

# 智能系统中的嵌入式多核操作系统

*Embedded Multi-core Operating System for Intelligent Systems*

ALLAN HE/何小庆

Allan@esbf.org

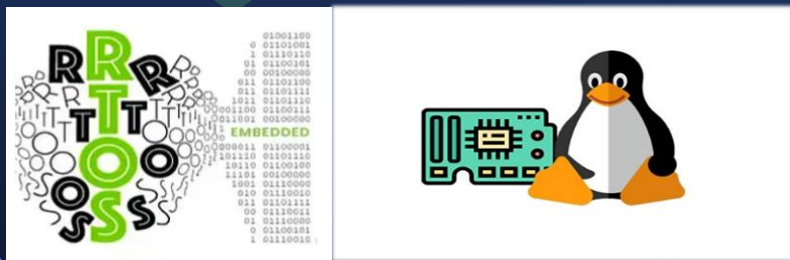
# 报告内容

- 引言:背景、现状、应用
- 多核芯片软硬件架构的概述
- 多核操作系统的虚拟化技术
- 嵌入式多核操作系统实例
- 嵌入式多核操作系统应用展望

# 第1:引言 产业发展大背景

## 万物智联的高性能嵌入式应用

- 嵌入式设备性能提升 (MCU->SoC、ARM64 /RISC-V)
- 嵌入式处理器芯片 (单核-同构多核-异构多核)
- 嵌入式系统功能升级 (实时控制+富功能应用)
- 嵌入式系统开发度增加 (功能安全+信息安全)

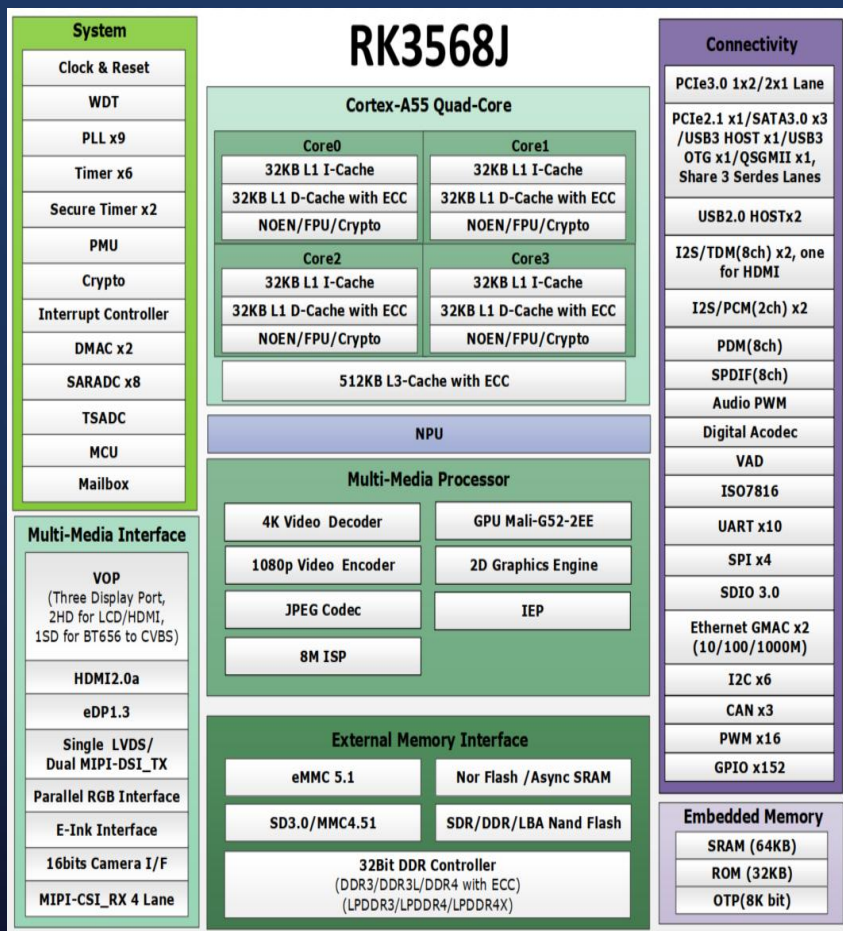


面向万物智联场景的新一代嵌入式混合、关键、分布式系统

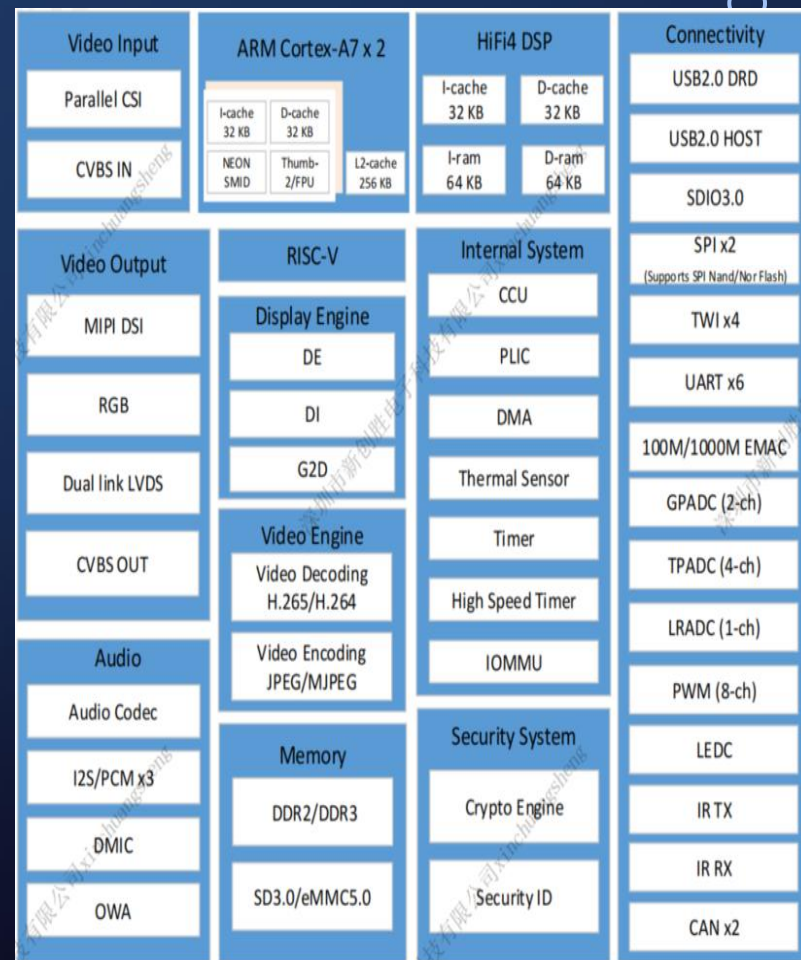
# 嵌入式处理器技术现状

## 全志T113-i

- 4x ARM Cortex-A65 up to 1.8GHz
- NPU: 1TOPS
- GPU: Mali-G52-2EE
- 3x Display, LVDS/HDMI/RGB
- 1x MIPI CSI, 1x DVP
- 1x PCIe3.0, 1x USB3.0 HOST
- 工商业储能EMS、小电流选线、通信管理机
- 运动控制器、AGV 机器人、医疗内窥镜



- 2x ARM Cortex-A7 up to 1.2GHz
- DSP: HIFI4 600MHz
- RISC-V: C906 1008MHz
- 2x LVDS DISPLAY
- 1x USB2.0 DRD, 1x USB2.0 HOST
- 1x EMAC
- 6x UART, 2x CAN
- 工业HMI、工业 PLC、物联网网关
- 示教器、户用储能 EMS/BMS、充电桩



