基于openEuler Embedded的麒麟信安嵌入式操作系统实践

湖南麒麟信安科技股份有限公司

邱文博

目录

麒麟信安嵌入式操作系统简介

- 公司简介
- 产品介绍
- 电力行业应用

麒麟信安®
KYLINSEC

湖南麒麟信安科技股份有限公司（简称“麒麟信安”）成立于2007年，是国内第一批从事操作系统产品研发和市场推广的企业。公司成立以来专注于国家关键信息基础设施领域相关技术的研发与应用，主要从事操作系统产品研发及技术服务，并以操作系统为根技术创新发展信息安全和云计算等产品及服务业务，致力于推进国产化安全应用。

重点行业领先的**国产操作系统**提供商

关键领域国产**云桌面**系统领先者

统一**数据安全加密防护系统**提供者

高可信操作系统国家地方联合工程研究中心

国家级“专精特新”小巨人企业

国家科技重大专项“核高基”承研单位

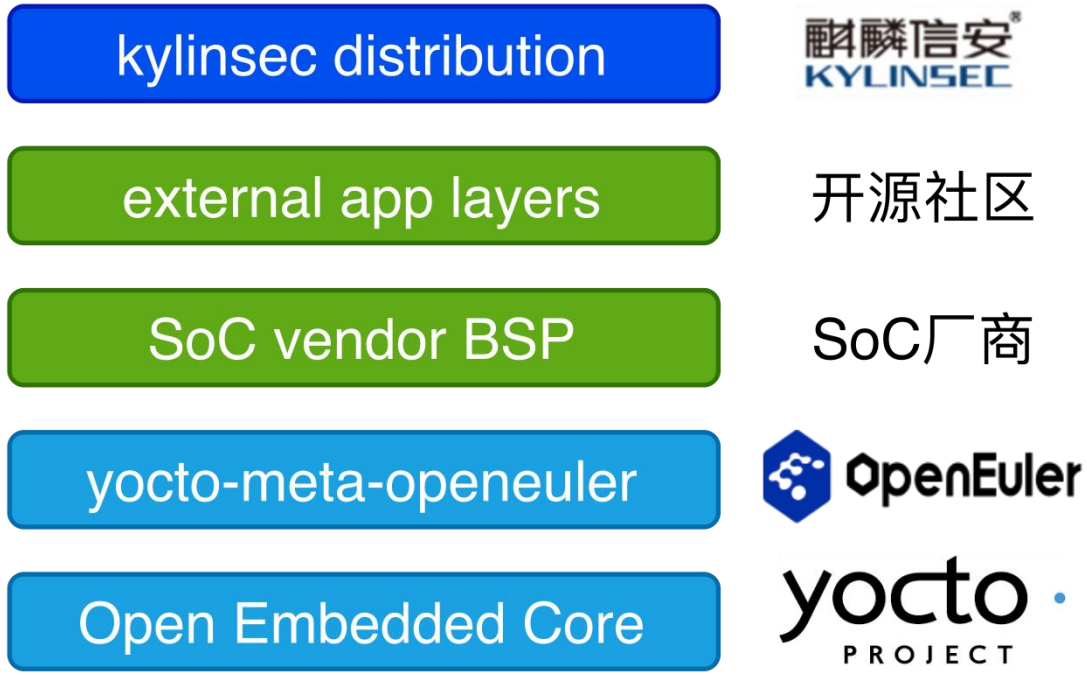
国家规划布局内重点软件企业

麒麟信安嵌入式操作系统

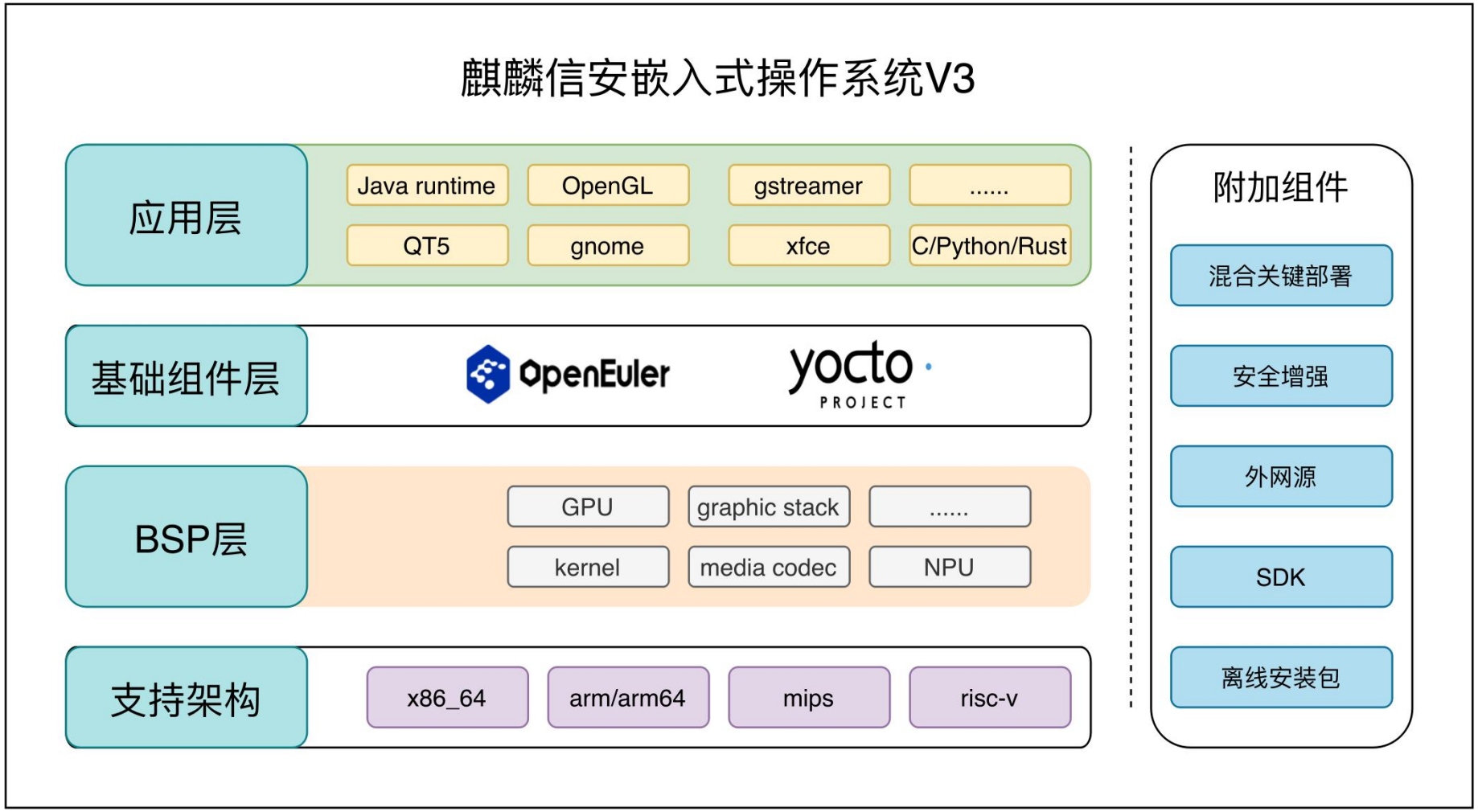
面向国防、电力和金融等重要行业工业控制系统，具备安全性、实时性和可靠性，通过内核安全增强、实时虚拟化、菜单式定制、链路级网络冗余等技术，有效保障工业控制领域信息系统安全运行。

- 基于openEuler Embedded的商用嵌入式发行版
- 原先基于Yocto社区，新版本基于欧拉社区
- 欧拉社区版与openEuler Embedded 23.09同步发布

