RISC-V虚拟化技术概览及实践

李天正

中国电信研究院

RISC-V虚拟化现状

RISC-V逐步进入云计算领域，虚拟化是云计算技术的基石

指令集规范

进展

- 虚拟化扩展规范已经发布正式版本
- IOMMU、AIA等规范逐步完善

问题

- 特性和规范仍待完善，如嵌套虚拟化、SBI对虚拟化的完整支持

硬件实现

进展

- 国内外厂商发布数款支持虚拟化的CPU IP
- 开源实现Rocket Chip、香山昆明湖

问题

- 在短期内仍缺少支持虚拟化的SoC产品

软件生态

进展

- QEMU、KVM等虚拟化基础软件适配
- Ubuntu、Debian等服务操作系统适配

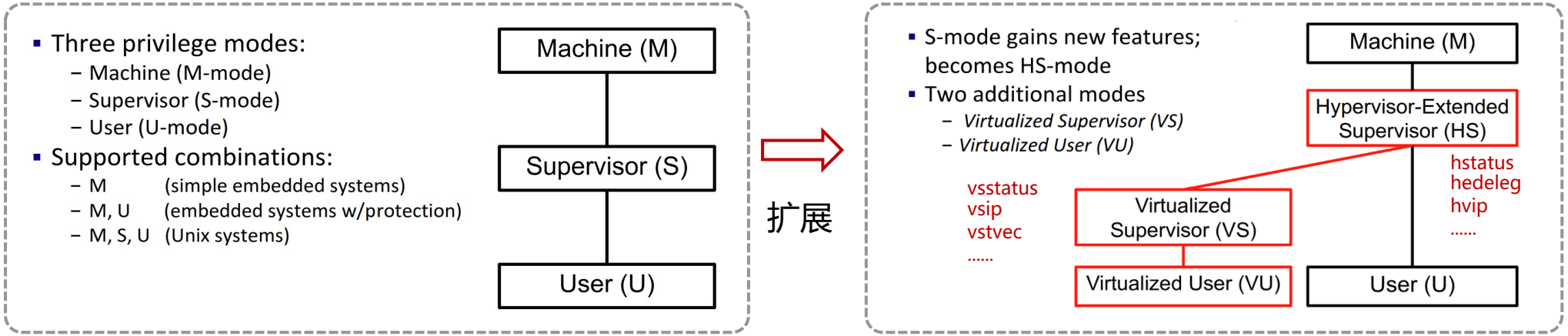
问题

- Xen、Firecracker等相关虚拟化软件仍未适配，虚拟化软件生态与x86和ARM仍存在差距

RISC-V虚拟化扩展

RISC-V特权架构规范提供了硬件虚拟化支持，即可选的 hypervisor 扩展（"H"），1.0版本于2021年12月发布，包括以下改动

- 增加新的特权模式，Hypervisor运行在HS模式，虚拟机则运行在VS和VU模式



- H扩展为HS和VS模式新增了一系列CSR，并修改了M模式的部分CSR
- 新增部分指令

Hypervisor Instructions	
HLV.width HLVX.HU/WU HSV.width	Hypervisor Virtual-Machine Load and Store Instructions
HFENCE.VVMA HFENCE.GVMA	Hypervisor Memory-Management Fence Instructions

- 两阶段地址转换
- 新增用于虚拟化的中断和异常

RISC-V虚拟化技术实践-TeleVM

➢ **轻量级虚拟机诞生背景：**

容器云、Serverless等应用不断发展，**当前容器轻量，但不够安全；虚拟机安全，但不够轻量。**目前KVM和QEMU 一起构成了当前业界主流的虚拟化方案，其中KVM位于Linux 内核，将硬件虚拟化的功能抽象成接口交给用户空间使用，QEMU则位于用户空间管控虚拟机。然而QEMU存在以下问题：

- **代码过重：**目前 QEMU已经有157万代码，存在臃肿、响应慢、占用资源多等问题
- **安全问题：**QEMU的CVE问题，其中有将近一半是因为内存问题导致的

