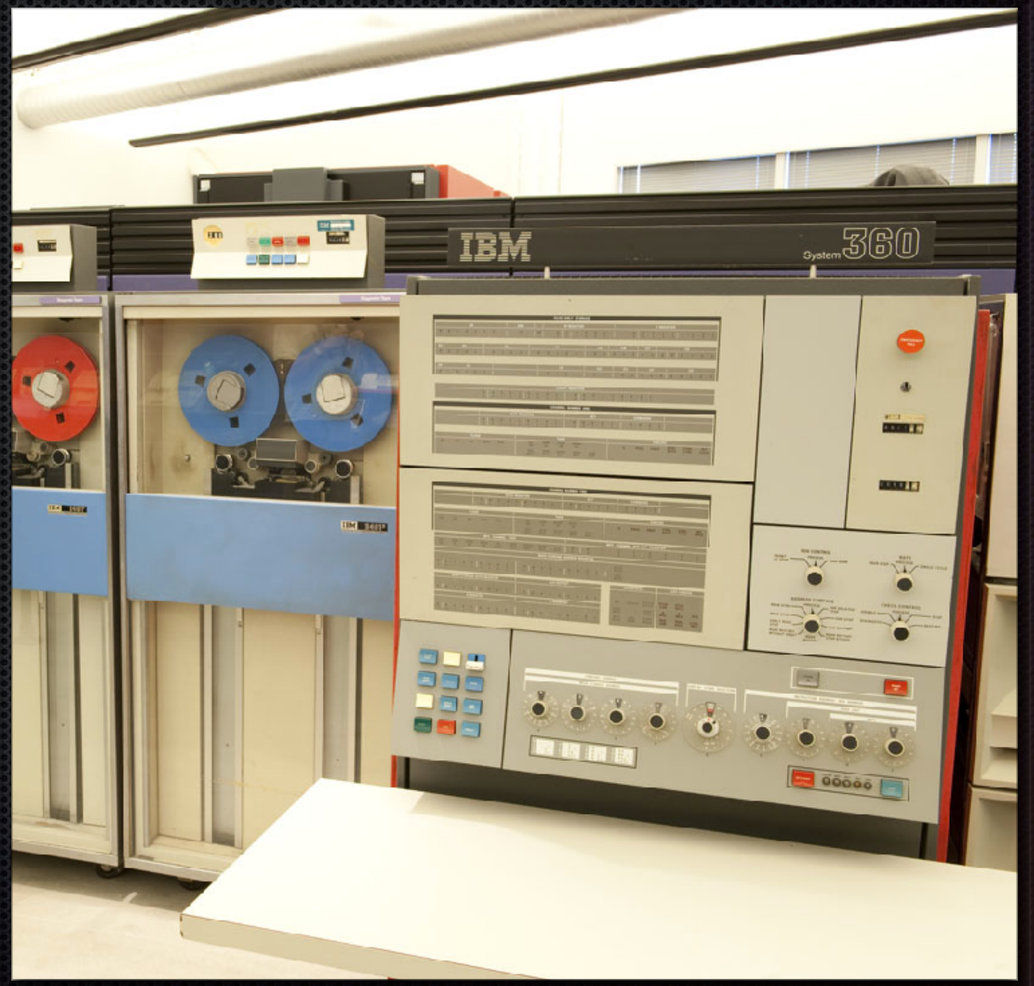
云计算核心技术分析 - 系统虚拟化

北京世纪诚链科技有限公司 赵祯龙

从IBM360说起

小沃森的决断

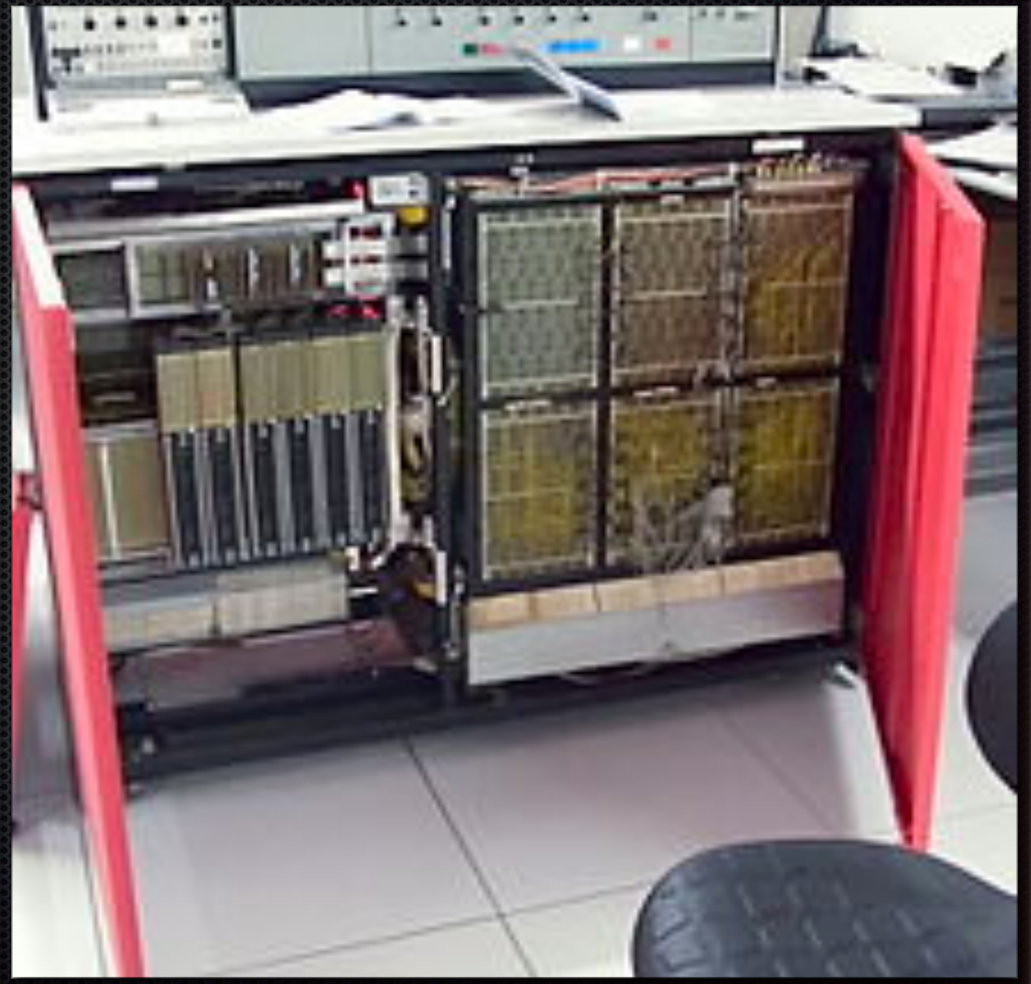
- 沃森父子的争论
- 1952年，小沃森任IBM总裁
- 投入50亿美元，研制IBM 360
  - 数目相当于马歇尔计划1/3
- 将计算机从政府军方推向民间
- 将计算机从科学计算推向商用



IBM 360

划时代的IBM 360

- 六万名新员工，五座新工厂
- 系统主架构师 G.M.Amdahi
- 项目经理 F.P. Brooks, Jr.
- 360/91 Team-leader Michael Flynn
- 20世纪最重要的三大商业成就之一
  - 波音707，福特T型车
- 贡献（远不止下面几条）
  - 提出ISA
  - 通用寄存器机器（GPR Machine）
  - Tomasulo算法
  - 软件工程



人月神话

- OS 360是一个批处理系统
- 许多用户希望使用分时系统
- IBM花了五千万美元研制TSS/360.
  - 非常庞大
  - 运行缓慢
  - 最后被弃用
- Brooks《人月神话》
- 三个系统理论
  - 《Linux/Unix设计思想》

VM 370

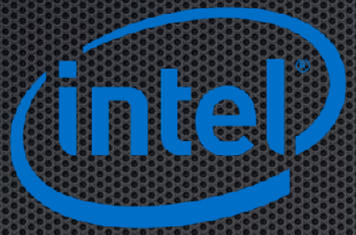
- TSS/360失败
- 麻省剑桥IBM研究中心研发出VM/370. 最初被命名为CP/CMS
- 《操作系统设计与实现(第三版)》Andrew S. Tanenabaum
  - 由于每台虚拟机都与裸机相同，所以在每台虚拟机上都可以运行一台裸机能够直接运行的任何类型的操作系统。不同的虚拟机可以运行不同的操作系统，而且实际上往往就是如此。有一些虚拟机运行OS/360的后续版本，从事着批处理或事务处理，而另一些虚拟机运行单用户、交互式系统供分时用户们使用，这个交互式系统被成为会话监控系统（Conversational Monitor System，CMS）。
  - 当一个CMS程序执行系统调用时，该调用被陷入到其虚拟机操作系统上，而不是VM/370上，似乎它运行在实际的机器上而不是在虚拟机上，CMS然后发出普通的硬件I/O指令由VM/370陷入，然后，作为对实际硬件模拟的一部分，VM/370完成指令。通过对多道程序设计功能和提供扩展机器两者的完全分离，每个部分都变得非常简单、非常灵活且容易维护。

发展趋势

- 在大型机和小型机的时代，机器成本高，虚拟化的需求也高
  - 虚拟化技术的发展比较早
- 随着个人PC的出现，分时共享的需求慢慢减弱
  - 虚拟化技术一段时间发展缓慢
- 服务器聚集后，资源利用率不高，有租户需求，应用轻量化
  - 虚拟化技术重新得到重视，再次发展

群星璀璨

vmware


intel


Xen


RISC-V


ARM


IBM


amazon
web services


KVM


redhat


Microsoft


docker

发展趋势

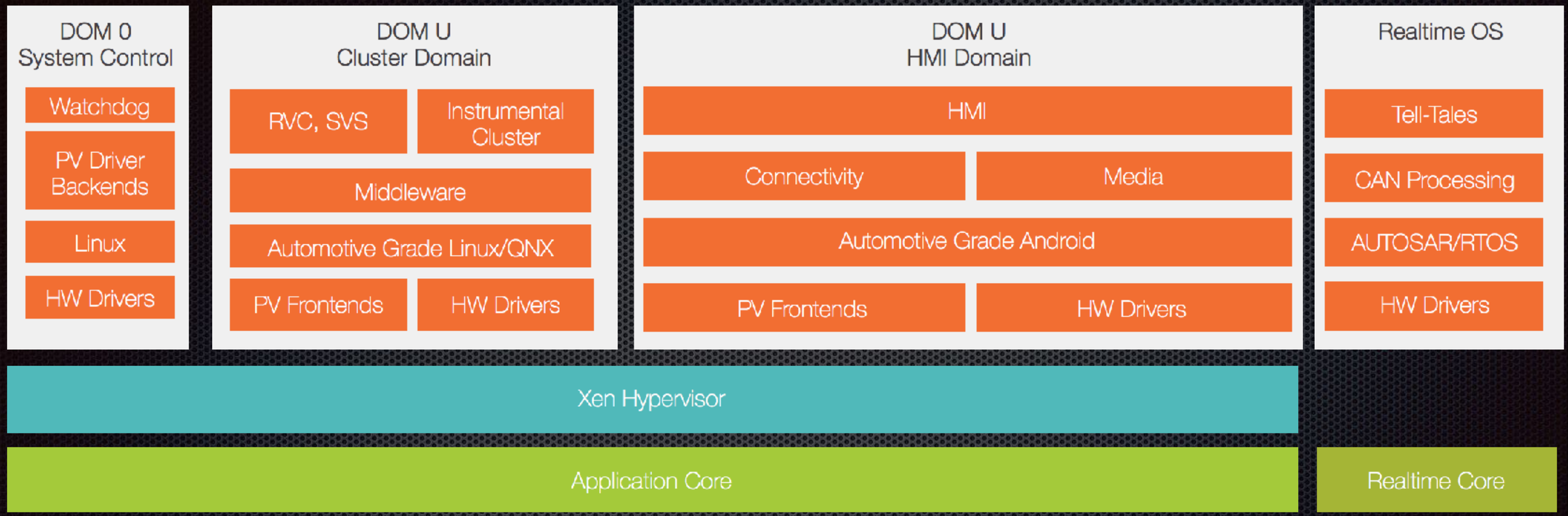
- 系统虚拟化技术地基已形成
  - 更广泛支持各类设备，提高性能是两个主题
- 嵌入式虚拟化 - 智能驾驶领域热度渐起，改进安全性和实时性是重要方向
- 服务器虚拟化 - 逐步走向软硬件协同设计，支撑更多工业级应用场景
- 容器虚拟化 - 利用现有硬件虚拟化特性增强容器安全隔离性

