL2**



内存虚拟化 – RISC-V架构

- **vsatp：Virtual Supervisor Address Translation and Protection Register (**vsatp**)**
- **hgatp：Hypervisor Guest Address Translation and Protection Register**
- **RISC-V的二阶段地址转换：**
  - **GVA -> GPA：vsatp**
  - **GPA -> HPA：hgatp**



大纲

- 硬件虚拟化介绍
- CPU虚拟化
- 内存虚拟化
- **中断虚拟化**
- 设备虚拟化

中断虚拟化

- 中断虚拟化介绍
- 各个平台的支持

中断虚拟化 – 介绍

- 中断处理流程：
  - 中断触发：中断源发生中断事件，向CPU发送中断信号
  - 检测中断：CPU检测到中断，停止当前执行流程，跳转到中断向量
  - 处理中断：CPU保存上下文，调用和执行中断处理函数，然后返回原有执行流程
- 中断来源不同，导致虚拟化方式不同
  - IPI：inter-processor interrupt
  - timer中断
  - external 中断：包括MSI

中断虚拟化 – 虚拟化方法1

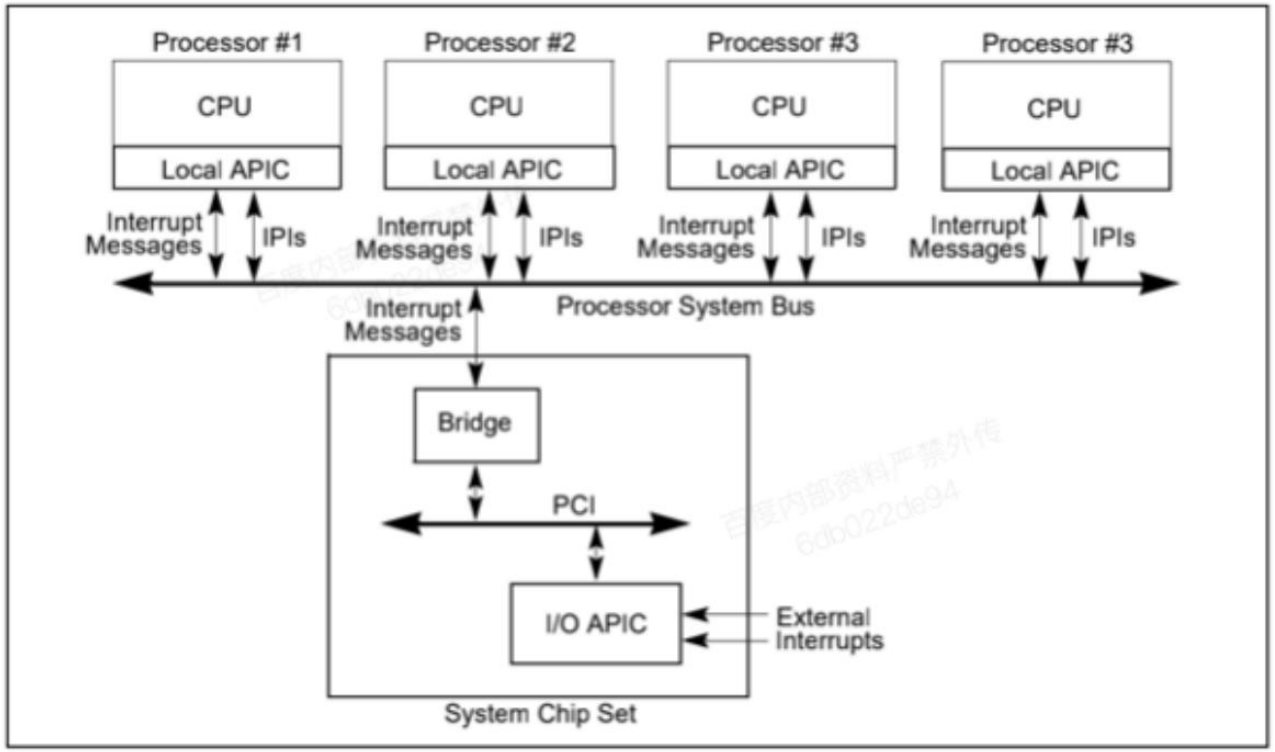
- 虚拟化方式1：发生vm exit和中断注入
  - 如果中断发生在当前pCPU上：如果vCPU处于运行状态，Guest CPU会发生vm exit进入Host CPU；
  - 如果中断发生在其他pCPU，需要向当前pCPU发送IPI
  - 在Host CPU检测到中断发生，把相关中断信息写入Guest CPU State，然后在vm entry后，触发Guest CPU发生中断，完成中断注入
  - vCPU在Guest CPU模式下完成处理中断
- 特点：
  - 中断发生时，vCPU会发生vm exit，效率较低
  - 在没有中断控制器的特殊支持时，大多数硬件都支持此种中断虚拟化

