嵌入式实时虚拟机ZVM关键核心技术进展介绍

汇报人：熊程来，openEuler SIG-Zephyr Maintainer

湖南大学

嵌入式与网络计算湖南省重点实验室谢国琪教授团队

2023-12-16

嵌入式实时虚拟机ZVM关键核心技术进展介绍

- **ZVM总体系统架构介绍**
  - ZVM设计思路；
  - ZVM架构设计；
- **ZVM系统功能特性**
  - ZVM系统功能特性；
  - ZVM系统新功能支持；
  - 性能测试对比；
- **未来应用场景**

嵌入式OS对富功能与硬实时的混合关键部署要求

双子星联动

富功能OS

- 智能座舱
- 数据可视化
- 场景可视化

Linux宏内核

Linux：功能强大
但实时性不足

实时控制OS

- 智能驾驶
- 底盘控制
- 动力控制

实时微内核

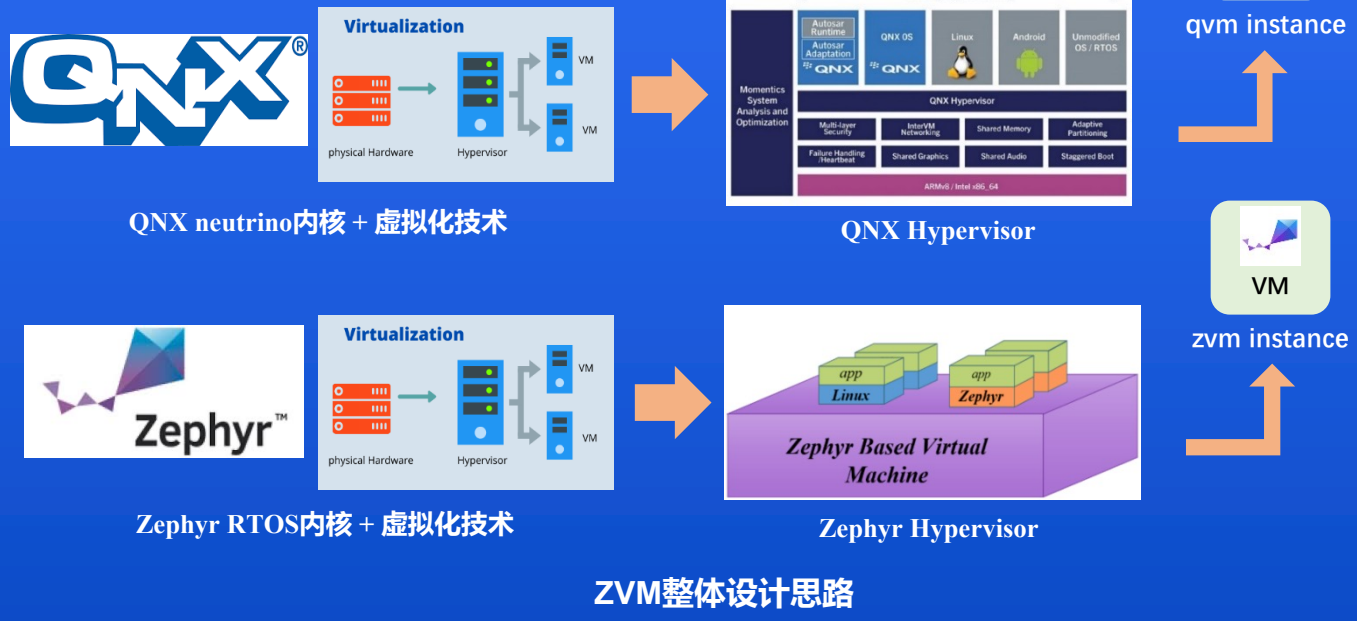
QNX：以闭源的方式来形成技术壁垒