➢ **轻量级虚拟机优势：**

精简了传统虚拟机（QEMU+KVM），既能**保障安全**，又具备**轻量化、低开销**的特点

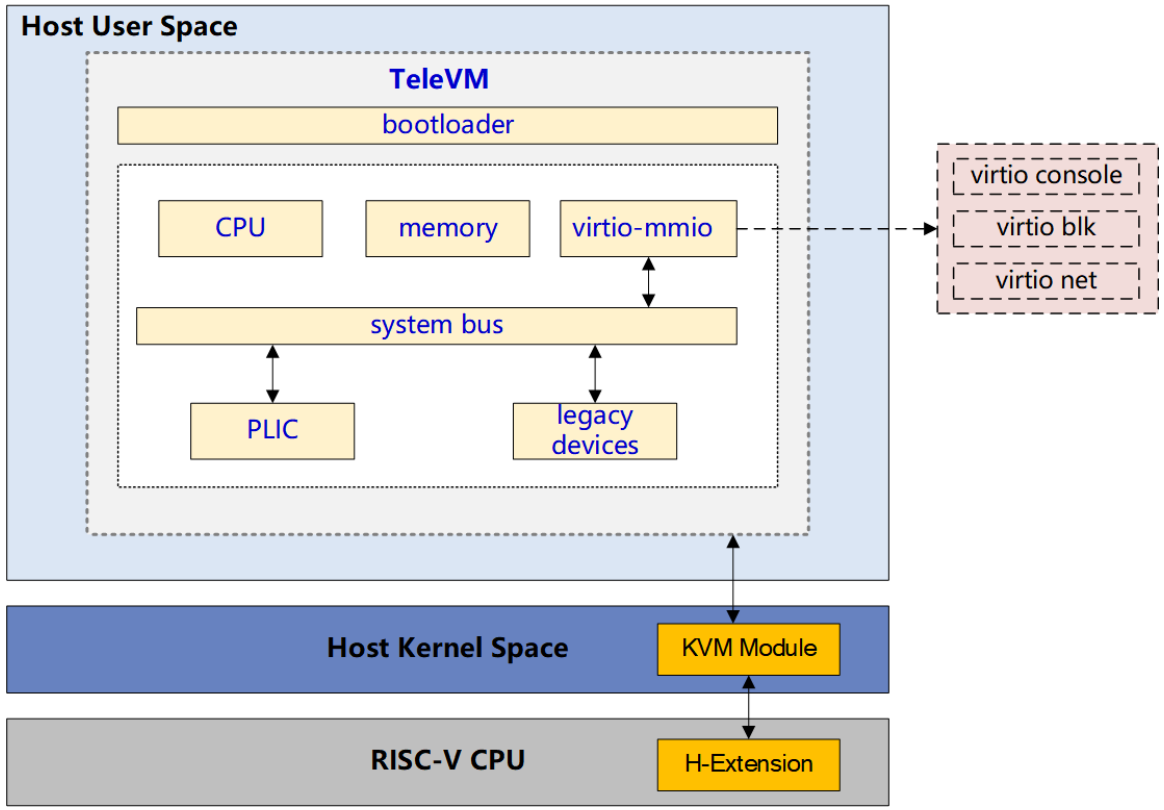
➢ **典型轻量级虚拟机开源项目**

- Firecracker（AWS）
- Stratovirt（OpenEuler）

目前缺少支持RISC-V架构的轻量级虚拟机

轻量级虚拟机TeleVM

- Rust-VMM工具链适配
- RISC-V CPU虚拟化
- RISC-V 内存虚拟化
- RISC-V BootLoader
- RISC-V 中断虚拟化
- RISC-V I/O虚拟化