“Ultimate mobile device” requirements

- Stability & Reliability
- Boot Time
- Security

Xen Embedded

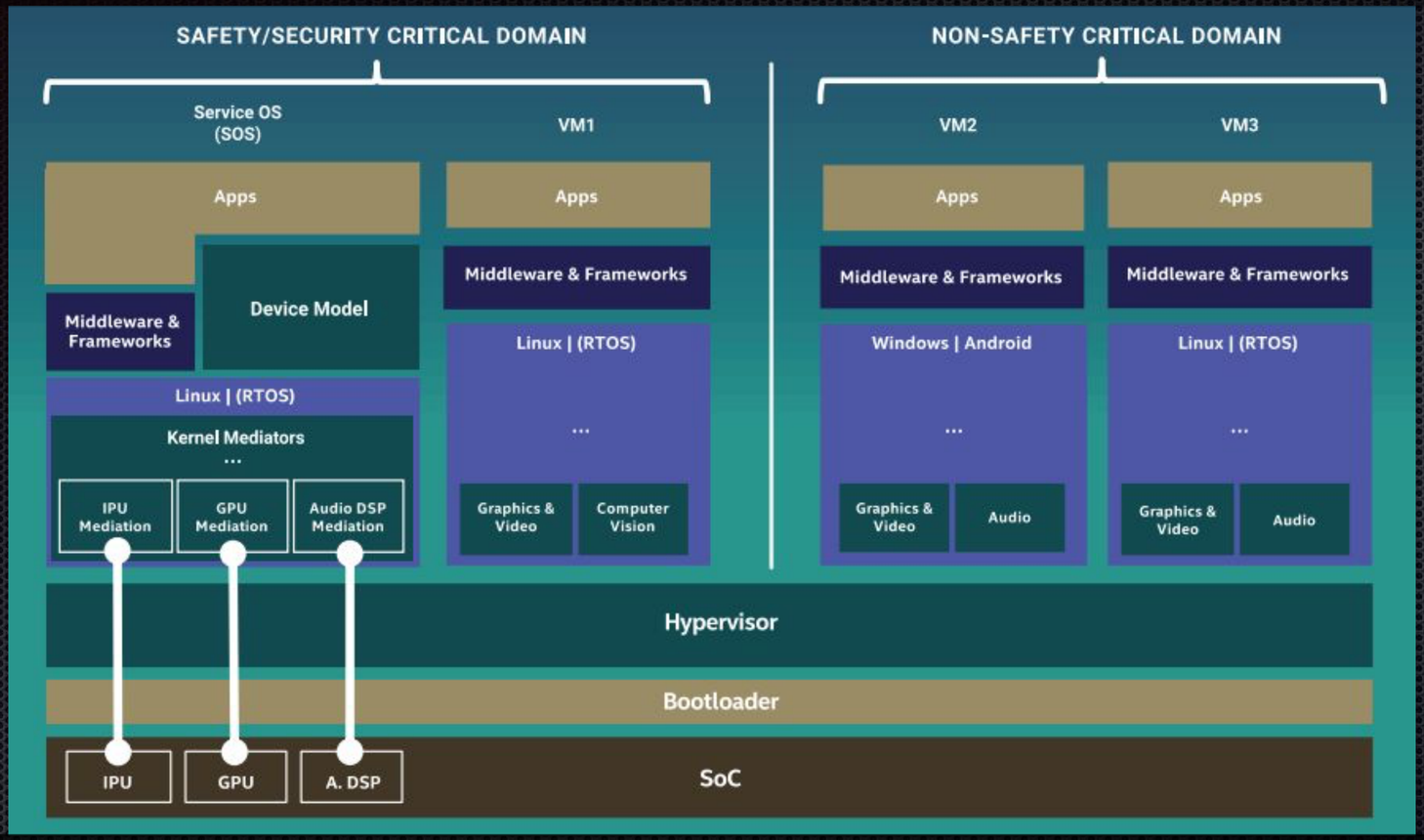
- Experimental PV ARM support on Nvidia made by Samsung
- ARM HW virtualization support from Xen 4.3
  - Interrupts mapping to DomU (for driver domains)
  - IOMEM mapping to DomU (for driver domains)
  - MMU SPT protection
  - PV drivers: HID, Audio, Framebuffer, etc.
  - Better DT support



Nautilus Platform Architecture

Based on Xen Embedded

GlobalLogic®



ACRN Overview

ACRN is a Big Little Hypervisor for IoT Development

ACRN Features

Small Footprint

- Optimized for resource constrained devices
- Few Lines of Code (LOC) of hypervisor: Approx. 25K vs. 156K LOC for datacenter-centric hypervisors.

Real Time

- Low latency
- Enables faster boot time
- Improves overall responsiveness with hardware communication

Built for Embedded IoT

- Virtualization beyond CPU, I/O, Networking, etc.
- Virtualization of Embedded IoT dev functions, ie: Graphics, Imaging, Audio, etc.
- Rich set of I/O mediators to share devices across multiple VMs

Adaptability

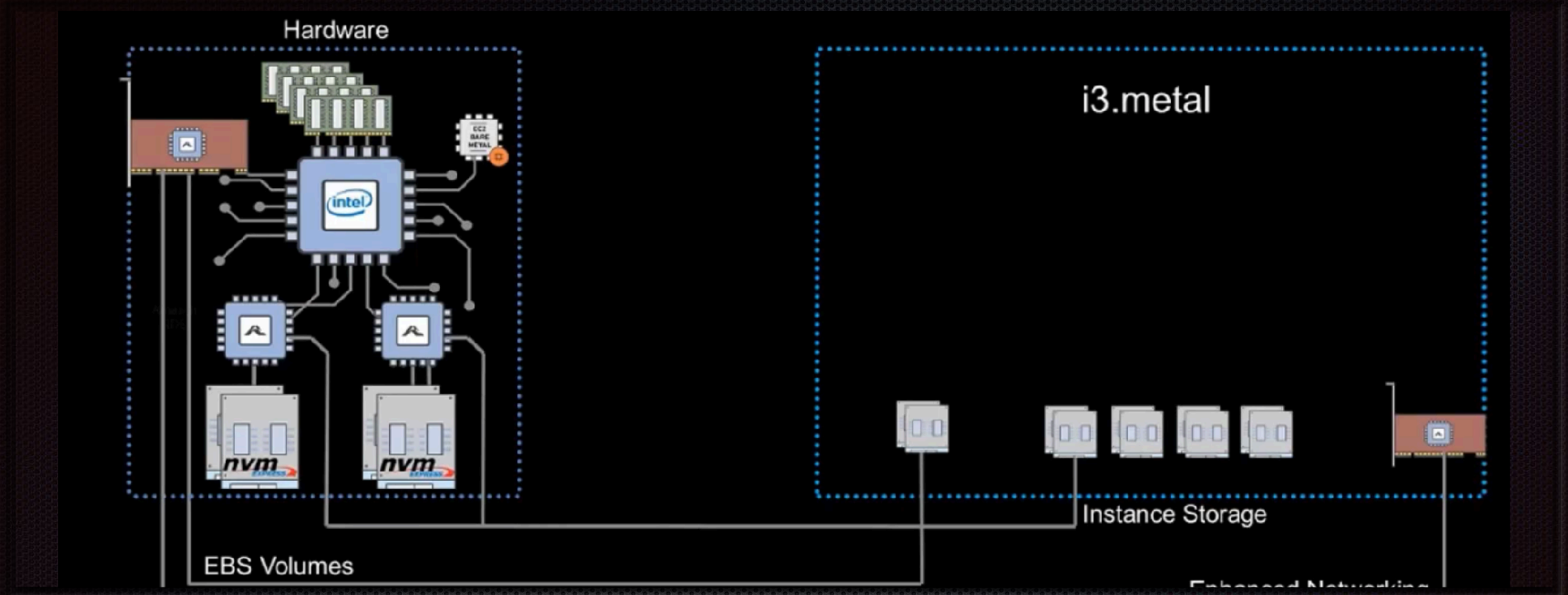
- Multi-OS support for guest operating systems like Linux and Android
- Applicable across many use cases

Open Source

- Scalable support
- Significant R&D and development cost savings
- Code transparency
- Collaborative SW development with industry leaders
- Permissive BSD licensing

Safety Criticality

- Safety critical workloads have priority
- Isolation of safety critical workloads
- Project is built with safety critical workload considerations in mind

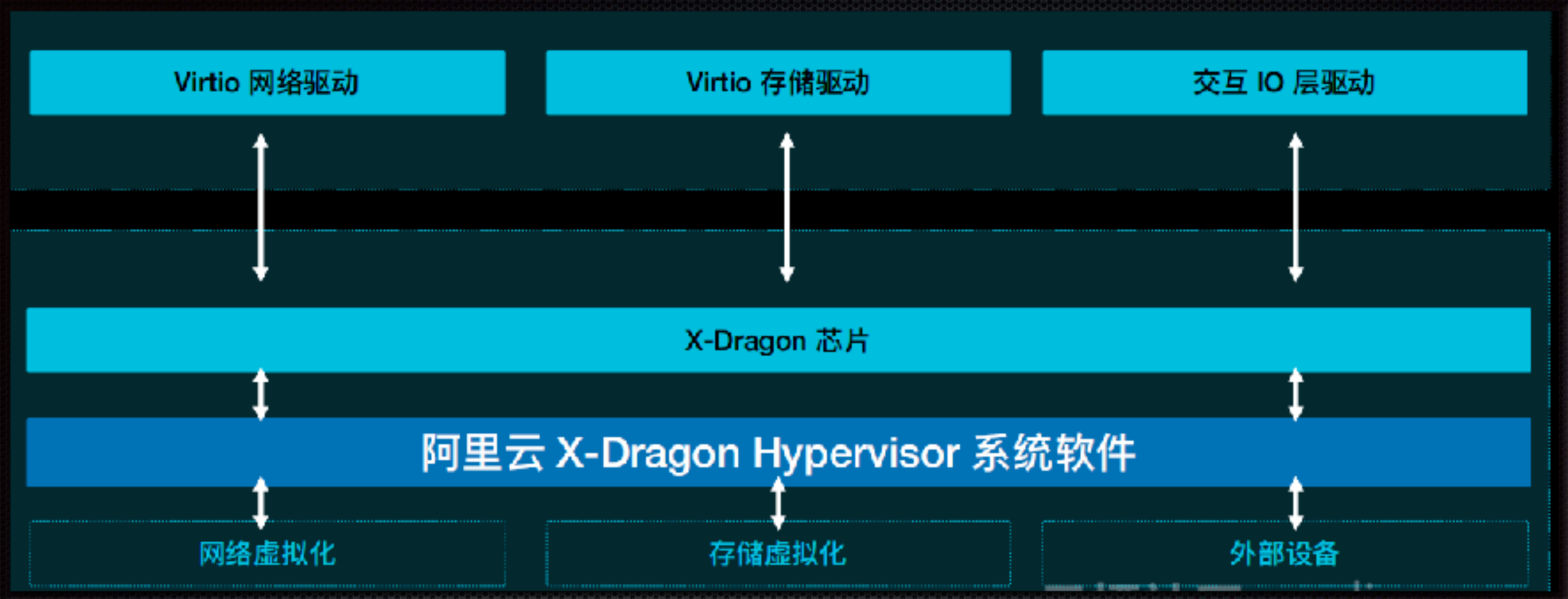


EC2 Bare Metal Instance

Amazon EC2 C5/C5d 和 M5/M5d 实例基于 Nitro 系统而构建，该系统是一系列由 AWS 构建的硬件和软件组件，可实现高性能、高可用性、高安全性和裸机功能，从而消除虚拟化开销。