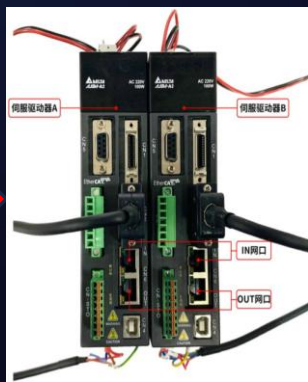
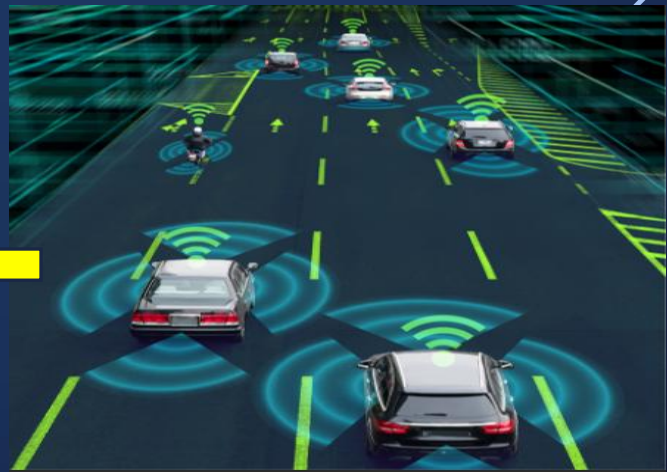
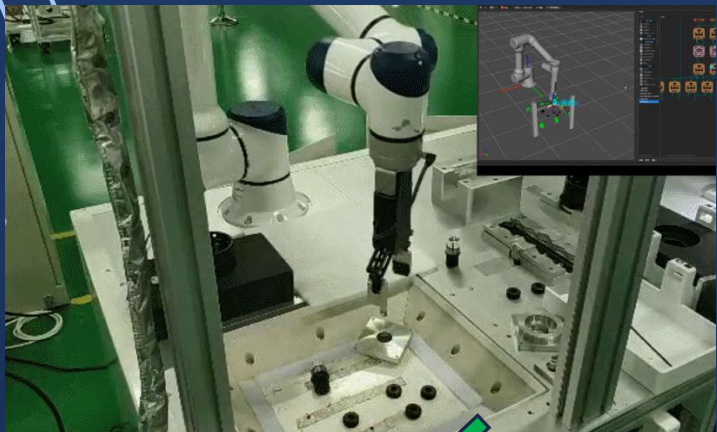
高算力多核处理器走千家万户

# 活跃的多核嵌入式处理器公司

- 海外芯片公司在高算力多核芯片和汽车智驾芯片上占有绝对的优势，配套的软件、工具和安全方案成熟。
- 国内在边缘和端侧智能芯片有成本和生态优势，在智驾领域正在积极跟进。



# 嵌入式多核操作系统的应用



# 《嵌入式实时操作系统 理论基础》介绍



## 第◆章 多核嵌入式系统



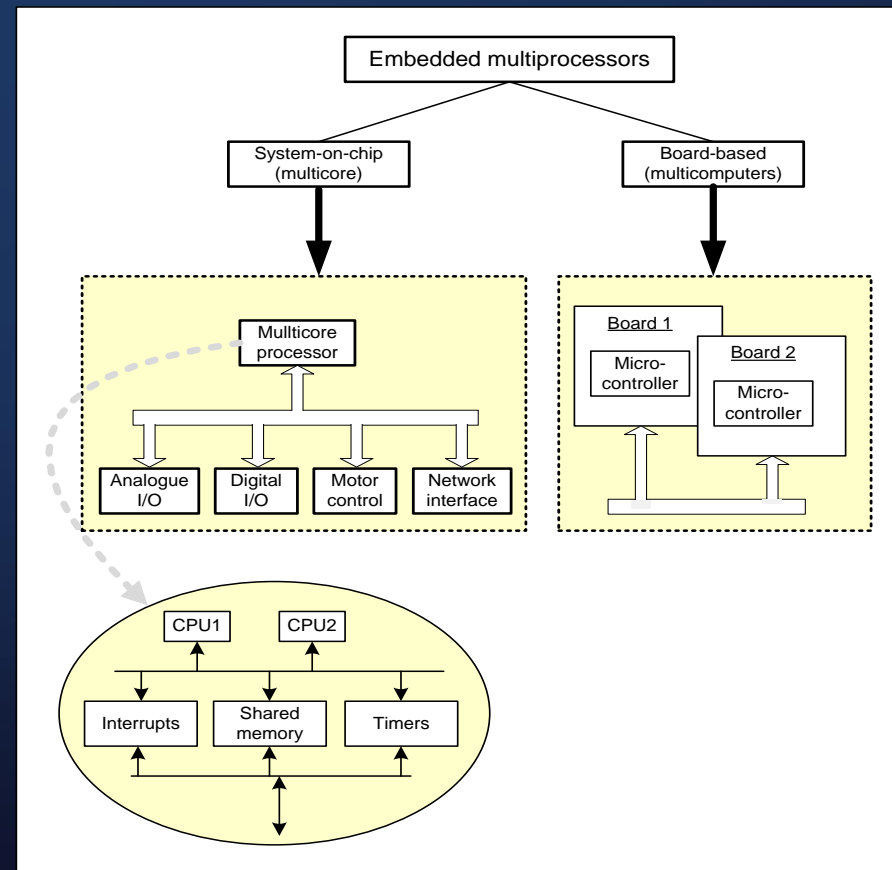
- 本书内容 (13章)
  - 实时操作系统基础
  - 调度——概念和实现
  - 使用互斥机制控制资源共享
  - 资源共享和争用问题
  - 任务间通信
  - 存储的使用和管理
  - **多处理器系统**
  - **分布式系统**
  - 调度策略的分析
  - 操作系统：基本结构和功能
  - RTOS的性能和基准测试
  - 多任务软件的测试和调试
  - **在关键系统中使用RTOS**



