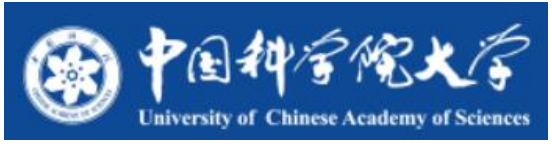
ISCAS
中国科学院软件研究所
Institute of Software,Chinese Academy of Sciences
中国科学院大学
University of Chinese Academy of Sciences

系统虚拟化

改编声明

- 本课程教学及PPT内容基于**上海交通大学并行与分布式系统研究所**发布的操作系统课程修改，原课程官网：
  - https://ipads.se.sjtu.edu.cn/courses/os/index.shtml
- 本课程修改人为**中国科学院软件研究所**，用于国科大操作系统课程教学。



计算设备集中与分散的变化

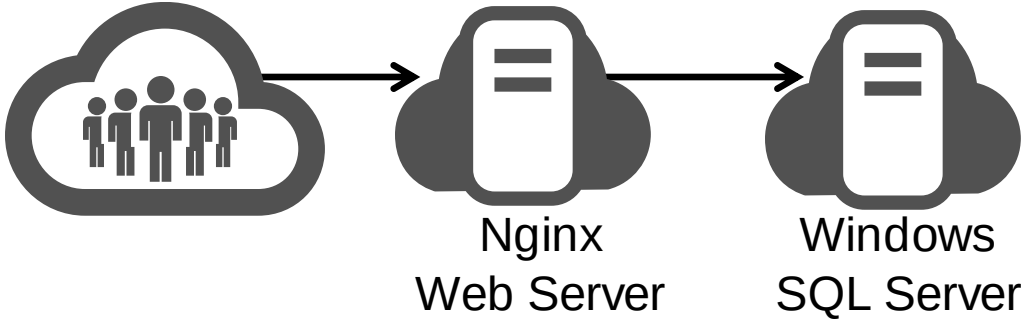
- **大型机时代**
  - 集中式计算资源，所有用户通过网络连接大型机，共享计算资源
  - 20世纪70年代，虚拟化技术已经兴起(!)
- **PC时代**
  - 摩尔定律：计算机性能提升、价格低廉化
  - Wintel联盟使个人计算机得到普及
  - 分布式计算资源，每个PC用户独占计算资源
  - 20世纪90年代，虚拟化技术沉寂
- **云时代**
  - 集中式计算资源，所有人通过网络连接，共享计算资源
  - 21世纪，虚拟化技术再次兴起

虚拟化复兴

- **云服务器代替物理服务器**
- **虚拟化复兴**
  - 以Vmware公司为代表的虚拟软件公司产品兴起
  - VMware：在虚拟化和云计算基础架构领域处于全球领先地位，提供服务 器、桌面虚拟化的解决方案；
  - Citrix：构建桌面、服务器、网络及移动端的虚拟化解决方案，借助各种 网络和云，在任何设备上无缝地交付应用、桌面、数据和服务；
  - 微软Hyper-v ：为广泛的用户提供更为熟悉以及成本效益更高的虚拟化基 础设施软件，以降低运作成本、提高硬件利用率、优化基础设施并提高

现代公司的IT部署方式：云

- **云服务器代替物理服务器**
  - 云服务器配置与物理服务器一致
  - 所有云服务器维护由服务商提供



云计算为云租户带来的优势

- 按需租赁、无需机房租赁费
- 无需雇佣物理服务器管理人员
- 可以快速低成本地升级服务器
- …

系统虚拟化是云计算的核心支撑技术

- **新引入的一个软件层**
  - 上层是操作系统（虚拟机）
  - 底层硬件与上层软件解耦
  - 上层软件可在不同硬件之间切换

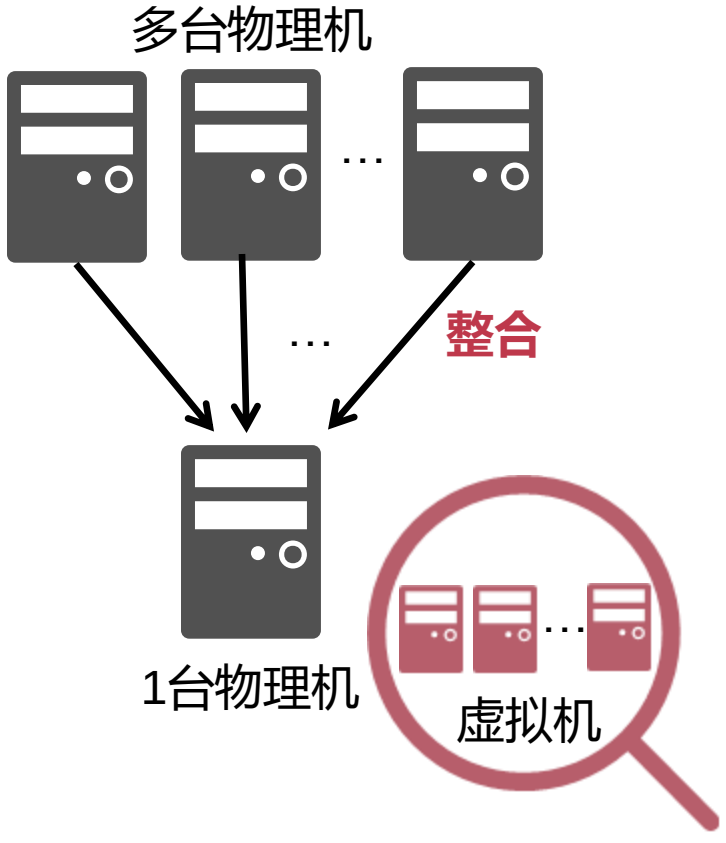
虚拟化带来的优势

- "*Any problem in computer science can be solved by another level of indirection*" **--- David Wheeler**

