Graphics Virtualization with L4Re on R-Car3

Adam Lackorzynski – Kernkonzept

GENIVI AMM, Munich, 16. 5. 2019

MICROKERNEL MADE IN GERMANY

Who we are

Kernkonzept is a young OS supplier:

- L4Re microkernel-based open source operating system framework
- Security, Safety, Real-time, Virtualization

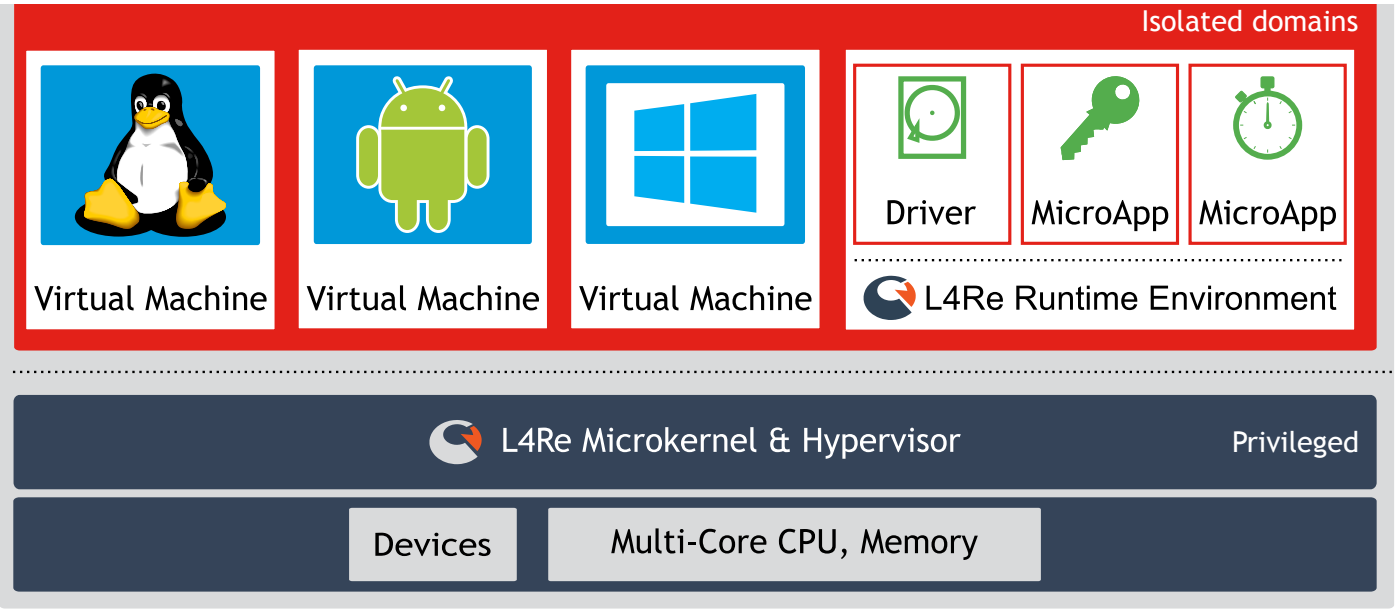
Based in Dresden, Germany

Providing secure OS platform throughout industry use-cases.

High-assurange security, Mobile systems, Automotive, Home appliances, Industrial monitoring and control, Industrie 4.0, ...