中断虚拟化 – 虚拟化方法2

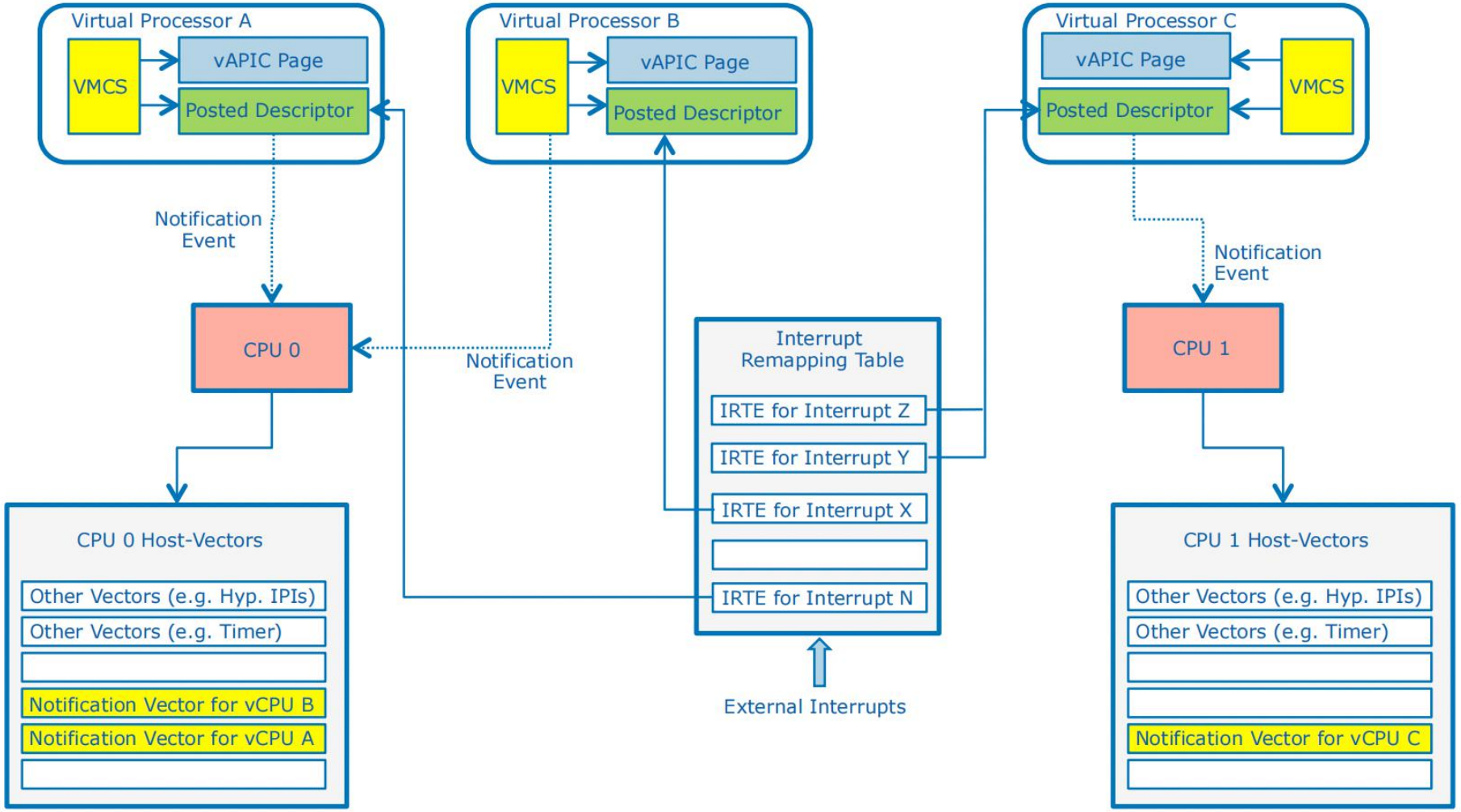
- 虚拟化方式2：中断直接投递给vCPU
  - 如果中断发生在当前pCPU上：如果vCPU处于运行状态，Guest CPU不会发生vm exit，中断直接投递给vCPU；如果vCPU处于等待状态，调用kvm_kick_cpu, 唤醒vCPU的线程，把中断信息投递到Guest CPU State，让vCPU进入Guest CPU模式
  - 如果中断发生在其他pCPU，需要向当前pCPU发送IPI
  - vCPU在Guest CPU模式下完成处理中断
- 特点：
  - 需要中断控制器的特殊支持
  - vCPU不需要产生vm exit，效率高

中断虚拟化 - x86架构(1)

- 中断控制器:
  - LAPIC
  - IOAPIC

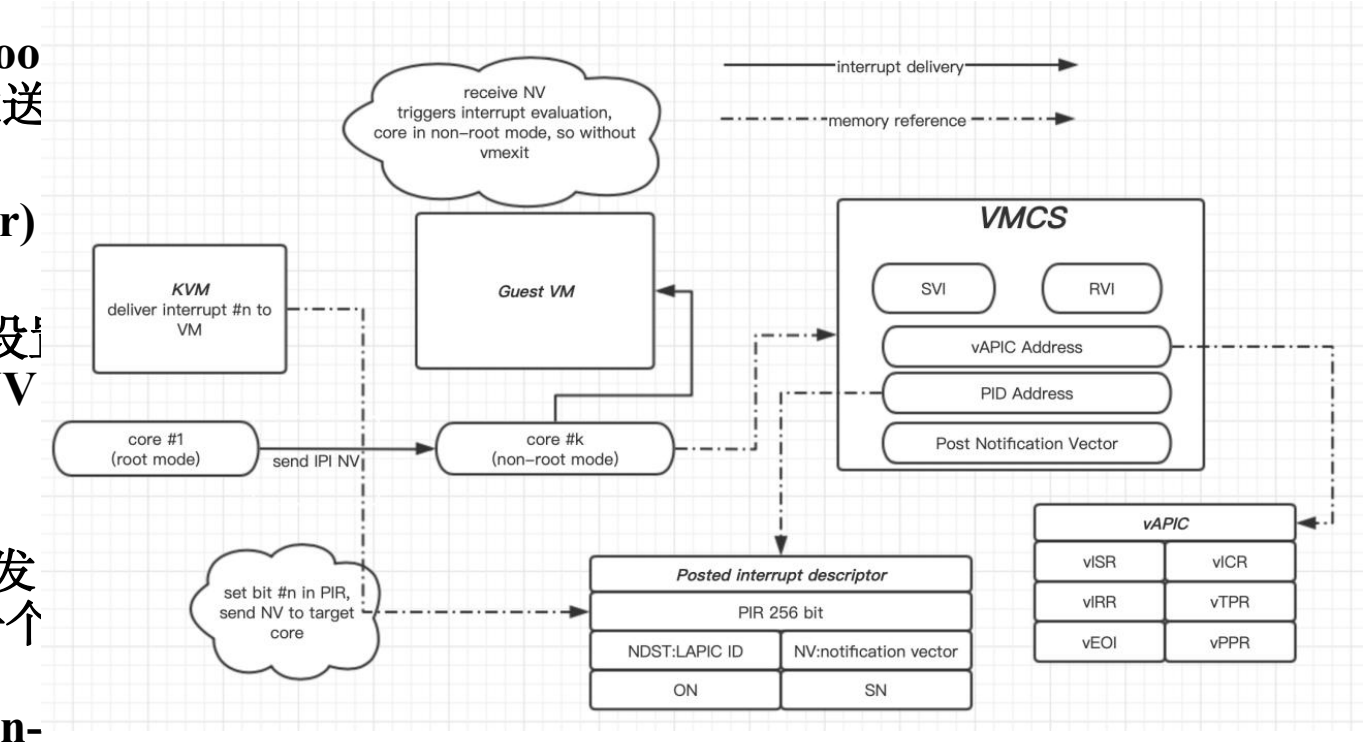


中断虚拟化 - x86 Posted Interrupt(1)