- 提供商业定制服务与技术支持
- 适配更多板卡
- 提供额外安全增强模块
- 提供外网源、SDK



系统架构图


麒麟信安嵌入式操作系统V3
应用层
Java runtime
OpenGL
gstreamer
.....
QT5
gnome
xfce
C/Python/Rust
基础组件层
OpenEuler
yocto
PROJECT
BSP层
GPU
graphic stack
.....
kernel
media codec
NPU
支持架构
x86_64
arm/arm64
mips
risc-v
附加组件
混合关键部署
安全增强
外网源
SDK
离线安装包

电力行业应用

麒麟信安嵌入式操作系统V3已在电力智能监控系统便携式运维网关中大规模应用，提高了电力系统的运维效率和可靠性，为电力行业的发展和安全提供了有力支持。

- 多种通信协议和接口，使便携式运维网关能够与各种电力设备和系统进行无缝连接和数据交换。
- 通过与电力设备的通信，实时采集和监测电力数据，为电力监控和故障诊断提供准确的数据支持。
- 支持远程控制和配置，使运维人员能够远程访问和管理电力设备，提高运维效率和响应速度。

在混合关键性系统（MCS）方面的拓展

- 混合关键性系统（MCS）简介
- 基于openAMP的MCS系统
- 基于Jailhouse的MCS系统

混合关键性系统（MCS）

OpenEuler

现状分析

现状：管理能力、丰富生态、高实时、高可靠、高安全要求不能同时满足。

传统方案：采用一颗性能较强的处理器运行Linux负责富功能，一颗微控制器/DSP/实时处理器运行实时操作系统负责实时控制或者信号处理。两者之间通过I/O、网络或片外总线的形式通信。

缺陷问题：集成度不高、通信速度受限、灵活性差、可维护性成本高。

MCS系统

定义：（混合）运行着不同关键性应用（任务）的软硬件系统。

关键性：系统异常是否会导致严重后果

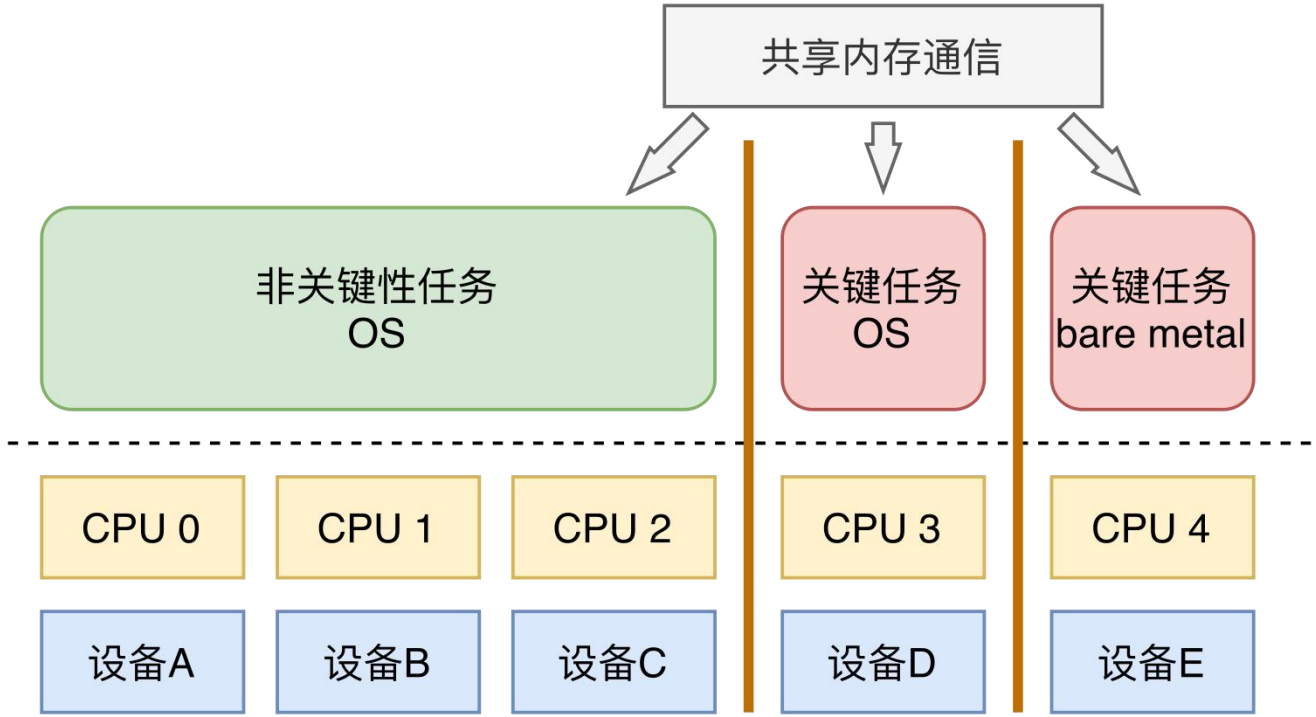
动机：

- 运行关键性任务的系统需要漫长的测试验证其安全性
- 关键性任务往往存在非关键性的部分
- 拆分运行任务的关键性与非关键性部分可以简化关键性任务的设计与验证流程
- 减少系统整体使用的组件

常见应用：飞控， 车机（自动驾驶，车载音响）

基于共享内存的MCS系统

基于共享内存的MCS系统通过利用共享内存方式实现多任务协作，具有高性能、低延迟特点。



缺陷和限制

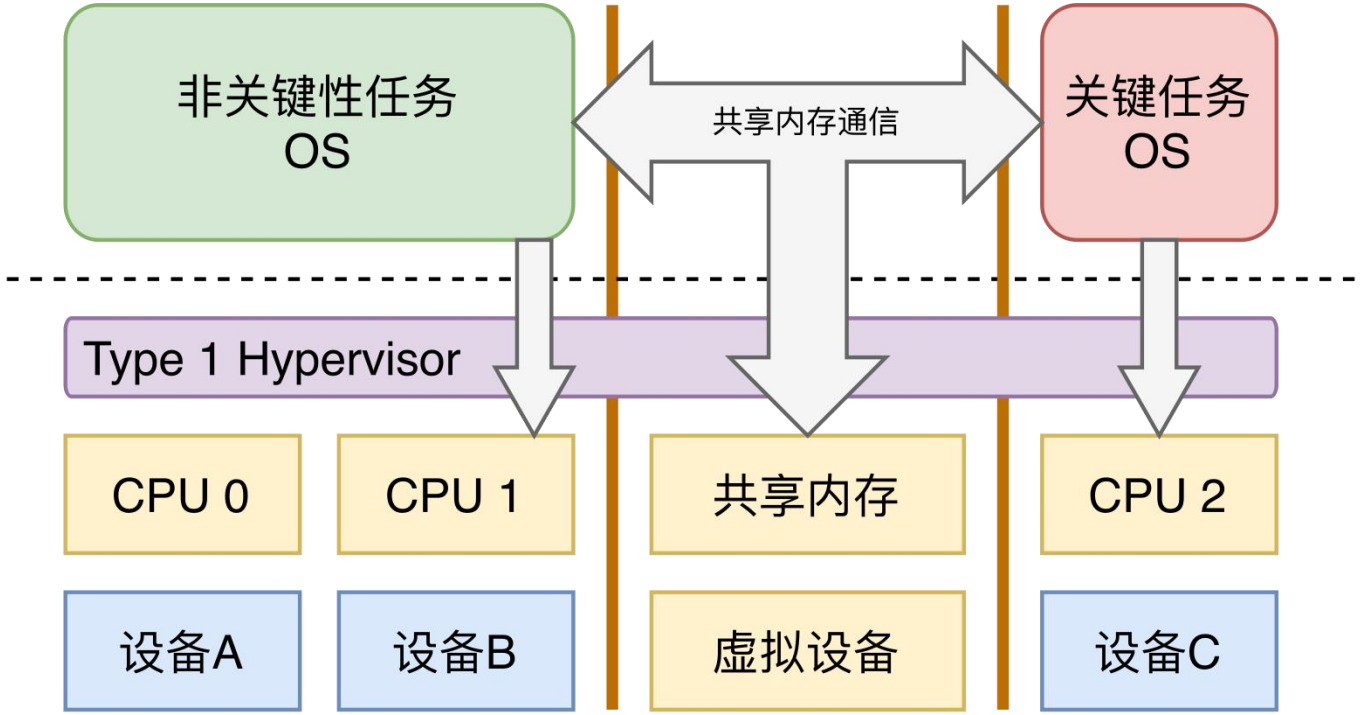
- **资源访问不受限制：**无法进行隔离访问的隔离控制。
- **系统之间不独立：**单个系统的程序异常可能导致整个系统出现无法恢复的异常。