满足关键场景对智能化、实时性及安全性的多重要求！

ZVM总体系统架构介绍

1 基于Zephyr RTOS的虚拟机设计思路

- Zephyr RTOS现有功能支持
- 核心虚拟化模块实现
- 可裁剪Zephyr 系统模块
- 运行虚拟机实例



ZVM整体设计思路

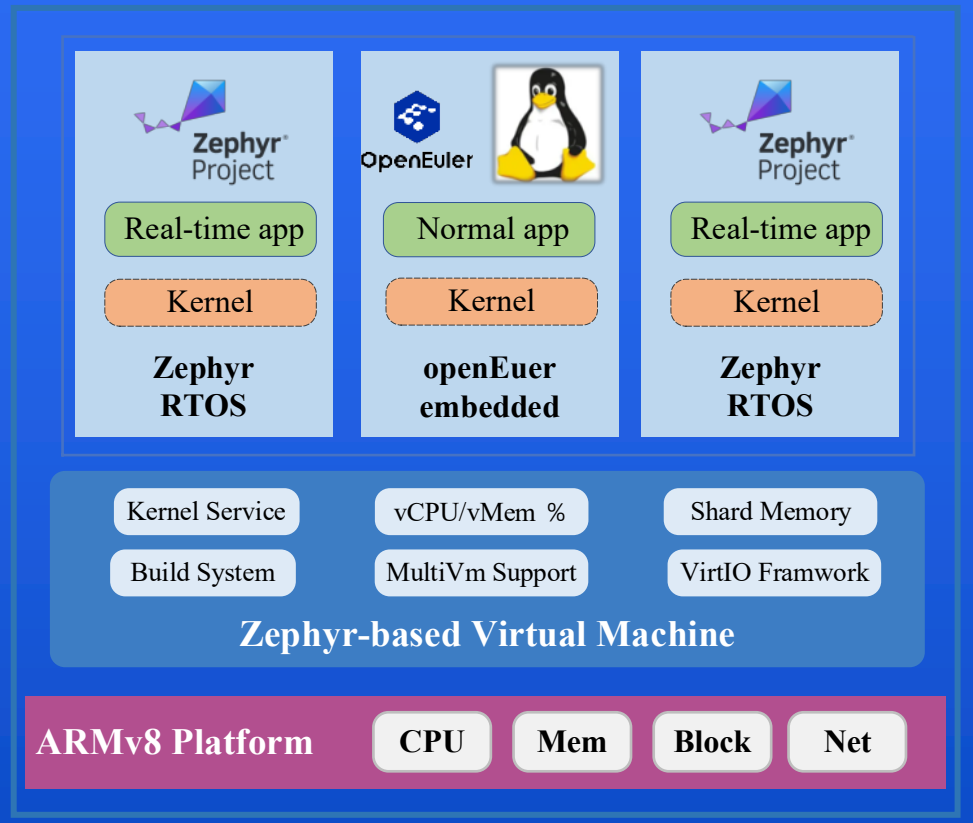
目标：满足开源、实时，灵活，智能特性的嵌入式虚拟机管理平台

ZVM总体系统架构介绍

2 基于Zephyr RTOS的虚拟机架构设计

分层架构设计：包括硬件、hypervisor、guest OS到应用的多层级系统：

- 硬件层：ARMv8处理器平台，包括处理器、内存、中断控制器等核心设备的支持；
- 虚拟化层：核心虚拟化功能模块化设计，虚拟化拓展模块设计，复用Zephyr RTOS内核服务及设备模型；
- 内核层：运行Zephyr RTOS、Linux等ARMv8架构内核；
- 应用层：客户虚拟机中运行实时或非实时任务；