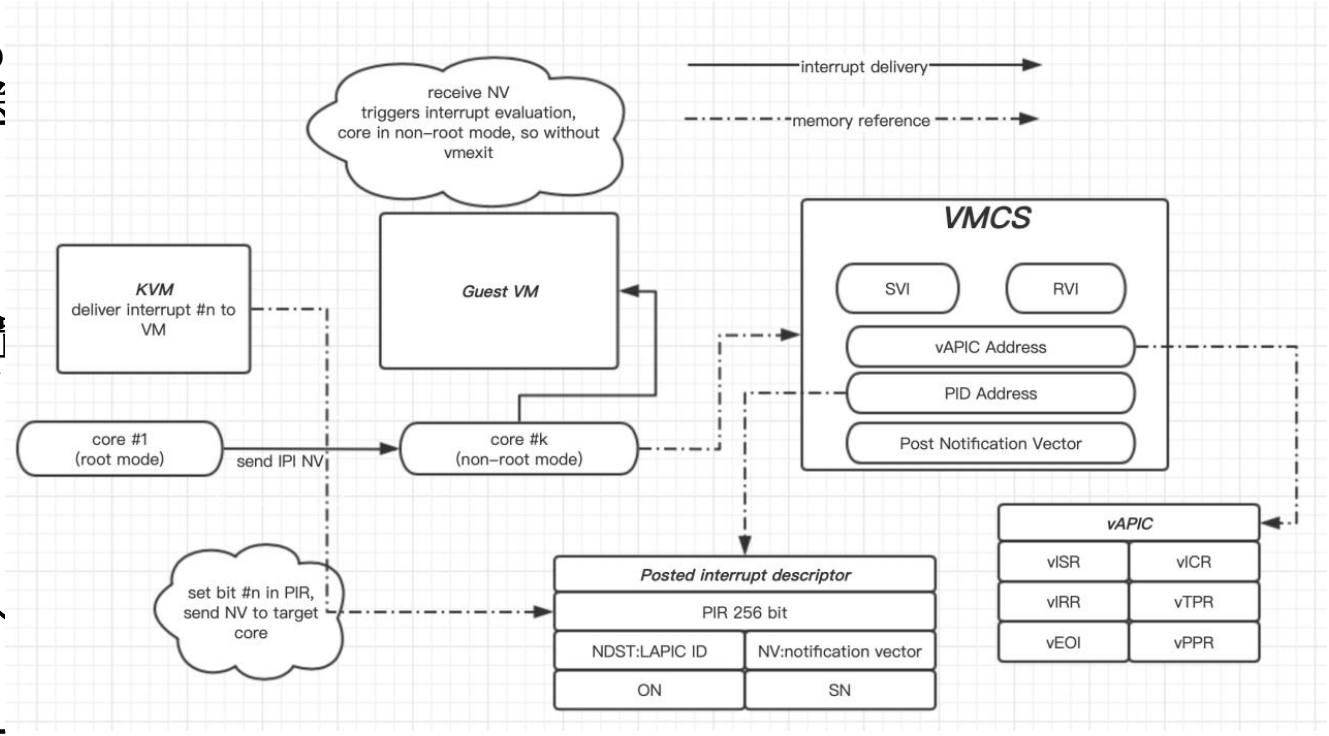
中断虚拟化 – x86 Posted Interrupt(2)

- 中断透传： VT-d Posted-Interrupts
  - 中断的注入：虚机在运行，处于non-roo模式，core#1向运行在core #k的虚机发送一个中断
  - vmcs 中的PNV(Post Notification Vector)初始化为POSTED_INTR_NV
  - core #1发现core #k 处于non-root模式设PIR，发送中断号为POSTED_INTR_NV的IPI给core #k
  - core #k 收到POSTED_INTR_NV中断发现和VMCS中的PNV一样则知道这是一个posted interrupt 无需vmexit，直接可以在non-root模式处理



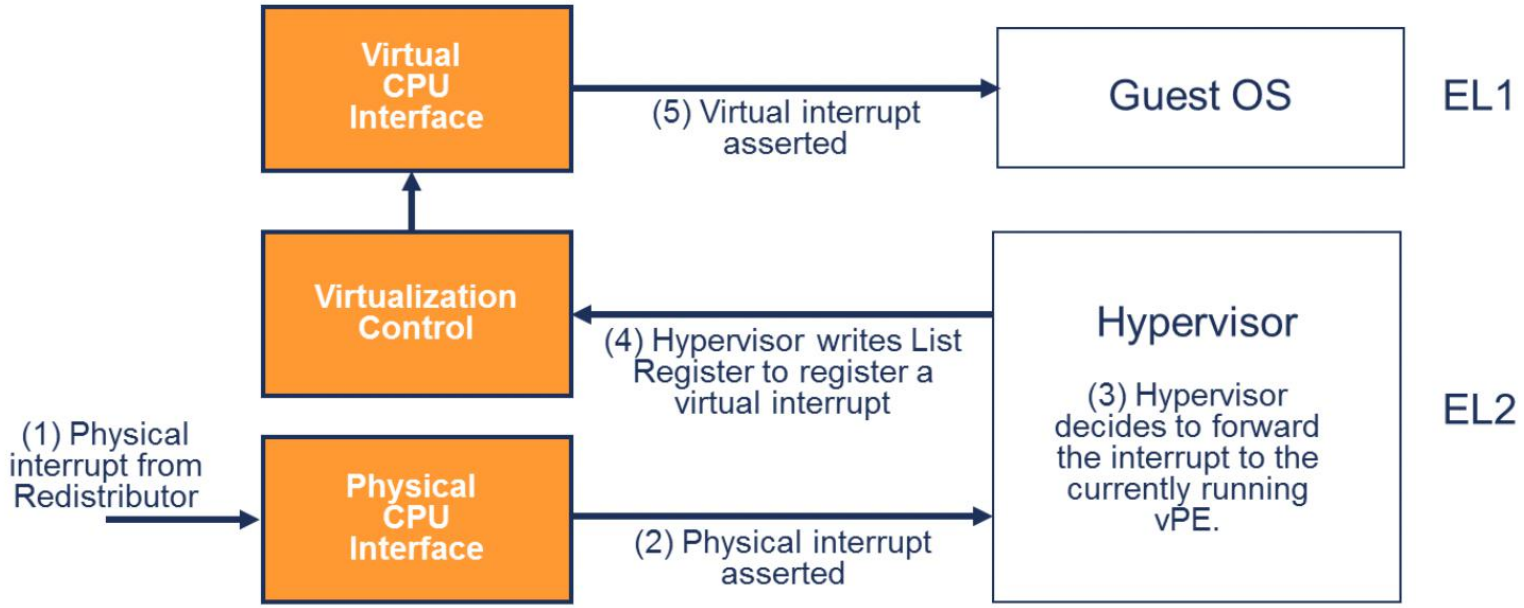
中断虚拟化 – x86 Posted Interrupt(3)

