KVM 内存管理机制串讲

zweiustc@gmail.com

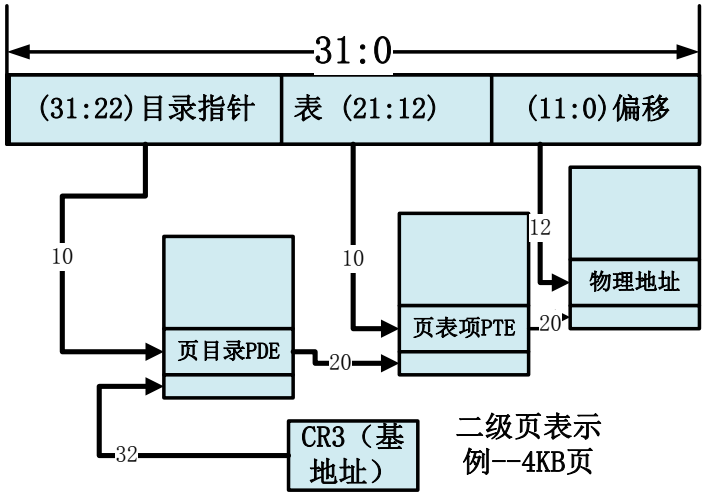
主要内容

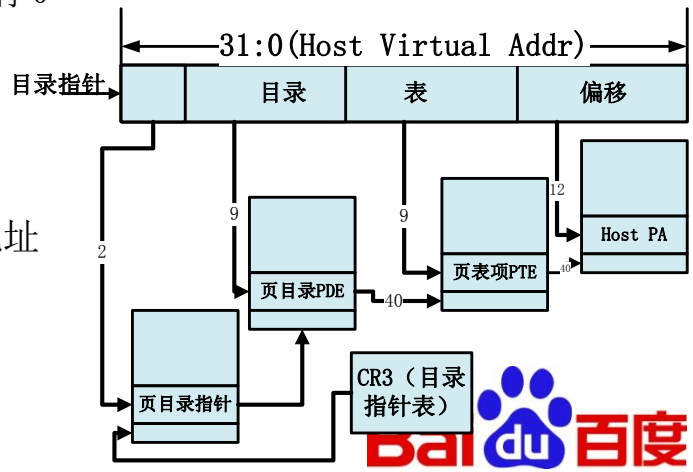
- 虚拟机地址空间
- 影子页表的实现
- EPT/NPT技术

Baidu 百度

多级页表结构

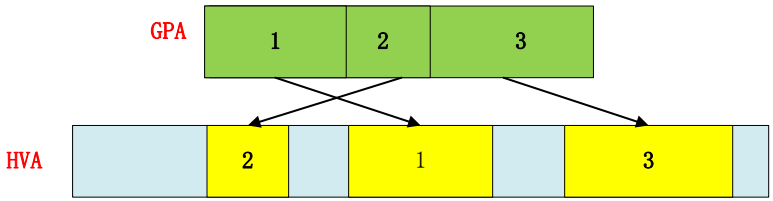
- PDE: 页表的物理地址（4B）
  - 1PD(4KB) = 1024 PDE
  - CPU用线性地址22-31索引
- PTE: 对应线性地址的PFN
  - 1PT(4KB) = 1024 PTE
  - CPU用线性地址21-12索引
- 关键字段P/D/A/RW：
  - P=0:物理页面在物理内存　P=1:不在-> 缺页处理
  - D（脏）:第一次写CPU置1，软件清0
  - A（访问）：指示是否访问过，CPU初次读写置 1 ，软件清 0
  - G :全局有效（重载CR,TLB有效）
- CR3寄存器：
  - PDBR 存页目录物理地址
  - 进程具有相同线性空间，CR3载入基地址映射不同物理地址
- TLB：
  - 缓存VFN到PFN的转换
  - VFN+offset ->PFN+offset