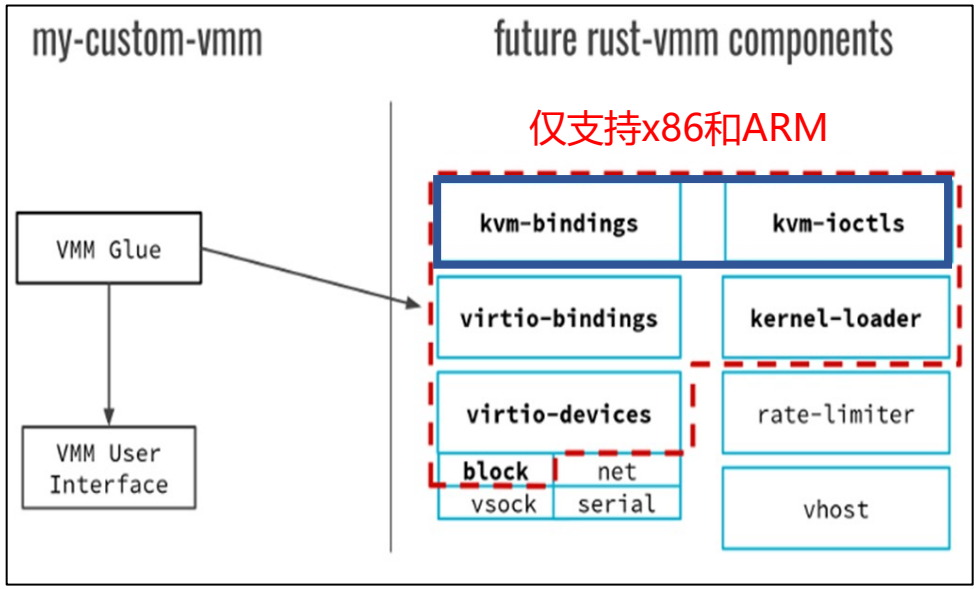
TeleVM整体架构

优势

- 强隔离性：基于RISC-V硬件虚拟化能力和KVM实现强隔离性
- 轻量低噪：相比QEMU+KVM传统方案启动时间降低80%，内存底噪减少90%
- 安全可信：基于Rust安全语言；架构精简减少攻击面
- 可扩展性：统一接口设计，可扩展设备模型和特性

Rust-VMM适配

- Rust-VMM是由AWS、谷歌等公司贡献的开源项目，为构建VMM程序提供了一组可重用的组件
- TeleVM使用kvm-ioctls和kvm-bindings与KVM交互



Rust-VMM组件

适配RISC-V架构

- kvm-ioctls: kvm api封装
- kvm-bindings: kvm数据结构封装

RISC-V CPU虚拟化

- CPU模块使用KVM实现对CPU的模拟，在用户态主要负责vCPU生命周期的管理和vCPU寄存器的初始化
- 针对RISC-V架构只能使用KVM提供的KVM_SET（GET）_ONE_REG对vCPU的单个寄存器进行操作
- vCPU在初始化阶段要设置pc，a0，a1寄存器以启动Guest OS；通过获取timer等寄存器的值得到vCPU信息

Encoding	Register	Description
0x80x0 0000 0200 0000	regs.pc	Program counter
0x80x0 0000 0200 0001	regs.ra	Return address
0x80x0 0000 0200 0002	regs.sp	Stack pointer
0x80x0 0000 0200 0003	regs.gp	Global pointer
0x80x0 0000 0200 0004	regs.tp	Task pointer
0x80x0 0000 0200 0005	regs.t0	Caller saved register 0
0x80x0 0000 0200 0006	regs.t1	Caller saved register 1
0x80x0 0000 0200 0007	regs.t2	Caller saved register 2
0x80x0 0000 0200 0008	regs.s0	Callee saved register 0
0x80x0 0000 0200 0009	regs.s1	Callee saved register 1
0x80x0 0000 0200 000a	regs.a0	Function argument (or return value) 0
0x80x0 0000 0200 000b	regs.a1	Function argument (or return value) 1
0x80x0 0000 0200 000c	regs.a2	Function argument 2
0x80x0 0000 0200 000d	regs.a3	Function argument 3
0x80x0 0000 0200 000e	regs.a4	Function argument 4
0x80x0 0000 0200 000f	regs.a5	Function argument 5
0x80x0 0000 0200 0010	regs.a6	Function argument 6
0x80x0 0000 0200 0011	regs.a7	Function argument 7
0x80x0 0000 0200 0012	regs.s2	Callee saved register 2