Amazon Nitro System

- Nitro Lightweight hypervisor
- Nitro Card
  - Storage / Networking / Management / Monitoring / Security
- Nitro Security Chip
  - Integrated into motherboard



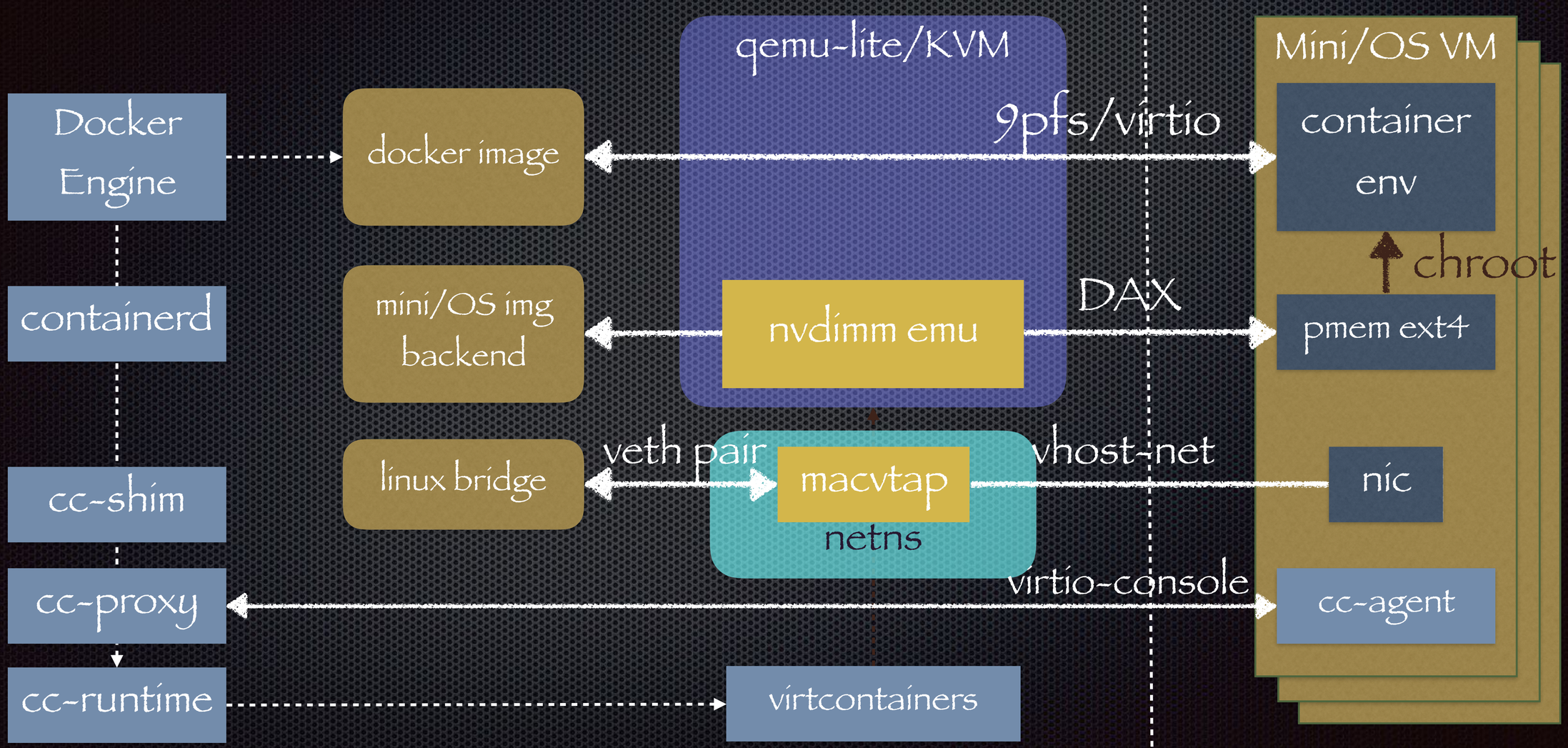
阿里云 X-Dragon裸金属服务器

弹性裸金属服务器（ECS Bare Metal Instance）是一种弹性可水平伸缩的高性能计算服务，计算性能与传统物理机无差别，具有安全物理隔离的特点，分钟级的交付周期

软硬件协同设计

芯片组和device model模拟的硬件化

Clear Container 3.0 Arch



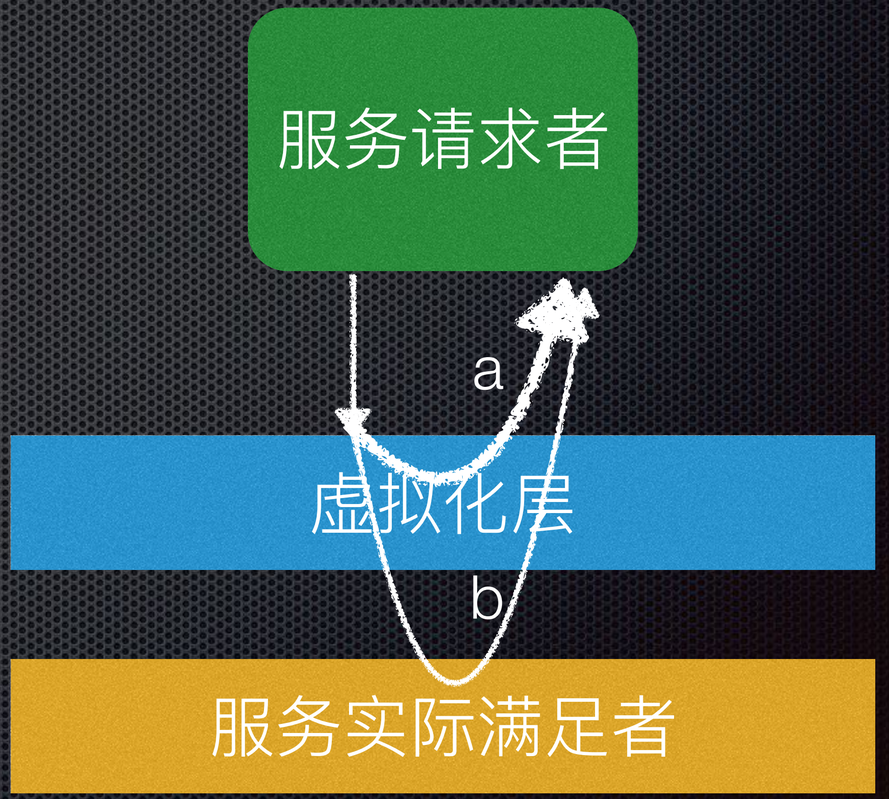
虚拟化概览

虚拟化是什么

- 虚拟化是资源的逻辑表示，将资源从一种形式重新表示成另一种形式
  - 不受物理资源的限制
  - 将一份资源抽象成多份
  - 将多份资源抽象成一份

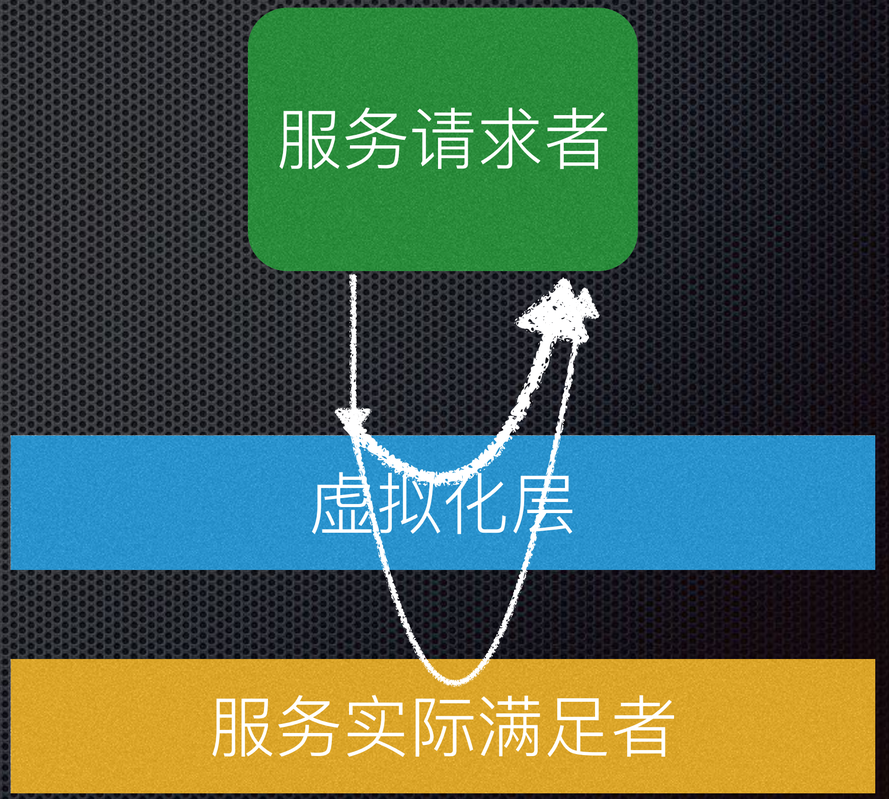
虚拟化是什么

- 请求被“虚拟化层”所模拟
- 请求需“服务实际满足者”完成
- 服务实际满足者可以将请求
  - 落在实际的物理资源上
  - 也可落在另外一层虚拟资源上



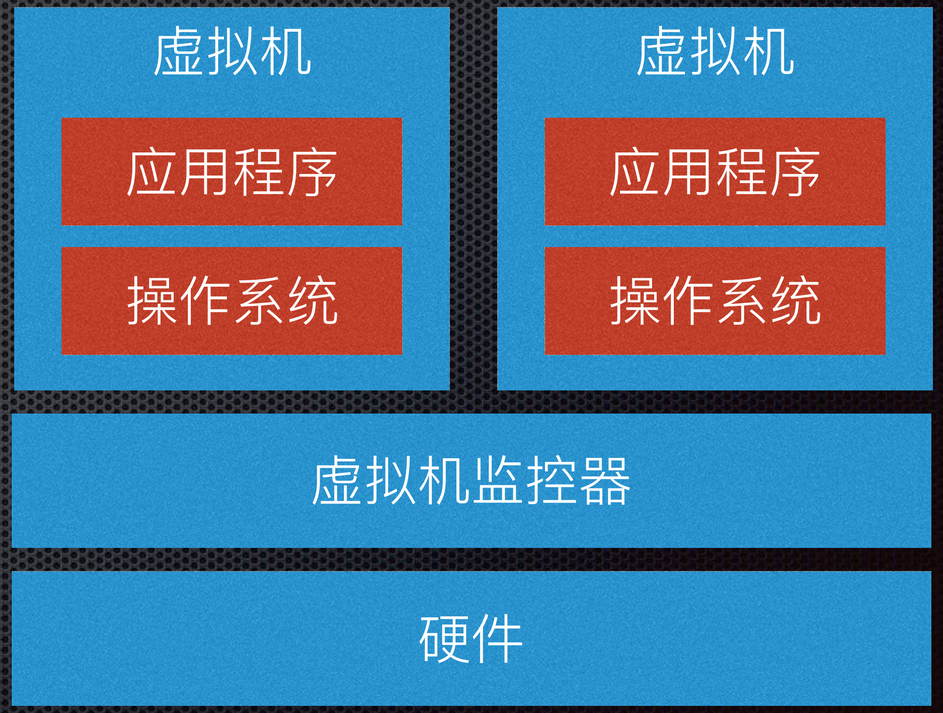
虚拟化是什么

- CPU地址映射
  - 服务请求者：内存访问
  - 虚拟化层：CPU内存映射
  - 实际服务满足者：物理内存
- 系统虚拟化
  - 服务请求者：Guest
  - 虚拟化层：VMM
  - 实际服务满足者：Host