基于分区虚拟化的MCS系统

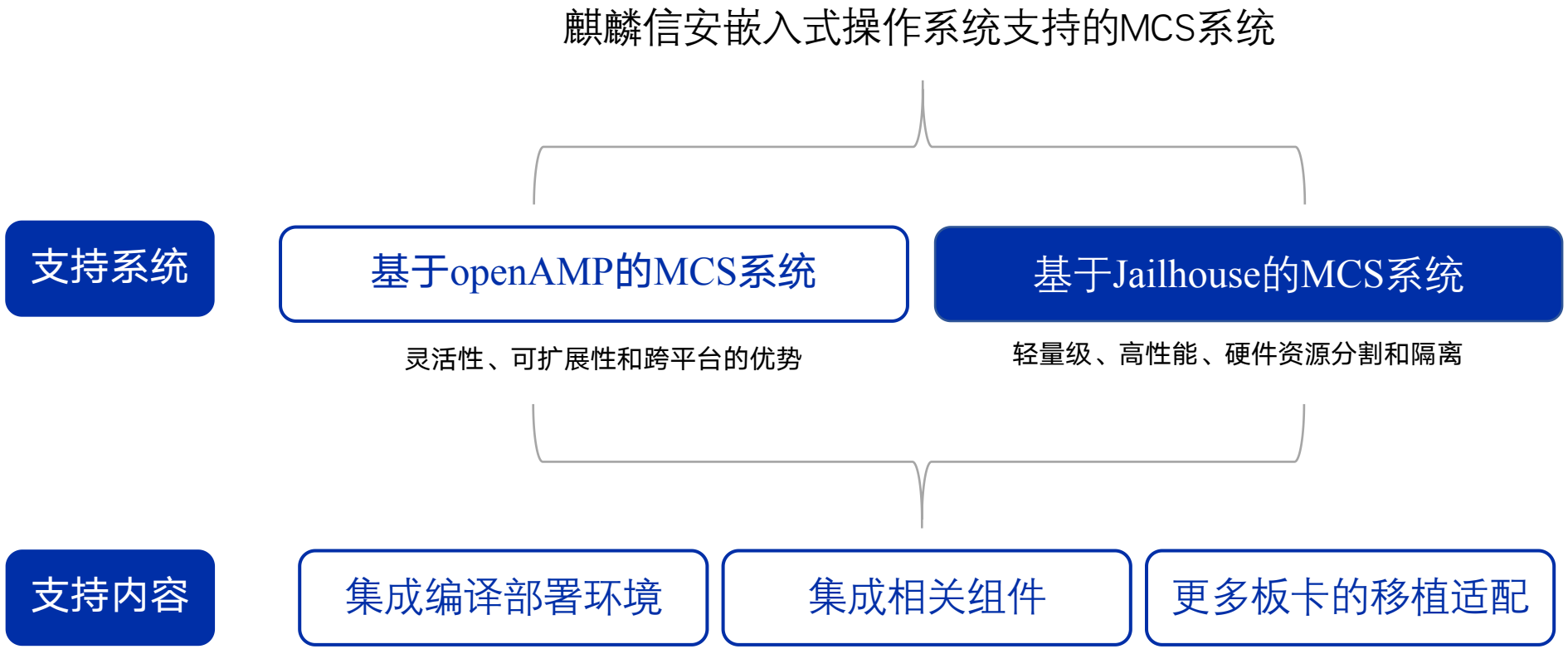
基于分区虚拟化的MCS系统通过分区虚拟化技术实现资源隔离，具有高效、可靠的特点。

缺陷和限制

- **资源浪费：**分区虚拟化对资源进行划分，没有调度机制，容易导致资源的浪费。
- **降低性能：**虚拟化开销可能会降低系统性能。对于实时性要求高的应用，虚拟化带来的延迟可能会影响系统响应能力。
- **运维成本高：**需要额外的管理和维护成本。



对MCS系统的支持


麒麟信安嵌入式操作系统支持的MCS系统
支持系统
基于openAMP的MCS系统
灵活性、可扩展性和跨平台的优势
基于Jailhouse的MCS系统
轻量级、高性能、硬件资源分割和隔离
支持内容
集成编译部署环境
集成相关组件
更多板卡的移植适配

openAMP简介

【术语解释】

- AMP：Asymmetric Multi-Processing，异构多处理器
- 典型例子：4个ARM A核 + 1个ARM M核心

【定义】

- openAMP是一整套实现异构多处理器计算系统的通用框架
- 基于上游Linux社区的多种组件，如RemoteProc、RPMsg、Virtio等

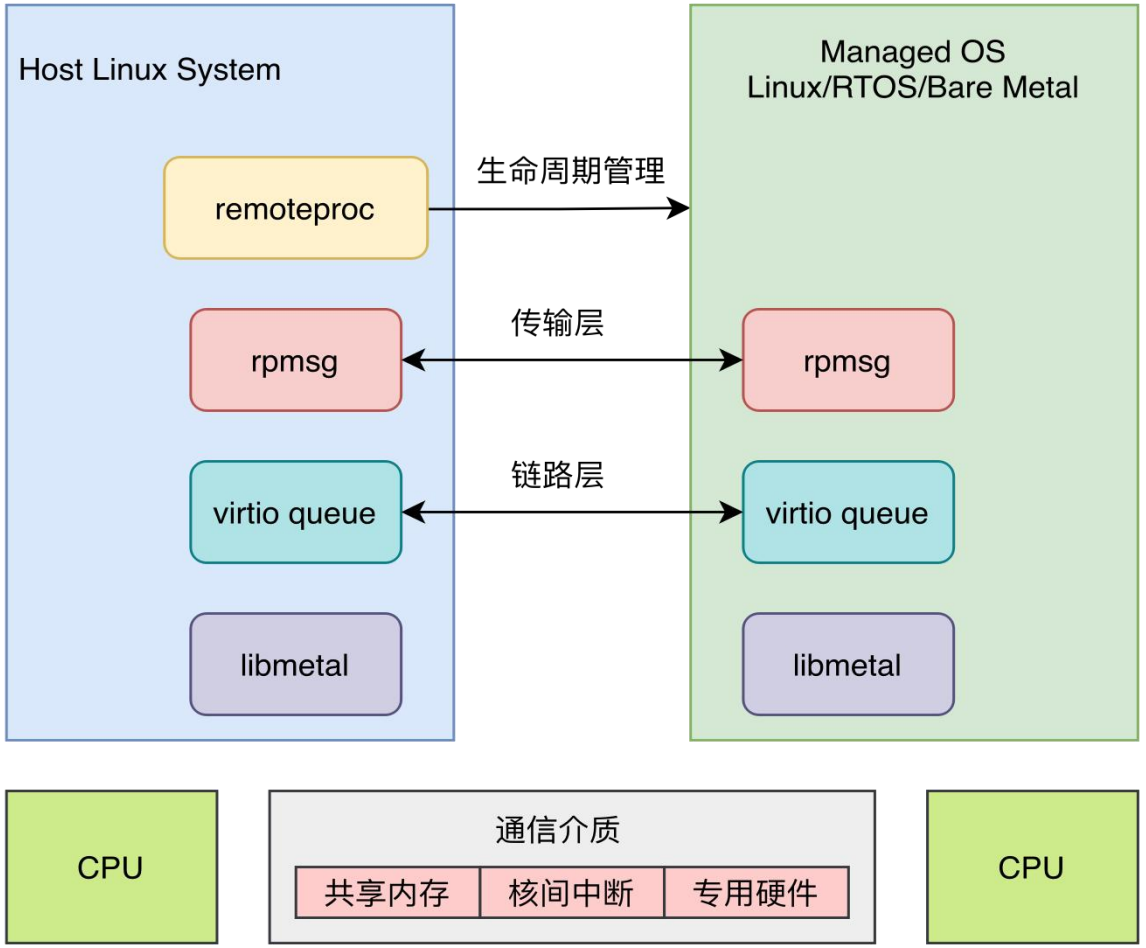
【功能介绍】

- 提供对远程核心的生命周期管理和核间通信框架
- 提供独立的基础库，协助RTOS核BareMetal应用程序的开发
- 保持与上游Linux社区提供的RemoteProc核RPMsg组件的兼容性
- 提供实现管理远程核心上执行的printf、scanf、open、close、read和write等调用的基础设施