- **1、服务器整合：提高资源利用率**
- **2、方便程序开发**
- **3、简化服务器管理**
- **…**

虚拟化优势-1：服务器整合

- **单个物理机资源利用率低**
  - CPU利用率通常仅<20%
- **利用系统虚拟化进行资源整合**
  - 一台物理机同时运行多台虚拟机
- **显著提升物理机资源利用率**
- **显著降低云服务提供商的成本**



虚拟化优势-2：方便程序开发

- **调试操作系统**
  - 单步调试操作系统
  - 查看当前虚拟硬件的状态
    - 寄存器中的值是否正确
    - 内存映射是否正确
  - 随时修改虚拟硬件的状态
- **测试应用程序的兼容性**
  - 可以在一台物理机上同时运行在不同的操作系统
  - 测试应用程序在不同操作系统上的兼容性

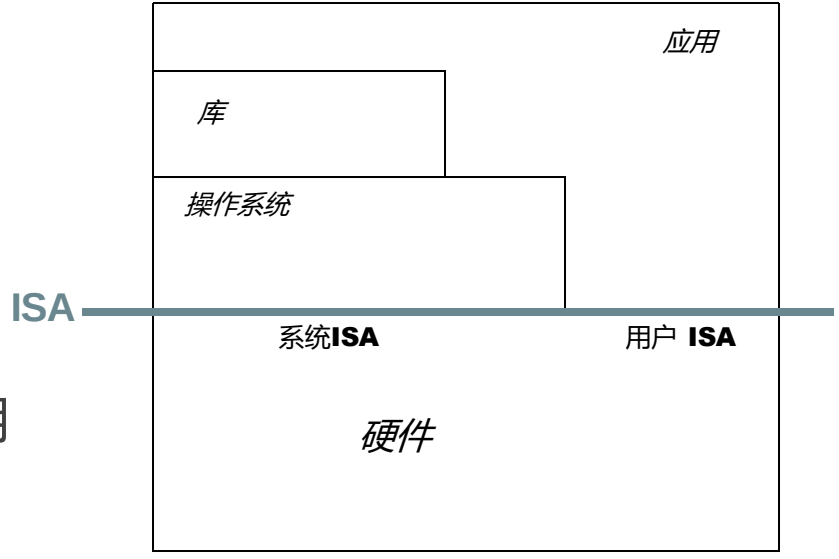
虚拟化优势-3：简化服务器管理

- **通过软件接口管理虚拟机**
  - 创建、开机、关机、销毁
  - 方便高效
- **虚拟机热迁移**
  - 方便物理机器的维护和升级

什么是系统虚拟化？

操作系统中的接口层次: ISA

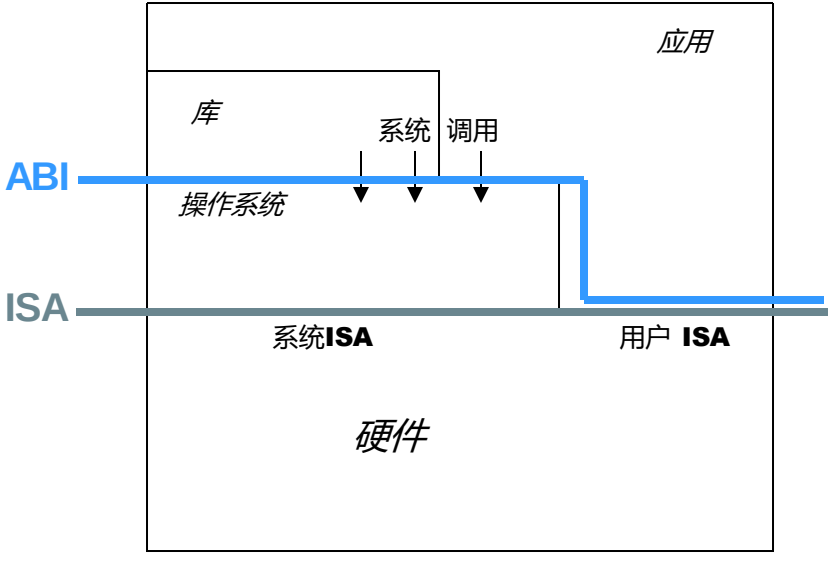
- **ISA层**
  - Instruction Set Architecture
  - 区分硬件和软件
  - 用户ISA
    - 用户态和内核态程序都可以使用
    - mov x0, sp
    - add x0, x0, #1
  - 系统ISA
    - 只有内核态程序可以使用
    - msr vbar_el1, x0



