用 Rust 语言编写 hypervisor

RVM: Rust Virtual Machine

贾越凯

清华大学计算机系

2023/2/24

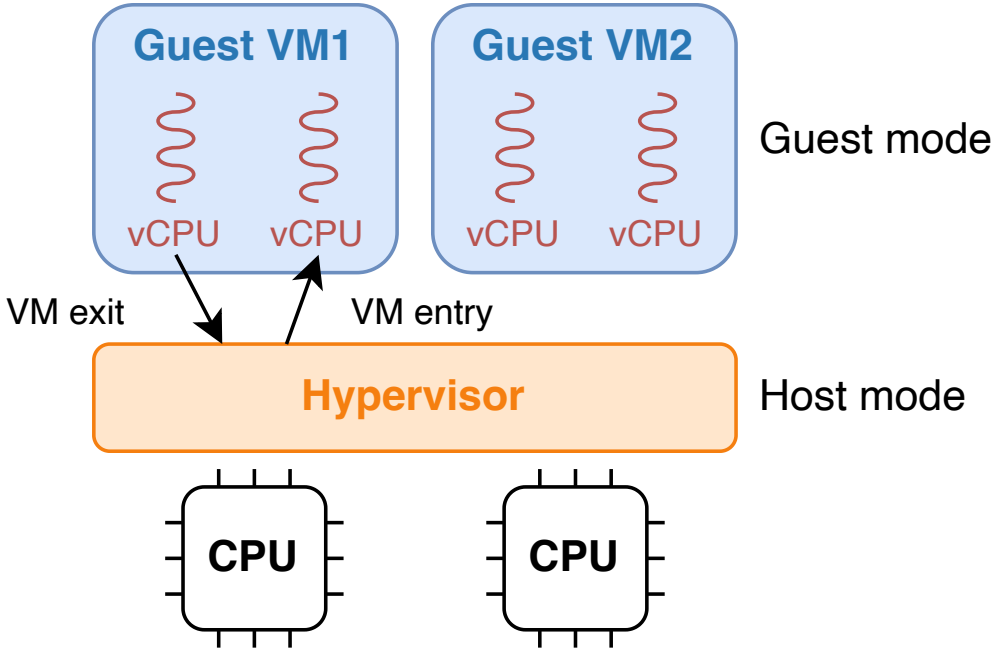
背景

- 虚拟化
  - 在一台计算机上创建一台或多台虚拟的计算机
  - 支持虚拟机中 OS 等软件的运行
- 好处
  - 资源划分：提升利用率
  - 资源隔离：提升安全性

虚拟化基本概念

- CPU 模式
  - **host**：最高特权级
  - **guest**：比 host 特权级低
- 软件
  - **hypervisor**：host 模式运行
  - **guest OS**：guest 模式运行
- 模式切换
  - **VM entry**：host → guest
  - **VM exit**：guest → host

- 状态的抽象
  - **vCPU**：guest 每 CPU 的状态
  - **Guest VM**：所有 vCPU + guest 全局机器状态



CPU 虚拟化

- 实现方式：
  - 比 guest 更高的特权级：trap-and-emulate
  - 只拦截敏感指令或事件：CPU、内存、I/O、中断
  - 软件/硬件
- 硬件虚拟化：
  - Intel VT-x (VMX)
  - AMD-V (SVM)
  - ARM EL2
  - RISC-V H-extension

内存虚拟化

- 页表：
  - 虚拟地址 (VA) → 物理地址 (PA)
- Guest 页表：
  - **guest 虚拟地址 (gVA) → guest 物理地址 (gPA)**
- 内存虚拟化：
  - guest 虚拟地址/物理地址 (gVA/gPA) → **host 物理地址 (hPA)**
- 实现方式：
  - 无特殊硬件：影子页表
  - 特殊硬件：嵌套页表

设备虚拟化

- CPU 与设备的交互 = I/O + 中断

	直通 (pass-through)	模拟 (emulation)	半虚拟化
实现方式	简单	复杂	复杂
性能开销	小	大	较小
多路复用	不支持	支持	支持
guest OS	无修改	无修改	需要修改