ZVM整体架构

ZVM系统功能特性

1 系统可配置性

- 支持配置ZVM的灵活配置，可在type-1到type-2型虚拟机的按需切换；

2 动态资源分配

- 支持运行时动态内存分配、启动时动态设备分配，自适应任务调度；

3 设备虚拟化

- 支持设备直通、完全虚拟化、半虚拟化virtIOv框架；

4 多类型Guest OS

- 支持社区Linux、openEuler Embedded、Zephyr RTOS等；

5 Guest OS间通信

- 支持基于共享内存的实时通信；

6 强实时性

- 复用Zephyr RTOS的强实时能力，支持抢占式、全局/分区调度算法；

7 端侧实时推理

- 支持Paddle Lite 端侧AI推理框架，支持ResNet50等多种AI模型；

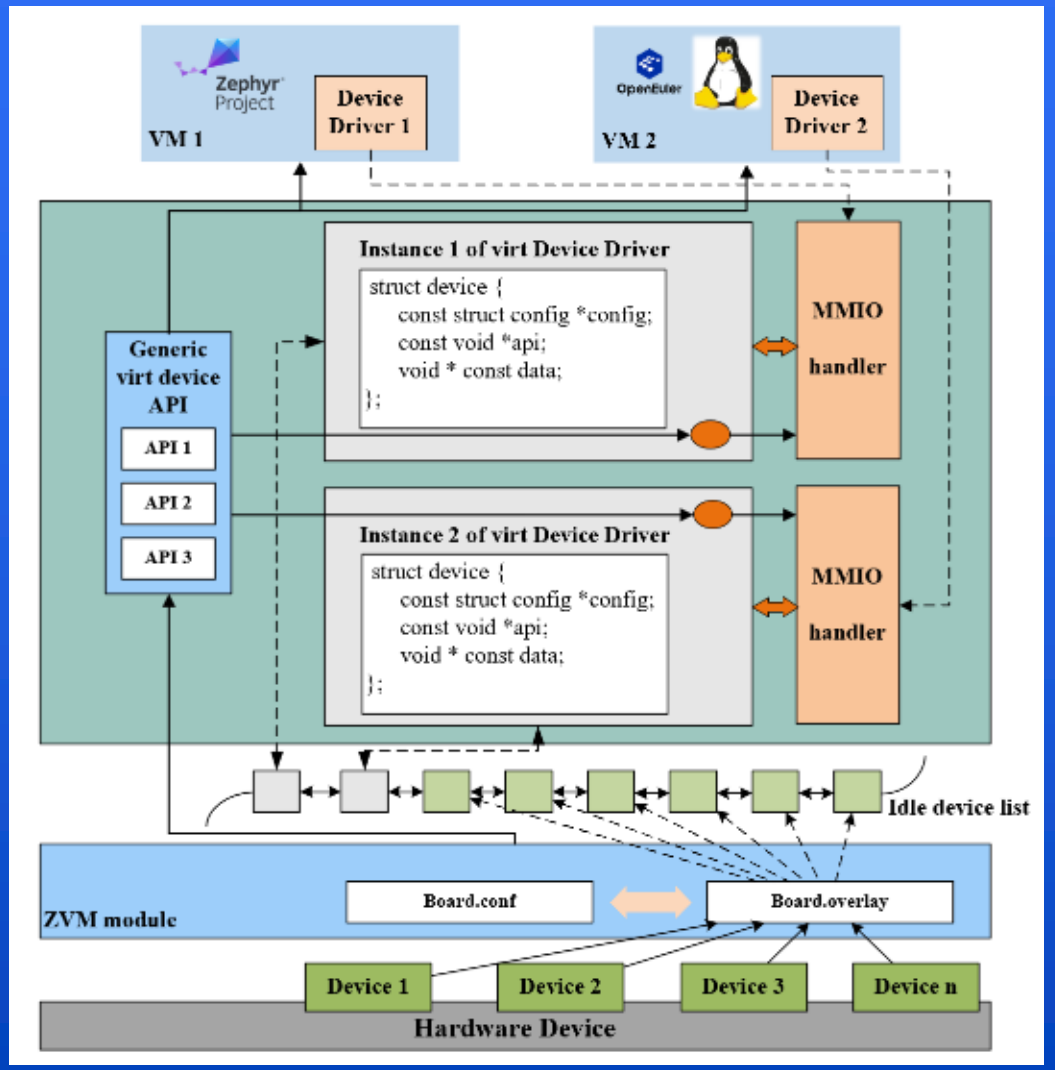
8 ARMv8硬件架构

- 支持RK3568等国产嵌入式SoC板卡，利用VHE特性减少性能损耗；

ZVM系统功能新特性

1 ZVM系统设备模型

- **支持核心设备完全虚拟化；**
- **支持半虚拟化方案；**
- **支持ZVM虚拟机的直通设备模型；**
  - 根据板卡overlay初始化板卡可用设备；
  - ZVM初始化idle deveice list；
  - VM初始化分配空闲设备；
  - 使用virt device driver实例虚拟化设备；
  - 建立虚拟机设备的MMIO映射；
  - 通过API -> handle函数处理设备中断；



