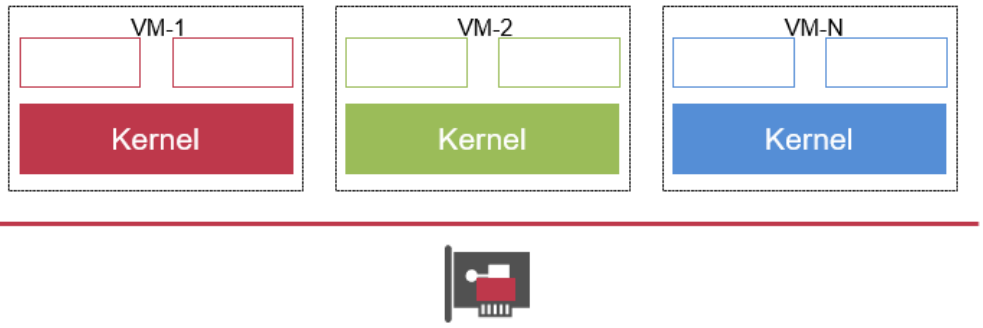
Lecture22 I/O Virtualization

1. 为什么需要I/O虚拟化

- 回顾：操作系统内核直接管理外部设备
  - PIO/MMIO
  - DMA
  - Interrupt
- 如果VM能直接管理物理设备

如果VM直接管理物理网卡

- **正确性问题：所有VM都直接访问网卡**
  - 所有VM都有相同的MAC地址、IP地址，无法正常收发网络包
- **安全性问题：恶意VM可以直接读取其他VM的数据**
  - 除了直接读取所有网络包，还可能通过DMA访问其他内存

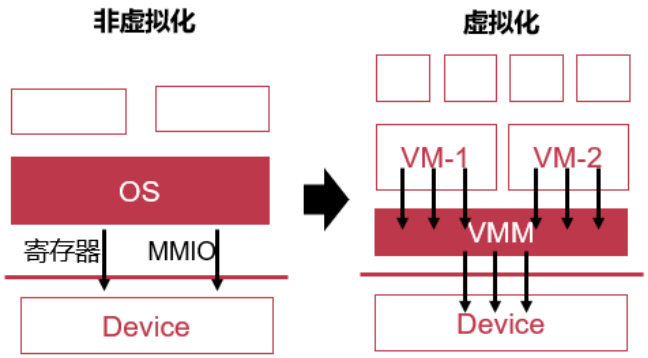


2. 怎么实现I/O虚拟化?

1. 设备模拟(Emulation)
2. 半虚拟化方式(Para-virtualization)
3. 设备直通(Pass-through)

方法1：设备模拟

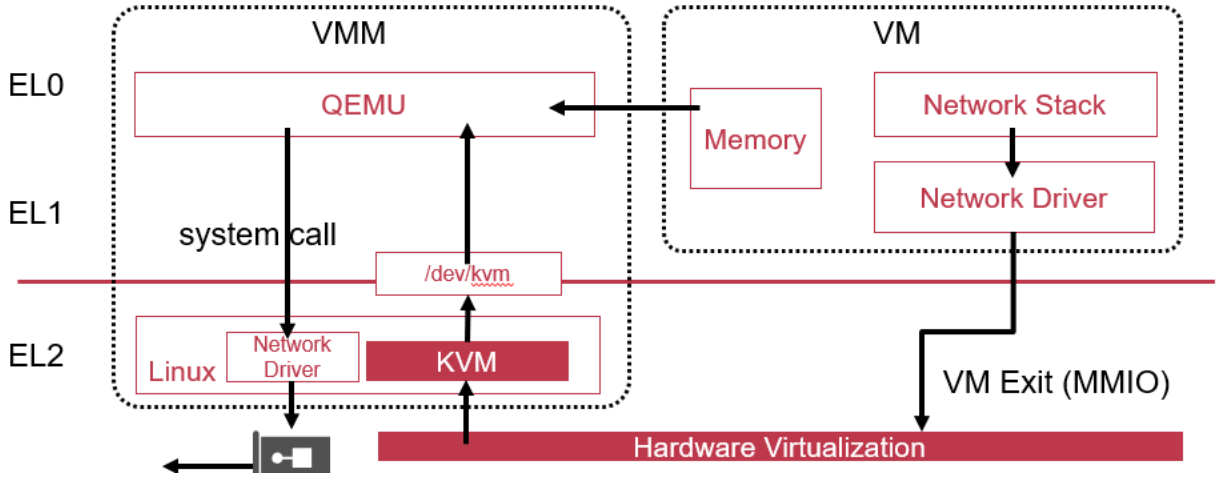
方法1：设备模拟



- **OS与设备交互的硬件接口**
  - 模拟寄存器(中断等)
  - 捕捉MMIO操作
- **硬件虚拟化的方式**
  - 硬件虚拟化捕捉PIO指令
  - MMIO对应内存在第二阶段页表中设置为invalid