- **中断透传： VT-d Posted-Interrupts**
  - **中断的注入： 虚机在运行，处于non-roo模式， core#1向运行在core #k的虚机发送一个中断**
  - **vmcs 中的PNV(Post Notification Vector)初始化为POSTED_INTR_NV**
  - **core #1发现core #k 处于non-root模式设PIR，发送中断号为POSTED_INTR_NV的IPI给core #k**
  - **core #k 收到POSTED_INTR_NV中断发现和VMCS中的PNV一样则知道这是一个posted interrupt 无需vmexit，直接可以在non-root模式处理**



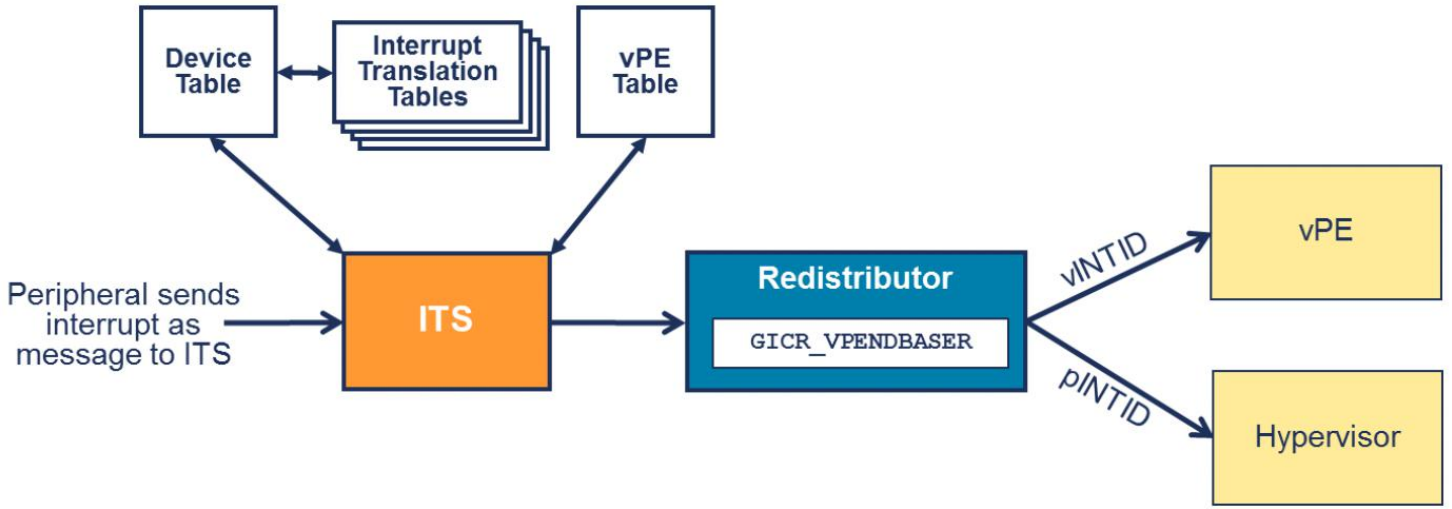
中断虚拟化 – ARMv8架构(1)

- 中断控制器：中断注入
  - GICv3



中断虚拟化 – ARMv8架构(2)