ISA – instruction set architecture

操作系统中的接口层次: ABI

- **ABI**
  - Application Binary Interface
  - 提供操作系统服务或硬件功能
  - 包含用户ISA和系统调用

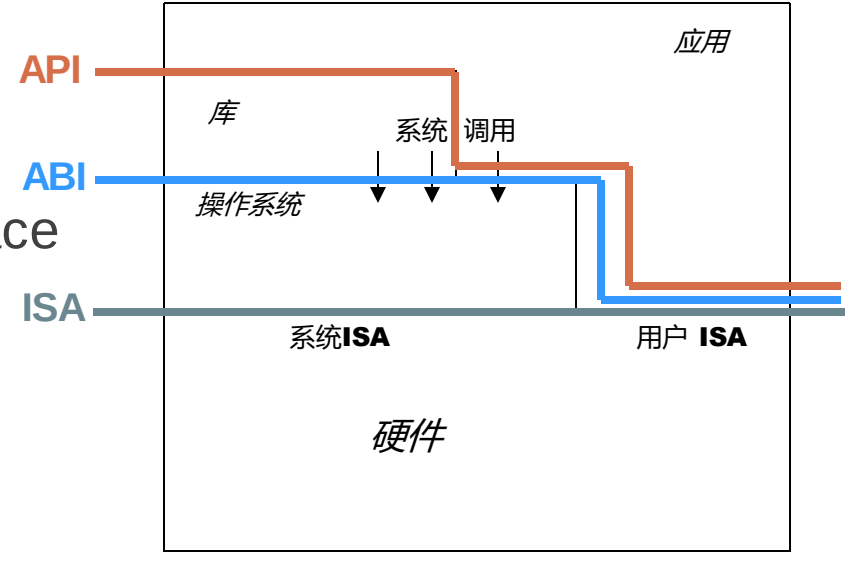


ABI – application binary interface

ISA – instruction set architecture

操作系统中的接口层次: API

- **API**
  - Application Programming Interface
  - 不同用户态库提供的接口
  - 包含库的接口和用户ISA
  - UNIX环境中的**clib**:
    - 支持UNIX/C编程语言



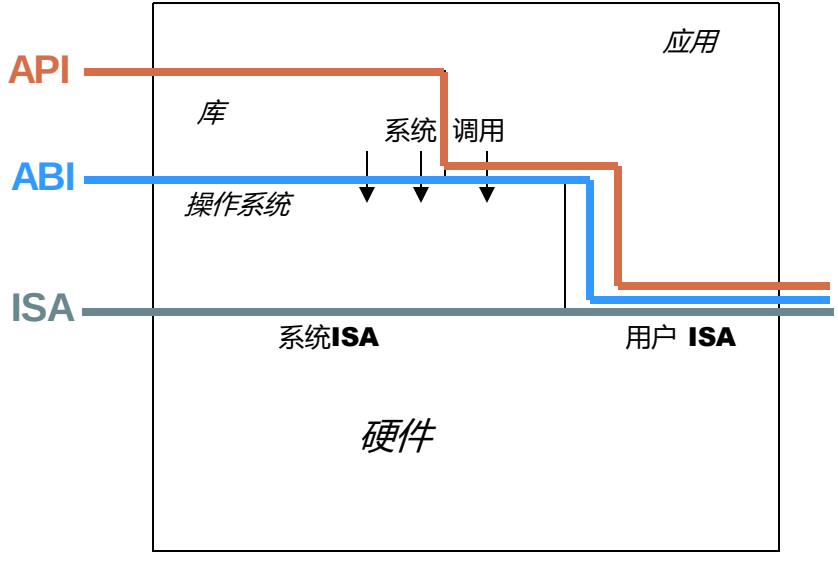
API – application programming interface

ABI – application binary interface

ISA – instruction set architecture

思考：这些程序用了哪层接口?

- Hello world
- Web game
- Dota
- Office 2016
- Windows 10
- Java applications
- PolyOS



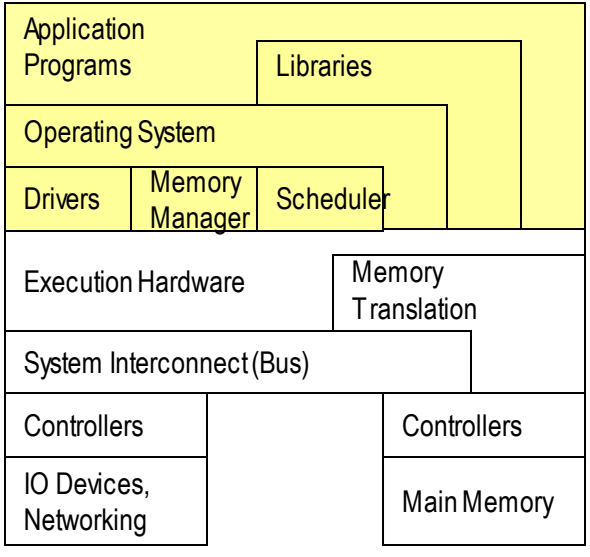
API – application programming interface

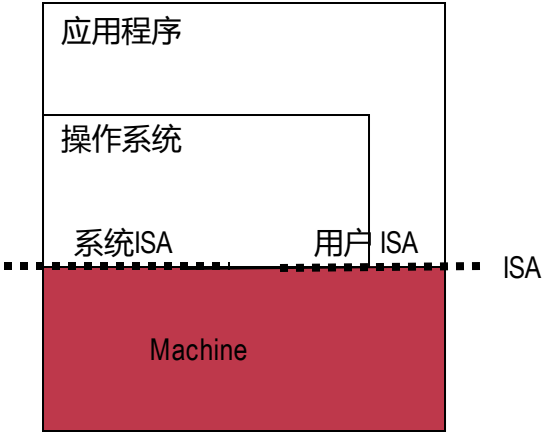
ABI – application binary interface

ISA – instruction set architecture

如何定义虚拟机?