# 第2: 多核处理器概述

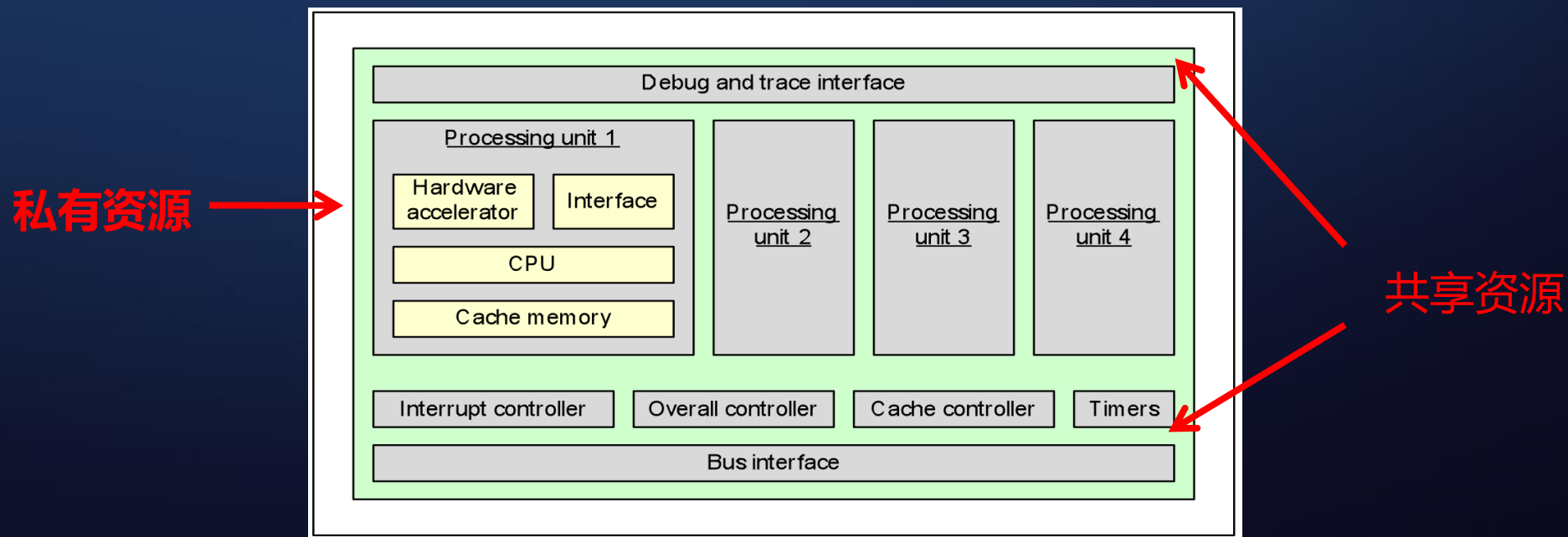
## 微处理器、多处理器和多核处理器

- 1971年 Intel公司设计出第一款微处理器 - 4004, 很快Intel又推出了8位的8008处理器和16位的8086处理器。
- 8086芯片是单核CPU, 随着需求的提高和功耗问题出现, 就有在一个芯片上建造两个或者多个核, 如 Intel i5 (14代) 6+8 核。
- 多处理器是由多个单板计算机组成的计算机系统。
- 多核处理器具有更高的计算密度和更强的并行处理能力, 单芯片尺寸还很小。多核处理器适合嵌入系统和嵌入式混合关键系统。



# 同构的多核处理器

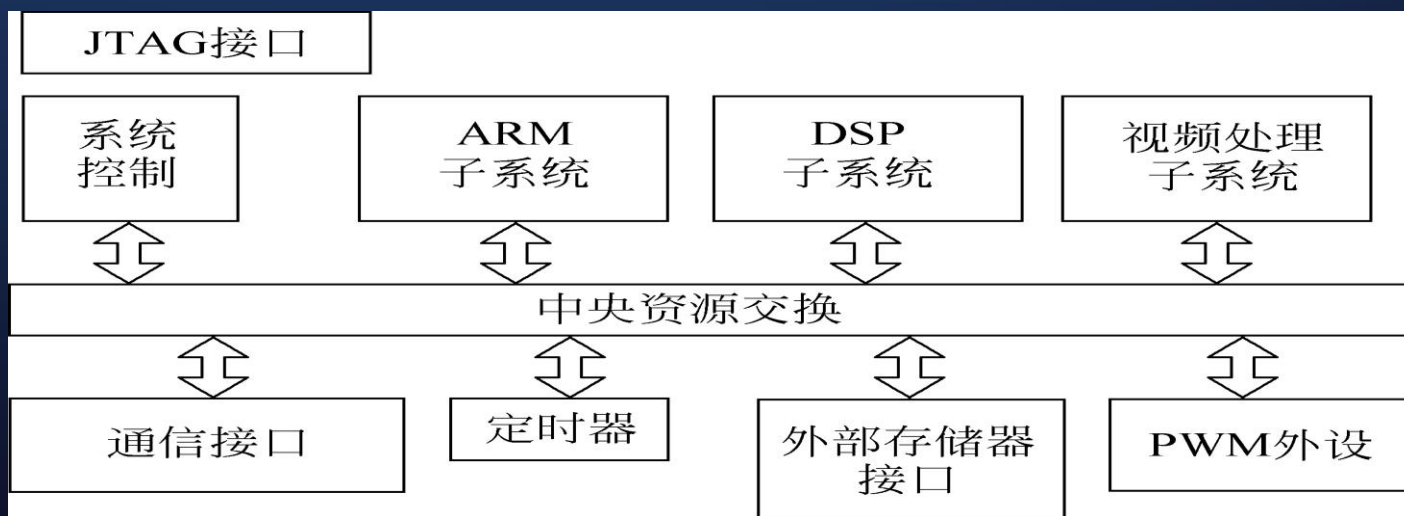
- **从硬件的角度看**，多核处理器可以分为同构和异构两种架构。
- **同构**
  - 所有的CPU的核心架构都一样。例如，瑞芯微RK3568、飞思卡尔的I.MX6D，它们有两个或者以上架构相同的ARM Cortex-A核。
  - RK3568 四核64位Cortex-A55 主频最高2.0GHz,内置1.0TOPS算力NPU



# 异构的多核处理器

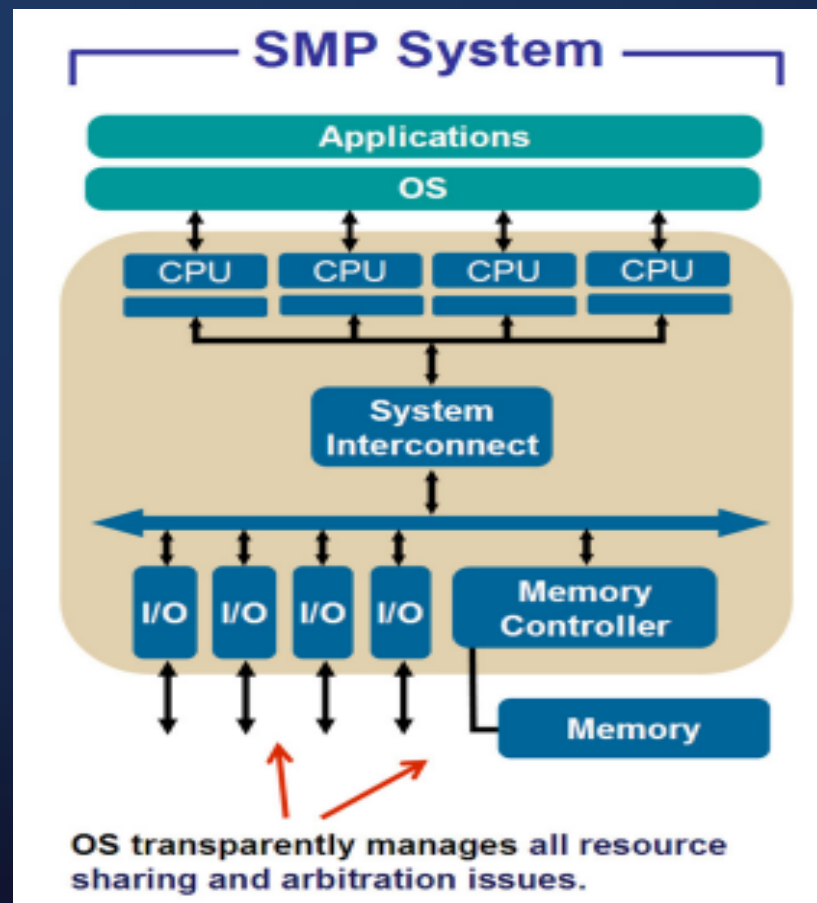
## ○ 异构

- CPU核心架构不一样称为异构。例如 TI TMS320DM8127有一个DSP C674x核和一个ARM Cortex-A8核, Xilinx的ZYNQ7000系列, 有两个ARM Cortex-A9核和FPGA (可配置MicroBlade), 这些处理器有不一样架构的CPU核, 所以都属于异构。
- 异构多核CPU 可根据负载, 平衡算力的使用, 比如实时性计算需求可由M/R核完成, 人机界面由A核完成等。