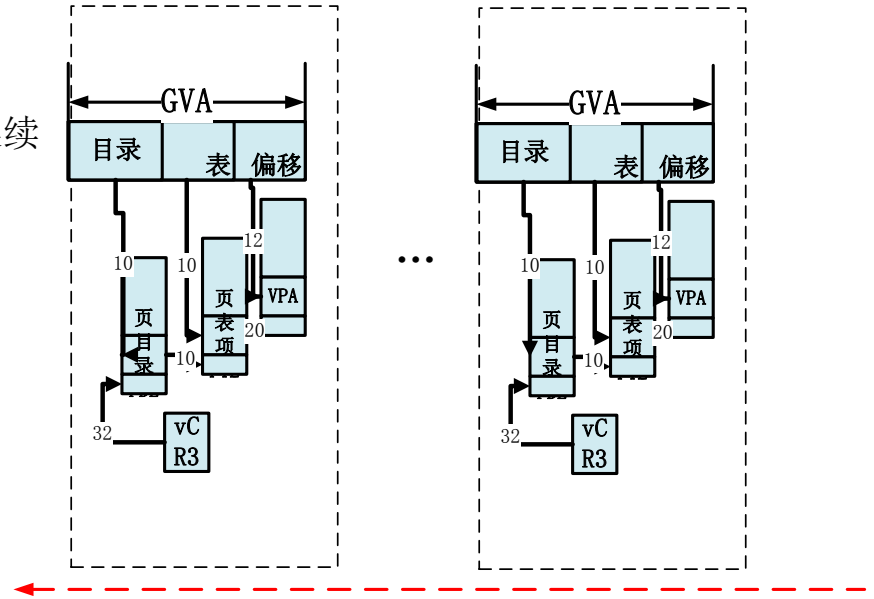




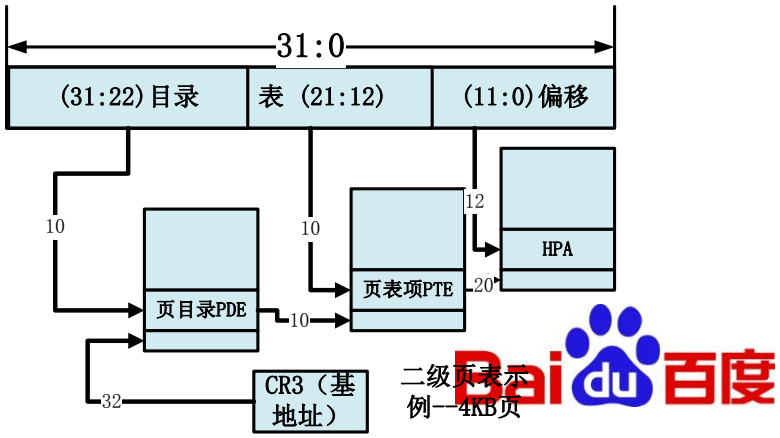
VM的地址空间

- 全虚拟化
  - Guest OS不变：物理地址从0开始/按页连续
  - vms虚拟地址空间一致，独占
- 客户机物理地址空间
  - 从0开始、连续
  - host虚拟地址(HVA)空间在guest的映射
  - 映射关系：由kvm_memory_slot标记