Partnering with Elektrobit for the automotive market

L4Re Operating System Framework Overview

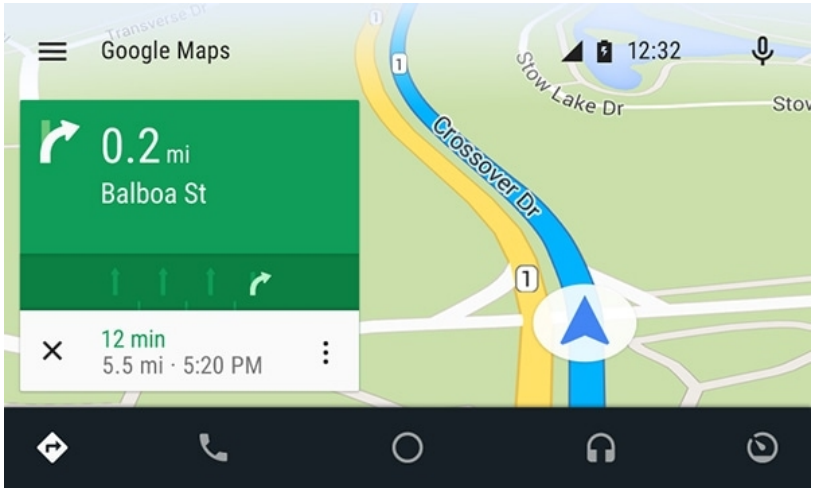


- Microkernel / small Hypervisor
- Small Components to build a system
- Everything in user-level, esp. drivers
- Small Application-specific Trusted Computing Base
- Strong isolation: real-time and secure
- Secure IPC
- Arm, x86, MIPS
- Open Source

Demo with EB Guide: Consolidated IVI & Dashboard

Cluster

Android Auto



Hypervisor

Hardware Platform

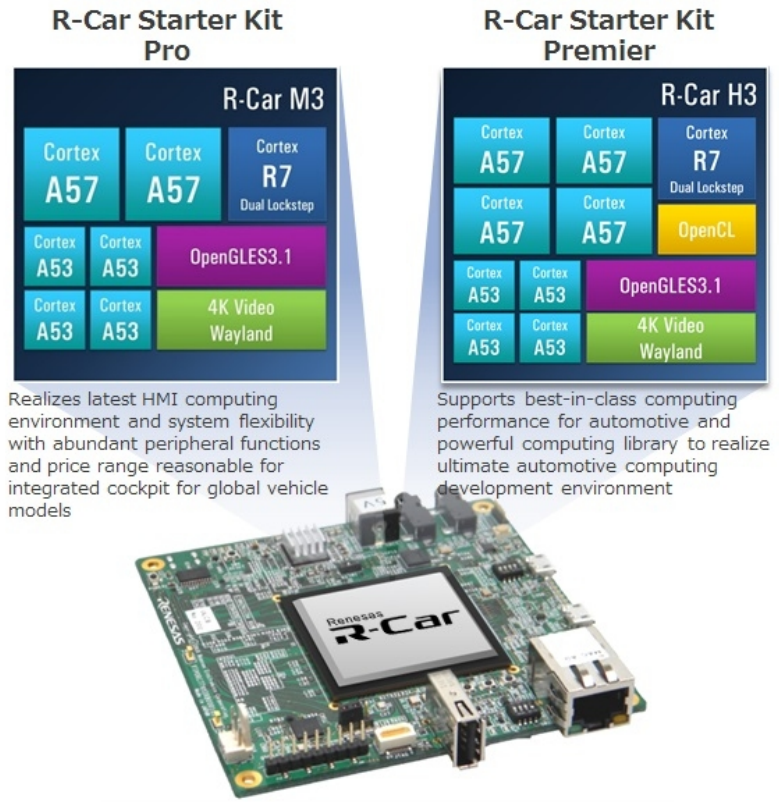
Source: https://d23rjziej2pu9i.cloudfront.net/wp-content/uploads/2018/10/29160352/Elektrobit_13034_04_Cluster-instrument-panel_DA_small.jpg

Source: https://www.android.com/intl/de_de/auto/

Demo Setup: Hardware Platform

Renesas R-Car3 H3 Salvator-XS

- Arm 4x Cortex-A57, 4x Cortex-A53
- Arm Dual-lockstep Cortex-R7
- Graphics: PowerVR Series6XT GX6650
- 4GB RAM
- Storage: eMMC / SD
- 2 HDMI connectors
- Audio



Source: https://www.renesas.com/

Demo Setup: Sharing of Resources & Devices

CPU sharing

Simple approach

- Static assignment of cores to VMs

Memory

- Static partitioning of memory to VMs
- Challenging to select proper regions
  - 32bit / 4GiB
  - Assumptions by guests / Linux
  - Details in next slides

Accessing and Sharing of Devices

Need to share between VMs

- Network
- Storage
- Graphics
- Input

Exclusive use by VMs - Path-through access

- One display per VM
- Audio: one VM only

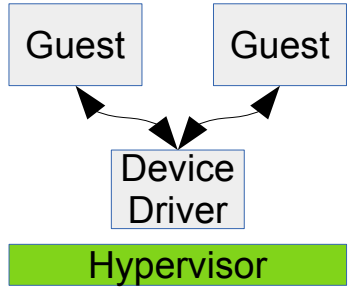
Ways to Share Devices between VMs

Where is the device driver?