RISC-V core registers

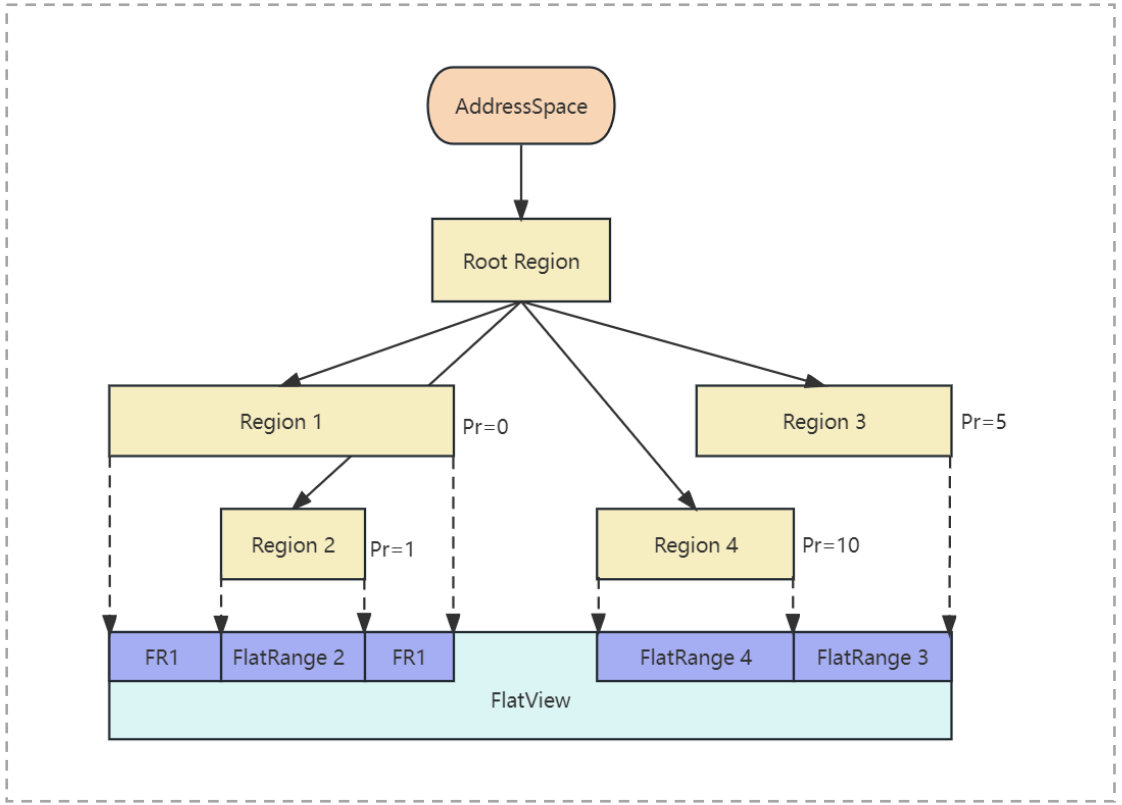
Encoding	Register	Description
0x8030 0000 0400 0000	frequency	Time base frequency (read-only)
0x8030 0000 0400 0001	time	Time value visible to Guest