系统虚拟化

- 系统虚拟化是虚拟化技术的一种
  - 抽象的粒度是整个计算机
  - 模拟出环境称为虚拟机



虚拟化的优势

虚拟机抽象				
封装	多实例	虚拟机间隔离	硬件无关	特权功能
虚拟机克隆 虚拟机快照 虚拟机挂起与恢复	优化资源调度	故障隔离	虚拟机迁移 兼容旧系统	时间记录与回放 入侵检测与防护
灾难恢复 便利软件测试和调试	服务器整合与节能	硬件维护、负载均衡		

虚拟化的分类

- 按照实现方法
- 按照资源类型
- 按照实现层次

按照实现方法

- 全虚拟化
  - 二进制扫描和动态翻译
  - 硬件辅助虚拟化
- 半虚拟化

按照资源类型

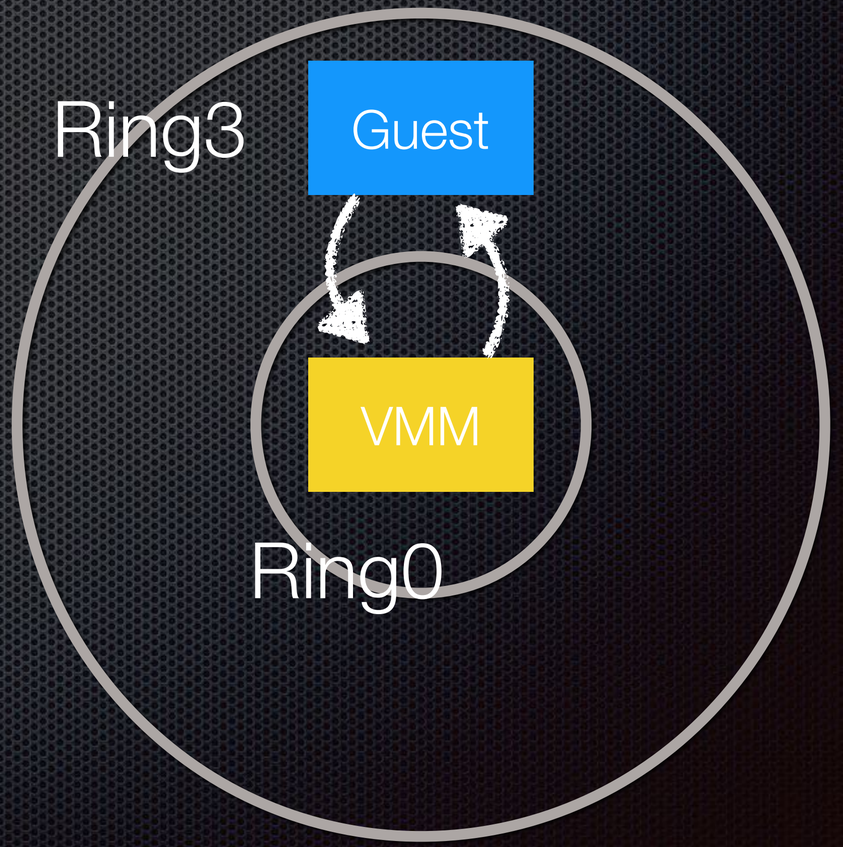
- 服务器虚拟化
- 网络虚拟化
- 存储虚拟化
- 桌面虚拟化
- 应用虚拟化

按照实现层次

- 应用程序级
  - JVM
- 系统库级
  - Wine，Cygwin
- 操作系统级
  - LXC，BSD Jail，Docker
- 硬件抽象级 - Guest与Host具有相同指令集
  - Xen，KVM，VMWare
- ISA级
  - Qemu

x86上的虚拟化技术

- 特权指令
  - 只能在最高特权级下执行的指令，Intel Ring0
- 敏感指令
  - 修改系统敏感信息的指令，如机器状态、时钟寄存器、存储系统、IO操作等
- 基本原理
  - guest在Ring3执行敏感指令，进而陷入到Ring0
  - VMM在Ring0捕捉到guest的操作，进而满足guest



x86指令集的虚拟化缺陷

- 如果所有的敏感指令都是特权指令
  - 指令集就能支持虚拟化
  - VM运行在Ring3，执行敏感指令是会产生异常，陷入VMM，VMM捕获之后进行模拟，此时资源可控
- 如果存在敏感指令不是特权指令，即在Ring3中可以修改系统敏感信息
  - VM运行在Ring3，某些执行敏感指令成功，系统全局信息被修改
  - x86有17条指令中招
- 有缺陷的指令必须要被替换掉

陷入/模拟

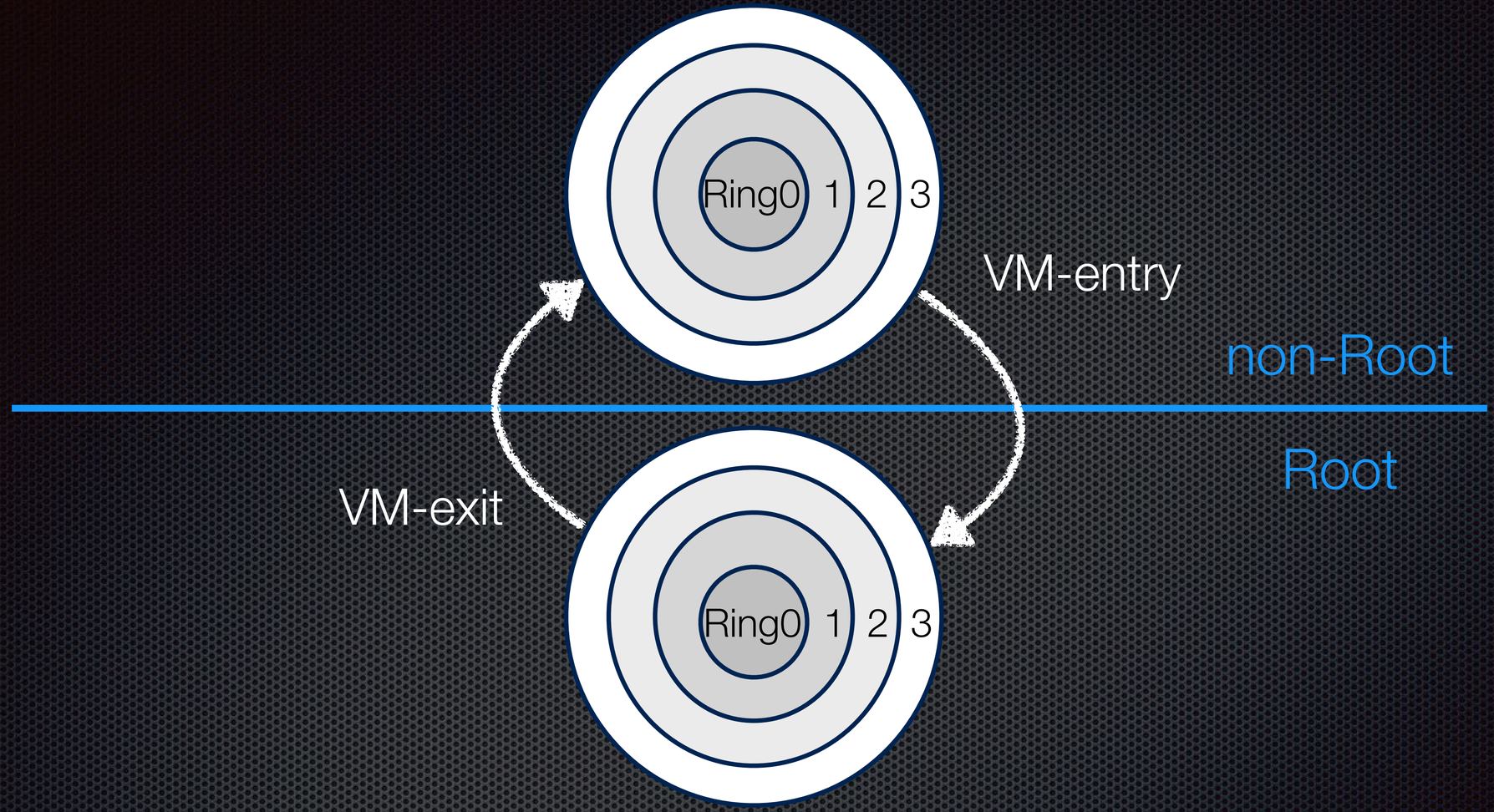
- 模拟的前提是Guest能陷入VMM
- VMM陷入利用处理器的保护机制，利用中断和异常
  - 基于处理器保护机制
  - 虚拟机主动触发异常，Hypercall
  - 异步中断，处理器内部中断源和外设中断源
    - 中断后控制流由中断门和中断向量决定
- 特权级是指令流的属性之一，如CS中的CPL
  - 只要指令流是受控的就可以执行，甚至在虚拟机中嵌套执行

二进制翻译

- 利用二进制翻译替换掉有缺陷的指令
- 优点
  - 不用修改GuestOS内核
  - 利用DBT的先天优势
- 缺点
  - 开销比较大，DBT有性能补偿
- 代表产品
  - VMware

半虚拟化

- PV : paravirtualization
- 在guest的kernel中，将所有的敏感操作全部替换掉
  - 需要修改guest kernel的代码
- pv-ops in linux kernel
  - 比如Guest把修改CR3的操作，修改为一个软中断
- hypercall in Xen



Intel VT-x的金蝉脱壳之计

实质是完成对Ring0的降权，所有非根模式的敏感指令都会陷入

修补了x86指令集的虚拟化缺陷，还保持了兼容

CPU硬件虚拟化

- 引入新的CPU模式用来运行Guest
- guest mode VS host mode
- non-root mode VS root mode on x86
- 该模式具有跟普通模式一样的特性，只是在触发一些“事件”时，会跳转到host mode，由host kernel处理该事件
- KVM 和 Xen HVM