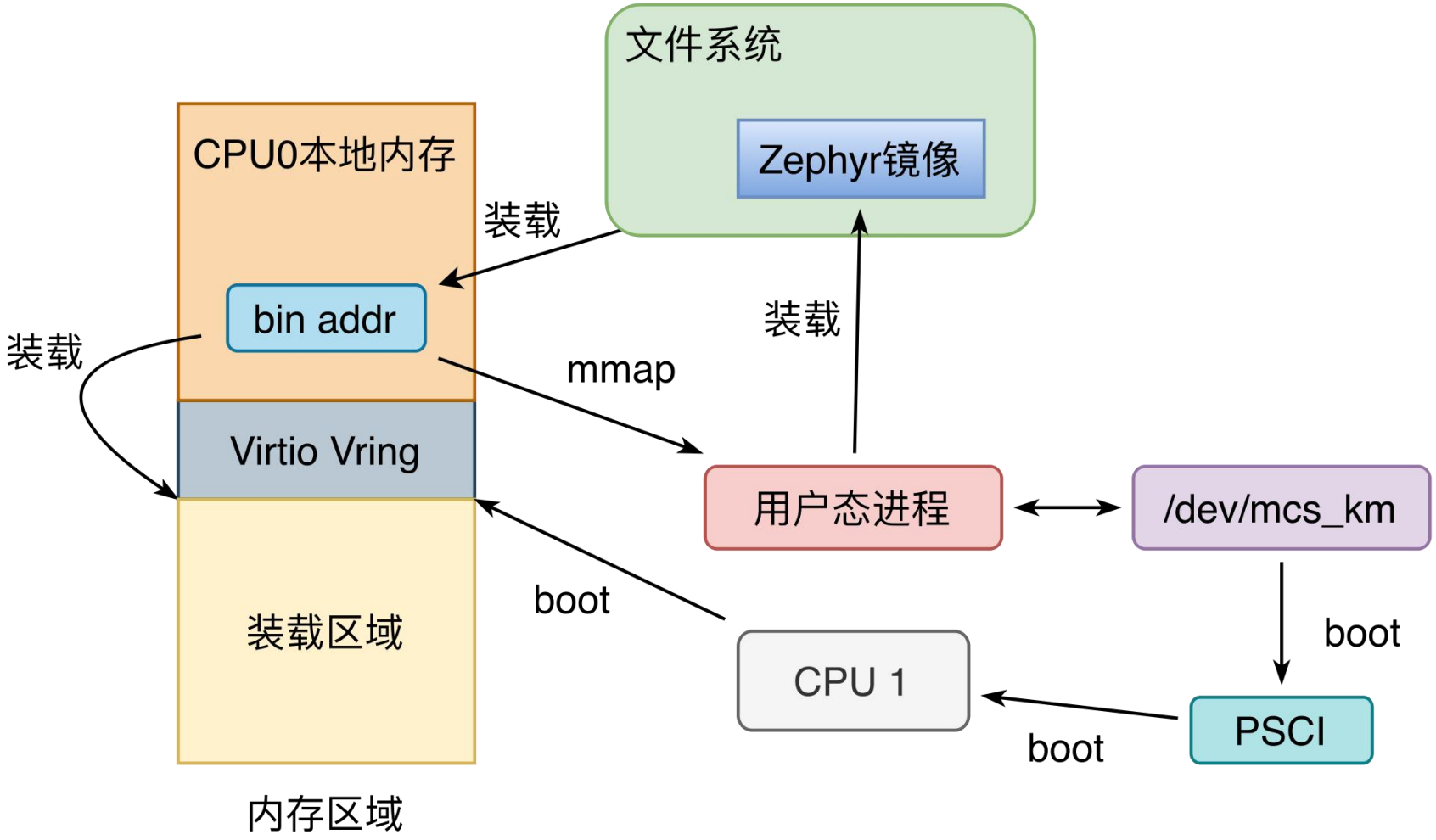
基于openAMP的MCS架构

基于openAMP的MCS架构具有灵活性、可扩展性和跨平台的优势，实现了多核协同工作、资源共享和通信协议的统一管理。

- 将openAMP框架应用到SMP系统中。
- 使用openAMP框架作为基础对于Linux+RTOS场景完成的实例化应用，包括了Linux侧和RTOS侧全流程的功能。
- 通过MMap映射并装载RTOS镜像，通过PSCI启动RTOS
- 使用openAMP协议框架，Linux 和RTOS之间将共享内存和IPI中断作为数据传输通路，在共享内存和IPI中断基础上抽象出Virtio的Vring作为链路层。
- 使用RPMsg消息格式，抽象出RPMsg Endpoint端点用于区分不同的数据服务。

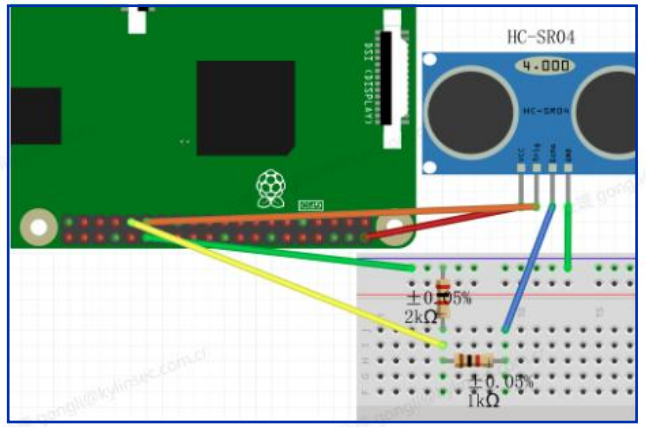


第二系统镜像装载执行流程



应用场景Demo

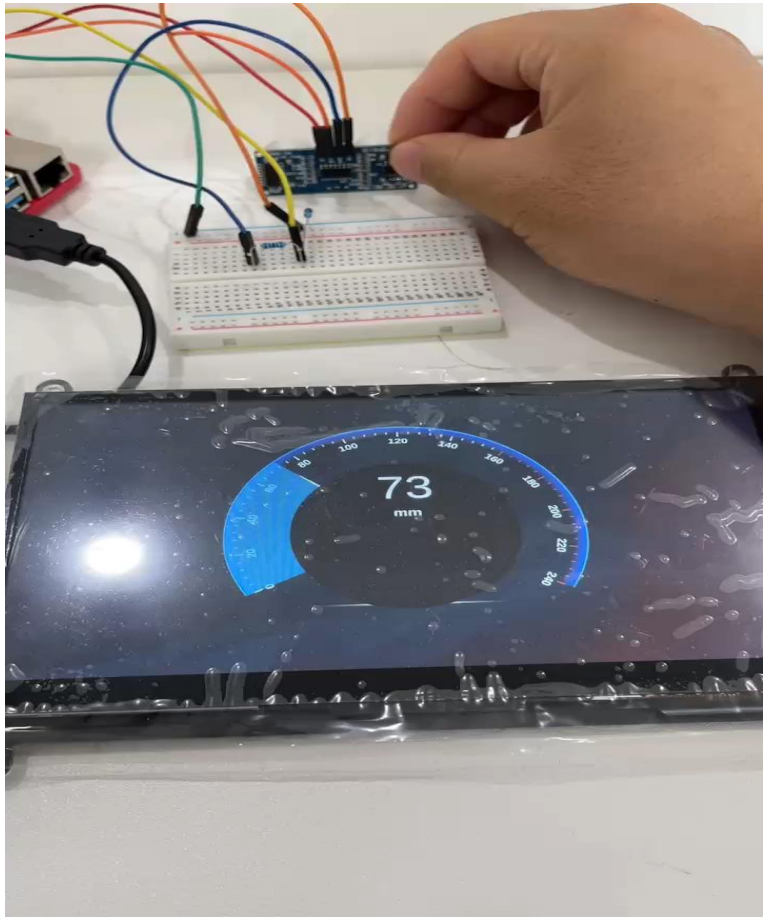
目标：实现一个基础MCS系统，其中，数据采集通过运行在独立核心上的Zephyr RTOS系统进行，数据的展示通过Host端的Linux系统上的应用程序完成。Demo系统实现了简单的传感器超生波测距。