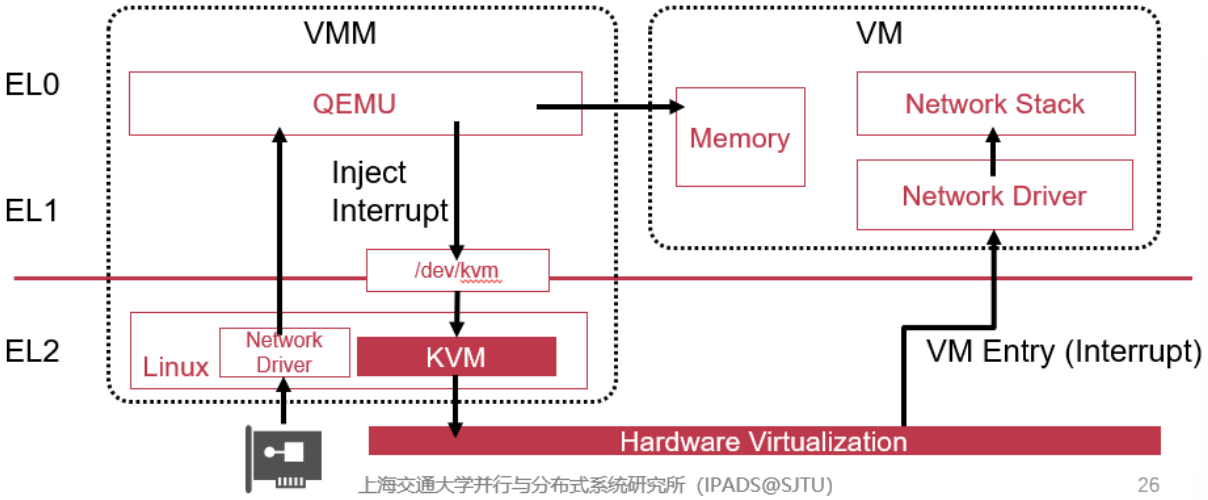
例：QEMU/KVM设备模拟1

- **以虚拟网卡举例——发包过程**



例：QEMU/KVM设备模拟2

- 以虚拟网卡举例——收包过程



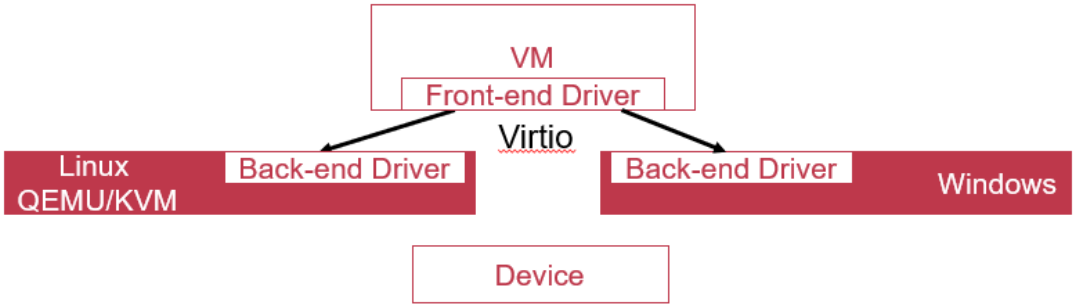
- 优点
  - 可以模拟任意设备
    - 选择流行设备，支持较为“久远”的OS
  - 允许在中间拦截(Interposition)：
    - 例如在QEMU层面检查网络内容
  - 不需要硬件修改
- 缺点
  - 性能不佳

方法2：半虚拟化方式

- 协同设计
  - 虚拟机“知道”自己运行在虚拟化环境
  - 虚拟机内运行前端(front-end)驱动
  - VMM内运行后端(back-end)驱动
- VMM主动提供Hyperacall给VM
- 通过共享内存传递指令和命令

VirtIO：Unified Para-virtualized I/O

- **标准化的半虚拟化I/O框架**
  - 通用的前端抽象
  - 标准化接口
  - 增加代码的跨平台重用



Virtqueue

- **VM和VMM之间传递I/O请求的队列**
- **3个部分**
  - Descriptor Table
    - 其中每一个descriptor描述了前后端共享的内存
    - 链表组织
  - Available Ring
    - 可用descriptor的索引，Ring Entry指向一个descriptor链表
  - Used Ring
    - 已用descriptor的索引