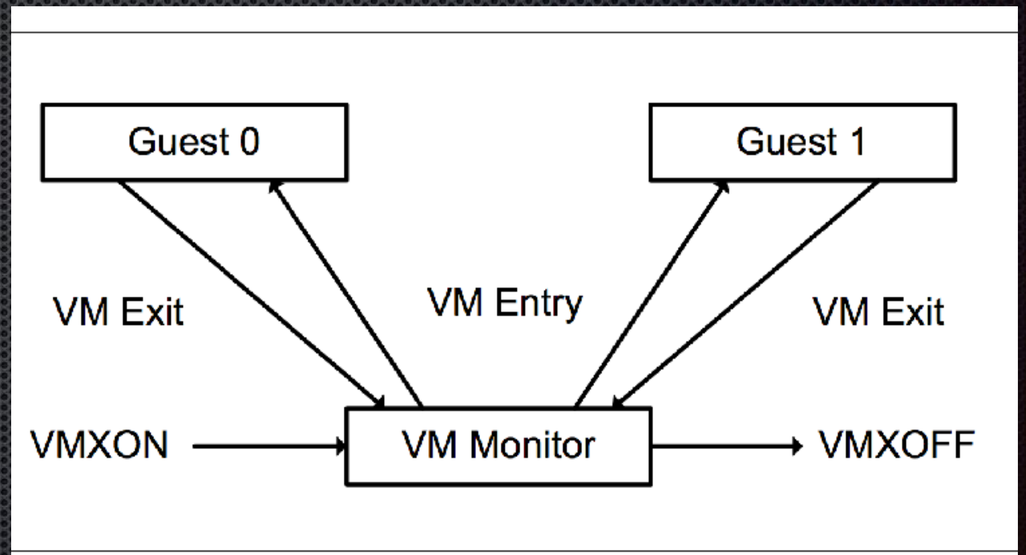
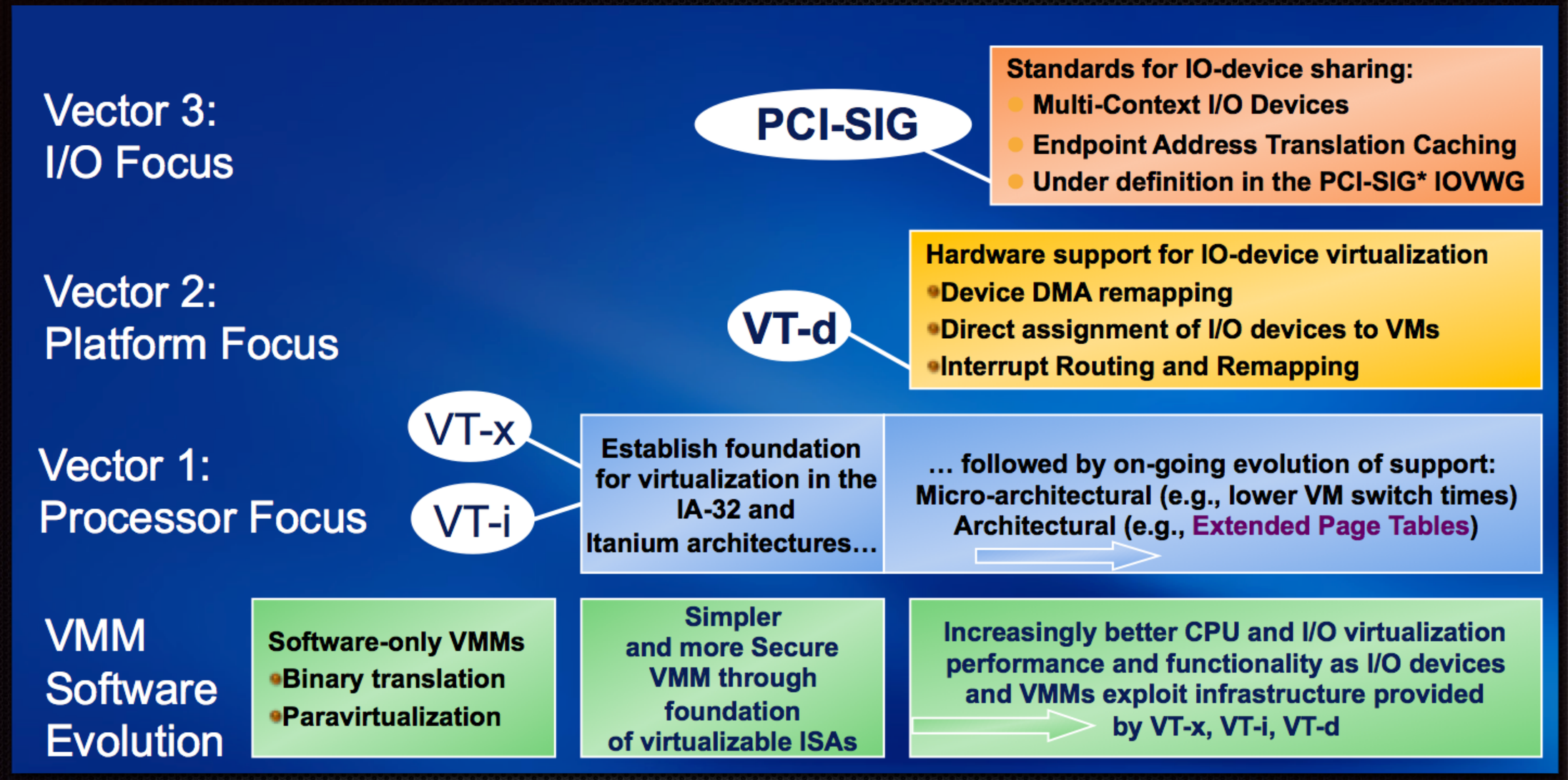
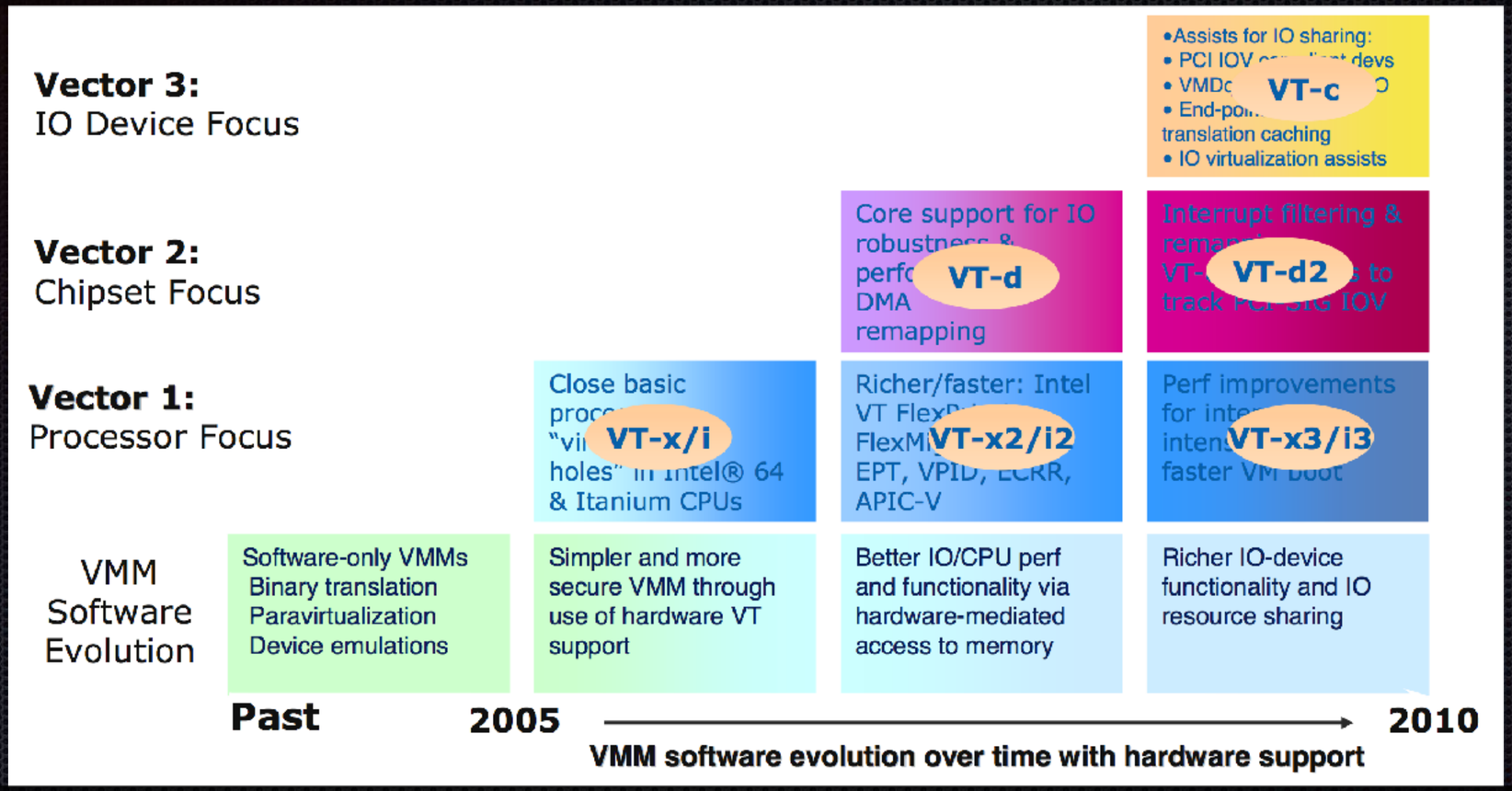


Figure 23-1. Interaction of a Virtual-Machine Monitor and Guests


Vector 3:
I/O Focus
PCI-SIG
Standards for IO-device sharing:
Multi-Context I/O Devices
Endpoint Address Translation Caching
Under definition in the PCI-SIG* IOVWG
Vector 2:
Platform Focus
VT-d
Hardware support for IO-device virtualization
Device DMA remapping
Direct assignment of I/O devices to VMs
Interrupt Routing and Remapping
Vector 1:
Processor Focus
VT-x
VT-i
Establish foundation for virtualization in the IA-32 and Itanium architectures…
… followed by on-going evolution of support:
Micro-architectural (e.g., lower VM switch times)
Architectural (e.g., Extended Page Tables)
VMM
Software
Evolution
Software-only VMMs
Binary translation
Paravirtualization
Simpler and more Secure VMM through foundation of virtualizable ISAs
Increasingly better CPU and I/O virtualization performance and functionality as I/O devices and VMMs exploit infrastructure provided by VT-x, VT-i, VT-d