0x8030 0000 0400 0002	compare	Time compare programmed by Guest
0x8030 0000 0400 0003	state	Time compare state (1 = ON or 0 = OFF)

RISC-V timer registers

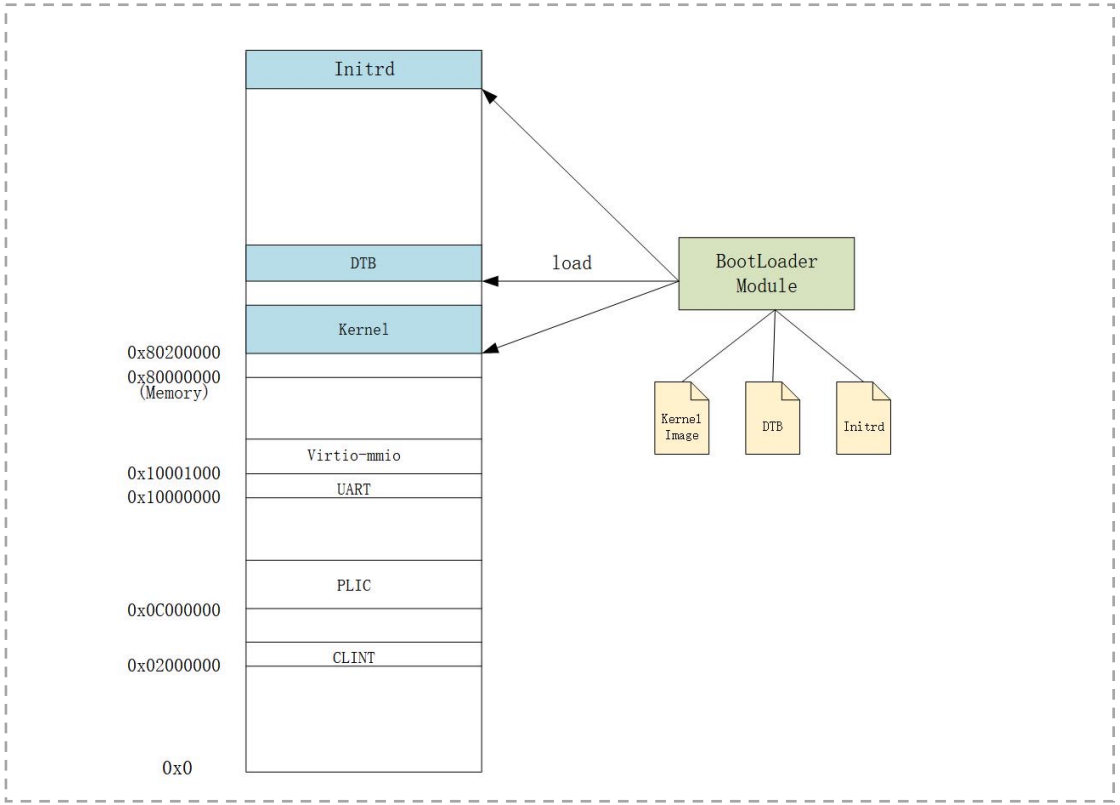
KVM API 文档：https://www.kernel.org/doc/html/latest/virt/kvm/api.html