主要组件：

- dashboard-gui：基于QT EGLFS的图形应用，独立线程接收并展示Zephyr系统传输来的数据及展示。
- mcs-km：内核驱动模块，负责提供基础设施和Zephyr系统生命周期管理。
- rpmsg_main：用户态应用，负责与内核模块通信，装载Zephyr系统镜像，并控制其生命周期。
- zephyr-image.bin：RTOS系统镜像，持续采集HC-SR04传感器的测量数据，简单处理后通过共享内存发送到Host Linux系统。

Demo运行演示



RTOS在做一些高可靠、高安全的控制算法时需要实时获取传感器数据作为输入并执行输出，但是结果数据人机交互并不需要太高的实时性。

RTOS本身不具备复杂的图形显示框架，Linux可以代理RTOS图形显示需求，RTOS本身完成传感器数据采集和控制。

延迟指标测试结果-树莓派4 1.5G - Zephyr

OpenEuler

测试项	指标（ns）	测试项	指标（ns）
线程上下文切换时间(实时线程)	197	信号量释放平均时间	190
线程上下文切换时间(协作线程)	194	信号量获取平均时间	88
中断切换到中断前的线程上下文	129	信号量获取（包含上下文切换）	351
中断切换到新的线程上下文	518	信号量释放（包含上下文切换）	555
创建线程（不启动）	7037	获取锁平均时间	92
启动线程	555	释放锁平均时间	6
挂起线程	407	内存申请平均时间	344
恢复线程	388	内存释放平均时间	165
中止线程	111		

基于Jailhouse的MCS系统

- Jailhouse的工作原理
- Jailhouse在Aarch64架构下的实现

Jailhouse项目简介

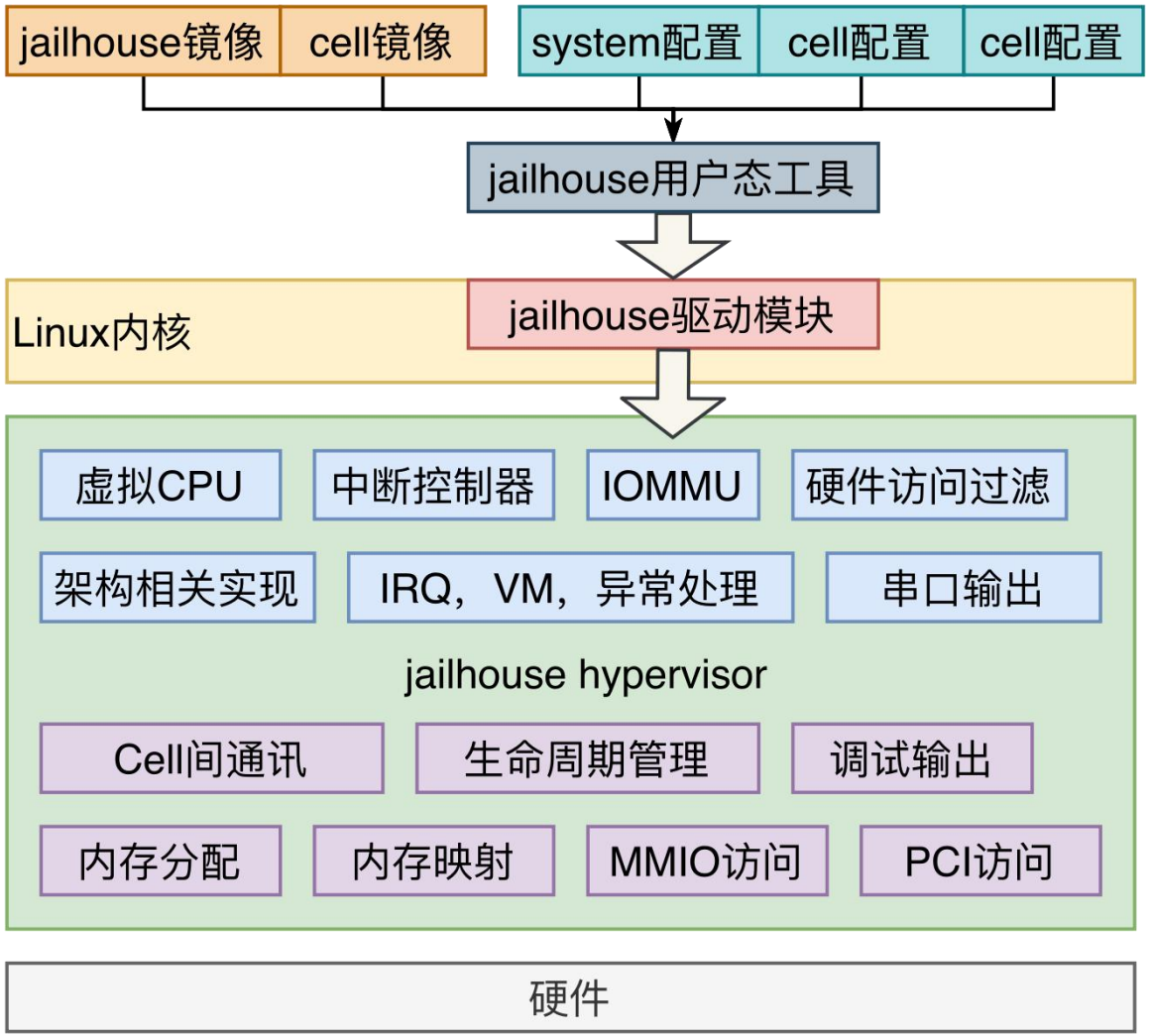
一个开源的虚拟化监狱系统，提供轻量级、高性能的硬件资源分割和隔离，适用于嵌入式和实时应用场景。

说明
基于分区虚拟化的MCS系统实现
静态分区虚拟化
无资源调度器实现
1比1资源分配
尽量不进行设备模拟
运行时最小化
硬实时支持，额外开销最小化
考虑隔离安全性，支持IOMMU等硬件特性