- Vmm管理内存
  - host物理内存管理/vm占用内存

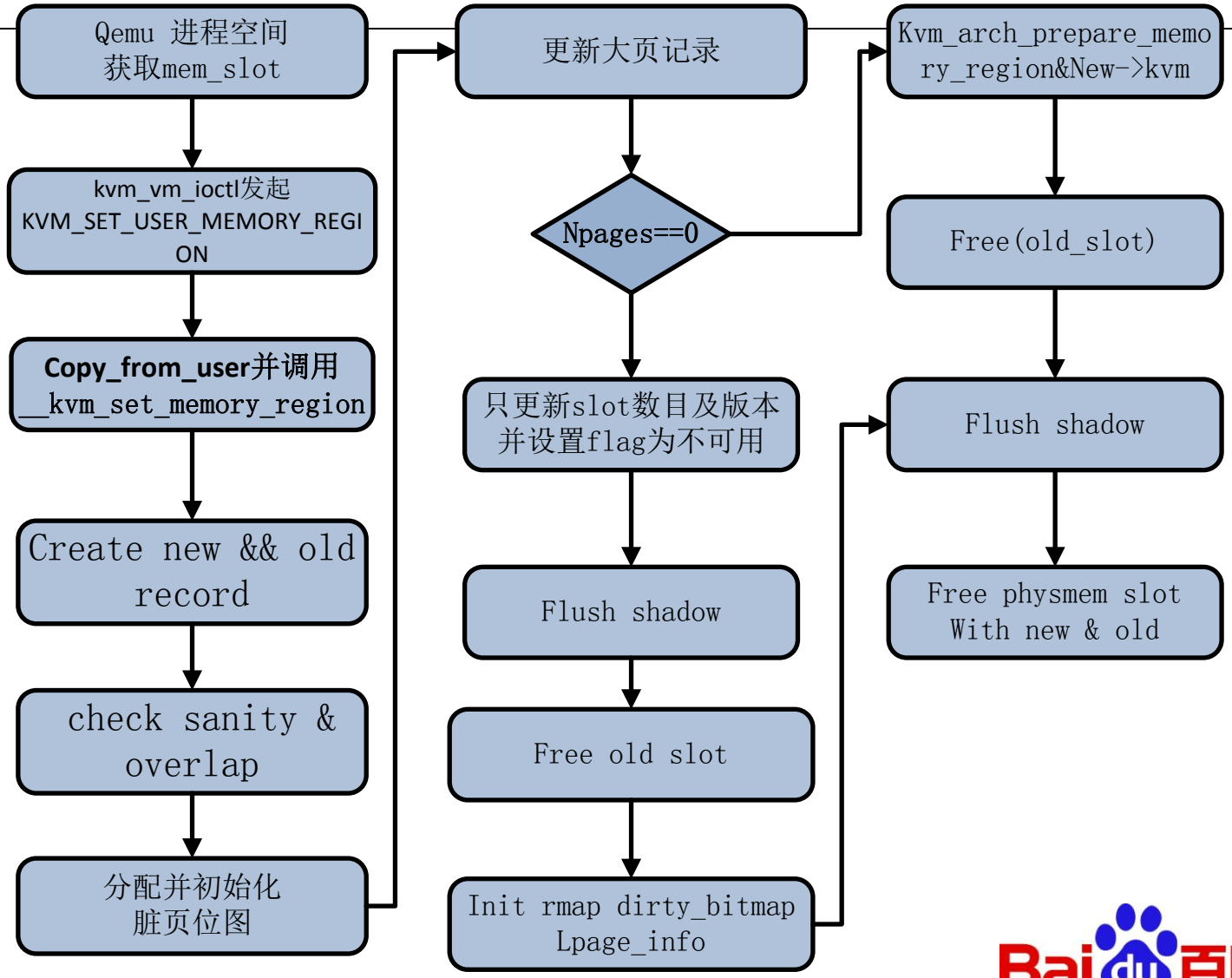


VM的地址空间

- KVM获取vm内存分布
  - qemu进程空间划分地址范围
  - KVM_SET_USER_MEMORY_REGION通知KVM有关guest内存
  - uest 内存映射userspace地址
  - rmap直译gva->hva映射
- Page fualt时分配物理页内存
  - vm_exit
  - get_user_pages内核接口，分配物理页
  - 映射hva
  - 获取gfn，vm_entry

Baidu 百度

VM的地址空间


Qemu 进程空间
获取mem_slot
kvm_vm_ioctl发起
KVM_SET_USER_MEMORY_REGI
ON
Copy_from_user并调用
__kvm_set_memory_region
Create new && old
record
check sanity &
overlap
分配并初始化
脏页位图
更新大页记录
Npages==0
只更新slot数目及版本
并设置flag为不可用
Flush shadow
Free old slot
Init rmap dirty_bitmap
Lpage_info
Kvm_arch_prepare_memo
ry_region&New->kvm
Free(old_slot)
Flush shadow
Free physmem slot
With new & old