RISC-V内存虚拟化与BootLoader

- 内存模块在用户态主要负责内存初始化和虚拟机的地址空间管理，地址空间设计采用树状结构和平坦视图结合的方案
- 为了减少启动时间采用直接加载内核的方式，由BootLoader模块计算内核、设备树和Initrd的加载地址并进行加载



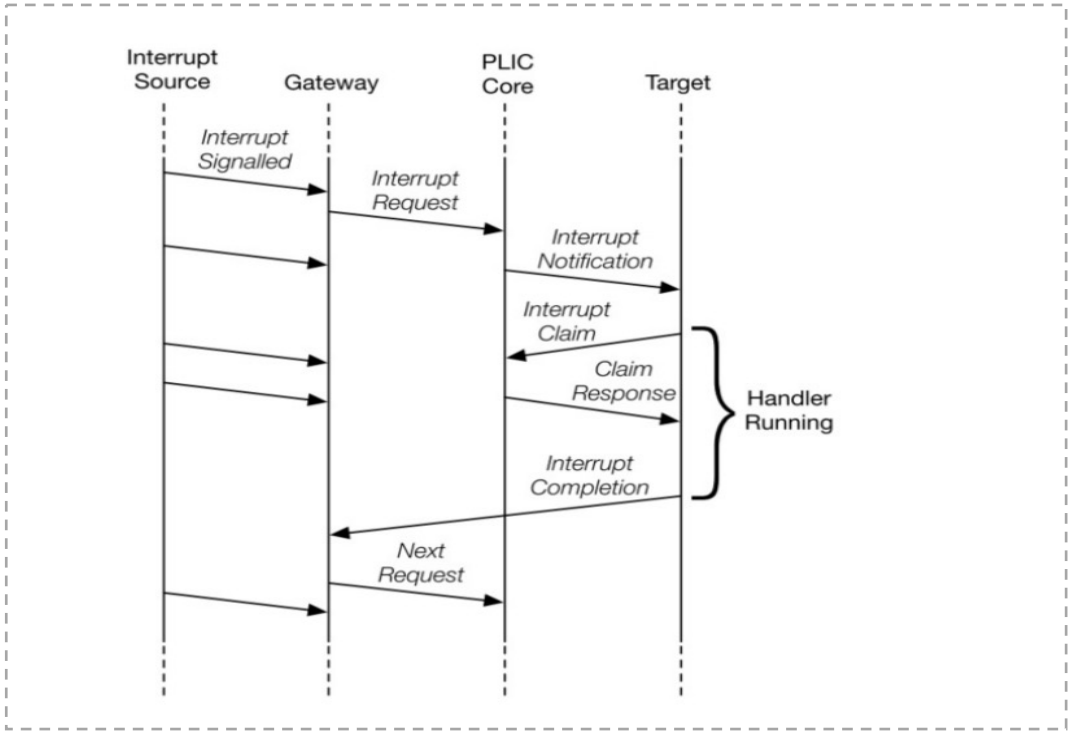
内存拓扑结构图



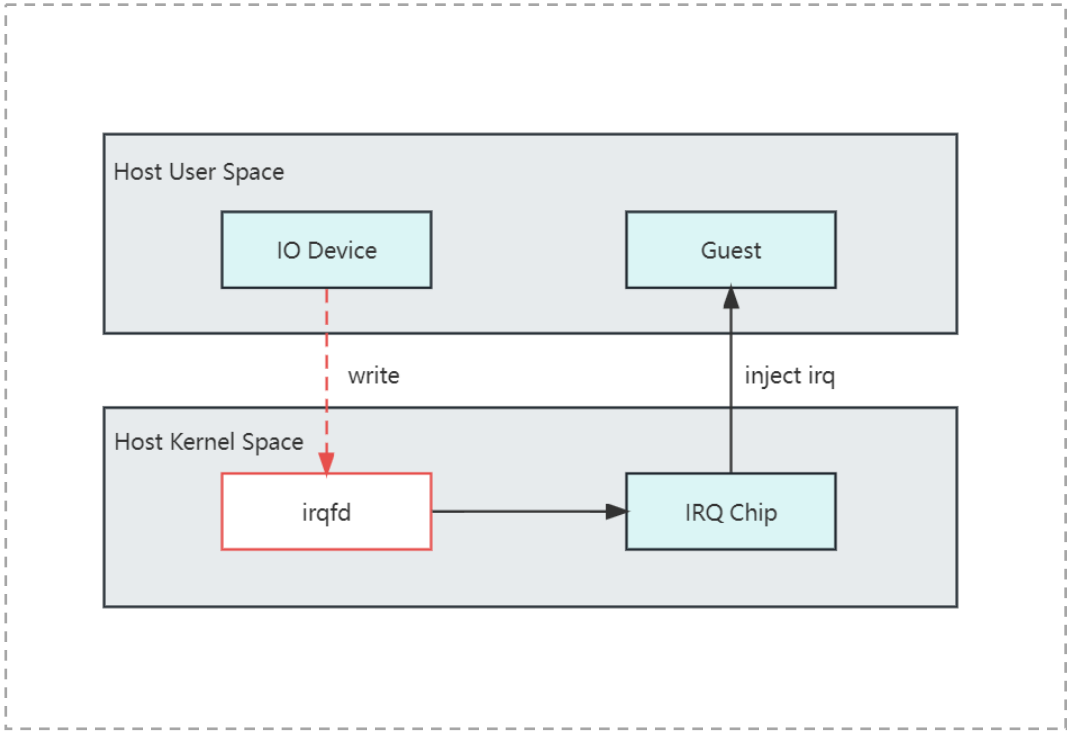
虚拟机地址空间布局

RISC-V中断虚拟化与IO虚拟化

- 当前RISC-V架构使用PLIC中断控制器，但是PLIC不支持中断虚拟化，需要在用户空间使用软件方式模拟PLIC的中断流程
- 针对PLIC的缺陷，RISC-V提出了AIA架构，标准已经发布正式版，相关补丁在逐步合并进内核中
- TeleVM中仅模拟少量传统设备和virtio-mmio设备；触发中断时只能使用KVM_INTERRUPT注入中断，存在额外的中断开销