- Hypervisor service
- VM

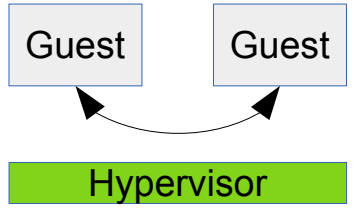
Hypervisor Driver Service

- Write driver from scratch (Heavy effort)
- Port some other driver (Considerable effort)



VM

- Easy to run driver, connect to other VM (Straight forward)



VirtIO for Inter-VM Communication

VirtIO as generic and widely available communication protocol

Support by common guests: Linux, Android, *BSD, QNX, …

- No need to provide guest support
- Works out of the box

Common device classes

- Console
- Storage / Block
- Network

Different design options...

VM as Device Driver

Easy access to drivers

Driver runs in native environment

Device pass-through to VM

Network:

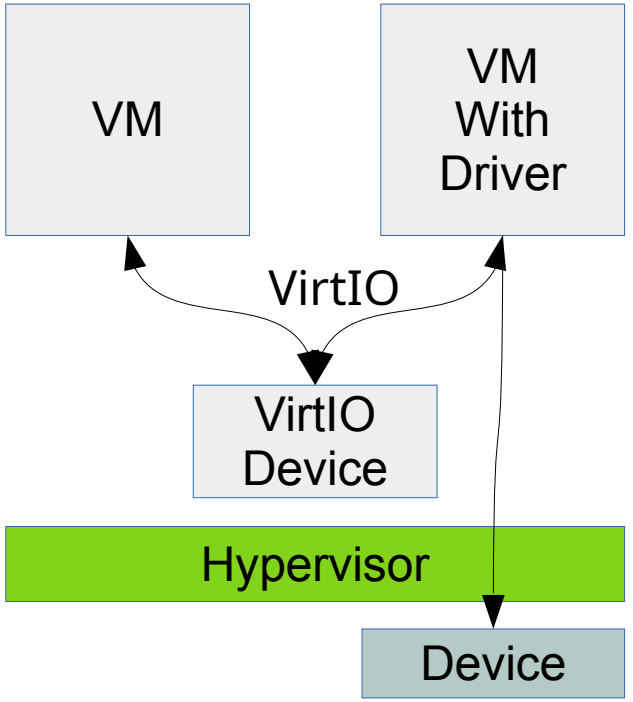
- Easy as both VMs are equal

Storage:

- Needs helper in driver VM

Properties:

- Availability: ✗
- Confidentiality + Integrity: ✗
- Confidentiality + Integrity with crypto: ✓