- 中断控制器： 中断透传
  - GICv4： **Direct Injection of Virtual LPIs**

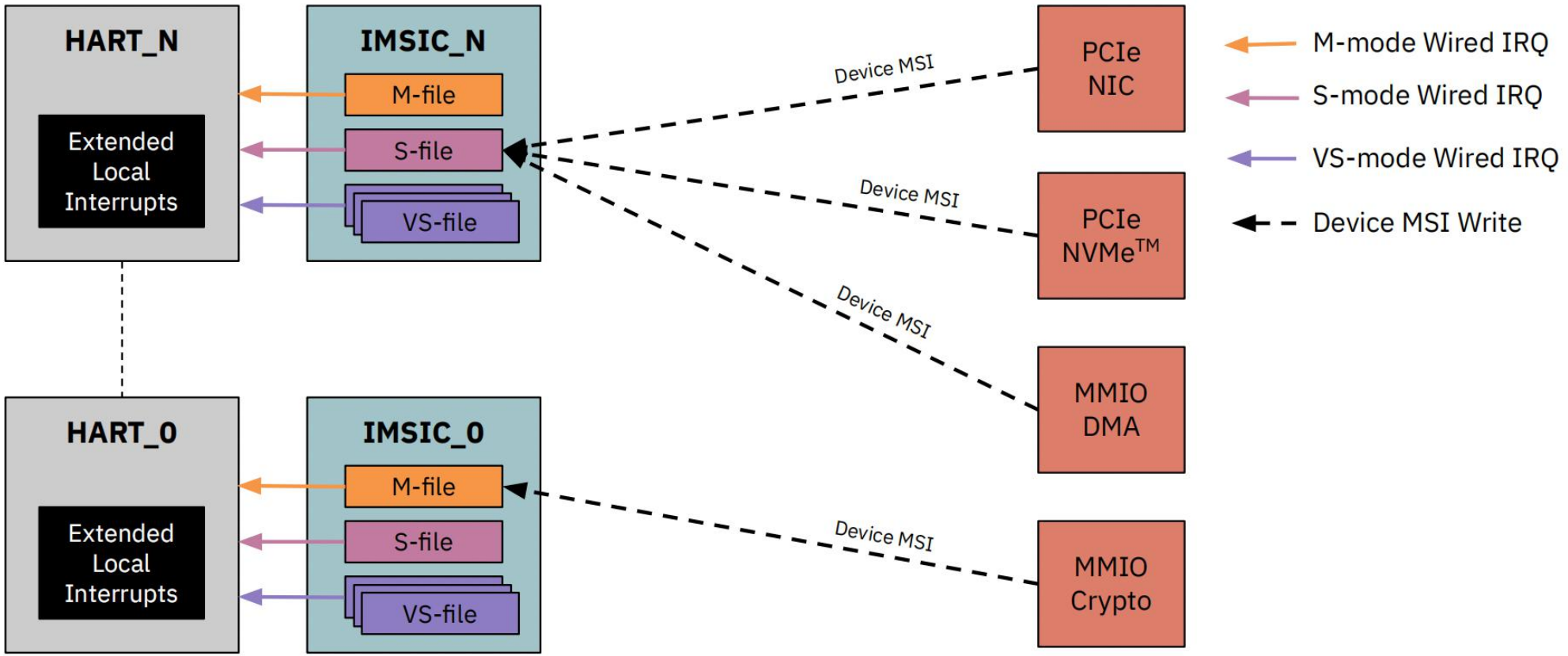


中断虚拟化 – RISC-V架构(1)

- **中断控制器：**
  - 现有的中断控制器PLIC
    - 不支持MSI
    - **不支持中断虚拟化**
- **RISC-V Advanced Interrupt Architecture： AIA，定义了两个控制器：**
  - Incoming Message Signaled Interrupt Controller (IMSIC)： 支持MSI； 支持IPI虚拟化
  - Advanced Platform Level Interrupt Controller (APLIC)： 支持Wired Interrupt

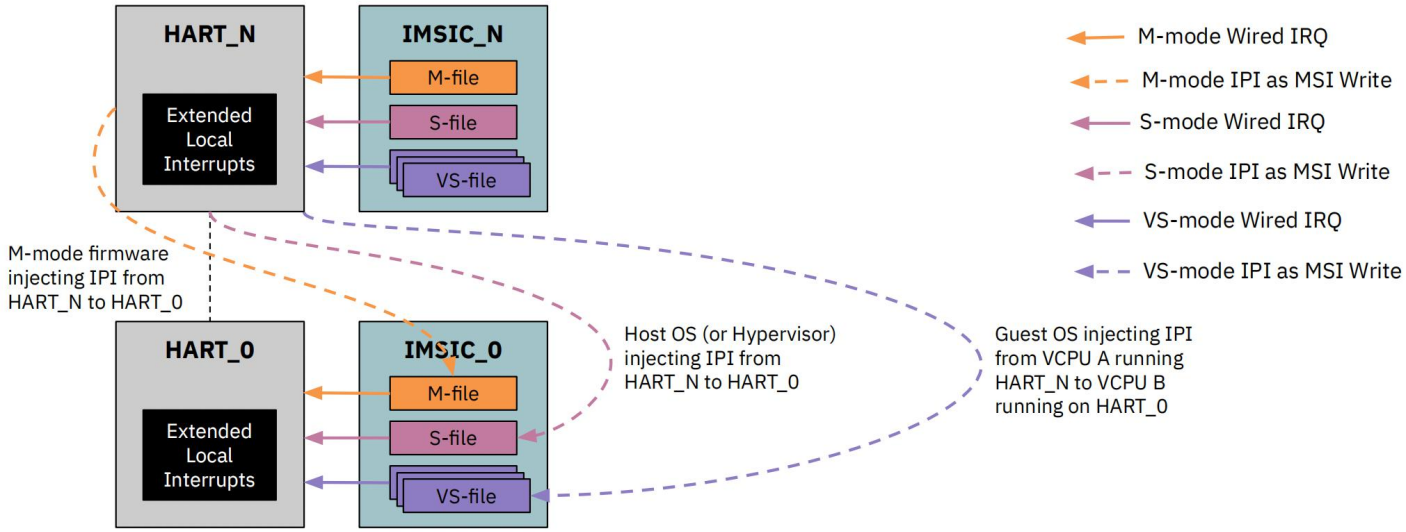
中断虚拟化 – RISC-V架构(2)