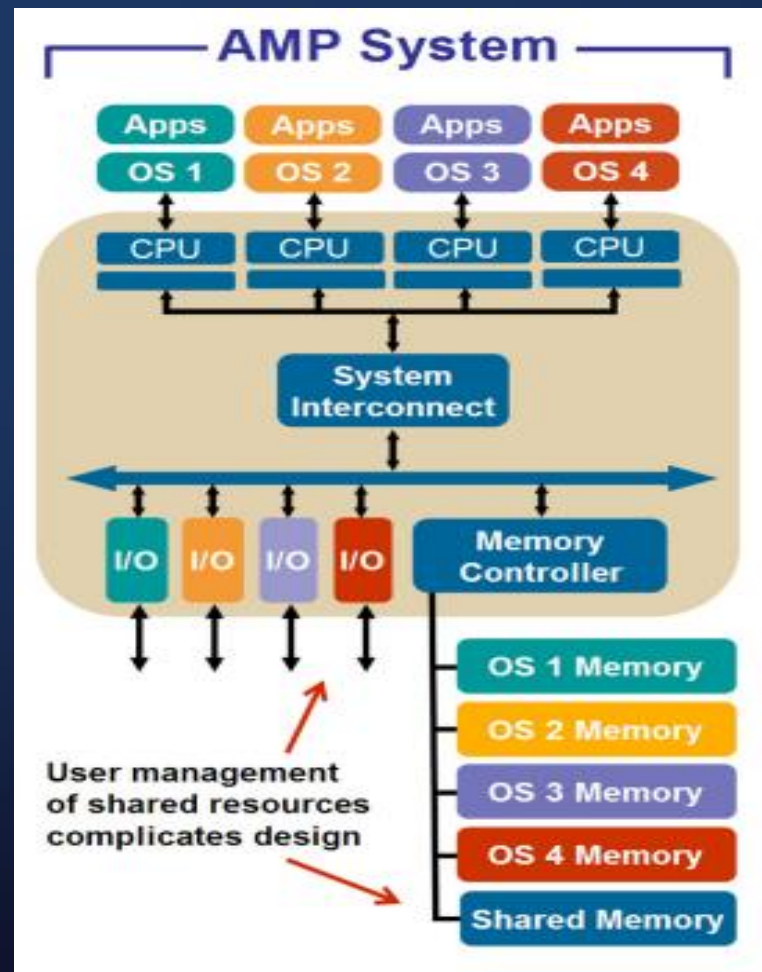
# 多核处理器软件架构 - SMP

- **从软件的角度看：**多核处理器平台的系统架构有：**SMP** (Symmetric Multi-Processing, 对称多处理)、**AMP** (Asymmetric Multi-Processing, 非对称多处理) 和**BMP** (bound multi-processing, 绑定多处理) 3种架构。
- SMP是指只有一个OS运行在多个CPU上，一个OS管理各个CPU内核，为各个内核分配工作负载，系统中所有的内核平等地访问内存和外设资源。
- 一般运行在SMP结构下的都是同构处理器。Windows和Linux, RTOS 中的 QNX和SylixOS 很好支持SMP结构。
- SMP结构下一个OS负责协调两个处理器，两个处理器共享内存，每个核运行的应用程序，通过MMU把它们映射到主存的不同物理位置上。



# 多核处理器软件架构 - AMP

- 每个CPU内核执行专门的任务，操作与单处理器设计完全相同。
- 系统的行为和可预测性与单处理器设计相似。
- 如果软件针对特定类型的硬件设计，在非对称多处理器上使用**异构AMP**。
- 在需要混合使用操作系统和/或调度方法的场景，在对称多处理器上使用**同构AMP**。
- AMP特点是各个OS都有本身独占的资源，其它资源由用户来指定多个系统共享或者专门分配给某一个系统来使用，系统之间通过共享的内存来完成通信。



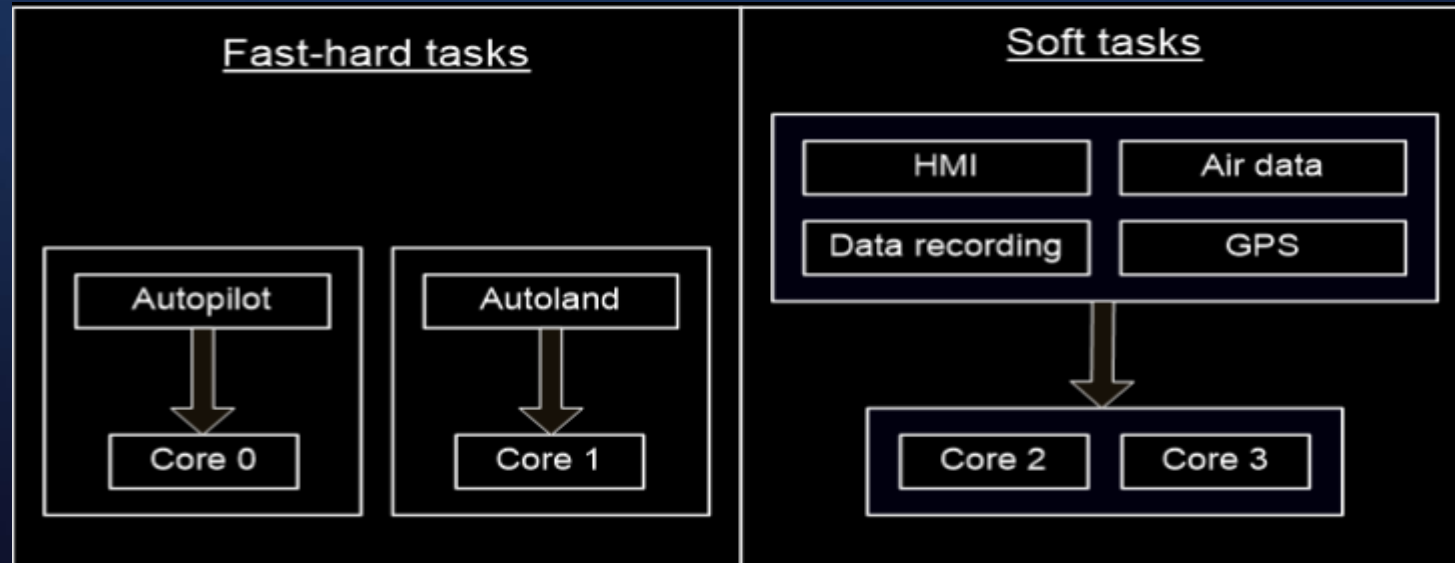
图片来自：Case Study — Making a Successful Transition to Multi-Core Processors /

# SMP在实时系统潜在的问题

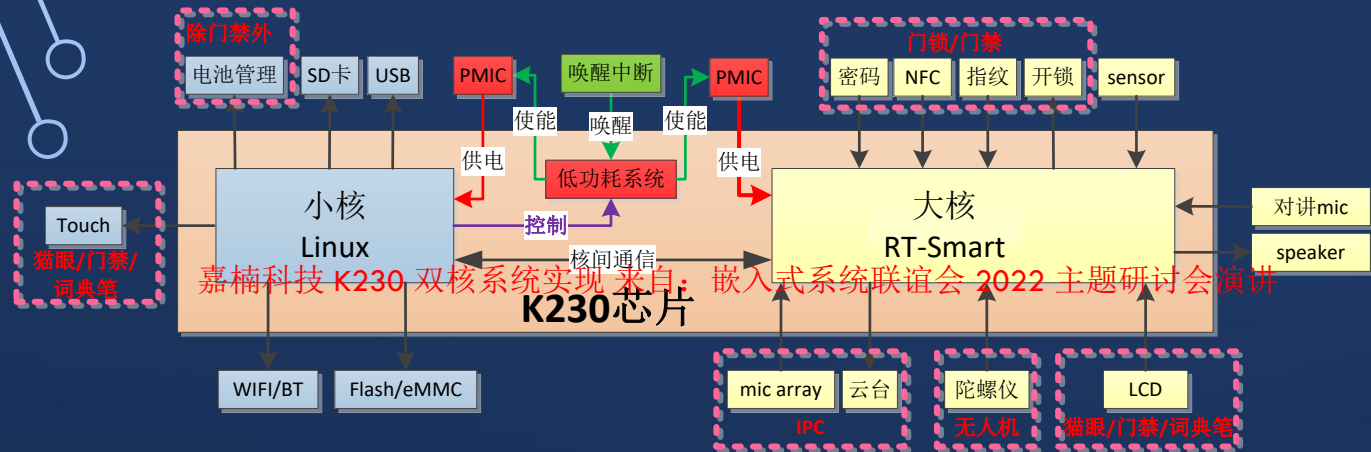
- 无法预测单个进程的执行顺序。
- 无法预测哪个CPU核执行哪个进程。
- 对于某些进程，可能无法保证实时行为。
- 如果从现有的单处理器设计移植到多处理器，可能会出现同步或互斥问题。
- 业界已开发绑定多处理（Osc v）以最大程度地减少此类问题。