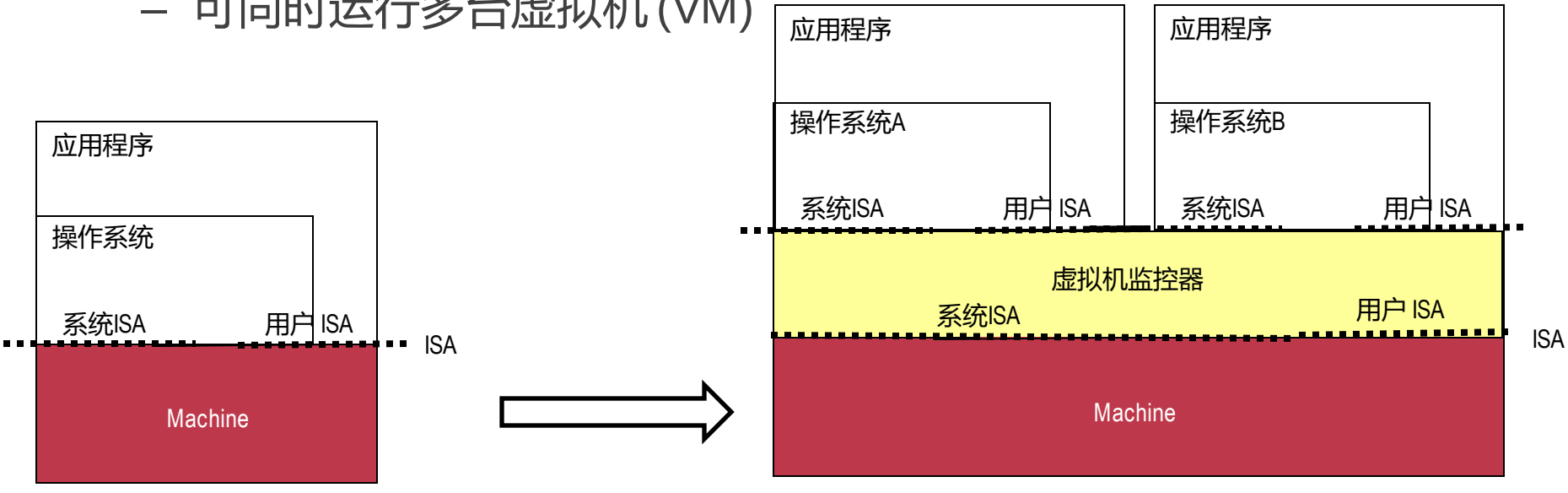
- **从操作系统角度看"Machine"**
  - ISA 提供了操作系统和Machine之间的界限





虚拟机和虚拟机监控器

- **虚拟机监控器 (VMM/Hypervisor)**
  - 向上层虚拟机暴露其所需要的ISA
  - 可同时运行多台虚拟机 (VM)



系统虚拟化的标准

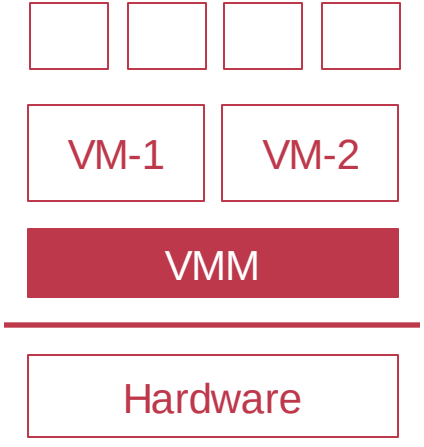
- **Popek & Goldberg, 1974 “Formal Requirements for Virtualizable Third Generation Architectures”**

- **高效系统虚拟化的三个特性**
  - 为虚拟机内程序提供与该程序原先执行的硬件完全一样的接口
  - 虚拟机只比在无虚拟化的情况下性能略差一点
  - 虚拟机监控器控制所有物理资源

虚拟机监控器的分类

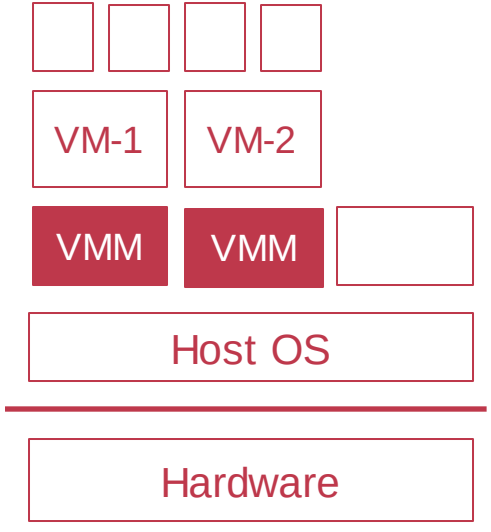
Type-1虚拟机监控器

- **VMM直接运行在硬件之上**
  - 充当操作系统的角色
  - 直接管理所有物理资源
    - 实现调度、内存管理、驱动等功能
- **性能损失较少**
- **例如Xen, VMware ESX Server**



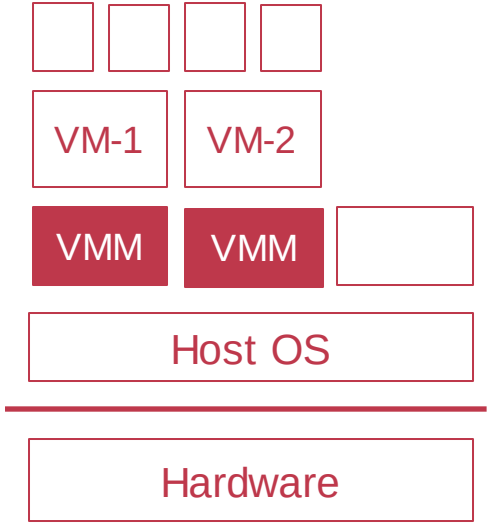
Type-2虚拟机监控器（基于Host OS）

- **VMM依托于主机操作系统**
  - 主机操作系统管理物理资源
  - 虚拟机监控器以进程/内核模块的形态运行
  - 易于实现和安装
  - 例如：QEMU/KVM
- **思考：**
  - Type-2类型有什么优势?