Intel硬件虚拟化

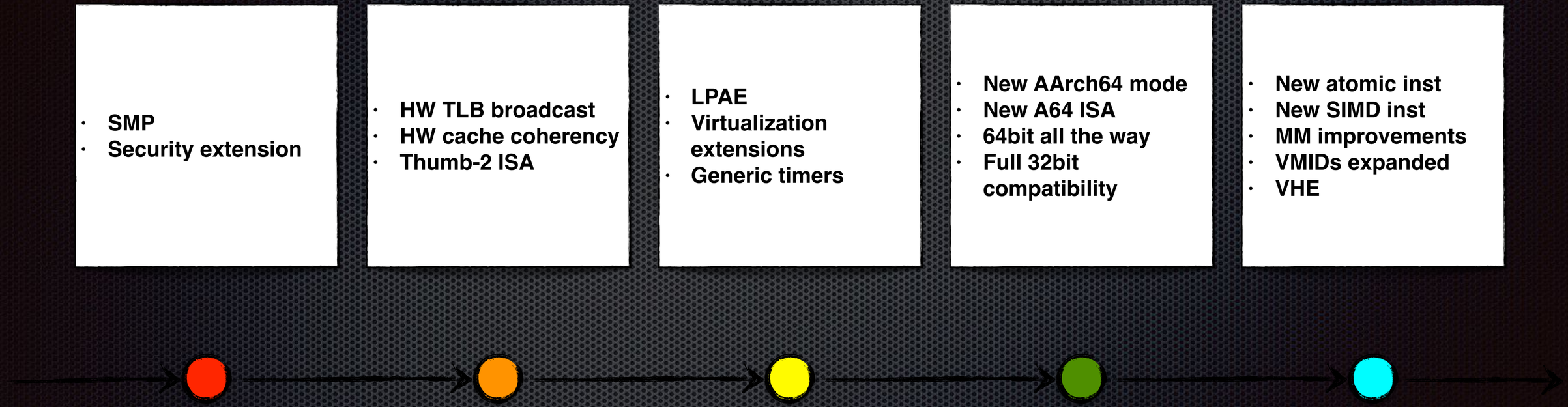


Intel® Virtualization Technology Evolution

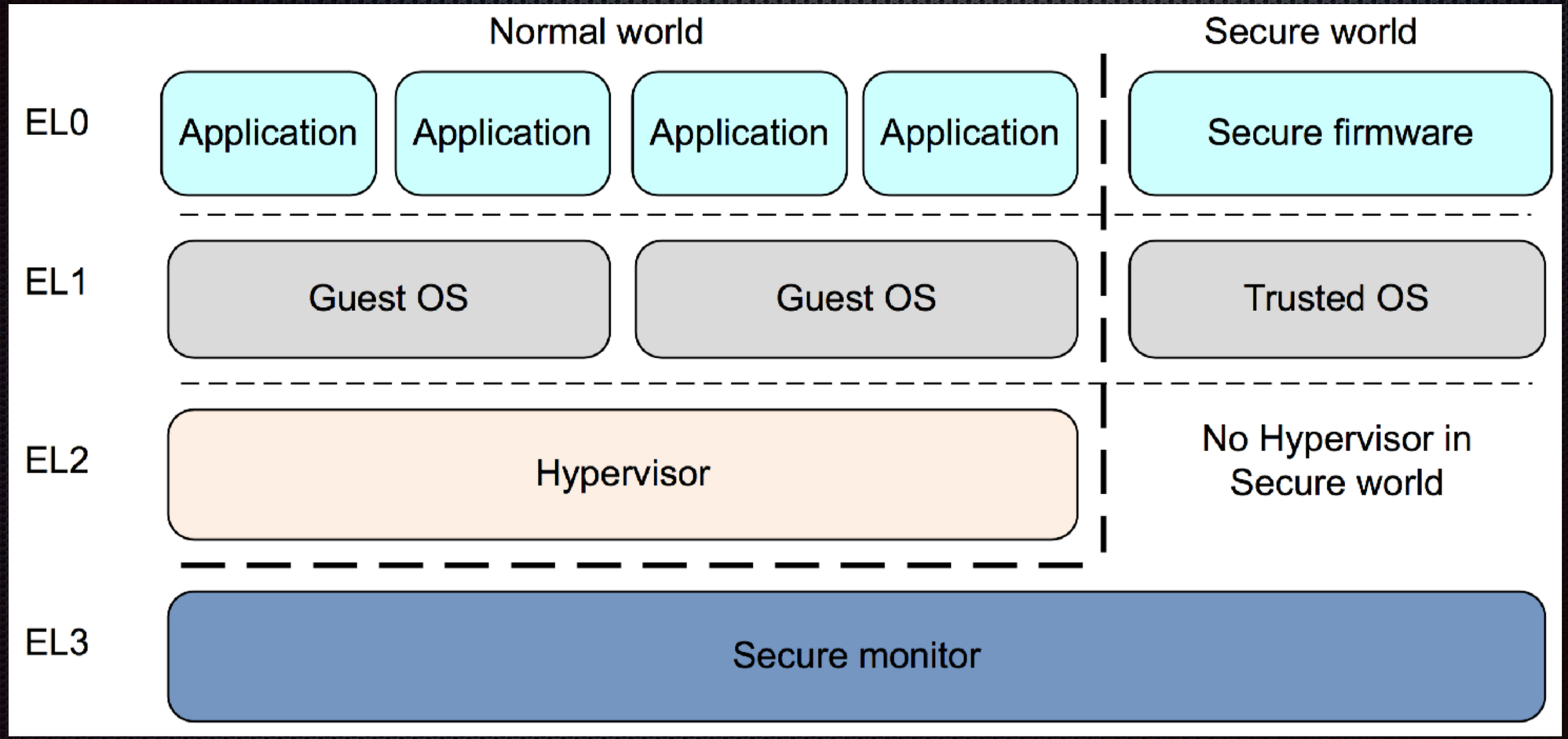
发展方向

从CPU，到芯片组，到外设

Evolution of ARM VE



Exception levels in AArch64

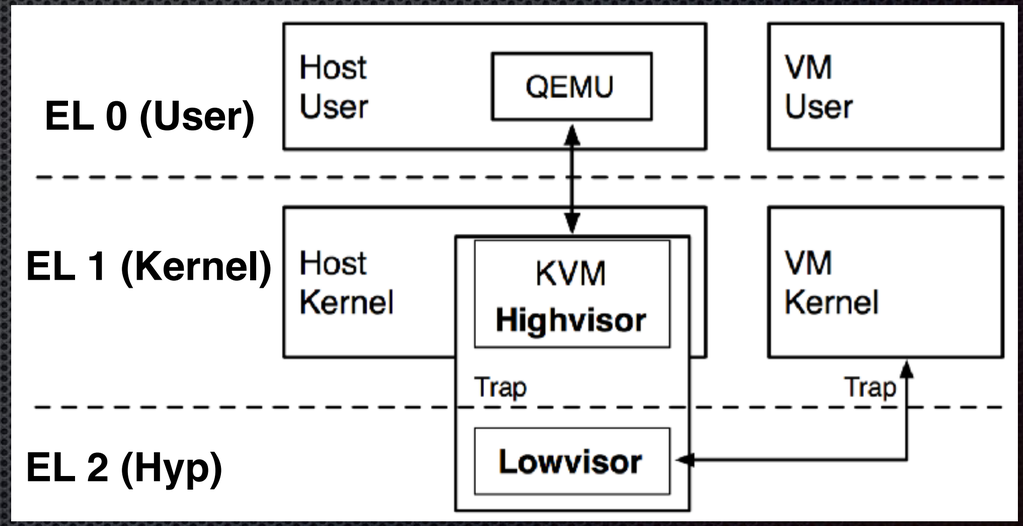


EL2 not superset of EL1

- EL2 is a completely different CPU mode from EL1
  - Different registers
  - Separate address space and different memory model
- Run Linux in EL2 requires too many changes
  - ARM Linux guest kernel should be able to run in EL1
- Potential performance problems in EL2
  - Only one Translation Table Base Register (TTBR0_EL2)

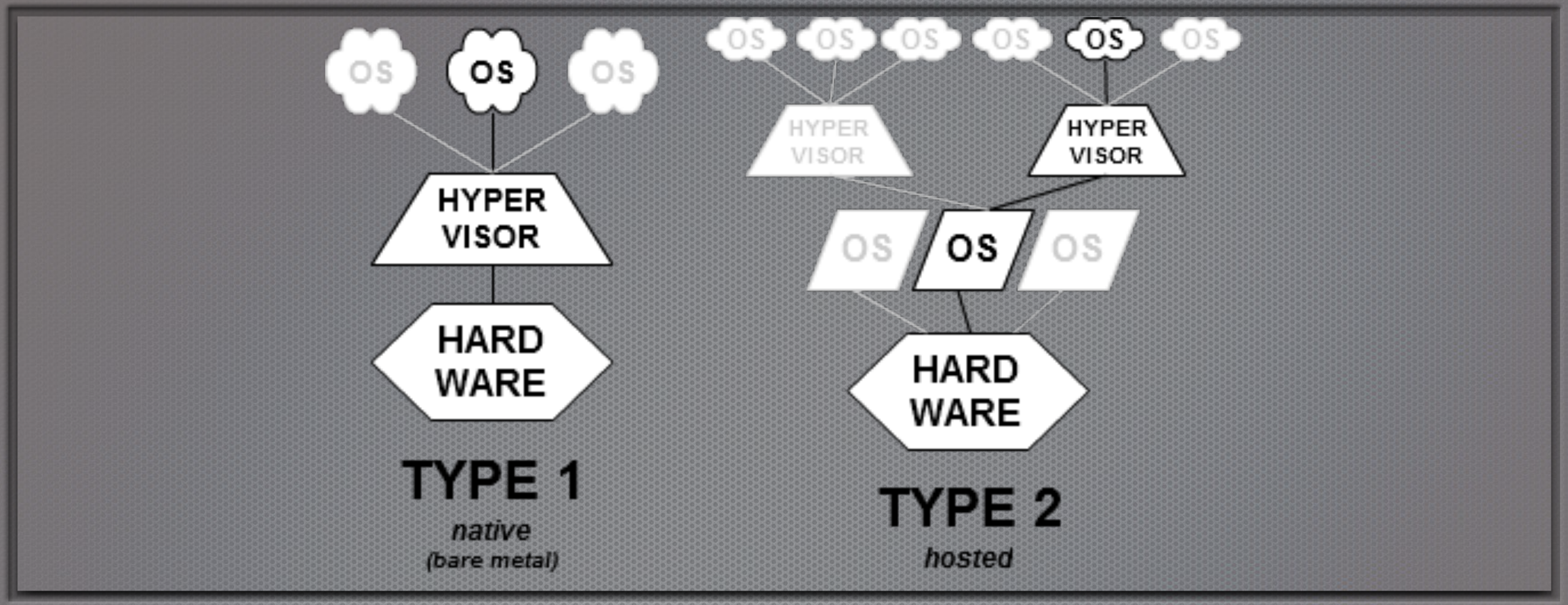
Split-Mode Virtualization

- Lowvisor in EL2
  - Configure hardware to set up execution context
  - Enforce protection and isolation
  - Context Switch
  - Receive interrupts and exceptions
- Highvisor in EL1
  - other functions in KVM



ARM and x86

- Separate mode vs Orthogonal privilege rings
  - EL2 is not a superset of normal privileged EL0/EL1
  - control registers must be programmed in EL2
  - overhead transitioning between EL2 and EL1
  - Linux kernel cannot directly run under EL2
- RISC-style vs CISC-style
  - x86 hardware automatically save to/restore from VMCS when context switching
  - ARM leaves it up to software to decide which state needs to be saved and restored
  - ARM has separate registers in VHE while x86 switching between root and non-root



VMM类型

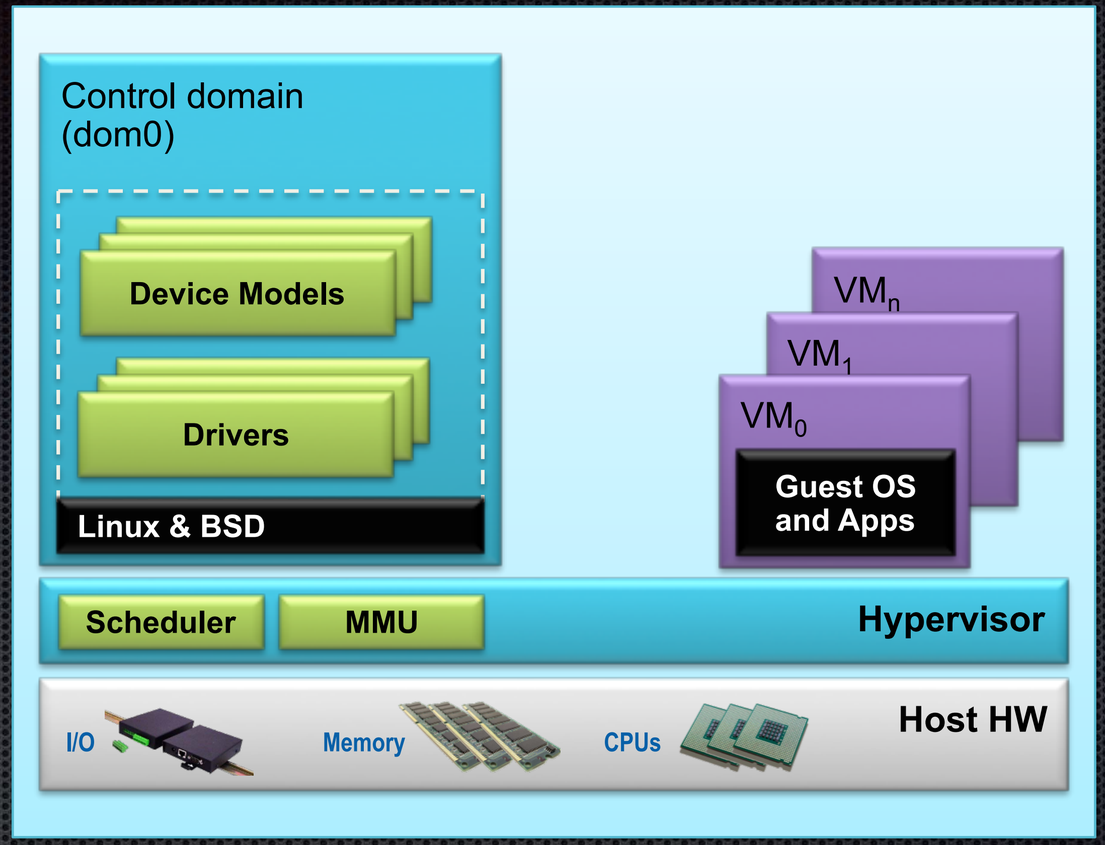
Type-I型 和 Type-II型

VMM举例

- VMware esxi
- 微软 Hyper-v
- XEN、XenServer
- KVM
- Virtual Box
- Power VM

Xen

- Xen是英国剑桥大学的一个研究项目，现在已经成为最著名的开源VMM之一，有自己独立的社区
- 2003年发布1.0，目前最新版本4.11
- 产品特点
  - 可移植性非常好
  - 提供了接近于物理机性能
  - 成熟稳定，为大多数公有云所使用