ZVM虚拟机直通设备模型

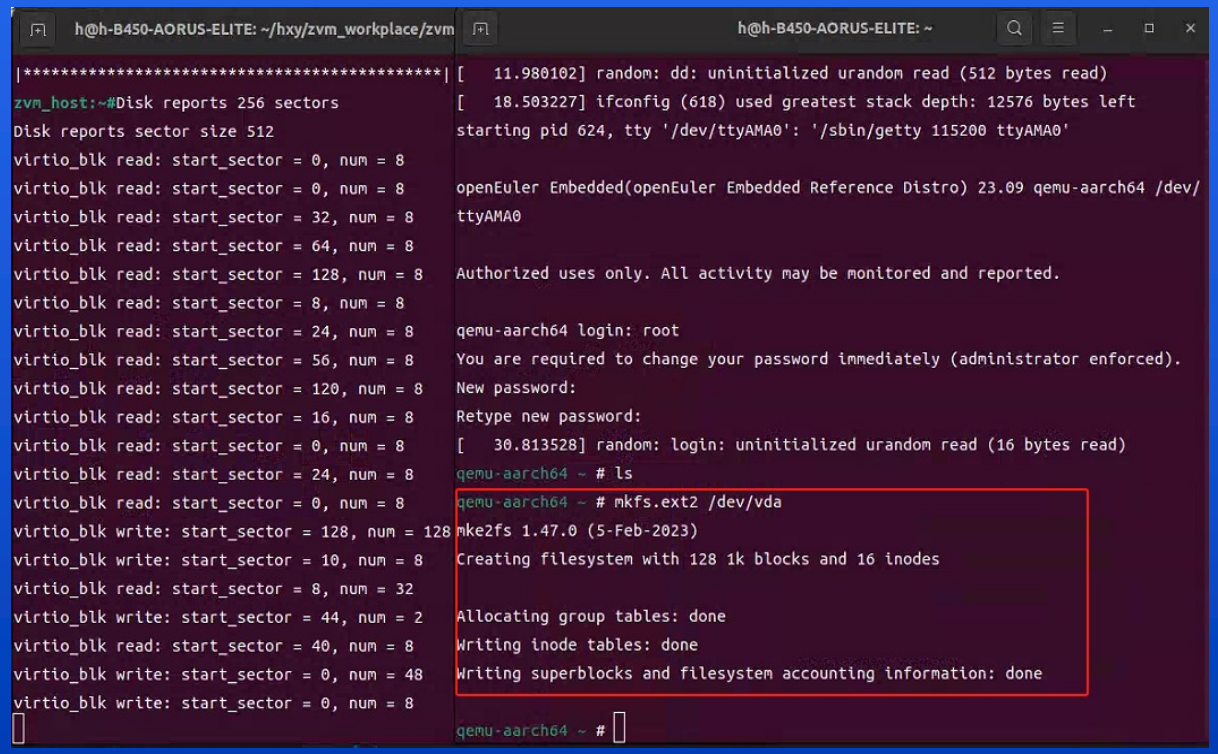
ZVM系统功能新特性

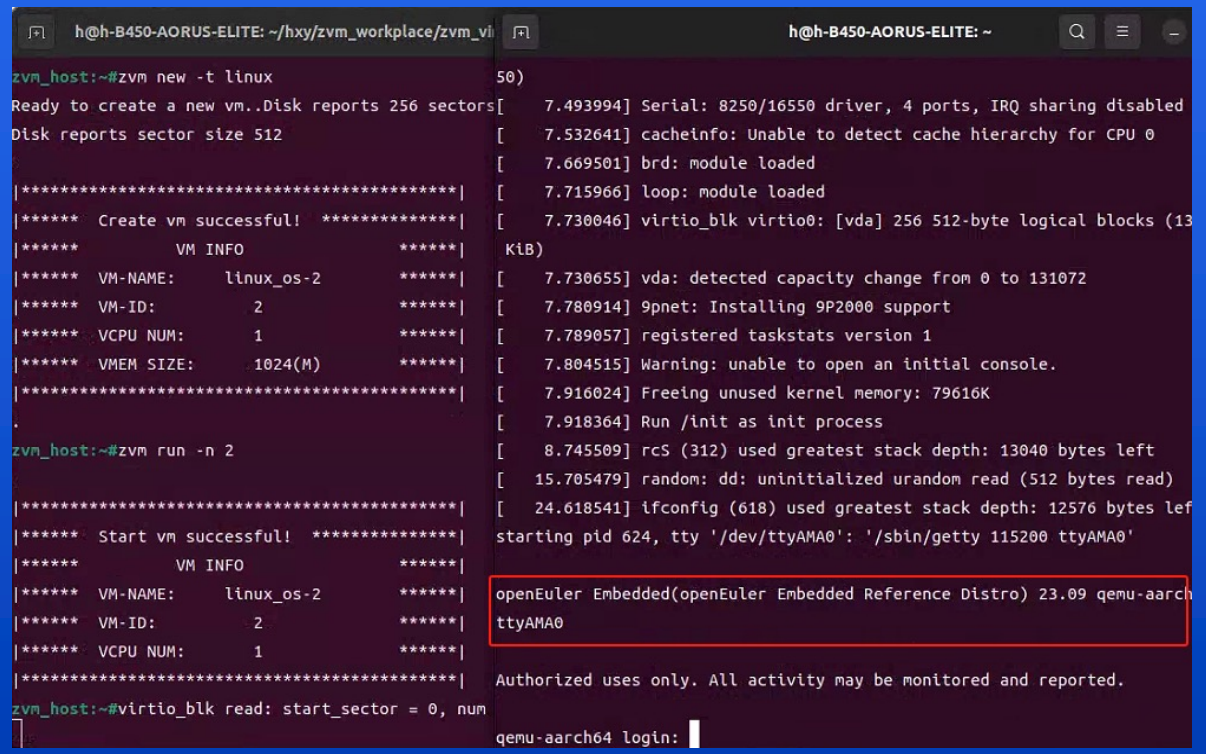
2 设备半虚拟化框架virtIO

- 符合OASIS virtIO V1.1规范，支持virtio_mmio传输方式，实现了virtIO块设备的后端支持

3 openEuler Embedded虚拟机支持

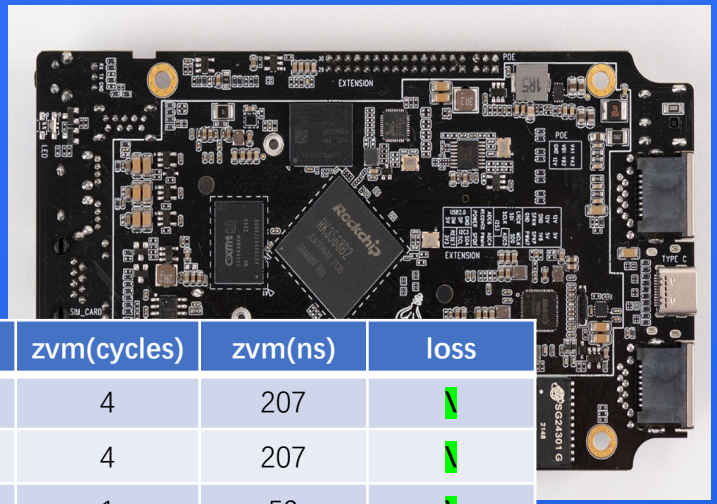
- 增添对openEuler Embeded虚拟机的支持，并支持启动个性化定制内核





ZVM性能测试与对比

处理器频率1.42Ghz, 计数器时钟频率24Mhz

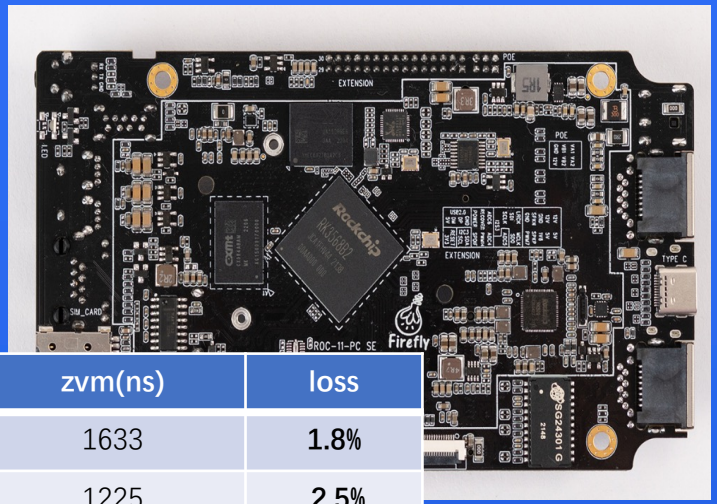


1 实时性对比，基于Zephyr Lantency_benchmark

items	board(cycles)	board(ns)	kvm(cycles)	kvm(ns)	loss	zvm(cycles)	zvm(ns)	loss
Preemptive threads ctx switch via k_yield (K -> K)	5	209	3	164	\	4	207	\
Cooperative threads ctx switch via k_yield (K -> K)	5	209	3	155	\	4	207	\
Switch from ISR back to interrupted thread	1	59	4	203	244.1%	1	59	\
Switch from ISR to another thread (kernel)	1	82	2	88	7.3%	1	82	\
Create kernel thread from kernel thread	33	1399	35	1459	4.3%	33	1396	\
Start kernel thread from kernel thread	6	268	7	321	19.8%	6	267	\
Suspend kernel thread from kernel thread	6	250	5	249	\	6	251	0.4%
Resume kernel thread from kernel thread	5	222	5	226	1.8%	5	221	\
Abort kernel thread from kernel thread	2	89	2	106	19.1%	2	88	\
Give a semaphore (no waiters) from kernel thread	0	3	0	2	\	0	3	\
Take a semaphore (no blocking) from kernel thread	429496	17895679	429496	17895672	\	429496	17895679	\
Take a semaphore (context switch K -> K)	5	241	5	227	\	5	244	1.2%
Give a semaphore (context switch K -> K)	7	296	5	237	\	7	296	\
Lock a mutex from kernel thread	429496	17895685	429496	17895673	\	429496	17895685	\
Unlock a mutex from kernel thread	429496	17895669	429496	17895660	\	429496	17895669	\
Average time for heap malloc	4	189	3	163	\	4	192	1.6%
Average time for heap free	3	163	3	148	\	3	157	\