# 多核处理器软件架构 – BMP

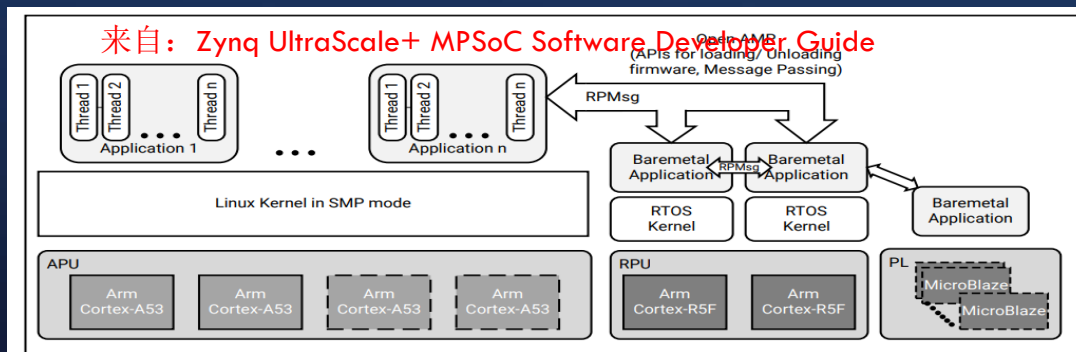
- BMP和SMP类似，也是由一个OS同时管理所有CPU内核，但是开发者可以指定某个任务在某个核中执行。
- BMP应用更多是在AMP的**混合的场景下 (hybrid)**，多个相同架构的内核被配置为一个SMP子系统，其他不同内核运行的是其它的OS，从整体来看就是AMP结构了。从逻辑上来分，这个SMP子系统看起来像是一个单核，可以把它包含在这个大的AMP系统中，这种混合模式即可以满足高性能需求、同时保证实时和安全性要求。



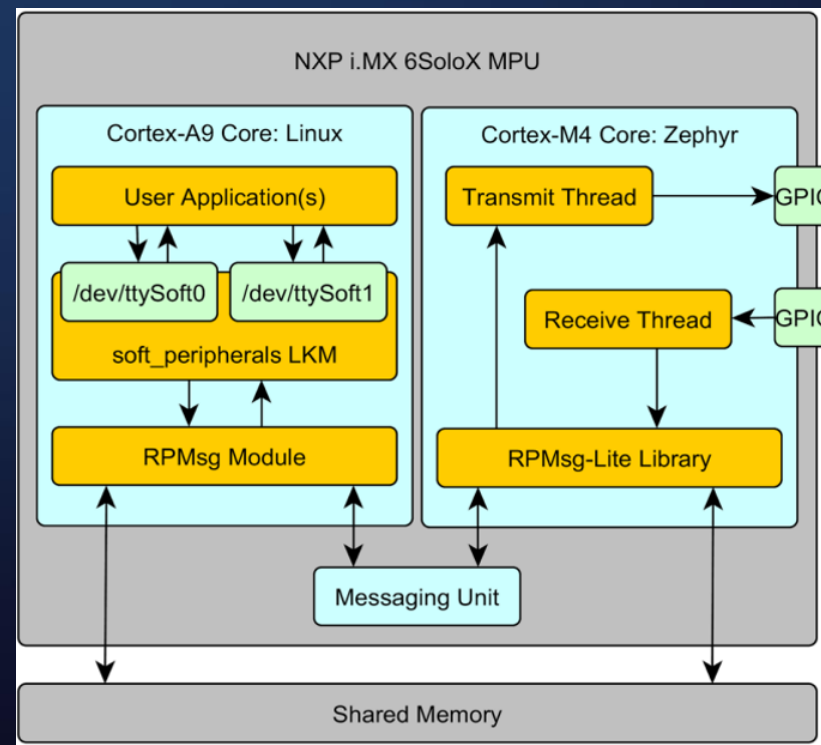
# AMP和SMP 解决方案



来自：国产操作系统在半导体AMHS 产业的应用 RT-Thread开发者大会演讲



**RT-Thread**  
基于TriCore FreeRTOS/RT-Thread SMP 汽车电子应用  
商业动态 | 支持车规级多核安全处理器



来自：Marek NOVAK 在 EW2019 演讲

# 第3: 虚拟化技术

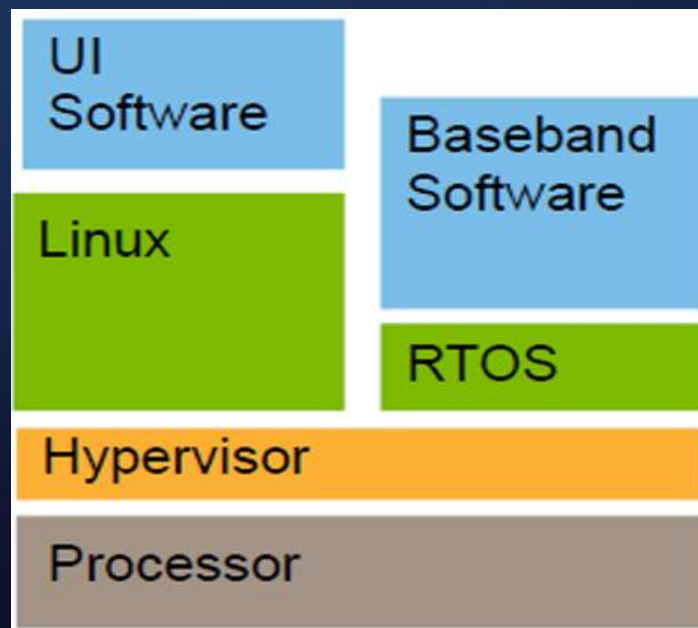
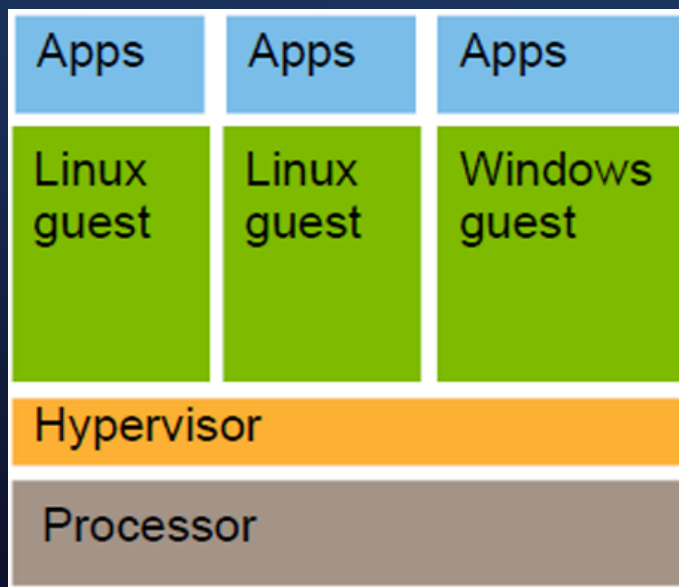
## 服务器虚拟化与嵌入式虚拟化的区别