VM的地址空间

- 反向映射的完成由qemu发起

kvm_arch_vm_ioctl:KVM_SET_MEMORY_REGION

参见：kvm_arch_prepare_memory_region（）的实现

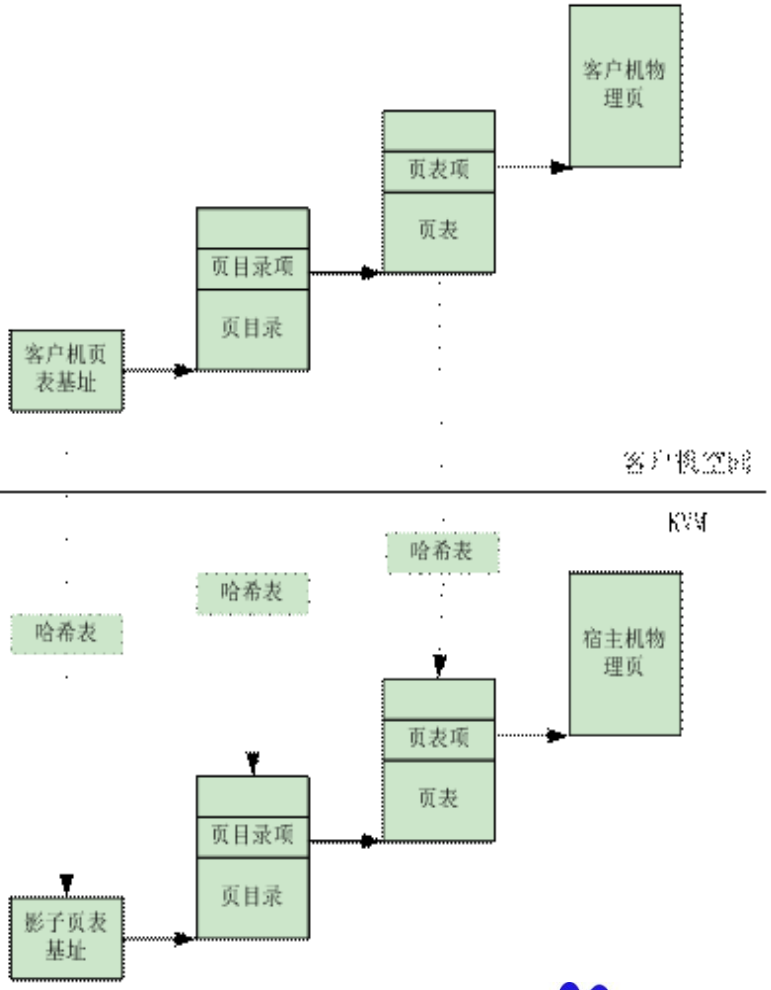
```
/*To keep backward compatibility with older userspace,
 *x86 needs to hanlde !user_alloc case.
 */
if (!user_alloc) {
    if (npages && !old.rmap) {
        unsigned long userspace_addr;

        down_write(&current->mm->mmap_sem);
        userspace_addr = do_mmap(NULL, 0,
                     npages * PAGE_SIZE,
                     PROT_READ | PROT_WRITE,
                     map_flags,
                     0);
        up_write(&current->mm->mmap_sem);

        if (IS_ERR((void *)userspace_addr))
            return PTR_ERR((void *)userspace_addr);

        memslot->userspace_addr = userspace_addr;
    }
}
```

主要内容

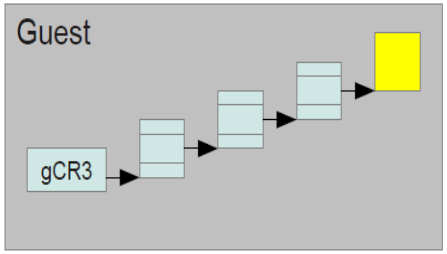
- 虚拟机地址空间
- 影子页表的实现
- EPT/NPT技术

影子页表

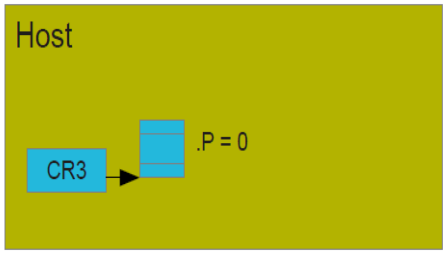
- 虚拟机的地址转换
  - 三级地址转换**GVA -> GPA -> HVA -> HPA**
  - 硬件MMU完成最后一级
- KVM方案
  - 软件层面：影子页表
  - 硬件层面：EPT/NPT
  - **bool tdp_enabled 标识**
- 影子页表
  - 功能：GVA->HPA映射
  - 客户机页表与影子页表一一对应：对应进程
  - 动态建立过程