- 半虚拟化 (para-virtualization)：让 guest 与 host 通过人为规定的接口进行通信，如 virtio

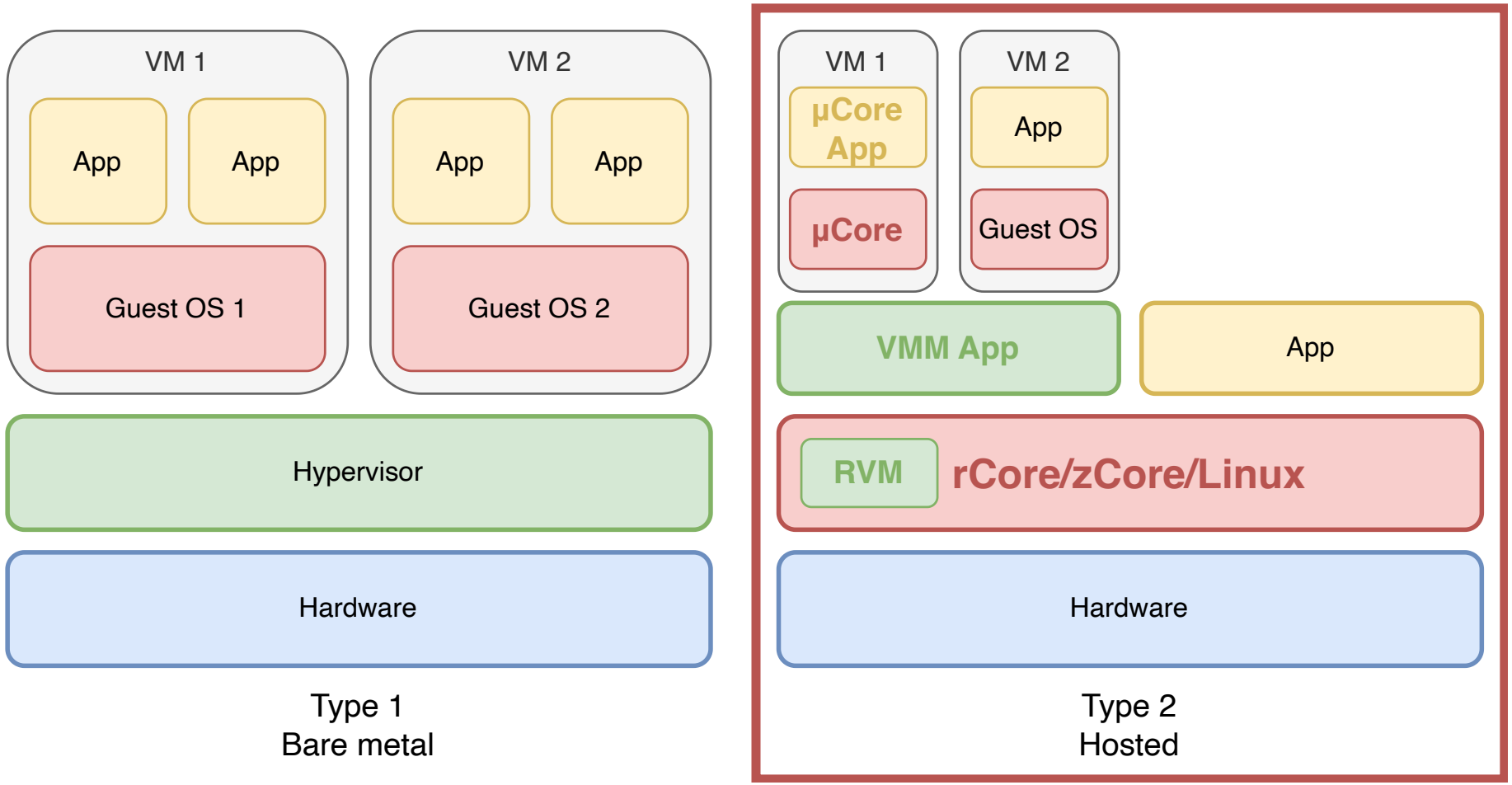
RVM 项目介绍

- RVM 是什么?
  - **R**ust/**R**core **V**irtual **M**achine
  - Rust 语言编写的实验与教学 hypervisor
- 发展历程
  - 第一代 (2020)：RVM
  - 第二代 (2021)：RVM1.5
  - 第三代 (2022)：RVM-Tutorial
  - ...