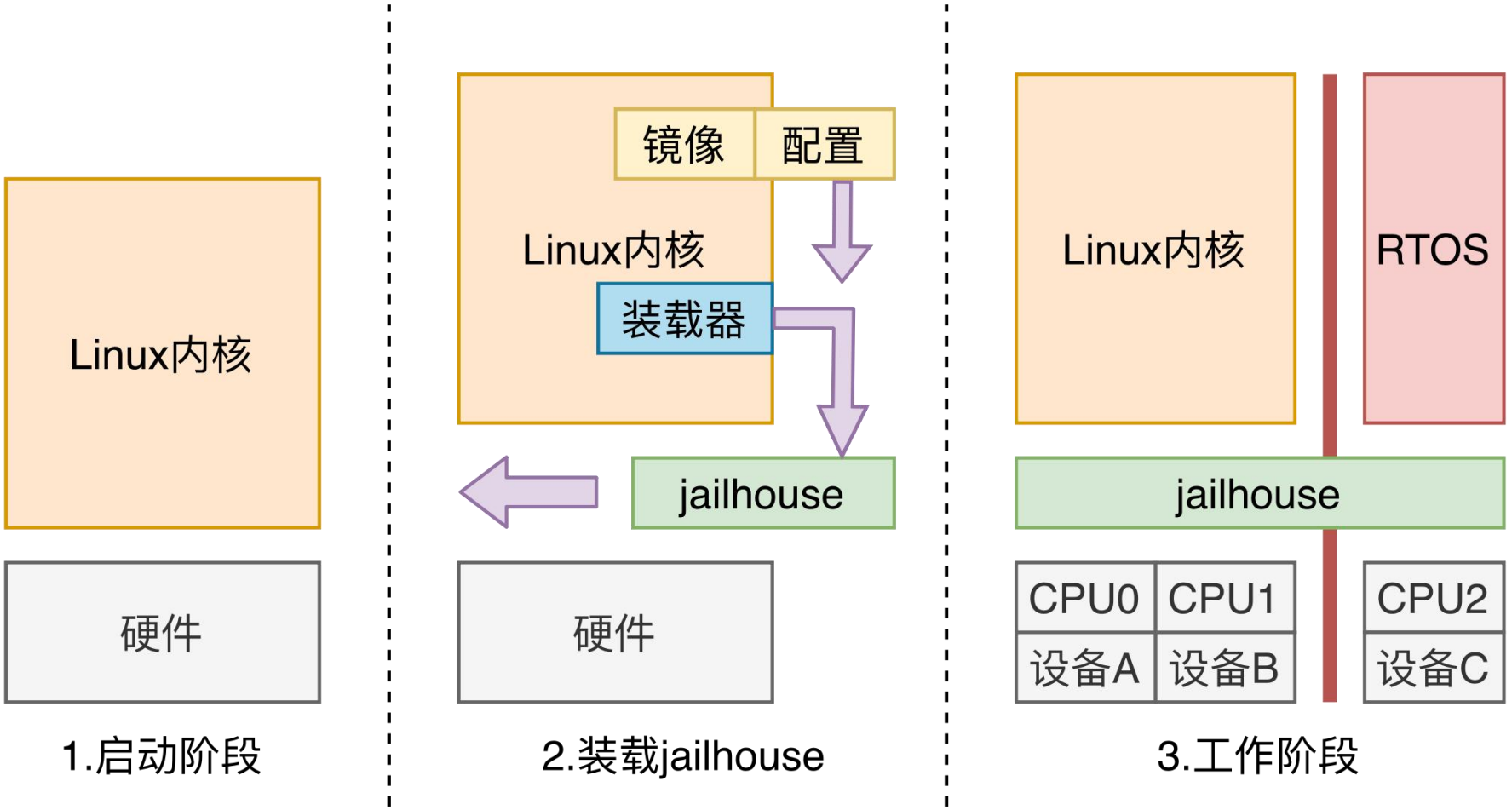
Jailhouse核心概念

核心概念	说明
hypervisor	虚拟机管理器，运行在操作系统之下。
hypervisor mode	CPU运行Hypervisor的特权级别。
guest mode	CPU运行OS或者特定应用的特权级别。
cell	Jailhouse对系统资源的一个划分。
root cell	运行着装载启动Jailhouse的Linux系统的Cell。
inmate	Cell上运行的OS或者Bare Metal程序。

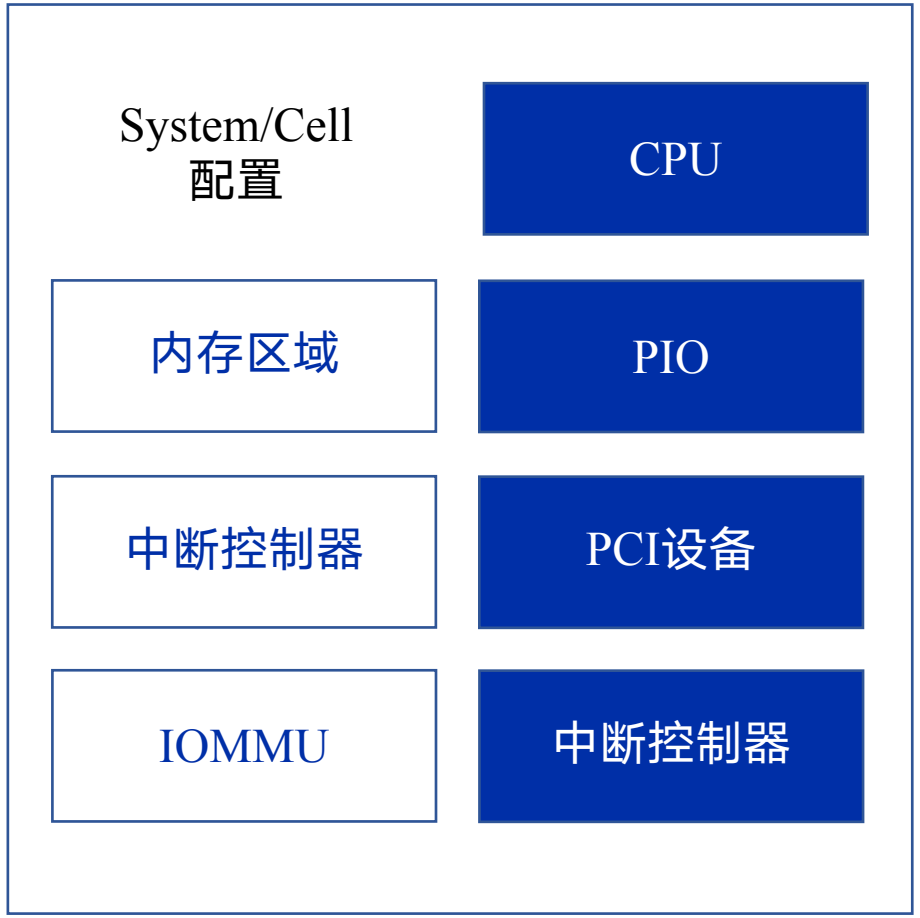
Jailhouse软件架构



Jailhouse装载流程



系统/Cell配置



目的：允许Hypervisor与Driver协作，将设备控制权转移给Hypervisor，使其可以将硬件资源划分并分配给其他的Cell。

实现方式：

- 二进制格式。
- 由.c文件通过GCC与Objcopy生成。
- 记录Hypervisor运行时所需配置。
- 记录系统硬件信息。
- 记录Cell所拥有的硬件。