影子页表



- Soft MMU
  - 影子页表初始化
  - 逐步建立过程
  - 与页表同步



- 影子页表的初始化
  - 在vcpu 的arch init中调用init_kvm_mmu
  - tdp_enabled参数指示使用影子页表 or EPT/NPT
  - 建立page table

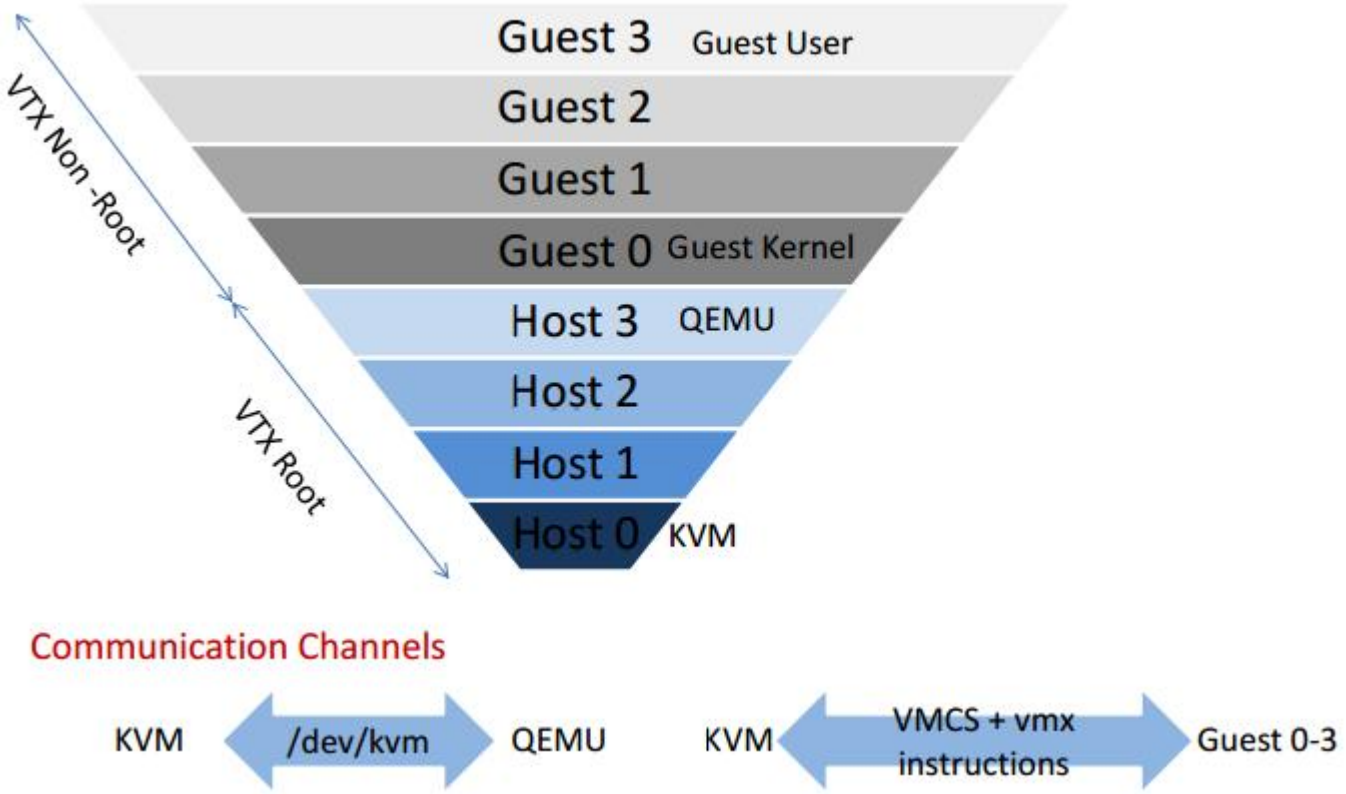
Baidu 百度

影子页表同步

- VMM拦截guest缺页异常
  - GUEST中VPA不存在，控制权返回guest OS
  - HPA存在，shadow page不一致
  - host将对应的内存页换出到硬盘上，spt捕捉并更新
- 缺页异常处理
  - 检查guest的Page Table访问权限,确定来源
  - 异常位于GUEST，返回guest OS
  - VMM引起：没同步或没有初始化