Type-2的优势

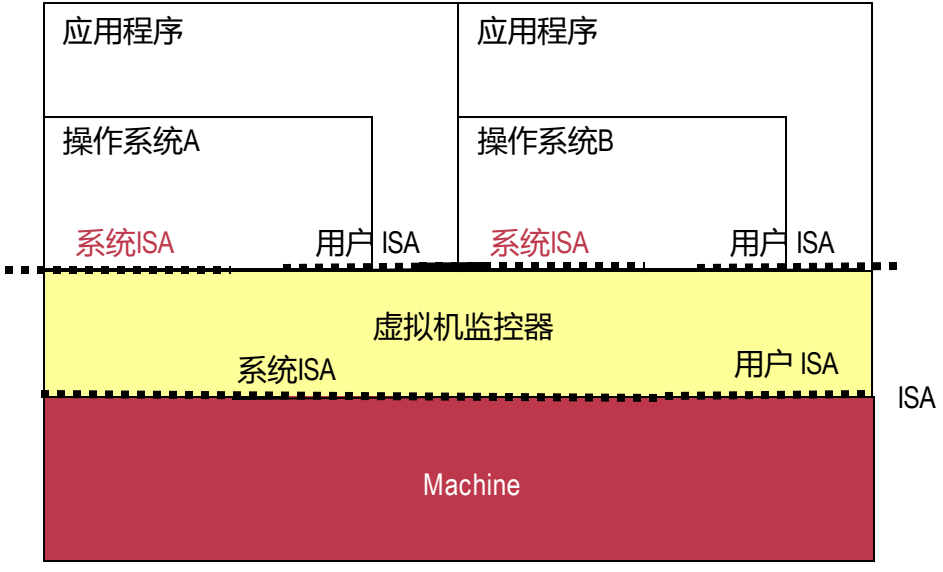
- **在已有的操作系统之上将虚拟机当做应用运行**
- **复用主机操作系统的大部分功能**
  - 文件系统
  - 驱动程序
  - 处理器调度
  - 物理内存管理



如何实现系统虚拟化？

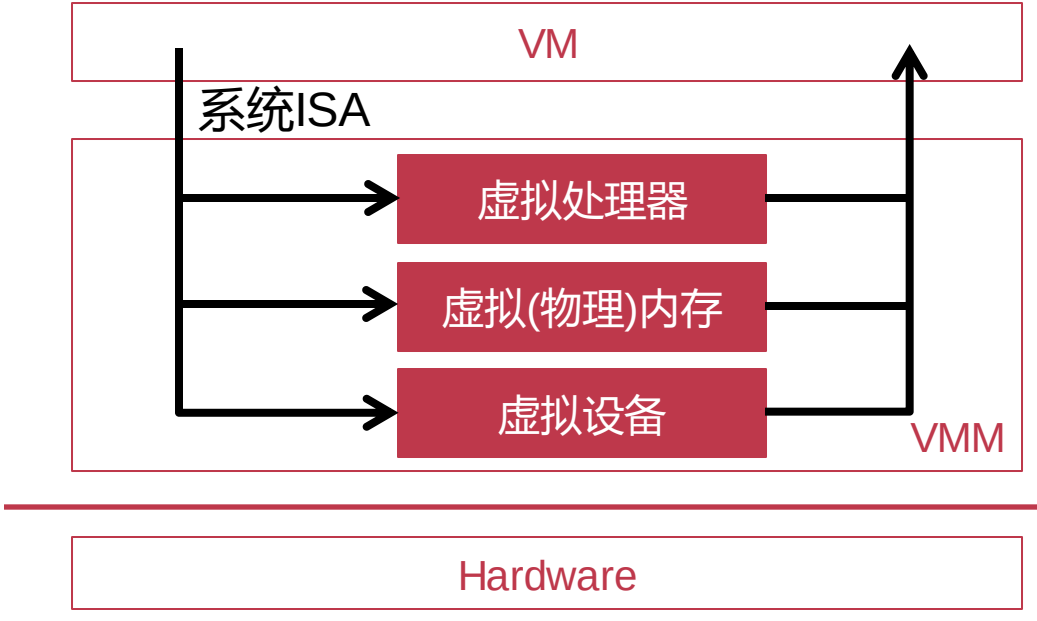
系统ISA：操作系统运行环境

- **读写敏感寄存器**
  - Satp、mhvec…
- **控制处理器行为**
  - 例如: WFI (陷入低功耗状态)
- **控制虚拟/物理内存**
  - 打开、配置、安装页表
- **控制外设**
  - DMA、中断



系统虚拟化的流程：Trap & Emulate

- **第一步（Trap）**
  - 捕捉所有系统ISA并陷入
- **第二步（Emulate）**
  - 由具体指令实现相应虚拟化
    - 控制虚拟处理器行为
    - 控制虚拟内存行为
    - 控制虚拟设备行为
- **第三步**
  - 回到虚拟机继续执行



系统虚拟化技术

- **处理器虚拟化**
  - 捕捉系统ISA
  - 控制虚拟处理器的行为
- **内存虚拟化**
  - 提供“假”物理内存的抽象
- **设备虚拟化**
  - 提供虚拟的I/O设备