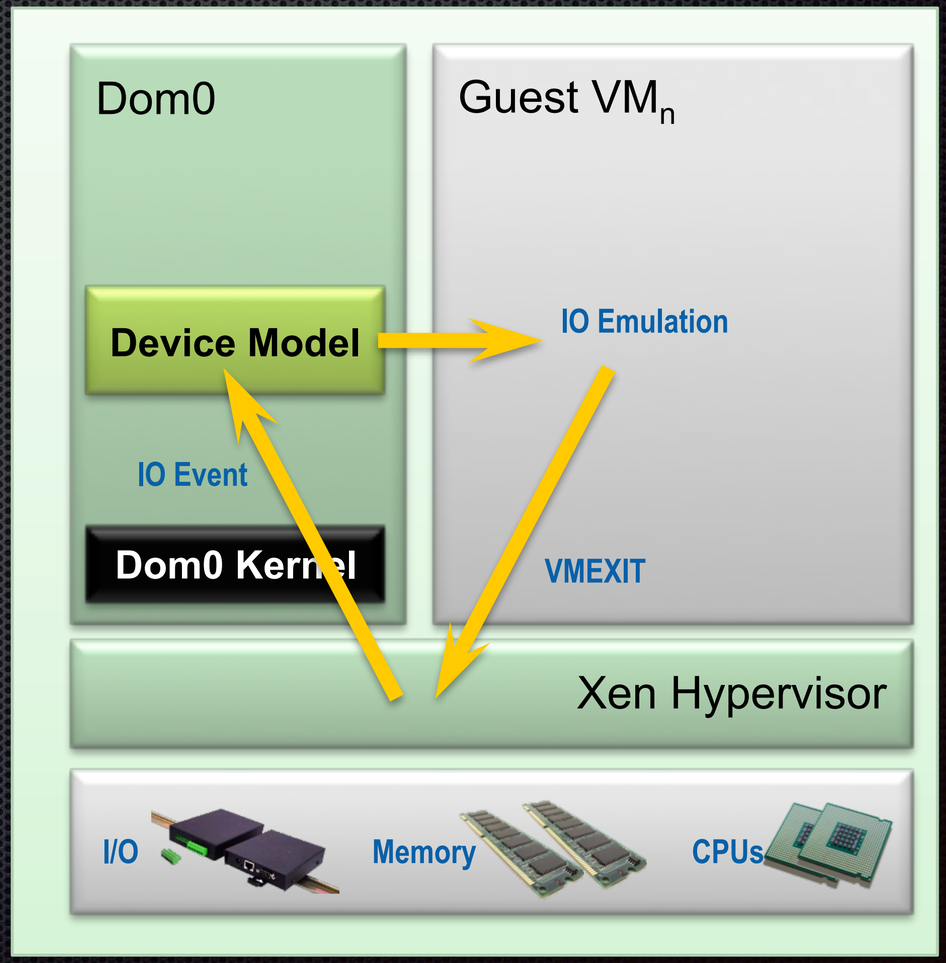


XEN架构-Type1

但是还有点不同，设备模型不在VMM中，在一个特殊的虚拟机中Domain 0

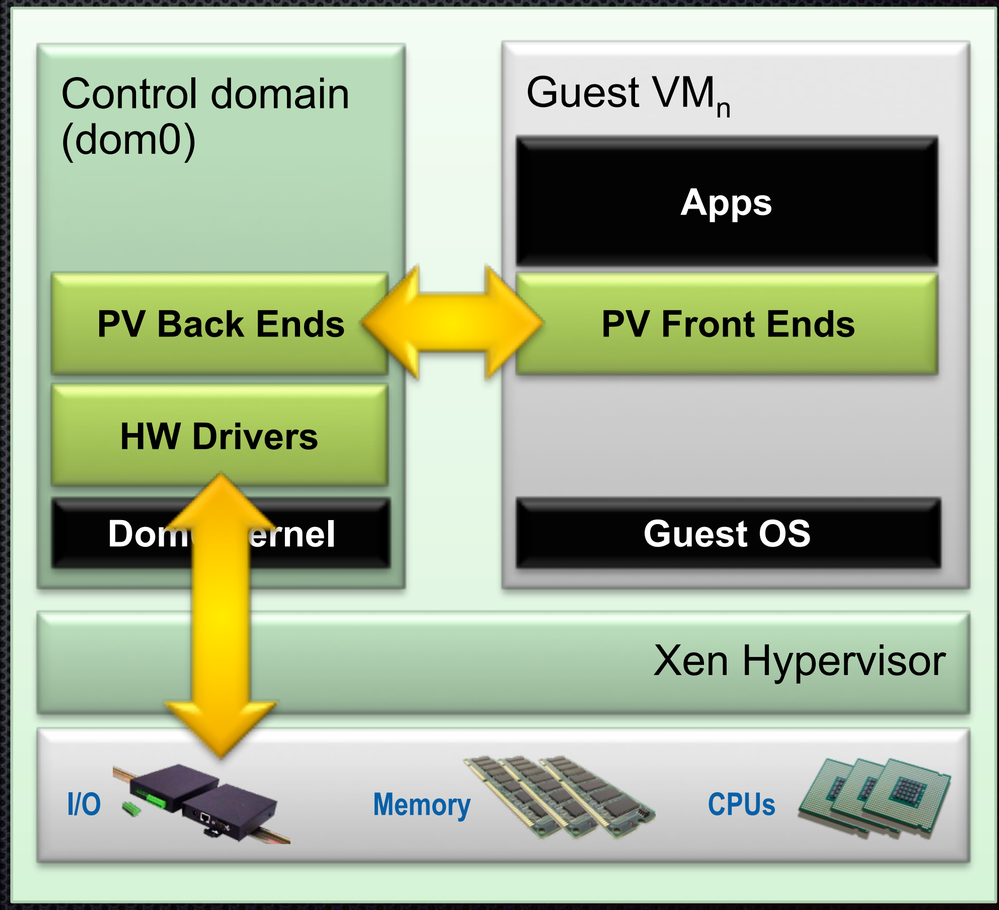
Xen HVM

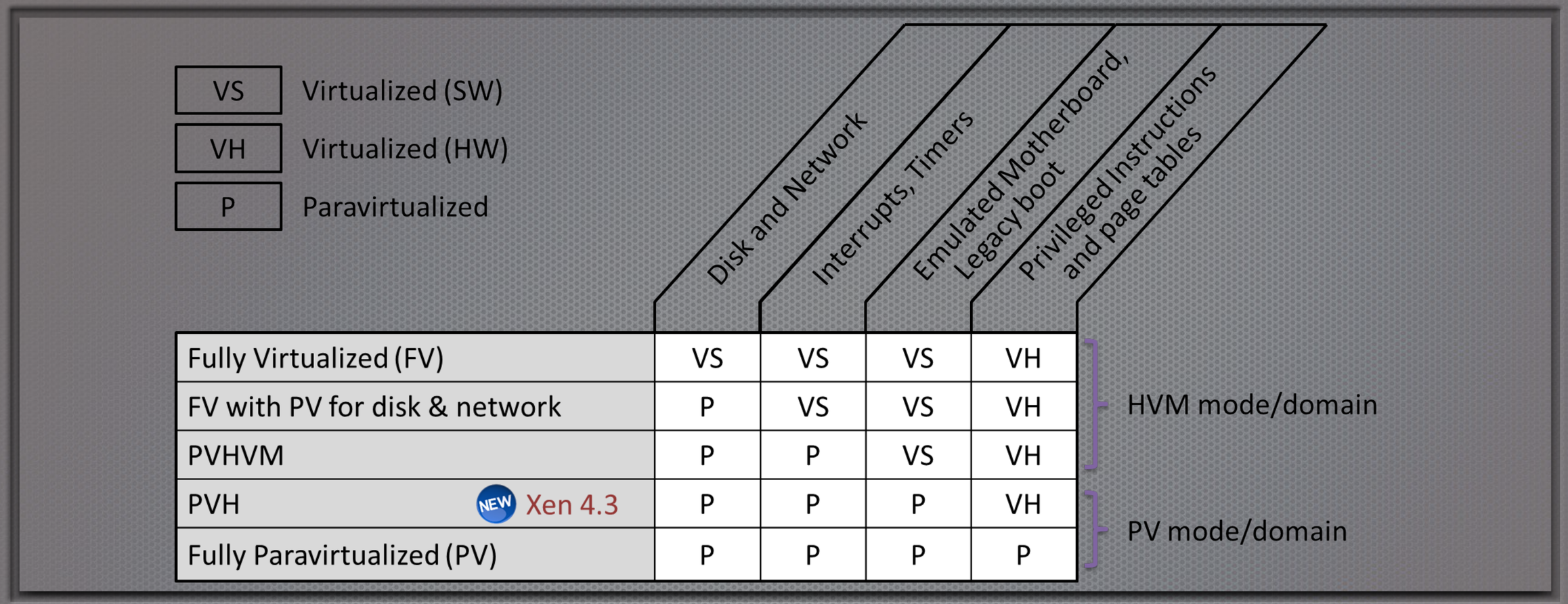
- Technology
  - Emulation using QEMU/Device Model (SW Virtualization)
- Disadvantages
  - Emulation slower than PV (mainly I/O devices)
- Advantages
  - No kernel support needed



Xen PV-on-HVM

- Technology
  - Paravirtualization
- Linux PV guests have limitations
  - limited set of virtual hardware
- Advantages
  - Fast
  - Works on most system (Linux & windows)