Baidu 百度

vm_entry与 vm_exit

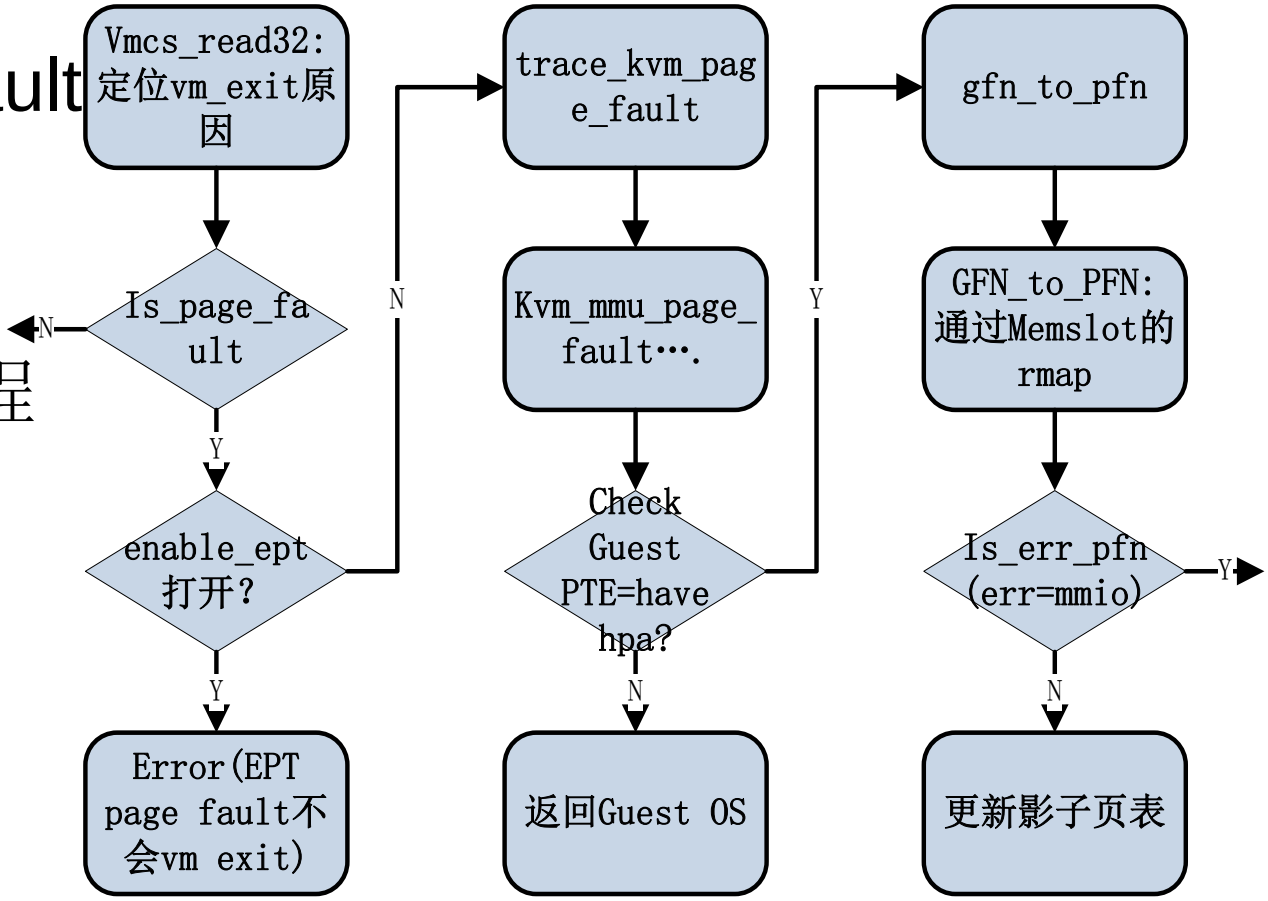


Non-Root -> Root = vm exit

Root -> Non- Root = vm entry

影子页表同步

- 拦截#Page Fault
  - PTE 增加写保护
  - 拦截CR3的加载
- Page_fualt流程



影子页表的检索

- 客户机页表与影子页表一一对应
  - 多个进程需要维护多个影子页表
- 映射由hash表完成
  - 客户机页目录 / 页表的客户机物理地址低 10 位作为键值进行索引
  - 遍历链表寻找页目录 /页表
  - 遍历失败则创立新的表项加入链表
- 进程切换
  - 客户机将页表基址（二级分页）载入CR3(特权指令)
  - KVM截获特权指令，寻找影子页表基地址输入客户机CR3
  - 客户机恢复运行（CR3指向新进程的影子页表）
- 开销
  - 多数地址访问KVM无需介入
  - 内存额外开销

Baidu 百度

虚拟内存的加载

- 虚拟机每次运行均对应vcpu_enter_guest
  - 内存管理相关：mmu_load/reload
- kvm_mmu_load
  - 预先分配重要的数据结构的内存：pte rmap …
  - 查询空闲页：超过阈值，则释放活动页
  - mmu_alloc_roots:分配root_hpa(pgd,权限相关，可参与代码)
    建立影子页表hash映射、更新空闲页面、设置页表写保护
- kvm_mmu_unload