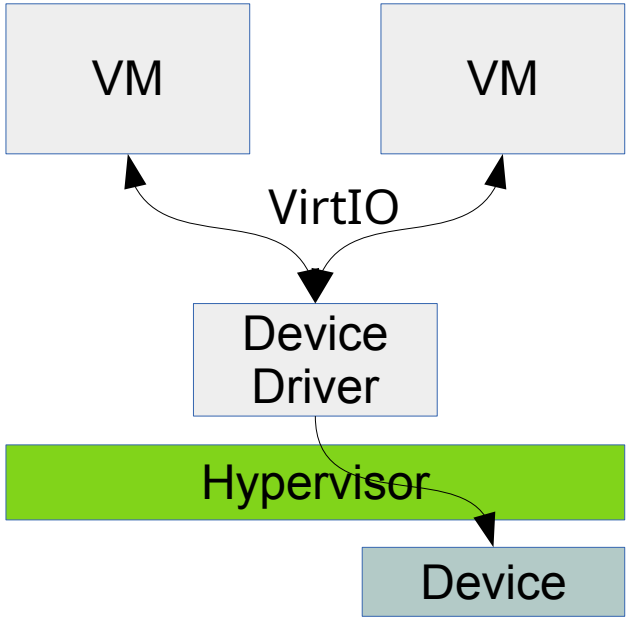


Driver in Hypervisor Component

Hypervisor component needed for security and safety

Availability: ✓

Confidentiality + Integrity: ✓



Devices with Virtualization Support

Device-passthrough to each VM

No hypervisor driver needed, no indirection

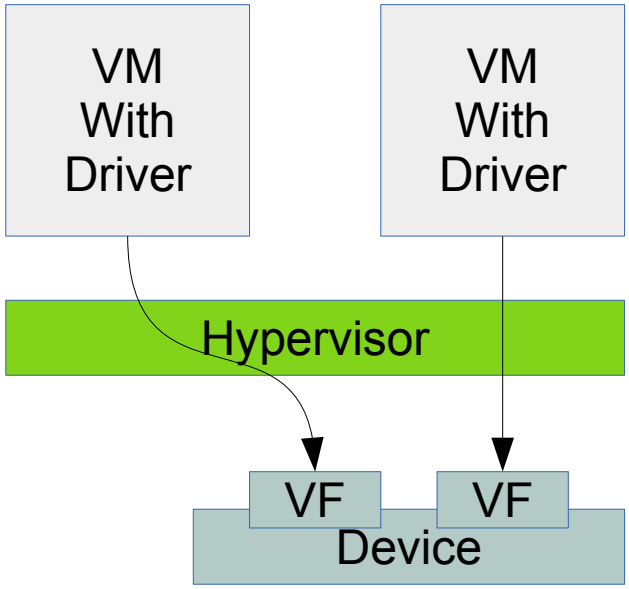
Best performance

Policy on client handling implemented by device

- E.g. QoS handling, might not fit

Requires virtualization-aware devices

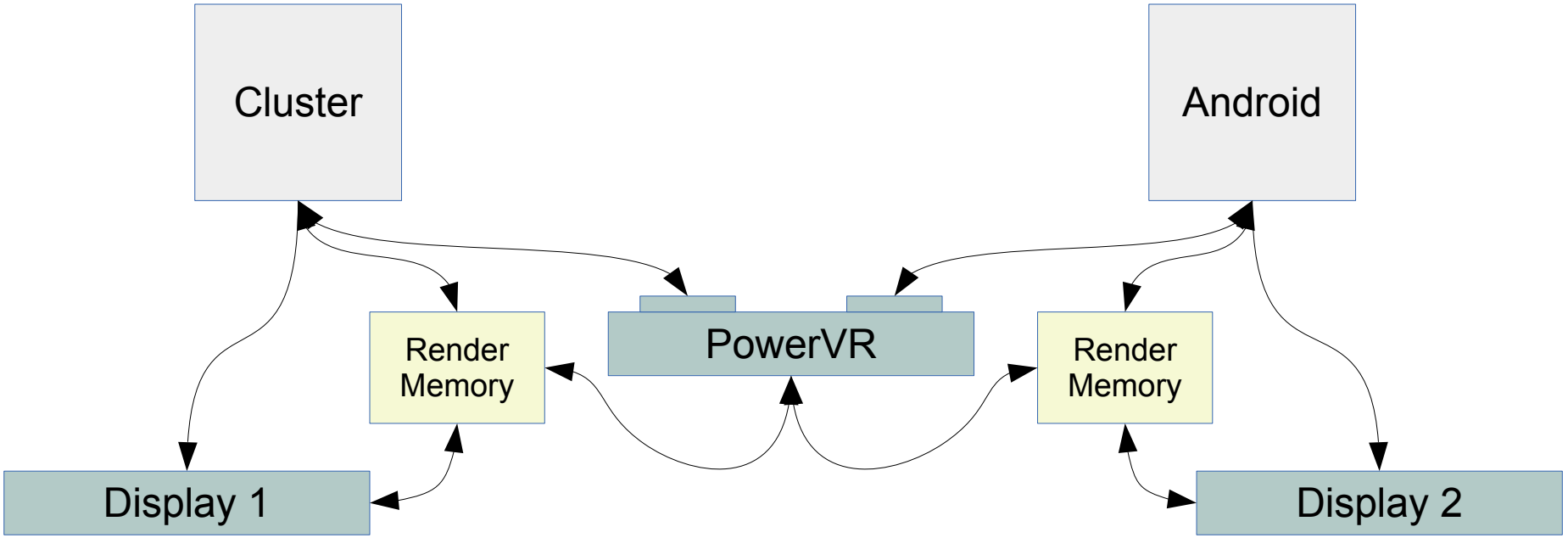
- Still seldom, esp. in embedded systems