系统/Cell配置

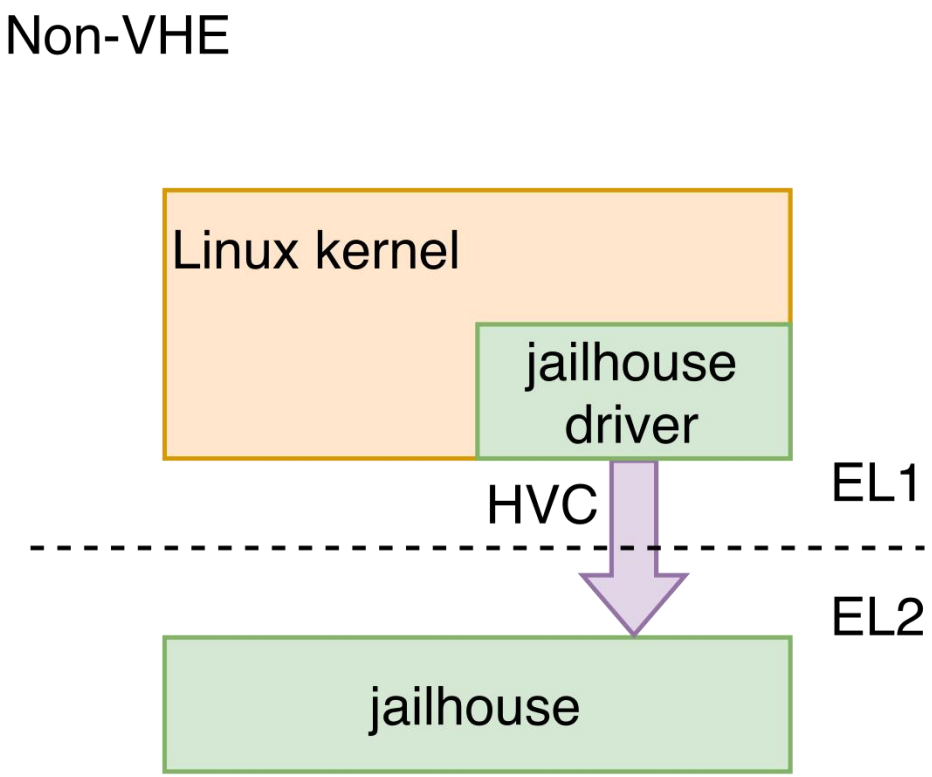
OpenEuler

```
struct {
    struct jailhouse_system header;
    __u64 cpus[1];
    struct jailhouse_memory mem_regions[14];
    struct jailhouse_irqchip irqchips[2];
    struct jailhouse_pci_device pci_devices[2];
} __attribute__((packed)) config = {
    .header = {
        .signature = JAILHOUSE_SYSTEM_SIGNATURE,
        .revision = JAILHOUSE_CONFIG_REVISION,
        .architecture = JAILHOUSE_ARM64,
        .flags = JAILHOUSE_SYS_VIRTUAL_DEBUG_CONSOLE,
        .hypervisor_memory = {
            .phys_start = 0x1fc00000,
            .size       = 0x00400000,
        },
        .debug_console = {
            .address = 0xfe215040,
            .size = 0x40,
            .type = JAILHOUSE_CON_TYPE_8250,
            .flags = JAILHOUSE_CON_ACCESS_MMIO |
                     JAILHOUSE_CON_REGDIST_4,
        },
```

```
    .cpus = {
        0b1111,
    },

    .mem_regions = {
        /* IVSHMEM shared memory regions for 00:00.0 (demo) */
        {
            .phys_start = 0x1faf0000,
            .virt_start = 0x1faf0000,
            .size = 0x1000,
            .flags = JAILHOUSE_MEM_READ,
        },
        {
            .phys_start = 0x1faf1000,
            .virt_start = 0x1faf1000,
            .size = 0x9000,
            .flags = JAILHOUSE_MEM_READ | JAILHOUSE_MEM_WRITE,
        },
        {
            .phys_start = 0x1fafa000,
            .virt_start = 0x1fafa000,
            .size = 0x2000,
            .flags = JAILHOUSE_MEM_READ | JAILHOUSE_MEM_WRITE,
        },
```

