Automotive Grade Linux
Open Source Low Level Hypervisor

Dominig ar Foll
Intel Open Source
Intel & Open Source
Intel Open Source
Some examples
What is an Hypervisor?

**Not a new technology**
- First used by IBM in 1967 in CP/CMS
- Allows to run multiple OS on the same HW
- Provides some level of isolation

**Type 1 and 2**
- Type 1 runs on bare metal (e.g. Jailhouse)
- Type 2 runs from a Host OS (e.g. VirtualBox)
- KVM blurs the models

**What for**
- Legacy code or alternative OS support
- Isolation (e.g. cyber security requirements)
- Real time sub-system
- Functional safety
AGL and Virtualisation

Virtualisation expert group
- Full virtualisation (kvm/xen)
- Container (name space enabling in AppFW)
- Low level virtualisation (Jailhouse)

Motivation
- Cyber Security
- Functional safety
- Enable legacy code, alternative OS
- Critical Real Time
Cyber Security

Crypto Locker
- May 12th, 2017 blocks Nissan and Renault factories
- Soon enough: our own cars

Private data
- Last trips
- Phone books
- High way remote tolls

Crime
- Remotely controlled “accidents”

Today our factories, tomorrow, our cars
Functional Safety

Reduces complexity
- Isolate critical code
- Simpler code vs full Linux

Still share the SoC
- HW virtualisation
- Multiple core
- Unknown: SoC level firmware

A Smart Coprocessor
- Improved controllability
- Heath check
- Feedback loop

ISO 26262

Functional Safety
Only use Hypervisors when you have to

AGL App/Middleware
• Built outside of the OS
• Installed under supervision from the OS

AGL Jailing system
• Smack security context
• Dynamic Privilege check (Cynara)
• Optional dedicated Name Space and c-group (container mode)

Legacy code
• Cost of port is often low
• Maintenance costs range up to 80% of the total SW costs

Do not add complexity when it’s not required
Hypervisor use shall remain minimal

Real time
- Real time is not fast response
- Linux PrempRT is your friend
- Micro controller / FPGA

Cyber Security
- Hidden security is dangerous
- AGL base is very strong
- Fast update requirement is mandatory

Run Apps as non root
- Low privilege UID
- Dynamic profile from Cloud

1. A feature well born & signed
2. Installed in the system
3. Served to users
4. Appreciated by users
The core AGL architecture

Cluster
- Head Unix
  - Direction Indication

Transport & ACL
- Cluster Virtual Signal
  - Engine-CAN-BUS
  - ABS

Entertainment
- Navigation Service
  - Carte handling
  - Localisation management
  - POI
- CAN-BUS Virtual Signal
  - CAN-BUS
  - LIN-BUS
- Geopositioning Virtual Signal
  - Gyro, Accelerometer
  - CAN GPS

Cloud
- My Car Portal
  - Payment
  - Subscriptions
  - Preference
- Maintenance Portal
  - Know Bugs
  - Maintenances
  - Service Packs

Multi ECU & Cloud Aware Architecture

Transport & ACL
- Cluster Virtual Signal
- CAN-BUS Virtual Signal
- Geopositioning Virtual Signal

Transport & ACL
- Navigation Service
- CAN-BUS Virtual Signal
- Geopositioning Virtual Signal

Transport & ACL
- Cluster Virtual Signal
- CAN-BUS Virtual Signal
- Geopositioning Virtual Signal
Example use case: emergency CAN alert

Default AGL mode
- CAN alert (e.g. brake failure)
- Read by CAN low level binding
- Push to subscribed App(s)
- App take action:
  - display error
  - limit speed
  - locate near service dealer
  - ...

No feedback
- Bug could alter expected behavior
- Bug could be in code (OS/App) or microcode.
- Complexity is too big for static compliance check.
Example use case : emergency CAN alert

With controller
- CAN alert message triggers
  - Push over CAN AGL Binder
  - App feedback request
  - Immediate secure speed mode via CAN
- App manages standard process
  - error message (cluster & heads-up)
  - deactivate cruise control
  - ...
- Send feedback via AGL transport layer

On No Feedback received
- Set Alert LED on Cluster via CAN or gpio
- Set alert buzzer via gpio
- ...

Note: An FPGA could provide the same service
Many valid use cases for Hypervisor

**Controller / feedback check**

- CAN
  - CAN firewall
  - CAN very fast response
  - CAN emergency message
  - ...

- Watchdog
  - Check system App/Middleware health
  - ...

**Real time**

- Benefit dedicated CPU and RAM allocation
- Run nonLinux OS (e.g. Autosar)
Jailhouse

What is it
- Open Source project
  - Originated from Siemens
  - Maintainer: Jan Kizka
  - https://github.com/siemens/jailhouse
  - Active project
- Aim
  - real time & safety tasks
  - Asymmetric Multiprocessing Platforms (AMP)
  - a side of Linux
  - Multi architecture (Intel & ARM)

Key values
- Strong & clean isolation
- Bare metal performances
- Open Source (GPLv2)
- Very small and simple (~3000 lines)
- Configured and initialised from Linux
Jailhouse is NOT

! = KVM

...
Standard Linux boot and update process

1. Boot phase
2. Partitioning phase
3. Operational phase
Jailhouse components

- **Cell Image**
- **Cell Config**
- **Jailhouse Image**
- **Jailhouse Management Tool**
- **/sys/devices/jailhouse**
- **/dev/jailhouse**
- **Linux Kernel**
- **Jailhouse Driver Module**
- **Jailhouse Hypervisor**
- **Hardware**
Jailhouse Management models

**Open Model**
- Linux (root cell) is in control
- Cells not involved in management decisions
- Sufficient if root cell is trusted

**Safety Model**
- Linux controls, but certain cells are configured to vote over management decisions
- Building block for safe operation
Status

What can be configured

- CPU
- Memory regions
  - Read-write-execute
  - DMA, IO, IRQ
  - Comm_Region
  - ...
- Work in progress
  - Cache region
  - Config tools
  - automatic conflict check
  - ARM64 support still new (Huawei over 1 year)

Typical RAM layout (generated and used for QEMU)

- RAM for Linux
- Hypervisor Code & Data
- RAM for non-root cells
- more RAM (optional)

Reserve during Linux boot

- RAM for Linux
- more RAM (optional)

Give to root cell

Reserve physical memory

- memmap=SIZESTADDRESS
- mem=PHYSICAL_SIZE_MINUS_RESERVATION
- Device tree (ARM/ARM64 only)

grub2 is your friend evil

- Use proper escaping in /etc/default/grub
  GRUB_CMDLINE_LINUX_DEFAULT="memmap=66M$0x3b000000"

Give to root cell (initial configuration)
Hypervisors Pros and Cons

**Pros**
- Simple and certifiable
- Bare metal performances
- Run any OS as guest
- Strong isolation

**Cons**
- No coding or configuration standards
- Debugging is far from simple
- Easy to badly use
- Still rely on a shared HW resource
- Micro code / cache conflicts can be a black zone in the certification process.
- Does not work on every SoC
- Most of them requires specific update strategies (not Jailhouse)
Q & A