- 中断控制器：IMSIC发送MSI



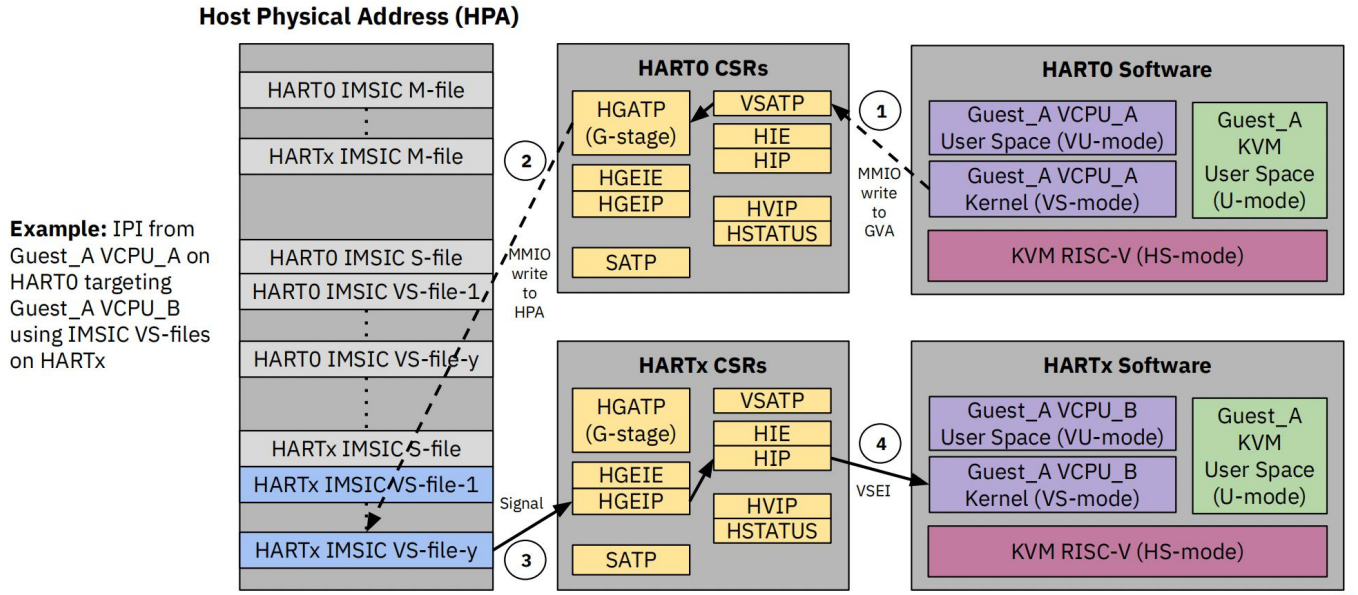
中断虚拟化 – RISC-V架构(3)

- 中断控制器：通过IMSIC发送IPI
  - Firmware
  - Hypervisor
  - Guest OS



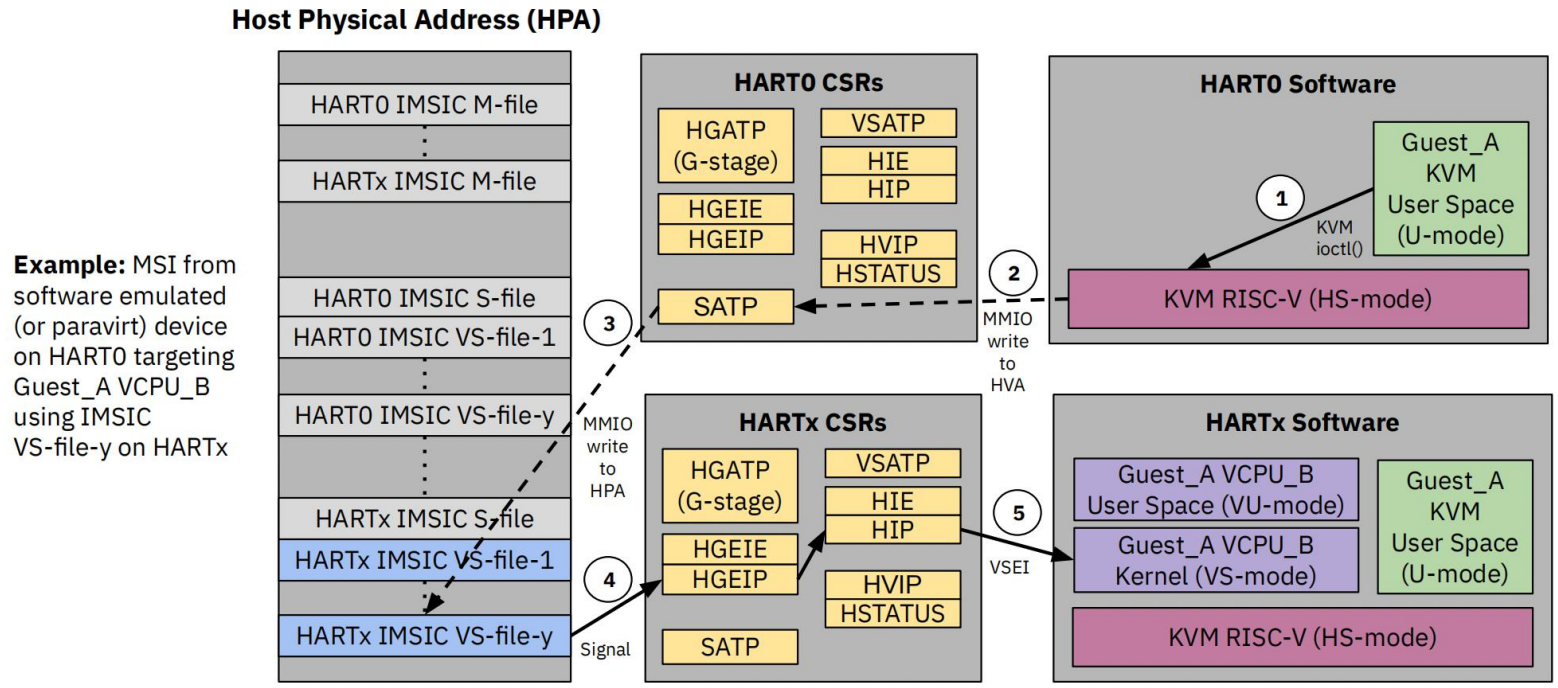
中断虚拟化 – RISC-V架构(4)

- 中断控制器：Guest OS A通过IMSIC发送IPI，从vCPU A到vCPU B的详细流程
  - HART0的Guest_A的vCPU A写MMIO到某GVA
  - GVA通过VSATP和HGATP，经过两次转换变为HPA
  - 写入HARTx的IMSIC的寄存器
  - 发送信号给HARTx的CSR
  - 触发HARTx的Guest_A的vCPU B的IPI



中断虚拟化 – RISC-V架构(5)

- 中断控制器：KVM通过IMSIC注入中断的详细流程
  - HART0的KVM写MMIO到某HVA
  - HVA通过SATP和转换变为HPA
  - 写入HARTx的IMSIC的寄存器
  - 发送信号给HARTx的CSR
  - 注入中断到HARTx的Guest_A的vCPU B