Linux内核调用Hypercall



debug console putc

cpu get info

disable

cell destroy

cell create

cell get state

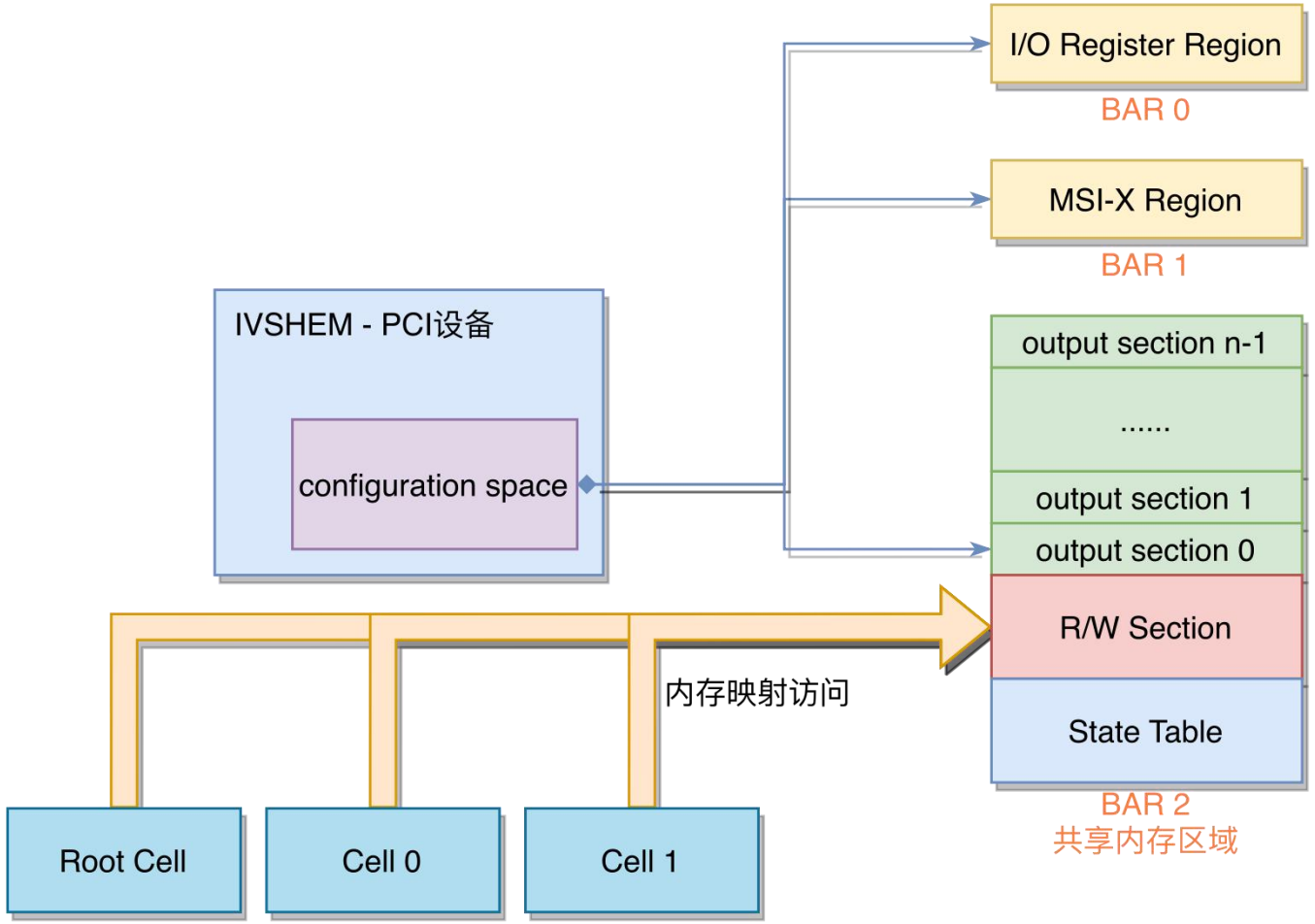
cell set loadable

hypervisor get info

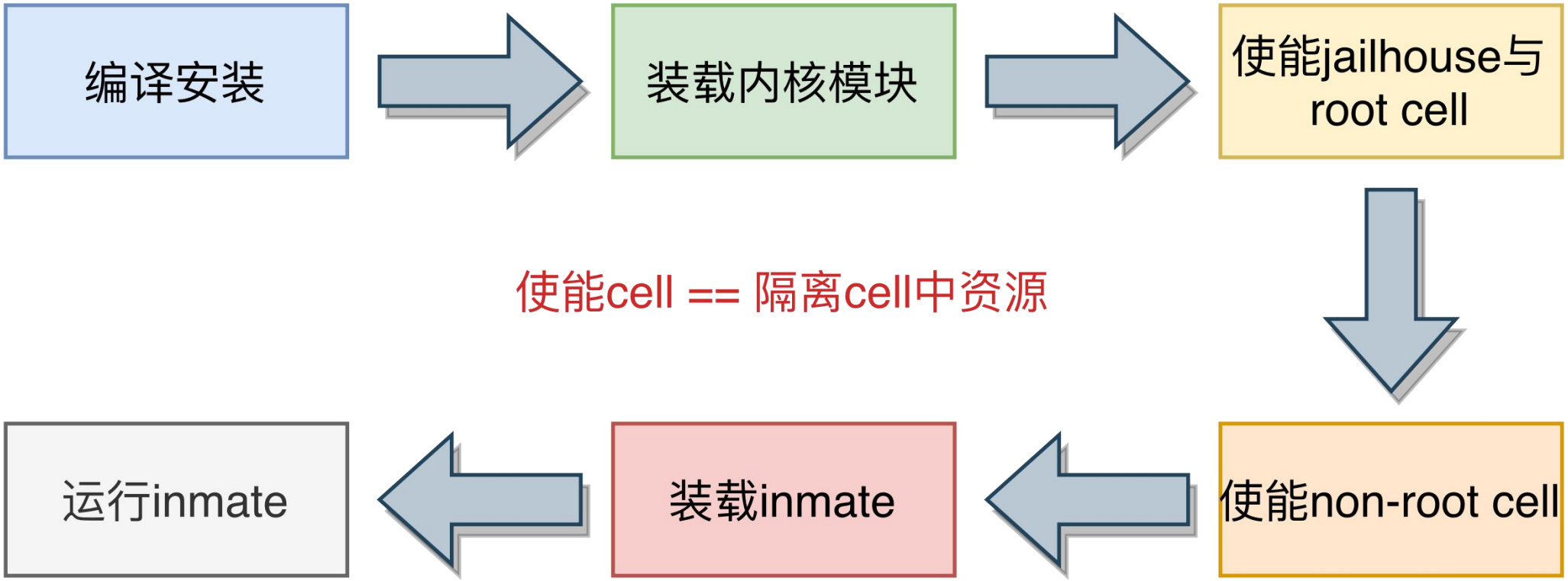
cell start

IVSHEM设备共享内存通信

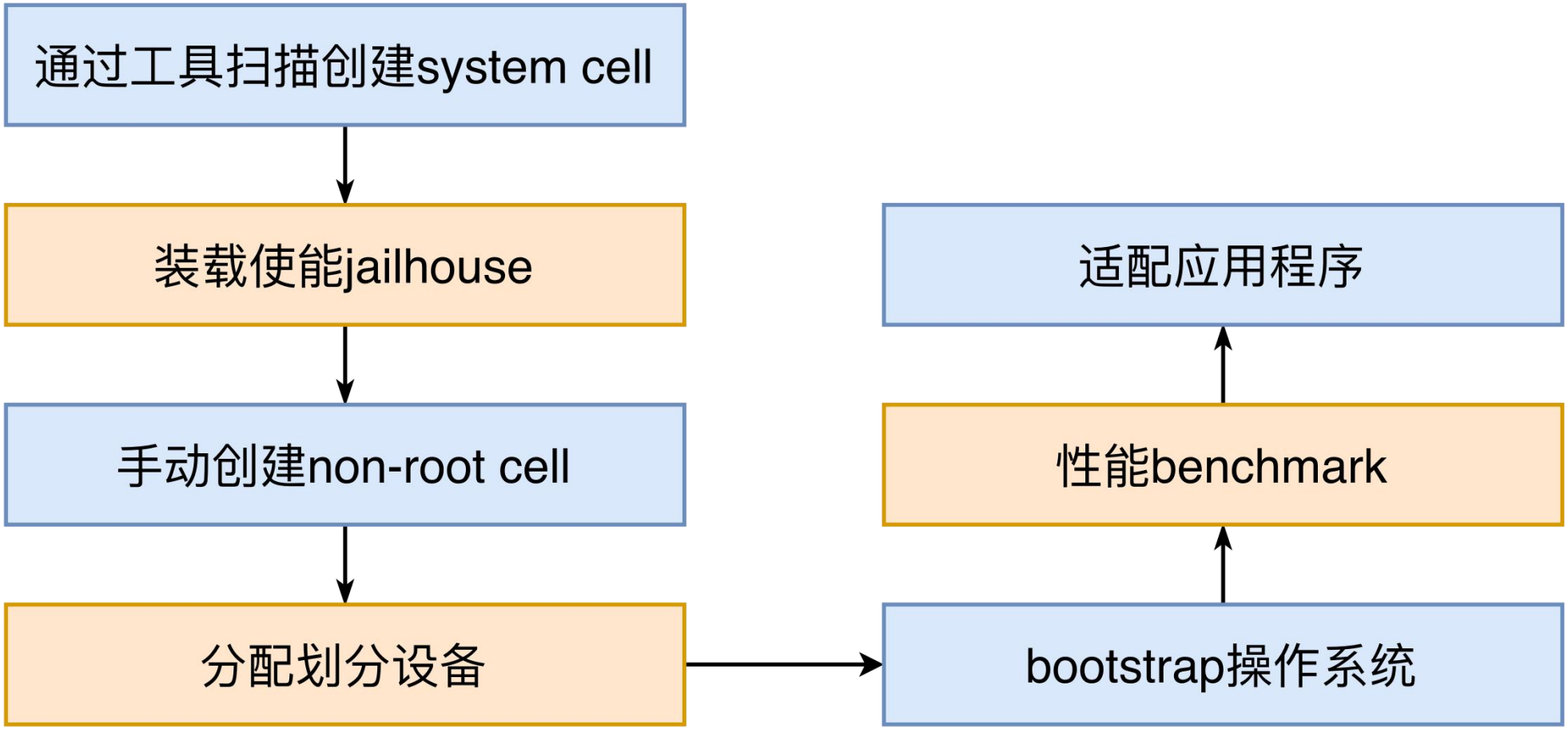
- 通过Cell配置构建共享内存设备
- Jailhouse控制设备映射与访问
- 支持多Peer通信
- 支持中断式通知机制
- 支持Ethernet通信协议
- 支持Virtio通信协议
- 提供框架自行实现设备



Jailhouse部署流程



Jailhouse适配流程



IVSHEM网络通信Demo

OpenEuler

```
root@demo:~# Cell "qemu-arm64-zephyr-demo" can be loaded
Started cell "qemu-arm64-zephyr-demo"
[00:00:00.000,000] <inf> ivshmem: PCIe: ID 0x4106110A, BDF 0x800, class-rev 0xFF000100
[00:00:00.000,000] <inf> ivshmem: MSI-X bar present: no
[00:00:00.000,000] <inf> ivshmem: SHMEM bar present: no
[00:00:00.000,000] <inf> ivshmem: State table size 0x1000
[00:00:00.000,000] <inf> ivshmem: RW section size 0x0
[00:00:00.000,000] <inf> ivshmem: Output section size 0x7F000
[00:00:00.000,000] <inf> ivshmem: Enabling INTx IRQ 141 (pin 2)
[00:00:00.000,000] <inf> ivshmem: ivshmem configured:
[00:00:00.010,000] <inf> ivshmem: - Registers at 0x10001000 (mapped to 0xEFEFE000)
[00:00:00.010,000] <inf> ivshmem: - Shared memory of 0xFF000 bytes at 0x7FB00000 (mapped to 0xEFDFF000)
[00:00:00.010,000] <inf> eth_ivshmem: ivshmem: id 1, max_peers 2
[00:00:00.010,000] <inf> eth_ivshmem: shmem queue: desc len 0x800, header size 0xD080, data size 0x71F80
[00:00:00.010,000] <inf> eth_ivshmem: MAC Address 22:87:85:4F:09:70
*** Booting Zephyr OS build zephyr-v3.4.0-3231-g76e4cd9dc40d ***


uart:~$ net ping 192.168.19.1
PING 192.168.19.1
28 bytes from 192.168.19.1 to 192.168.19.2: icmp_seq=1 ttl=64 time=3.71 ms
28 bytes from 192.168.19.1 to 192.168.19.2: icmp_seq=2 ttl=64 time=1.83 ms
28 bytes from 192.168.19.1 to 192.168.19.2: icmp_seq=3 ttl=64 time=0.92 ms
uart:~$
```


OpenEuler