  - 调用场景：vcpu_destroy/ free_vcpus/vcpu_enter_guest

Baidu 百度

虚拟内存的释放

- KVM_SET_NR_MMU_PAGES控制字
  - 内存活动页超过设定值，需要释放
  - 当前进程内存中释放page,循环检查
- kvm_mmu_change_mmu_pages()完成
  - 回收sp子页面，unacount并释放page
  - 回收原则：前一个活动页（LRU）
  - PTE更新 parent信息更新
- arch参量的更新
  - max_mmu_pages n_requested_mmu_pages

Baidu 百度

虚拟内存的释放

- kvm_mmu_load(reload)
  - 查询空闲页：超过阈值，则释放影子页
- kvm_mmu_unload
  - 调用场景：vcpu_destroy/ free_vcpus
  - 回收影子页表内存
  - 更新root_hpa为invalid状态
  - 区分PAE是否启用

主要内容

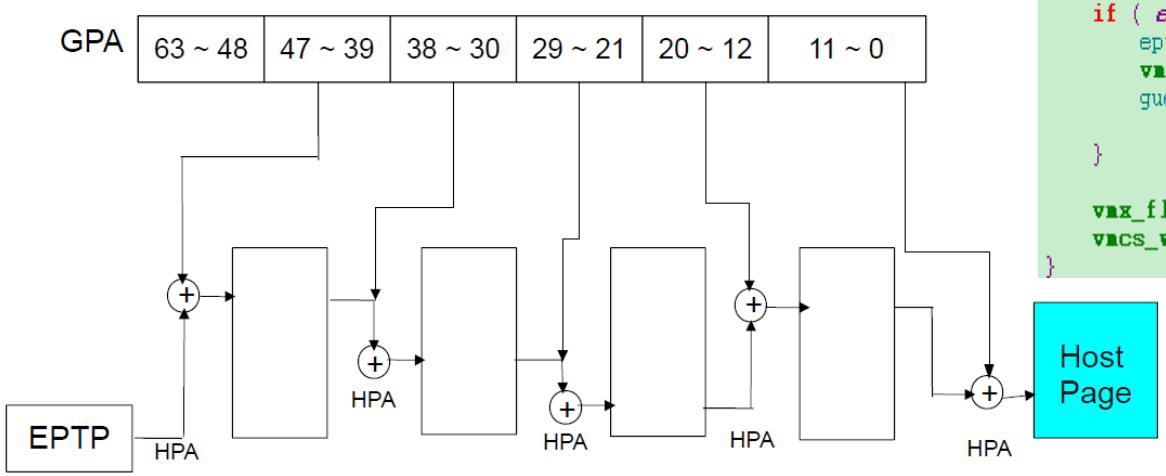
- 虚拟机地址空间
- 影子页表的实现
- EPT/NPT技术

Baidu 百度

EPT/NPT技术

- 硬件辅助内存虚拟化：二维地址翻译
- EPT页表：gpa->hpa
- 硬件自动完成：gva->gpa（客户机CR3） gpa->hpa
- EPT页表载入专门的EPT页表指针寄存器EPTP
  - — EPT PF: a.gpa->hva b.为hva分配物理页
    c.KVM更新EPT
  - — 单个VM只需一个EPT页表