大纲

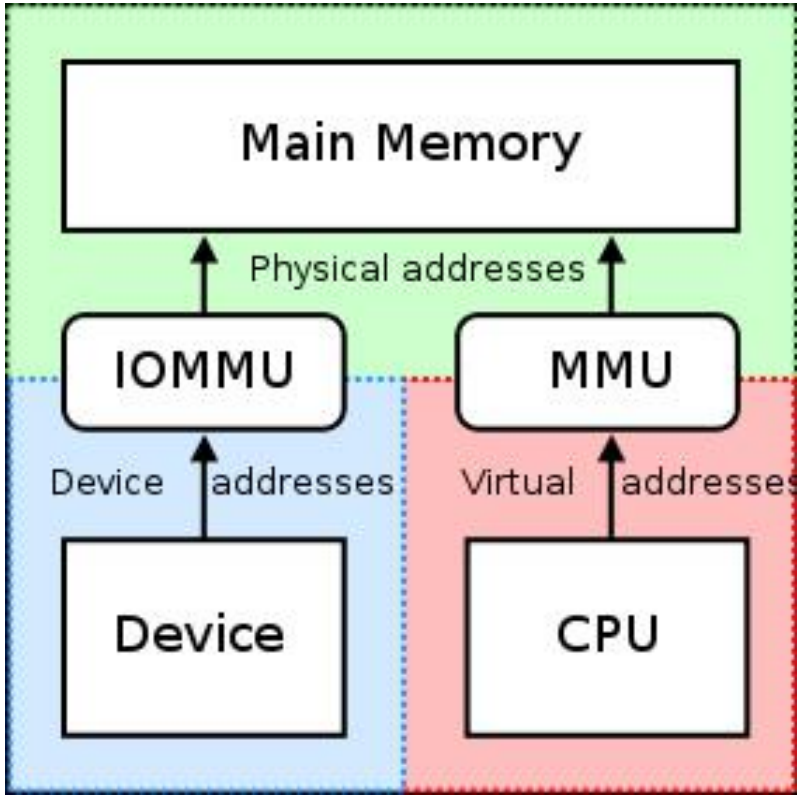
- 硬件虚拟化介绍
- CPU虚拟化
- 内存虚拟化
- 中断虚拟化
- **设备虚拟化**

设备虚拟化

- **IOMMU介绍**
- **设备虚拟化介绍**
- **设备透传**

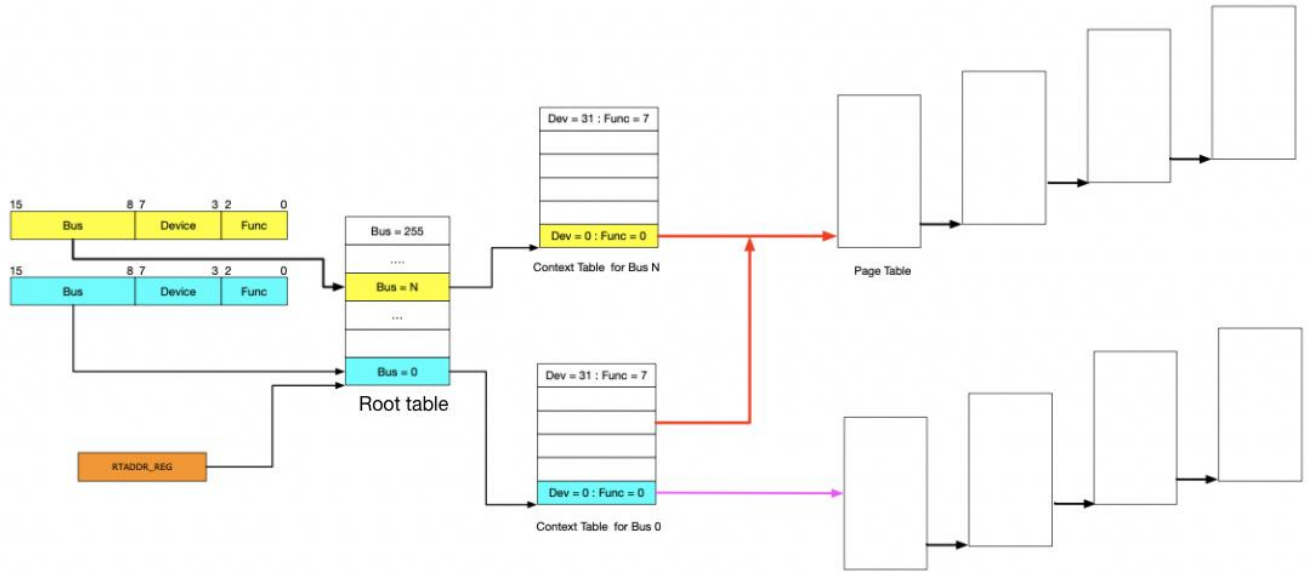
设备虚拟化 – IOMMU介绍

- **IOMMU：Input–Output Memory Management Unit**
  - **MMU：CPU通过VA访问内存**
  - **IOMMU：设备通过IO VA访问内存**
- **IOMMU作用**
  - **DMA设备的虚拟地址转换。完成IOVA -> HPA的转换。把设备发送的IOVA转换成HPA。**
  - **DMA设备的内存写保护**
  - **Interrupt remapping和虚拟化**
  - **IO设备可以共享页表**



设备虚拟化 - IOMMU的地址转换

- IOMMU地址转换流程
  - 首先需要建立IOMMU页表，并将页表基址配置到正确的Context Entry中。
  - 在设备发起DMA请求时，会将自己的Source Identifier(包含Bus、Device、Func)包含在请求中
  - IOMMU根据这个标识，以RTADDR_REG指向空间为基地址，然后利用Bus、Device、Func在Context Table中找到对应的Context Entry，即页表基址
  - 最后利用页表即可将设备请求的虚拟地址（IOVA）翻译成物理地址