第一代 RVM

- https://github.com/rcore-os/RVM
- 参考 Zircon (Google Fuchsia) hypervisor 模块
- 平台：x86_64 (Intel)
- 运行模式 (**Type-2**)：
  - 在 host rCore 中运行 guest μCore
  - 在 host zCore 中运行，能通过 Zircon hypervisor 相关测例
  - 在 host Linux 中运行，作为一个 kernel module，代替 KVM 运行 guest μCore

第一代 RVM


VM 1
App
App
Guest OS 1
VM 2
App
App
Guest OS 2
Hypervisor
Hardware
Type 1
Bare metal
VM 1
µCore App
µCore
VM 2
App
Guest OS
VMM App
App
RVM
rCore/zCore/Linux
Hardware
Type 2
Hosted

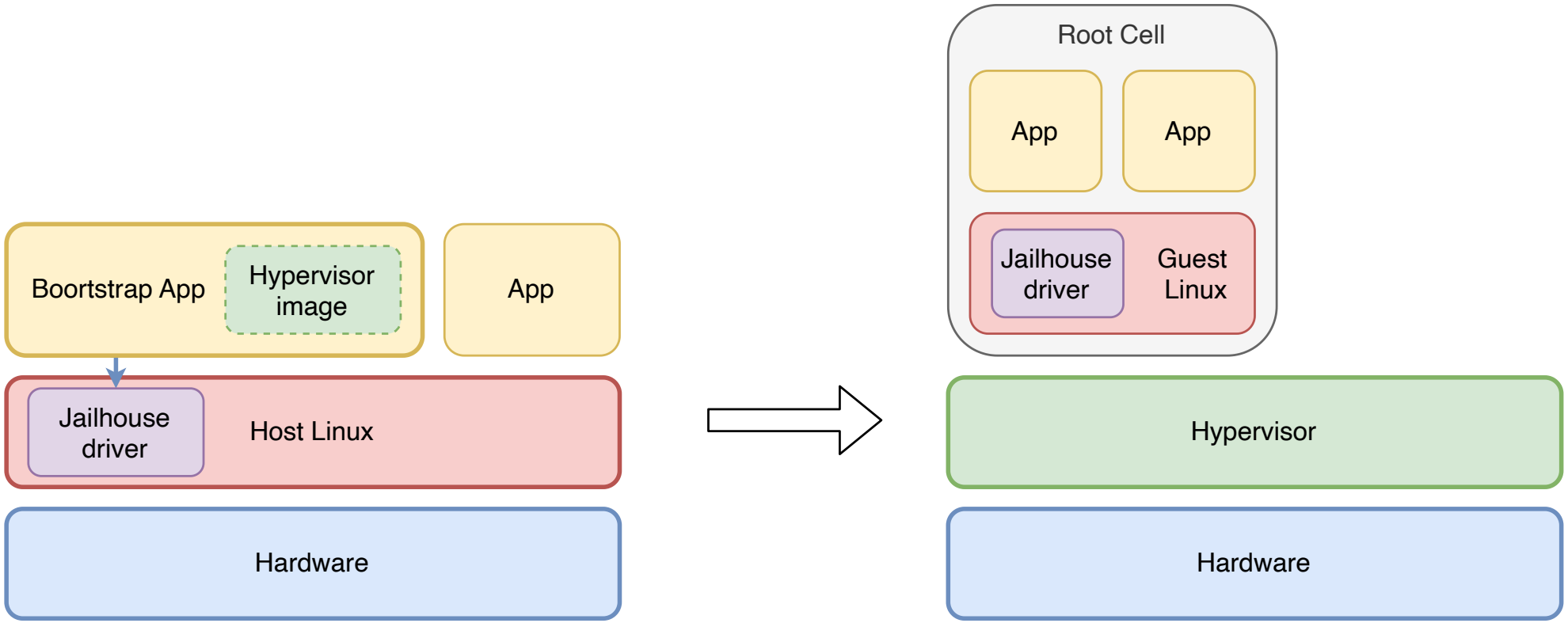
第一代 RVM

- 功能
  - 单核，单 vCPU，单 guest
  - 设备模拟：
    - 串口、IDE 磁盘、PIT 时钟
    - 转发给用户态 VMM 处理
- 不足：
  - 与 host OS 耦合较大
  - 不支持复杂 guest OS (如 Linux)
  - 初次尝试，代码结构有待优化

第二代：RVM1.5

- https://github.com/rcore-os/RVM1.5
- 参考 Jailhouse hypervisor
- 平台：x86_64 (Intel/AMD)
- 运行模式 (**Type-1.5**)：
  1. 先启动 host Linux
  2. 通过一个 kernel module 加载 hypervisor 并启动
  3. Linux 被**降权**为 guest 模式
  4. 返回 guest Linux 继续执行