○服务器：许多相似的Guest

- 不关注虚拟机管理程序大小
- 虚拟机调度是时间片轮转
- 调度机制和设备驱动-公平和共享

○嵌入式：1 HLOS(Linux) + 1 RTOS

- 虚拟机管理程序资源受限
- 中断延迟很重要
- 调度机制和设备驱动-实时和直通



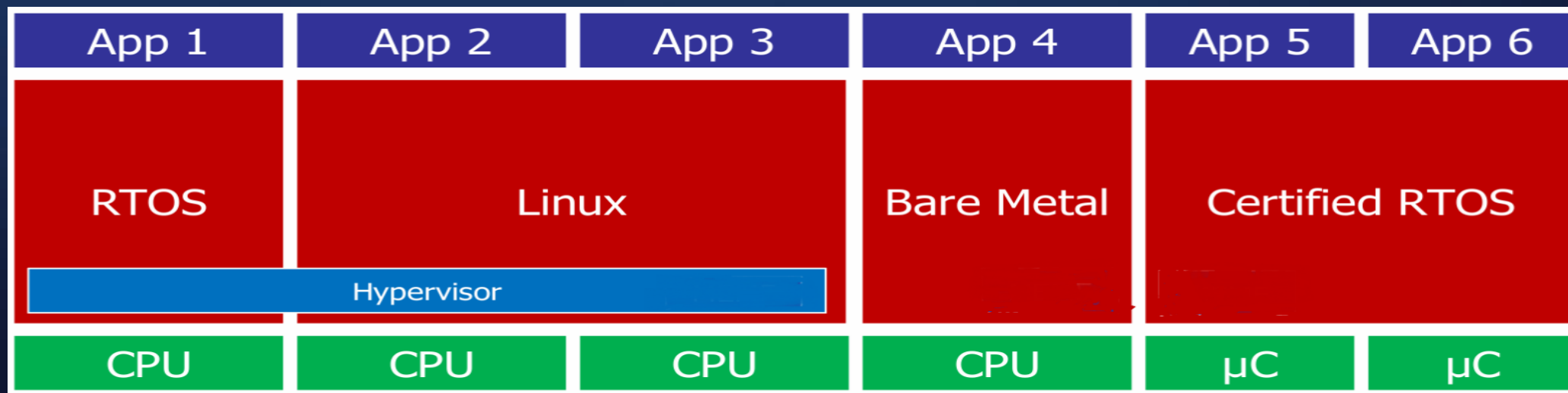
# 虚拟化解决方案

- 常见的开源虚拟化解决方案包括KVM和Xen，嵌入式系统 Jailhouse和Bao。许多商业操作系统提供虚拟化解决方案，例如Wind River VxWorks的虚拟化，Green Hills INTEGRIT 的多虚拟机监视器。这些示例都属于全虚拟化与静态分区虚拟机，类型1或类型2虚拟机。
- 国内的嵌入式虚拟化项目：vmRT-Thread、Intewell-V、ZVM 和 Rust-Shyper



# 嵌入式虚拟化 - 混合关键系统的底座

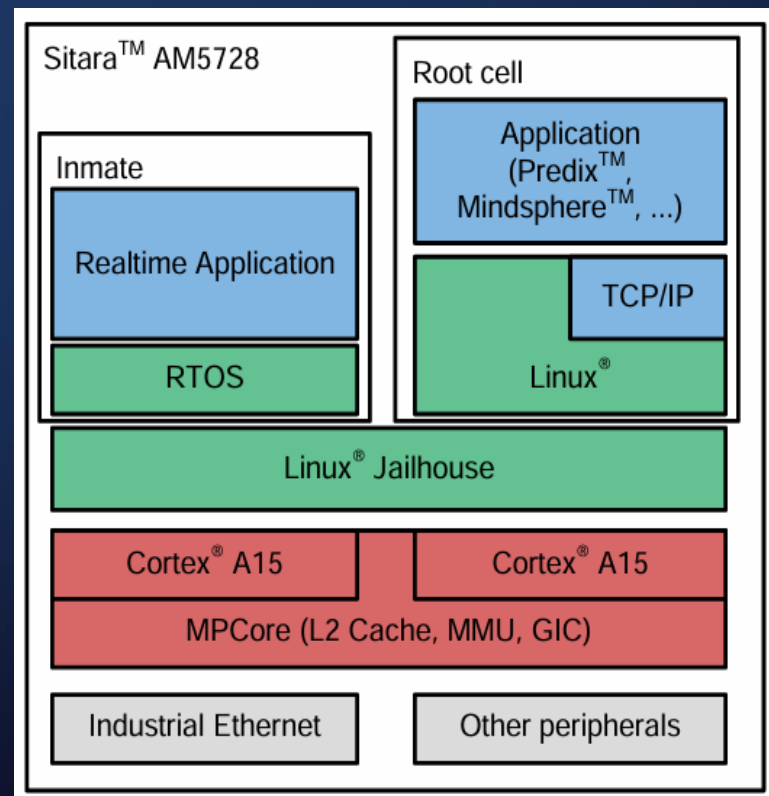
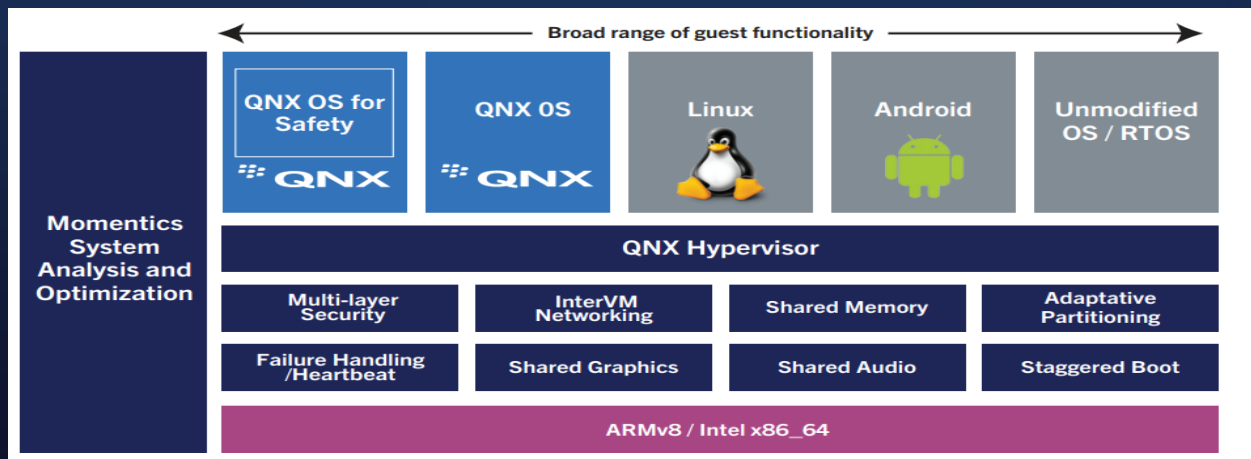
- 应对：实时与非实时的需求，安全认证的需要。
- 应用：工业（汽车）、军工和医疗。
- 认证：费用、时间、代码的考量，RTOS的预认证成熟（比如SafeRTOS和QNX）。



# 嵌入式虚拟化案例和解决方案



## 混合关键性系统架构

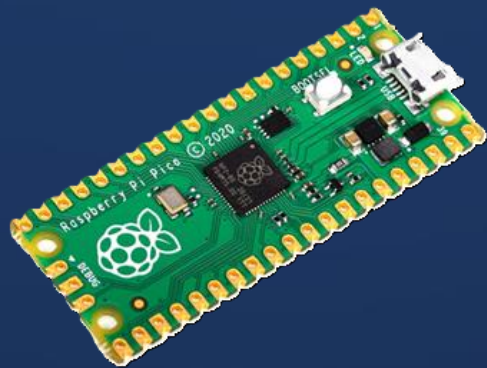


AM572x上的Jailhouse Hypervisor  
工业参考设计

# 第4: 多核操作系统实例

## 在树莓派 PICO上使用 FreeRTOS实现 SMP

- RP2040微控制器
- 133MHz双核Arm Cortex M0+
- 264KB SRAM
- 2MB Flash
- GPIO、UART、SPI和I2C支持
- 温度传感器
- WiFi和蓝牙支持 (Pico W和WH)