回顾：RISC-V的特权级

- **0: User**
- **1: Supervisor**
  - Exception processing facility
  - Virtual memory management
- **2: Hypervisor**
  - Virtualizes physical memory
- **3: Machine**
  - Has access to all hardware features

Level	Name	Abbreviation
0	User	U
1	Supervisor	S
2	Hypervisor	H
3	Machine	M

虚拟化：一种直接的实现方法

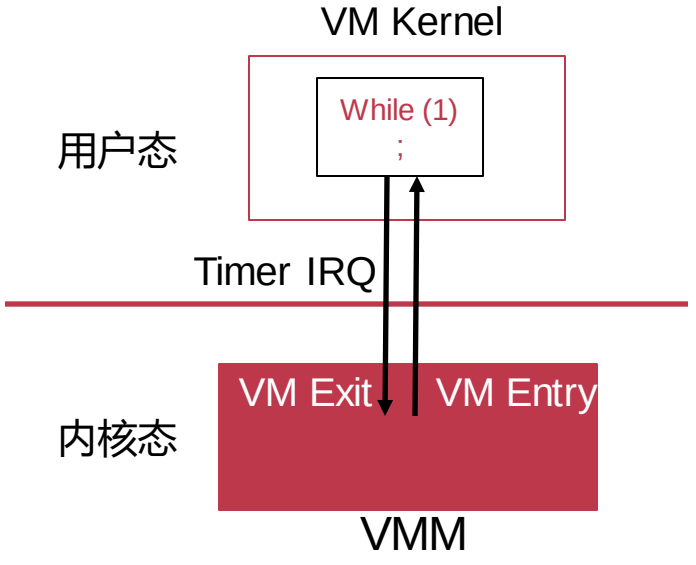
- **把虚拟机当做应用程序**
  - 将虚拟机监控器运行在hypervisor层
  - 将客户操作系统和其上的进程都运行在user mode
  - 当操作系统执行系统ISA指令时下陷
    - 写入satp
    - 执行WFI指令
    - …

虚拟化功能：迭代演进、分步理解

- 第一版：支持只有内核态的虚拟机
- 第二版：支持虚拟机内的时钟中断
- 第三版：支持虚拟机内单一用户态线程
- 第四版：支持虚拟机内多个用户态线程
- 第五版：支持多个虚拟机间的分时复用
- 第六版：支持多个物理CPU
- 第七版：支持多个虚拟CPU

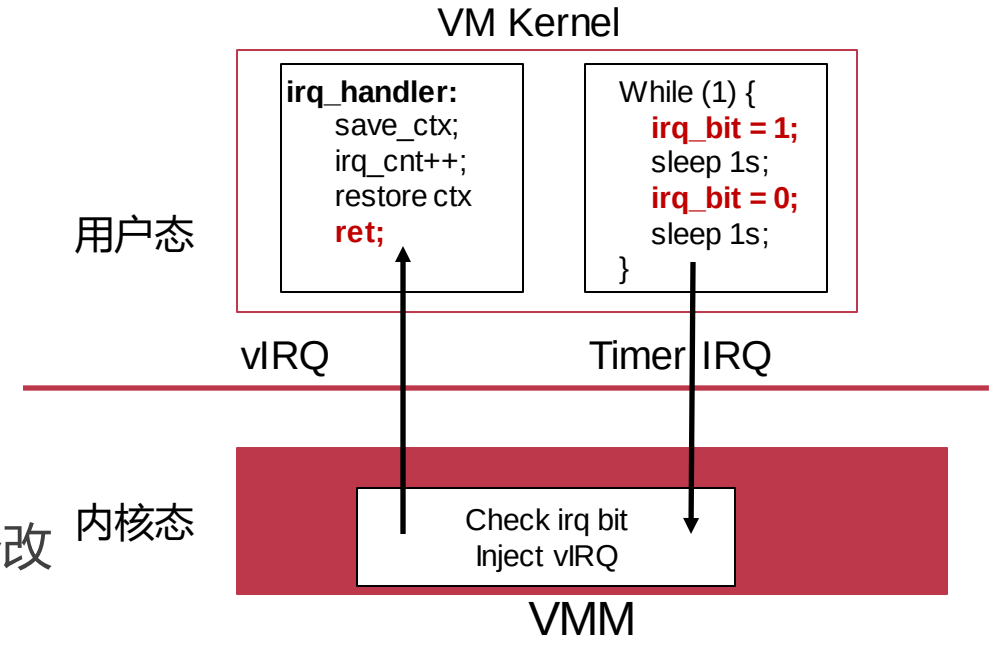
第一版：虚拟机只在内核态运行简单代码

- **VM的能力**
  - 只支持一个VM
  - 没有内核态与用户态的切换
  - 只有内核态，且仅运行用户ISA的指令（与用户态没有区别）
- **VMM的实现**
  - 处理时钟中断造成的VM Exit