Xen支持的虚拟化类型

KVM

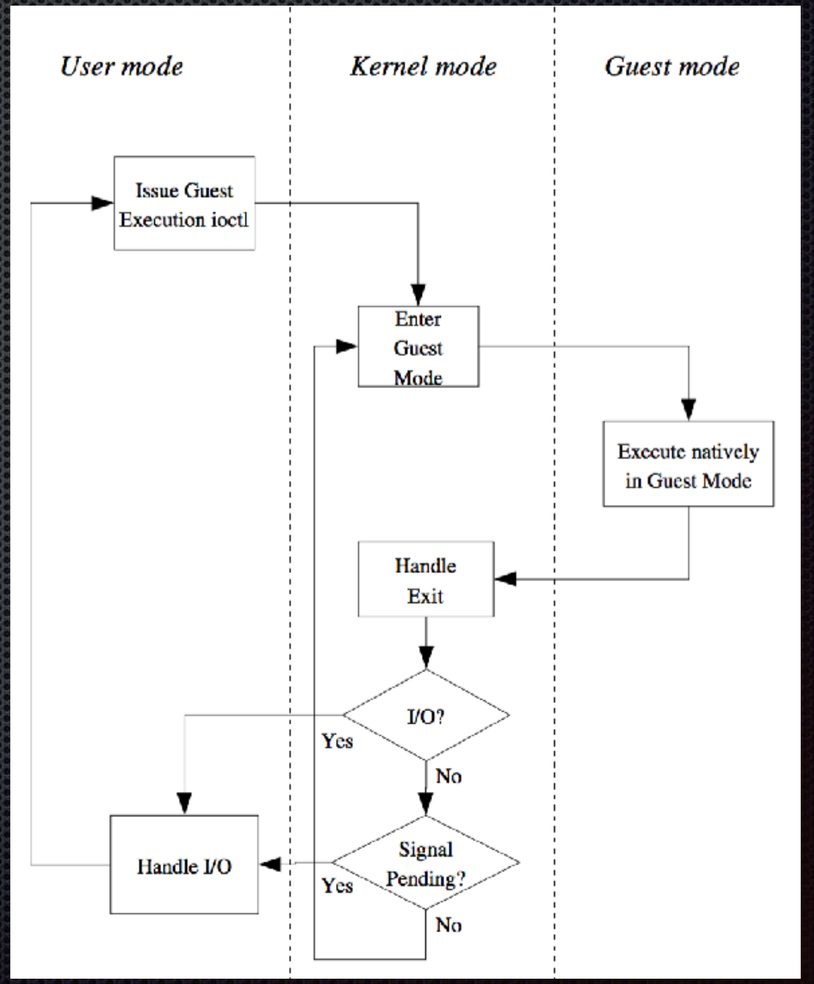
- 基于内核的虚拟机 Kernel-based Virtual Machine
- 起初由 Quramnet 开发，该公司于 2008年被 Red Hat 收购
- 它是Linux 的一个内核模块，该内核模块使得 Linux 变成了一个 Hypervisor
  - Linux 2.6.20进入主线，遵循GPL
- 它需要支持硬件虚拟化的 CPU
  - x86 / s390 / Powerpc / ARM 等

KVM

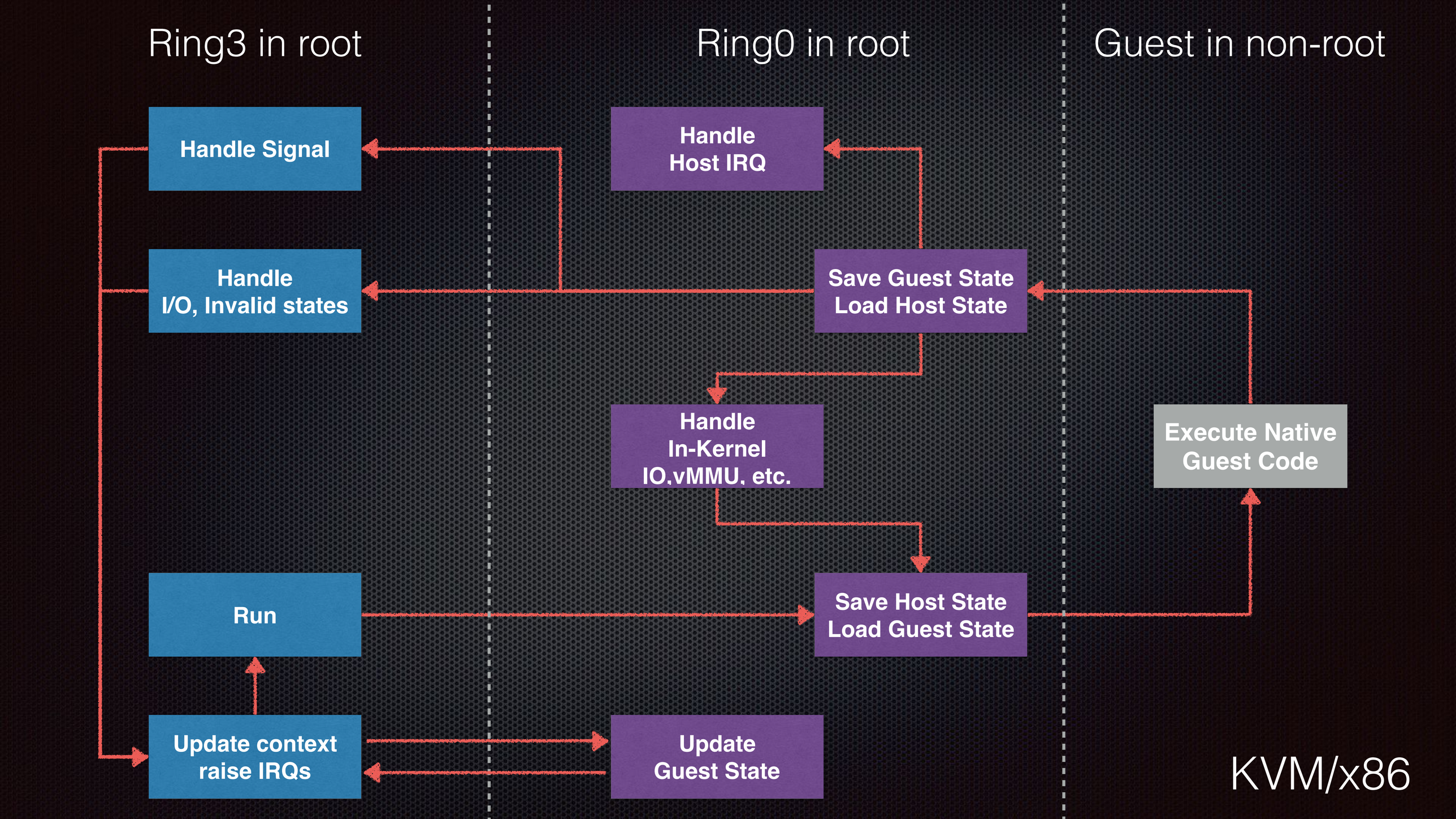
- VMM如何操纵虚拟机
  - KVM_API
- 哪些事件要传递给VMM
  - 虚拟机访问了本该VMM维护的资源，而产生的事件

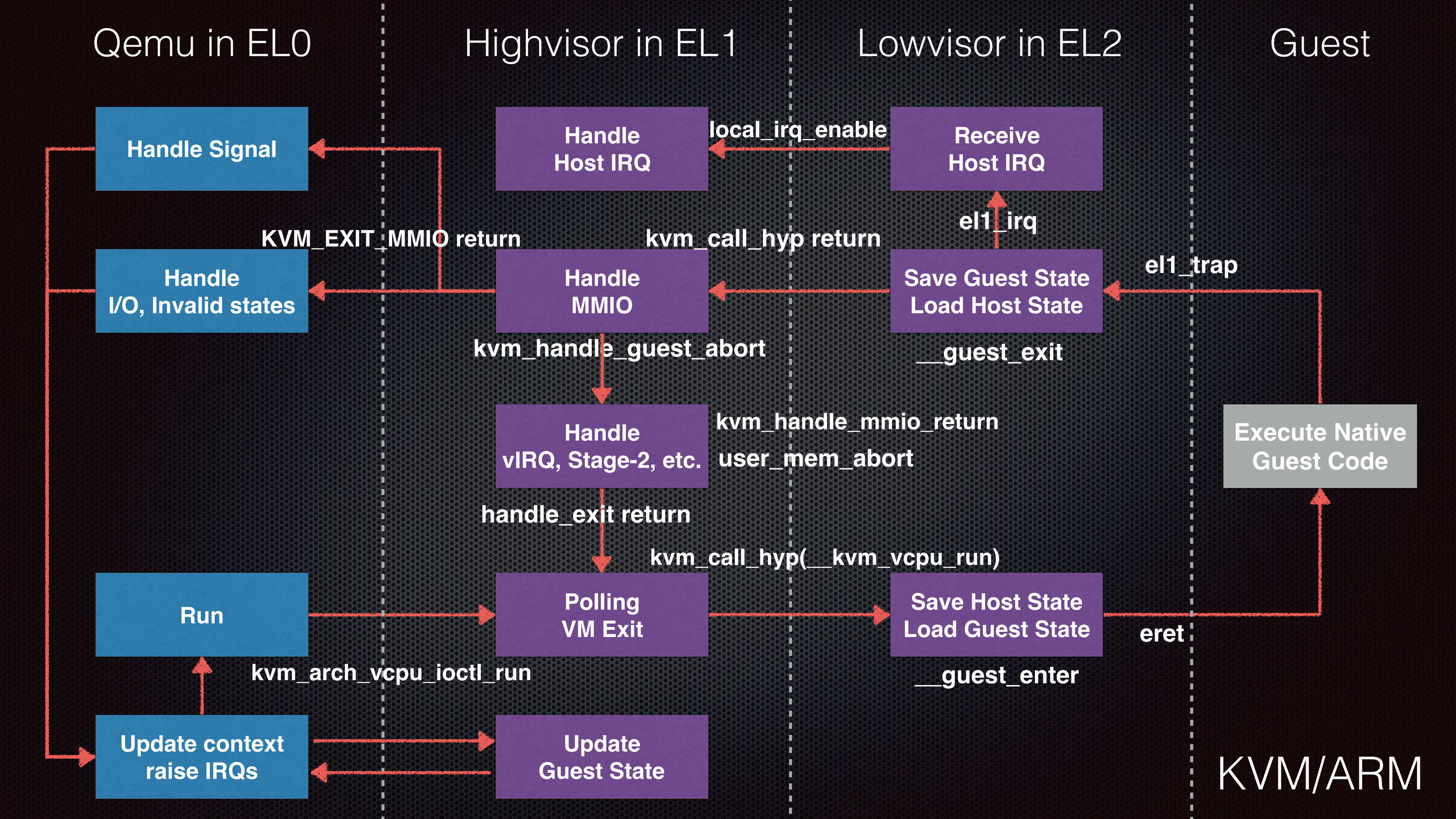
KVM

- kvm_fd = open(“/dev/kvm”);
- vm_fd = ioctl(kvm_fd, KVM_CREATE_VM, 0);
- posix_memalign(&mem, PAGE_SIZE, 0x20000000);
- ioctl(kvm_fd, KVM_SET_USER_MEMORY_REGION, mem);
- vcpu_fd = ioctl(vm_fd, KVM_CREATE_VCPU, 0);
- thread_start(run_vcpu(vcpu_fd));
- while (1)
  - process_io();



kvm: the Linux Virtual Machine Monitor


Ring3 in root
Ring0 in root
Guest in non-root
Handle Signal
Handle Host IRQ
Handle I/O, Invalid states
Save Guest State Load Host State
Handle In-Kernel IO,vMMU, etc.
Execute Native Guest Code
Run
Save Host State Load Guest State
Update context raise IRQs
Update Guest State
KVM/x86


Qemu in EL0
Highvisor in EL1
Lowvisor in EL2
Guest
Handle Signal
Handle Host IRQ
local_irq_enable
Receive Host IRQ
el1_irq
KVM_EXIT_MMIO return
kvm_call_hyp return
Handle I/O, Invalid states
Handle MMIO
Save Guest State Load Host State
el1_trap
kvm_handle_guest_abort
__guest_exit
Handle vIRQ, Stage-2, etc.
kvm_handle_mmio_return
user_mem_abort
Execute Native Guest Code
handle_exit return
kvm_call_hyp(__kvm_vcpu_run)
Run
Polling VM Exit
Save Host State Load Guest State
eret
kvm_arch_vcpu_ioctl_run
__guest_enter
Update context raise IRQs
Update Guest State
KVM/ARM