- FreeRTOS是一款历史悠久的开源轻量级RTOS, 支持40余种体系结构。
- FreeRTOS非官方的SMP支持: Espressif的ESP-IDF针对ESP32双核处理器SoC FreeRTOS进行了定制。
- 2021年宣布的FreeRTOS官方SMP分支支持多个相同体系结构的处理器核; 处理器核必须都能够访问同一片内存空间。
- 使用FreeRTOS的SMP时, 一个FreeRTOS实例跨多个处理器核心调度RTOS任务。
- 和标准发行版相比, SMP分支额外增加8个与处理器亲和性和抢占有关的API以及数个配置宏。
- 2024年12月 FreeRTOS 11.0 正式支持 SMP 并入主线。

本实例由何灵渊完成  
《嵌入式软件精解》和《嵌入式实时操作系统-理论基础》译者

# SMP和任务调度

- SMP中的调度方法和单核一致，但是同时可以有等同于处理器核数量的任务处于运行状态。
  - 这意味着，可能同时有较高和较低优先级的任务在不同核上运行，即低优先级的任务不一定要等待所有高优先级任务结束才能运行。
  - 例如，有一个高优先级任务和两个中等优先级任务处于“就绪”状态，SMP调度器需要选择两个任务，每个核心对应一个任务。首先，选择较高优先级任务在第一个核心上运行。这样就剩下了两个中等优先级的任务作为可运行的任务，因此会在第二个核心上轮询运行。结果是高优先级和中等优先级的任务同时运行。
- 同优先级的任务由于核亲和度不同，不一定会严格按序轮询执行。
  - 在所有核上都可以运行的任务依然能够按序轮询。

# 代码分析样例1：多任务和处理器核亲和性

- 创建A、B、C、D四个任务，优先级相同。
- A、B通过两种不同API设置处理器亲和性，只能分别运行在核0/1上。C、D不设置亲和性。
- 运行中A、B只会分别运行在核0和核1上，C、D运行的核不定。
- 样例输出：右图



```
Task A core 0 tick 5700 mask 1
Task D core 1 tick 5701 mask -1
Task B core 1 tick 5701 mask 2
Task C core 1 tick 5800 mask -1
Task A core 0 tick 5800 mask 1
Task D core 0 tick 5801 mask -1
Task B core 1 tick 5801 mask 2
...
Task A core 0 tick 6200 mask 1
Task D core 0 tick 6201 mask -1
Task B core 1 tick 6201 mask 2
Task C core 1 tick 6300 mask -1
Task A core 0 tick 6300 mask 1
Task D core 1 tick 6301 mask -1
```

# 代码分析样例 2：禁用抢占

- 创建A、B、C三个任务，B和C的优先级较A高。任务A只能运行在核0上，B、C不设置亲和性。
- 任务A持续执行，每3秒周期性禁用或允许抢占。任务B、C每1秒会进入可调度状态。
- A禁用抢占时，B、C会在核1上依次运行；A允许抢占时，B、C同时占用两个核运行，即A被抢占。
- 样例输出：右图

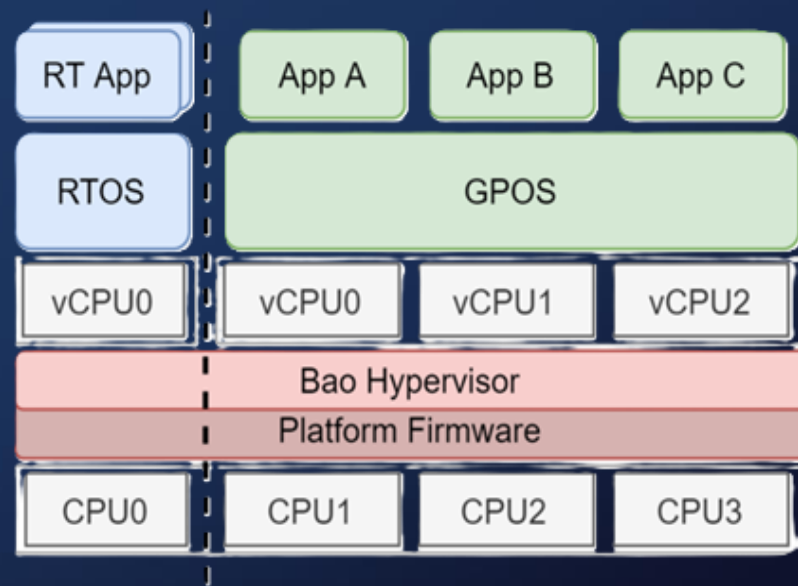


```
Task A priority 1 core 0 mask 1 tick 15005 - preemptable
Task B priority 2 core 1 mask -1 tick 16000
Task C priority 2 core 0 mask -1 tick 16001
Task B priority 2 core 1 mask -1 tick 17000
Task C priority 2 core 0 mask -1 tick 17001
Task B priority 2 core 1 mask -1 tick 18000
Task C priority 2 core 0 mask -1 tick 18001
Task A priority 1 core 0 mask 1 tick 18006 - no preempt
Task B priority 2 core 1 mask -1 tick 19000
Task C priority 2 core 1 mask -1 tick 19001
Task B priority 2 core 1 mask -1 tick 20000
Task C priority 2 core 1 mask -1 tick 20001
Task B priority 2 core 1 mask -1 tick 21000
Task C priority 2 core 1 mask -1 tick 21001
```

# 在树莓派 4B 上通过 Bao 实现虚拟机实例

## 什么是Bao虚拟机?

- 针对嵌入式处理器，支持Arm v8和RISC-V 64位架构。
  - Bao要求体系结构必须支持硬件虚拟化。
- 不依赖于其他大型的GPOS（如 Linux）独立运行。
- 通过硬件静态分区分配虚拟机的CPU、内存和外设：
  - 内存在初始化时静态分配。
  - 虚拟CPU静态固定分配到物理CPU上，不会在物理CPU间迁移；一个物理CPU上的多个虚拟CPU实现简单的先入先出调度。
  - 客户机I/O全部为直通。
  - 虚拟中断会直接映射到物理中断上。
- 贯彻最低特权理念：虚拟机的虚拟CPU不能访问其他硬件分区的物理内存；Bao不能直接访问虚拟机的物理内存。
- Bao的开销很小，最大仅为2%左右。



Bao双客户机配置

本实例由何灵渊完成

《嵌入式软件精解》和《嵌入式实时操作系统-理论基础》译者

# Bao实例：准备环境、构建和运行

- 下载和解压Arm工具链，树莓派4需要aarch64-none-elf工具链
- 安装必备的软件。
- 构建必需固件和demo。
- 构建完成后屏幕会输出具体的文件复制命令，运行它们将生成的文件复制到SD卡上。
- 使用串口调试硬件连接树莓派的UART接口到USB口；通过以太网接口连接树莓派到主机，并设置主机以太网适配器使用静态IP。
- 将SD卡插入树莓派4的卡槽，打开电源，观察串口输出。看到UBoot显示“Hit any key to stop autoboot”时立即在键盘上输入任何字符，然后在UBoot命令行输入命令载入Bao：  
fatload mmc 0 0x200000 bao.bin; go 0x200000
- 此时串口应该看到Bao启动，然后Zephyr启动，程序线程开始打印数字（右图）。