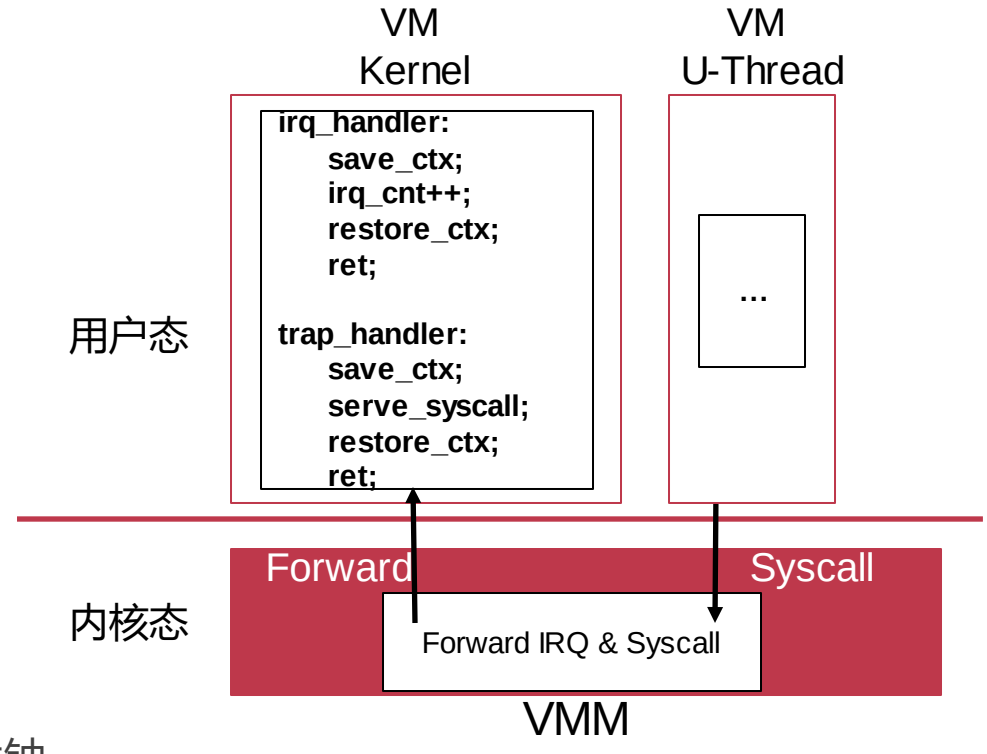
第二版：虚拟机内部支持时钟中断

- **VM的能力**
  - **设置irq_handler**
  - **开关时钟中断 (irq_bit)**
  - **运行时钟中断处理函数**
- **VMM的实现**
  - 捕捉VM对irq_handler的修改
  - 捕捉VM对irq_bit的修改
  - 根据irq_bit决定插入虚拟时钟中断vIRQ并调用irq_handler



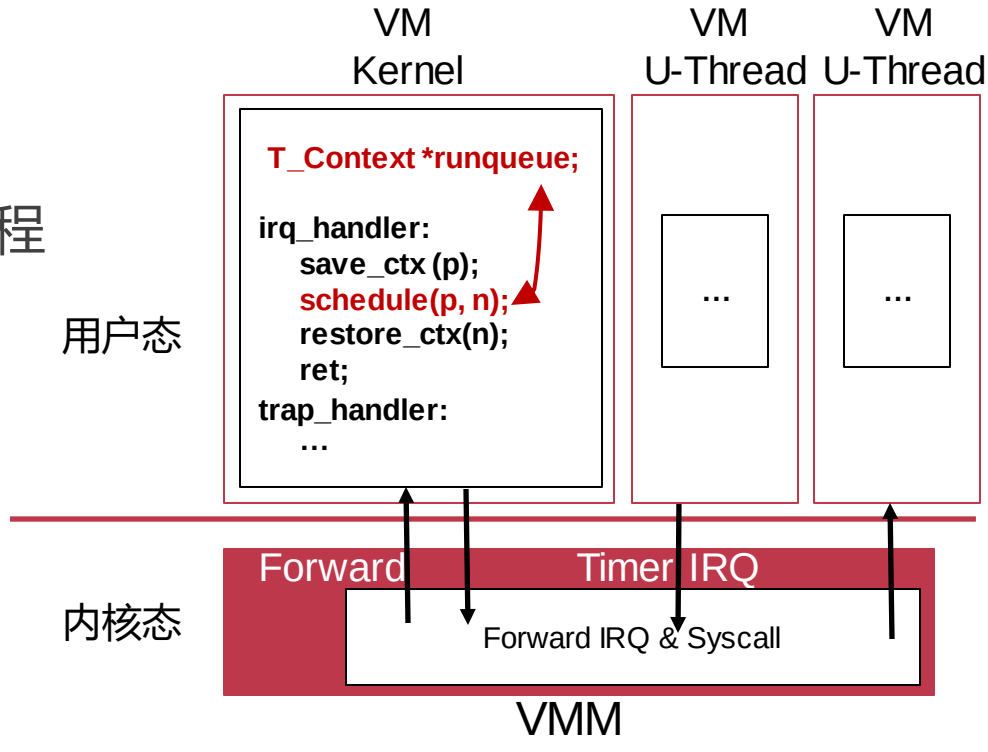
第三版：虚拟机内支持运行单一用户态线程

- **VM的能力**
  - **虚拟机包含内核态与用户态**
  - **用户态运行一个用户态线程**
    - **U-Thread**
  - **用户态线程可调用内核syscall**
  - **用户态线程可被时钟中断打断**
- **VMM的实现**
  - 捕捉并转发U-Thread系统调用syscall
  - 转发syscall至VM内核
  - 捕捉并转发U-Thread执行时的时钟中断