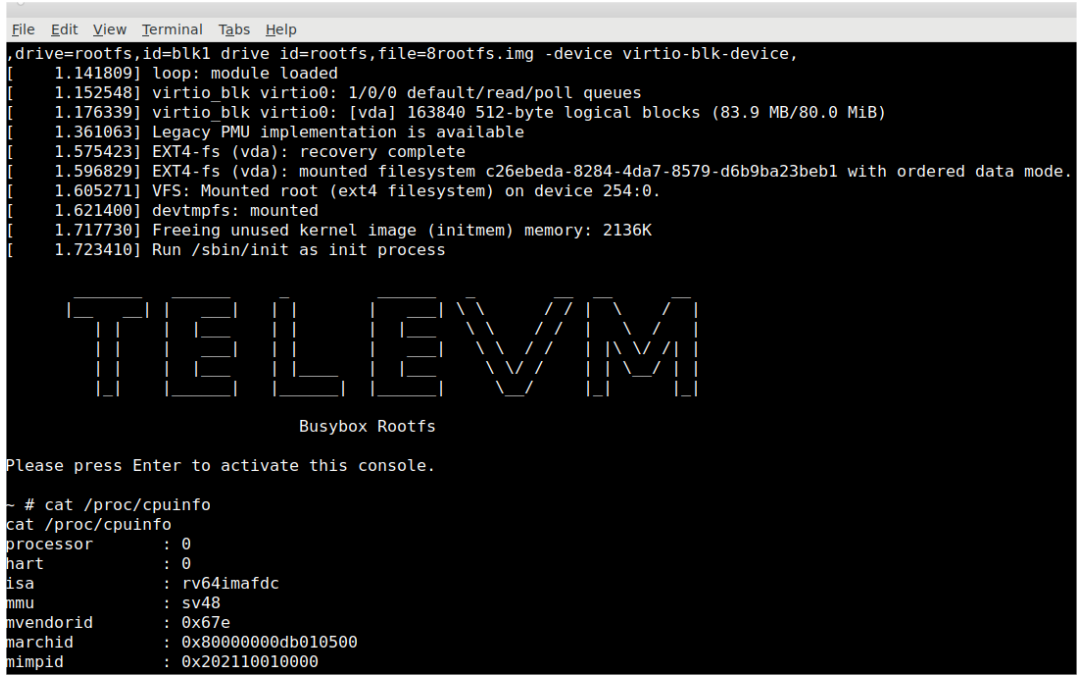
PLIC中断处理流程



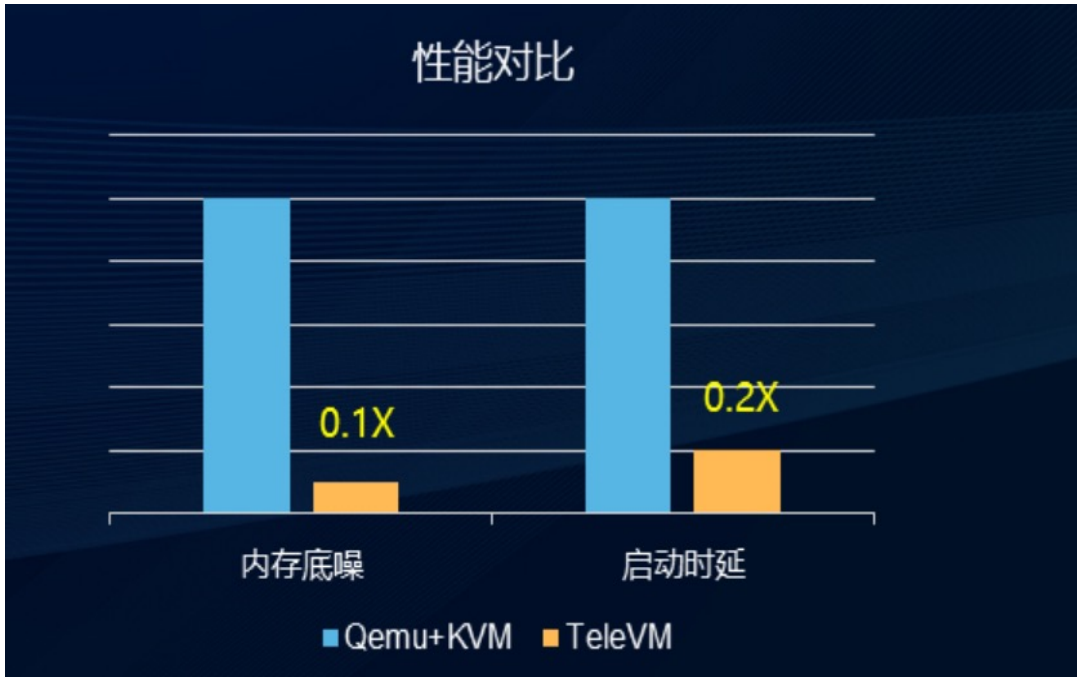
PLIC无法使用irqfd向Guest直接注入中断

测试结果

- 在实现了H扩展的RISC-V CPU IP核上对轻量级虚拟机进行了验证测试，对比QEMU+KVM启动时间减少了80%，内存底噪减少了90%



在合作伙伴RISC-V CPU IP核上验证成功



虚拟化关键指标对比

谢 谢!