EPT/NPT




```
static void vmx_set_cr3(struct kvm_vcpu *vcpu, unsigned long cr3)
{
    unsigned long guest_cr3;
    u64 eptp;

    guest_cr3 = cr3;
    if ( enable_ept ) {
        eptp = construct_eptp(cr3);
        vmcs_write64(EPT_POINTER, eptp);
        guest_cr3 = is_paging(vcpu) ? vcpu->arch.cr3 :
            vcpu->kvm->arch.ept_identity_map_addr;
    }

    vmx_flush_tlb(vcpu);
    vmcs_writel(GUEST_CR3, guest_cr3);
}
```

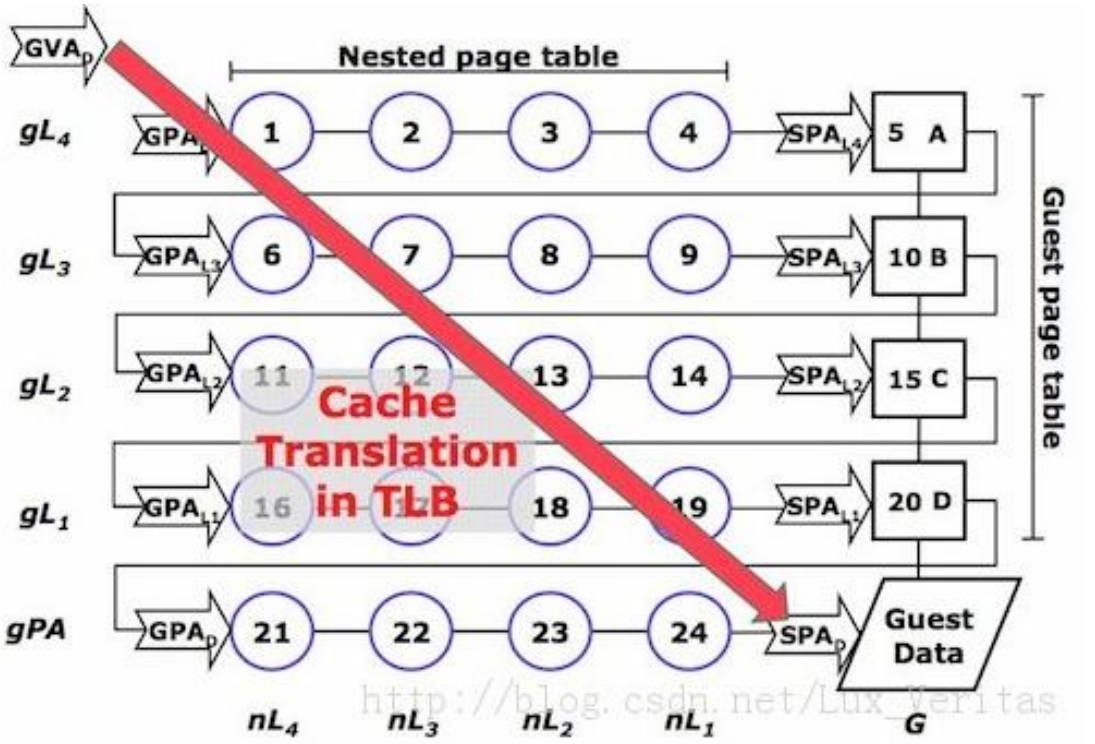
Baidu 百度

NPT 寻址举例

- 区别：

客户机每次寻找下级页表地址均要查询一次NPT

客户机N级分页 + NPT的M级分布



最差情况下的内存访问次数？

Q&A