第二代：RVM1.5



RVM 1.5

第二代：RVM1.5

- 功能：
  - 多核，多 vCPU，单 guest
  - 启动/关闭 hypervisor
- 特点：
  - **方便**：以 Type-2 方式加载，以 Type-1 方式运行
  - **轻量**：设备访问、调度全部交给 guest Linux 处理
  - **高效**：直通设备 I/O，几乎不会发生 VM exit
  - **安全**：hypervisor 拥有最高权限，监控 guest Linux 特权操作
  - **灵活**：利用 hypervisor 可实现附加功能

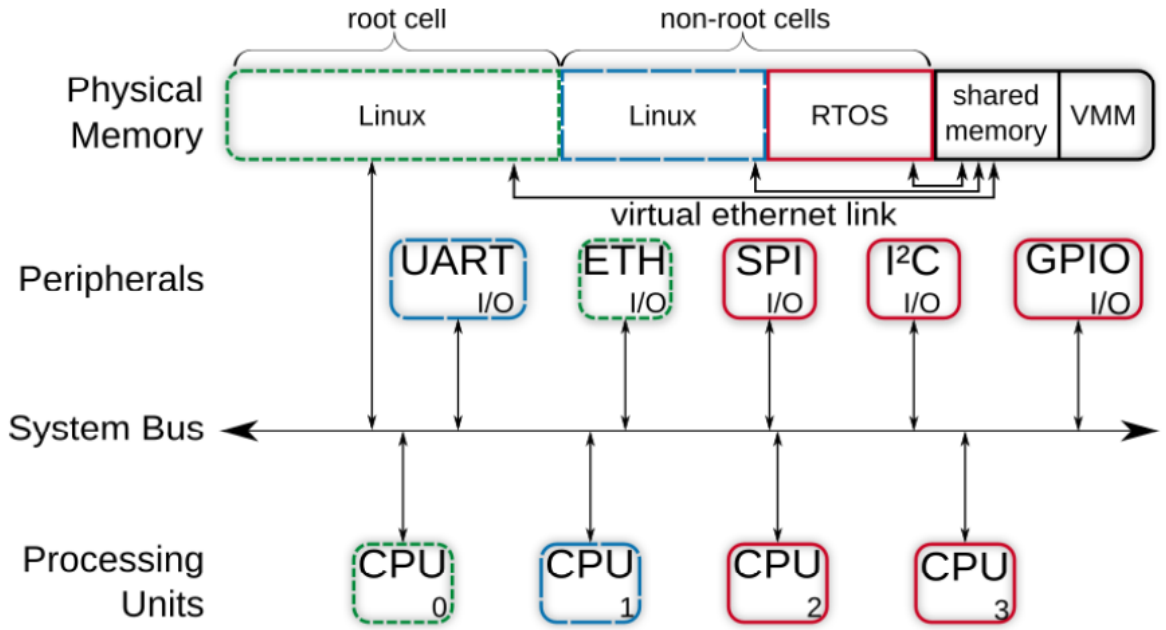
第二代：RVM1.5

研究成果

1. **RVM1.5 + 可信执行环境 (TEE)**
   - HyperEnclave: An Open and Cross-platform Trusted Execution Environment (USENIX ATC '22)
   - 用虚拟化模拟 TEE 原语，在非 Intel 平台安全运行 SGX 应用
2. **RVM1.5 + RTOS**
   - Linux 被降权后，将部分 CPU 分给 RTOS
   - 复杂 syscall 转发给 Linux 处理
   - 普通应用不干扰实时应用运行