ZVM性能测试与对比

处理器频率1.42Ghz, 计数器时钟频率24Mhz



2 系统性能，基于Zephyr sys_kernel benchmark

items	board(ns)	kvm(ns)	loss	zvm(ns)	loss
k_sem_init/ksem_take/ksem_give	1604	4112	156.4%	1633	1.8%
k_sem_init/ksem_take/**k_yield**/ksem_give	1195	3937	229.5%	1225	2.5%
k_sem_init/ksem_take/ksem_give/**ksem_give/ksem_take**	875	4316	393.3%	962	9.9%
k_lifo_init/k_lifo_get/k_lifo_put	1400	5920	322.9%	1370	\
k_lifo_init/k_lifo_get/**klifo_get**/k_lifo_put/**k_yield**	1283	5804	352.4%	1312	2.3%
k_lifo_init/k_lifo_get/k_lifo_put/**klifo_get/k_lifo_put**	962	4608	379.0%	991	3.0%
k_fifo_init/k_fifo_get/k_lifo_put	1254	6008	379.1%	1283	2.3%
k_fifo_init/k_fifo_get/**k_fifo_get**/k_lifo_put/**k_yield**	1312	5116	289.9%	1283	\
k_fifo_init/k_fifo_get/k_fifo_put/**kfifo_get/k_fifo_put**	1225	5891	380.9%	1225	\
k_stack_init/k_stack_pop/k_stack_push	1370	4533	230.9%	1370	\
k_stack_init/k_stack_pop/**k_stack_pop**/k_stack_push/**k_yield**	1195	5804	385.7%	1225	2.5%
k_stack_init/k_stack_pop/k_stack_push/**k_stack_pop/k_stack_push**	933	4416	373.3%	933	\
k_mem_slab_alloc	145	300	106.9%	175	20.7%
k_mem_slab_free	116	262	125.9%	116	\

未来发展规划

1 ZVM未来发展规划（应用）

（1）增加对TSN的支持：

在虚拟机中**实现TSN通信协议**，保证车辆内多个域之间通信的**高可靠、低时延和确定性**，以满足严苛的实时要求和多个应用场景的需求。

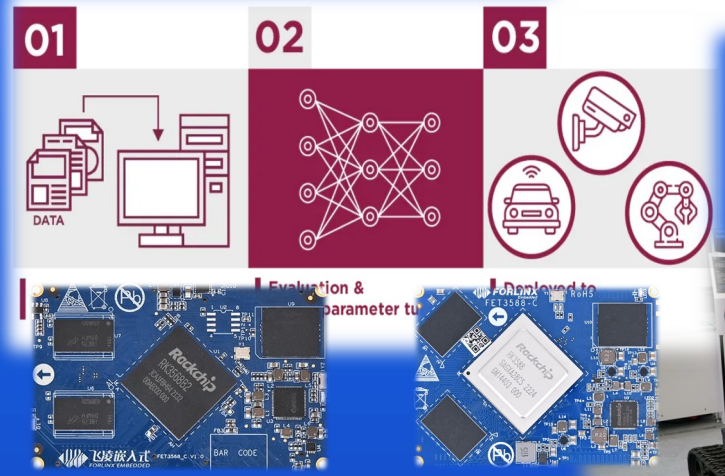
（2）增加端侧AI推理的支持：

为ZVM适配更多国产高性能嵌入式板卡，为**AI模型的推理**提供充足的硬件资源。充分利用底层RTOS的实时特性，加速AI推理。

（3）增加对智能机器人控制系统的支持：

在智能机器人平台上搭建控制系统和人机交互系统，保证**实时控制**机器人的同时，实现较好的**人机交互**能力。

车载TSN网络



端侧AI推理支持

机器人控制系统

THANKS

湖南大学

嵌入式与网络计算湖南省重点实验室

开放原子开源基金会 OPENATOM FOUNDATION | OpenEuler