Hypervisor Design(s)
2018-04-18 GENIVI AMM (Munich)
Dr. Ralph Sasse, Lead Solution Engineer
Typically Linux

Dom0 Linux

DomU (e.g. RTOS)

DomU (e.g. Linux)

XEN Toolstack

Qemu pDRV

App

App

Linux Kernel

RTOS

Linux Kernel

v DRV

p DRV

App

EL0

EL1

EL2

XEN Microkernel

Dev

Dev

VM Control Blocks

VMcb0

VMcb1

VMcb2
KVM (on ARM)

Note: this design may change with ARM v8.3
COQOS

EL0

VM0 (e.g. Linux)

EL1

VM1 (e.g. RTOS)

EL2

VM2 (e.g. Android)

Note: Example core assignment
No core sharing (cores are dedicated to VMs)

Linux loads Jailhouse Hypervisor
Differences in the Designs

**Master VM vs. Equal VMs**
- One VM (Linux or RTOS) is master (e.g. Xen, KVM)
- All VMs are equal

**Static vs. Dynamic Configuration**
- Statically configured → offline check possible
- At run-time configured → more flexibility, but more difficult to validate, API

**Core sharing vs. Dedicated Cores**
- Core sharing requires a scheduler in the Hypervisor

**Small vs. functionality**
- Small is easy to qualify, but restrictions in usage apply (less flexible)

**HW Virtualization Extension (EL2) vs. Para-virtualization**
- Para-virtualization means Maintenance Effort and Performance

**MMU vs. MPU**
- Intended for different Use Cases
  - Small vs. huge SoC
  - RTOS vs. GPOS
Possible VM Allocation Patterns

1 core == 1 VM

Specialized cores (e.g. I/O)

N cores == 1 VM ➔ aka SMP; distribution requires inter-core communication

Shared cores ➔ requires VM scheduling

1 VM == 1 supplier

1 VM == 1 functionality

1 VM == 1 criticality (ASIL) ➔ reduced effort for qualification

1 VM == 1 timing (real-time/best effort)