第二代：RVM1.5

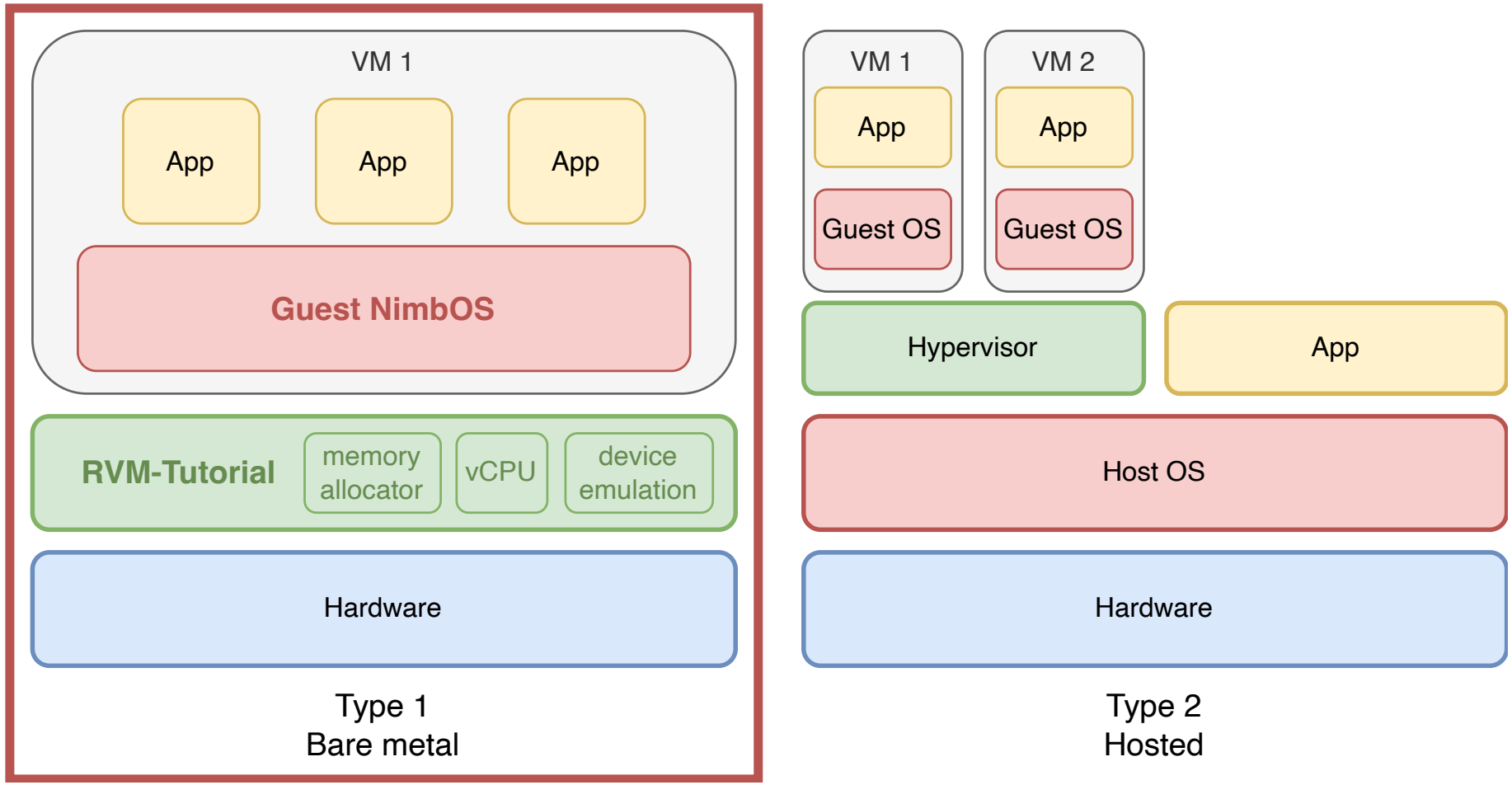
- 不足：
  - 目前仅支持单 guest (root cell)
  - 设备都是直通，无法分给多 guest



第三代：RVM-Tutorial

- https://github.com/equation314/RVM-Tutorial
- 平台：x86_64 (Intel)
- 特点：
  - 对前代 RVM 的整理与优化
  - 代码精简，实现简单，面向教学 (6 个 step)
- 运行模式 (**Type-1**)：
  - 无需完整 host OS，可直接在 baremetal 环境运行
  - 仅依赖少量 OS 服务 (如内存分配)

第三代：RVM-Tutorial



第三代：RVM-Tutorial

- 功能：
  - Guest OS: NimbOS
  - 单核，单 vCPU，单 guest
  - 简单的设备模拟：
    - 串口 I/O
    - APIC 时钟

总结与展望

- 移植：
  1. 将 RVM1.5/RVM-Tutorial 移植到 ARM/RISC-V 平台
- 功能扩展：
  2. 为 RVM1.5 添加多 guest 支持
  3. 为 RVM-Tutorial 添加多核、多 vCPU、多 guest 支持
  4. 在 RVM-Tutorial 上运行 guest Linux
- 模块化：
  5. 对 RVM-Tutorial 的模块化改造
  6. 将 RVM-Tutorial 做成 Linux Kernel Module (*兼容 KVM API 以运行 QEMU*)
- 嵌套虚拟化：
  7. NestedRVM：在 host RVM 上运行 guest RVM，再运行 guest OS

Thanks!