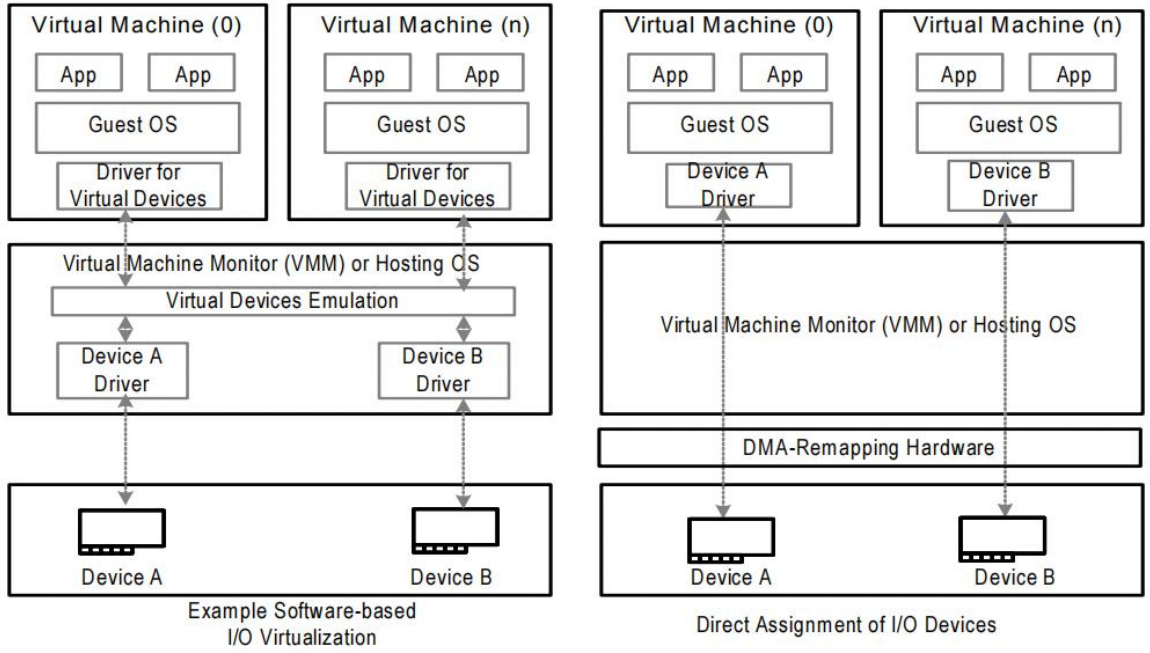


设备虚拟化 – 不同体系的IOMMU支持

- **X86 IOMMU：包括dma remapping和 interrupt remapping**
  - **Intel：VT-D**
  - **AMD：AMD IOMMU**
- **ARMv8：SMMUv3**
- **RISC-V：RISC-V IOMMU**

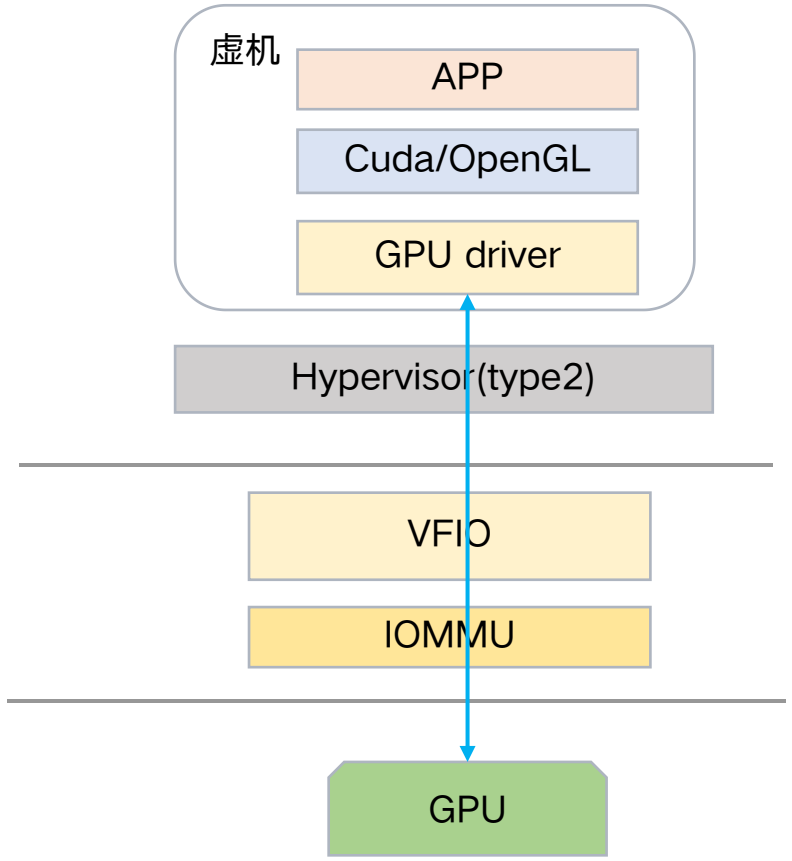
设备虚拟化 – 设备透传的基础

- **IOMMU是设备透传的基础：为什么？**
- **VFIO：Virtual Function I/O是Linux下利用IOMMU构建设备直通方案。用户态进程可以使用VFIO驱动直接访问硬件，并且由于整个过程是在IOMMU的保护下进行因此十分安全，而且非特权用户也是可以直接使用。**



设备虚拟化 - 设备透传的应用

- 设备透传的作用：
  - 透传GPU/网络设备给虚拟机
  - 虚机的GPU/网络驱动，不需要做任何修改就可以直接使用透传设备
  - 性能损耗最小
- 应用场合：云服务器透传各种设备，包括但不限于：
  - GPU设备透传
  - 网卡透传
  - 磁盘透传



CPU虚拟化 – 不同体系的比较

Privilege Level	RISC-V		ARMv8		x86	
Low	User	0	EL0	Guest App/Host App	VMX non-root mode 3	Guest App
	Supervisor	1	EL1	Guest OS	VMX non-root mode 0	Guest OS
	Hypervisor	2	EL2	Hypervisor/Host OS	VMX root mode 3	Host App
High	Machine	3	EL3	Secure monitor	VMX root mode 0	Hypervisor/Host OS