第四版：虚拟机内部支持多个用户态线程

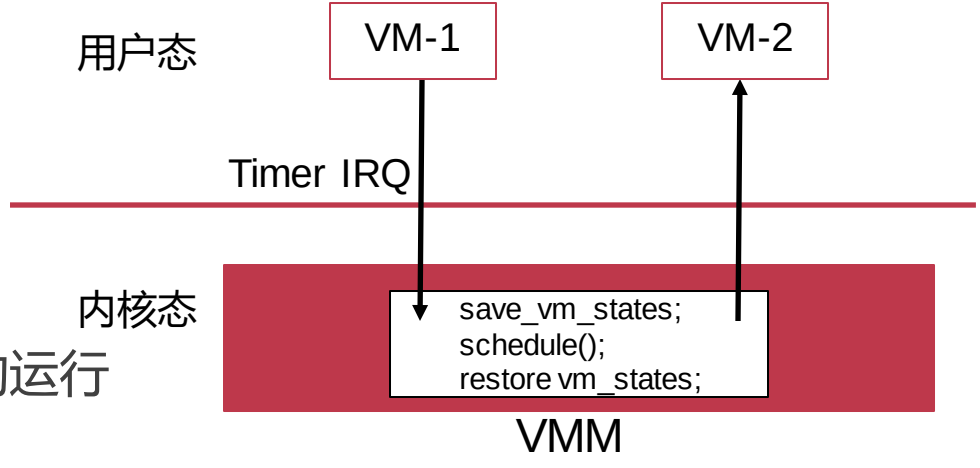
- **VM的能力**
  - 用户态运行**多个**用户态线程
  - **内核可调度用户态线程**
- **VMM的实现**
  - 与第三版相同



思考：Fork bomb是否会影响VMM?

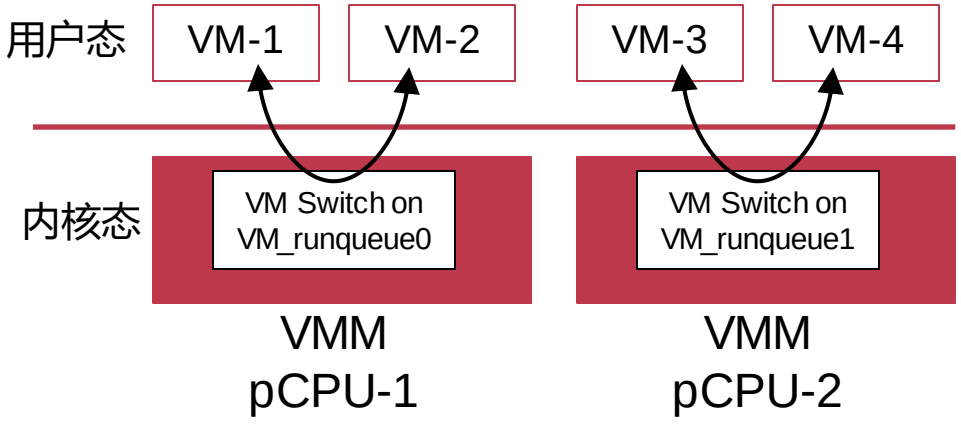
第五版：支持多个虚拟机间的分时复用

- **VM的能力**
  - **支持多个VM**
- **VMM的实现**
  - 每个VM对应一个内核线程
  - 维护VM_runqueue队列
    - 每个元素对应一个VM的运行状态
  - 由VMM实现VM间切换
    - 保存和恢复VM寄存器



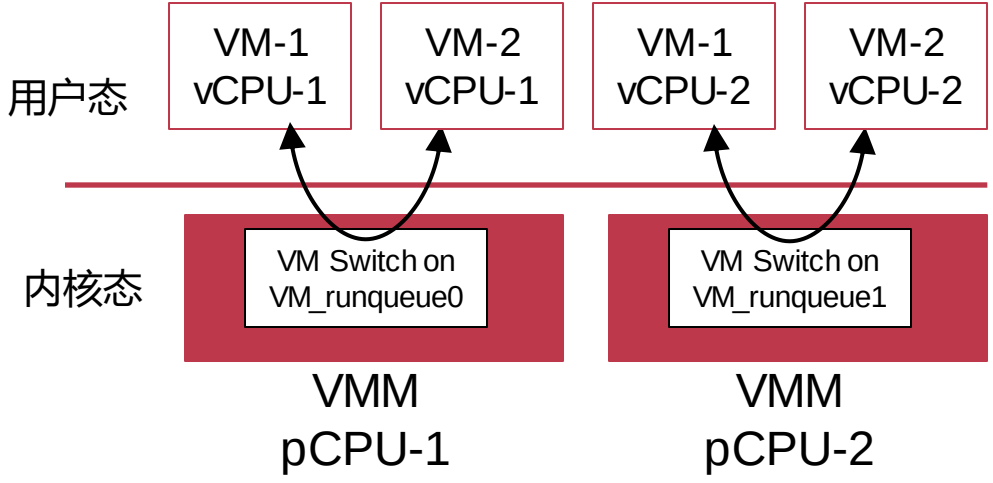
第六版：VMM支持多个物理CPU

- **VM的能力**
  - 与第五版相同
- **VMM的实现（基于第五版）**
  - 为每个pCPU维护不同的VM_runqueue



第七版：虚拟机支持多个虚拟CPU

- VM的能力（与第六版的区别）
  - 虚拟机有多个Virtual CPU (vCPU)
- VMM的实现
  - 在VM_runqueue中标记出VM和vCPU的类型



回顾：虚拟化功能的迭代演进

	用户态ISA	时钟中断	用户线程	多用户线程	多虚拟机	多物理CPU	多虚拟CPU
版本一	√						
版本二	√	√					
版本三	√	√	√				
版本四	√	√	√	√			
版本五	√	√	√	√	√		
版本六	√	√	√	√	√	√	
版本七	√	√	√	√	√	√	√