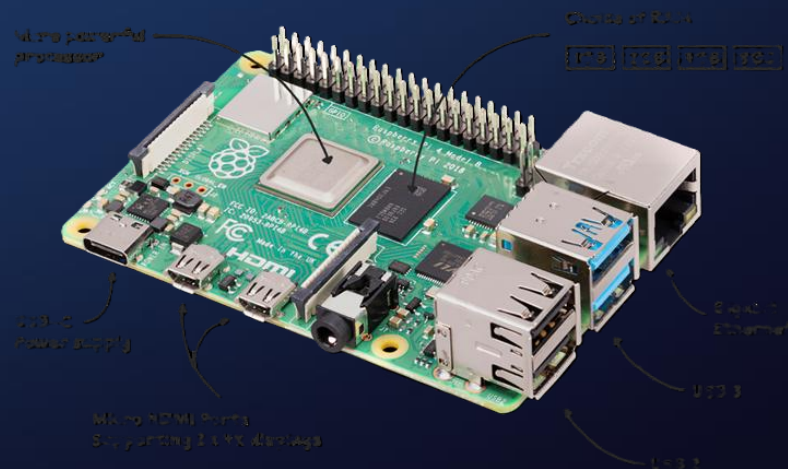
```
U-Boot 2022.10 (Mar 18 2025 - 23:37:11 -0700)

DRAM: 3.9 GiB
RPI 4 Model B (0xc03115)
Core: 209 devices, 16 uclasses, devicetree: board
MMC: mmcnr@7e300000: 1, mmc@7e340000: 0
Loading Environment from FAT... Unable to read "uboot.env" from mmc0:1...
In: serial
Out: serial
Err: serial
Net: eth0: ethernet@7d580000
PCIe BRCM: link up, 5.0 Gbps x1 (SSC)
starting USB...
Bus xhci_pci: Register 5000420 NbrPorts 5
Starting the controller
USB XHCI 1.00
scanning bus xhci_pci for devices... 2 USB Device(s) found
       scanning usb for storage devices... 0 Storage Device(s) found
Hit any key to stop autoboot: 0
U-Boot> fatload mmc 0 0x200000 bao.bin; go 0x200000
50270212 bytes read in 2130 ms (22.5 MiB/s)
## Starting application at 0x00200000 ...
Bao Hypervisor
*** Booting Zephyr OS build v3.5.0-rc1 ***
Zephyr Bao Demo!
Thread1 (cpu0): 0
Thread2 (cpu0): 0
Thread1 (cpu0): 1
Thread2 (cpu0): 1
Thread1 (cpu0): 2
Thread2 (cpu0): 2
Thread1 (cpu0): 3
Thread2 (cpu0): 3
```

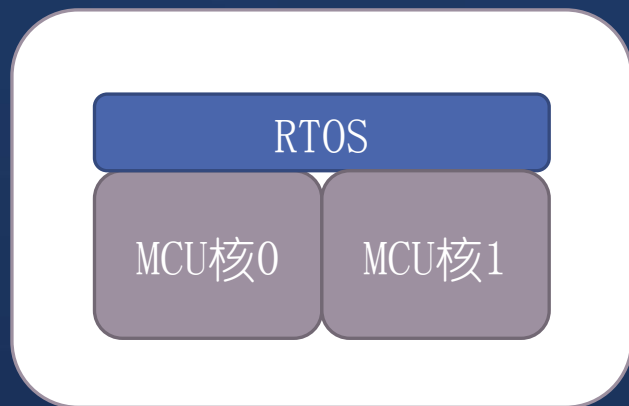
# Bao实例：虚拟机间通信

- 样例中Bao上运行的第二台虚拟机运行Linux (CPU1)。因为仅Zephyr通过设备直通使用UART接口，因此没有串口输出。
- Linux环境可以通过以太网SSH访问
- 在Linux中，Bao创建了设备/dev/baoipc0用于向Zephyr传递消息。Zephyr中通过中断处理代码接收和打印消息。
- 在Linux中运行：  
echo "Hello, Zephyr!" > /dev/baoipc0
- 串口输出可以看到Zephyr接收并打印了这条消息 (右图)。
- 至此我们验证了Bao上Zephyr+Linux的运行和Bao提供的虚拟机间通信机制。

```
Thread1 (cpu0): 1928
Thread2 (cpu0): 1928
message from linux: Hello, Zephyr!
Thread1 (cpu0): 1929
Thread2 (cpu0): 1929
```

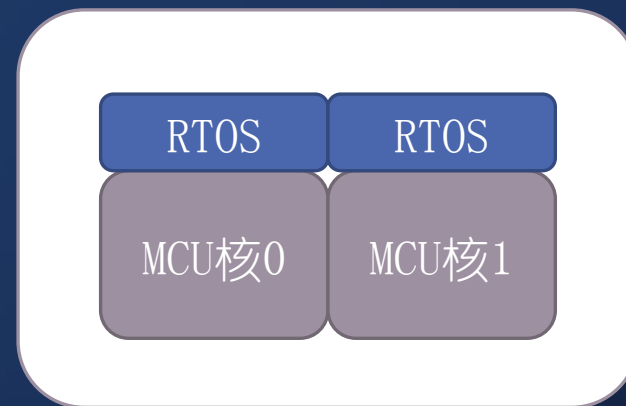


## 第4 :应用展望 SMP还是AMP ?



### 对称多处理 (SMP)

- 一个RTOS在多个处理器核上运行任务
- 任务负载由多个核分担, 动态任务调度
- 处理器核可以共享内存区域
- 调度的不确定性
- 需要安全认证时的考量- 静态分配



### 非对称多处理 (AMP)

- 每个处理器核上运行一个 GPOS/RTOS
- 每个核的行为和单核系统类似
- 进一步分为同构和异构AMP
- 虽有openAMP 开源协议架构, 核间通信不标准
- 多RTOS 或者 RTOS+ BareMetal 部署适合安全认证

# SMP/AMP/BMP的 选择原则

- 如果高性能是主要需要的应用场景，SMP模式是最佳解决方案。但是简单地将一个单处理器软件移植到多处理器平台上，可能无法很好完成这一任务。程序需要重新编写才能适应并行计算的环境，毫无疑问计算密集的应用适合于SMP模式的系统。
- AMP模式的特性让它广泛适应实时嵌入式系统，它将性能提升与可靠性完美匹配到一起。为单处理器编写的程序可以很好移植到多处理器，开发者不需要担心太多。
- 混合模式处理器，看起来是在SMP和BMP 之间的一个平衡，这样的实现适用于复杂的嵌入式系统应用，设计相对复杂、调试和部署有一定难度。

参考：《嵌入式实时操作系统-理论基础》第7章：多处理器系统

**从简原则：bearmetal – RTOS , AMP- SMP –BMP/混合模式**

# 嵌入式虚拟化技术的挑战

- 兼容性 – 需要整合数个OS 和应用，OS版本不一 **兼容性是一大问题。**
- 实时性 – 虚拟机引入Hypervisor 必然带来性能和资源消耗，如何保证实时性是一大问题。
- 安全隔离 – 虚拟机一大功能是隔离应用和负载，实际效果非常重要。
- 平台支持 – Hypervisor依赖平台和处理器，多数商业和开源虚拟机支持Arm，x86 和RISC-V 支持不充分。

来自：王洪波 《嵌入式虚拟化技术与应用》

- 随着处理器芯片的算力提升和集成度提高，以及行业应用越来越复杂，基于虚拟化技术的开发平台，因其在解决实时、安全性同时，又能提供丰富软硬件支持，还具备一定的隔离性和更好灵活性，未来应用前景广阔。

来自：熊谱翔" 机器人上的虚拟化混合部署探索和实践

- 虚拟化的分布式架构带来的问题：体积庞大、**调试困难**、维护部署复杂

来自：张云飞 嵌入式虚拟化技术探索及实践

**嵌入式虚拟化是发展趋势、广泛使用才开始，需要很好的平台和技术支持**



Embedded OS in China  
国产嵌入式操作系统产业论坛

欢迎参加

QUESTIONS 问题?

THANKS 感谢聆听



Allan@esbf.org