Graphics on the R-Car3

Graphics: Imagination PowerVR Series6XT GX6650

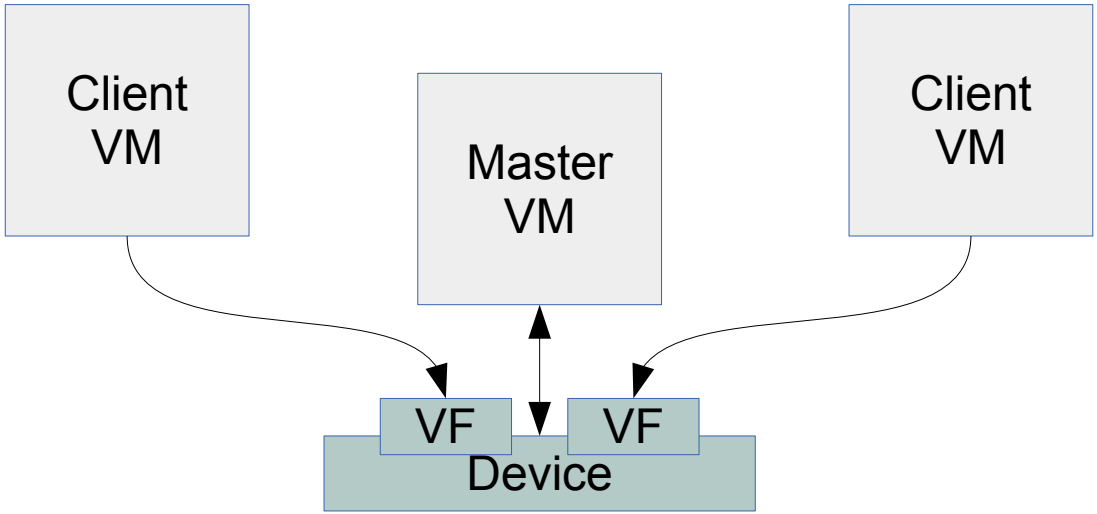
Virtualization capable



Graphics on the R-Car3

Caveats

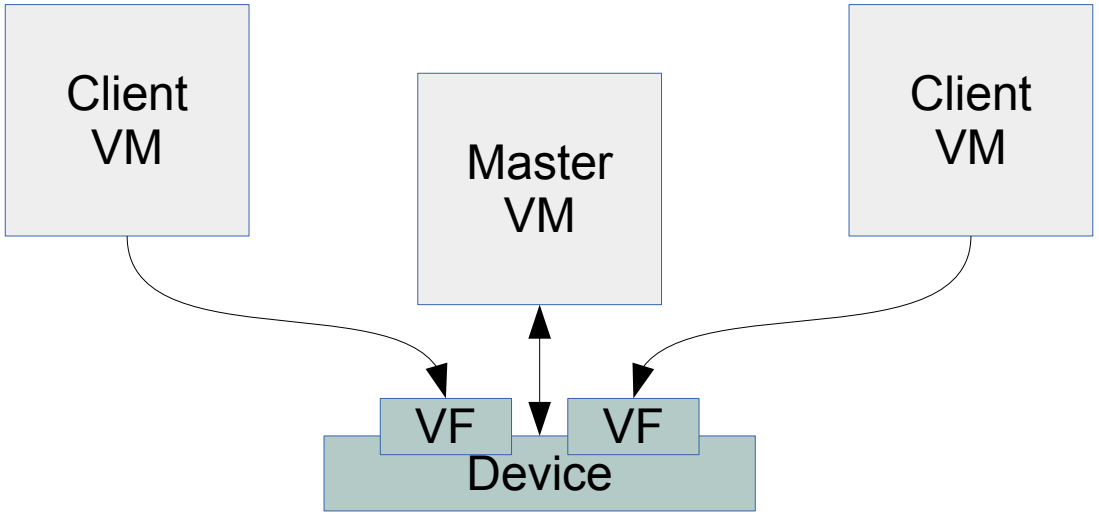
- Need to configure graphics controller – Done in 3rd VM (master)
- Needs proprietary & binary-only drivers in master VM and client VM



Graphics on the R-Car3 - Drivers

Drivers need to have the exact same version between master and client

- It was difficult to get builds of same versions for both Linux and Android



Memory Setup of VMs

Native Linux configurations assume specific physical memory layout

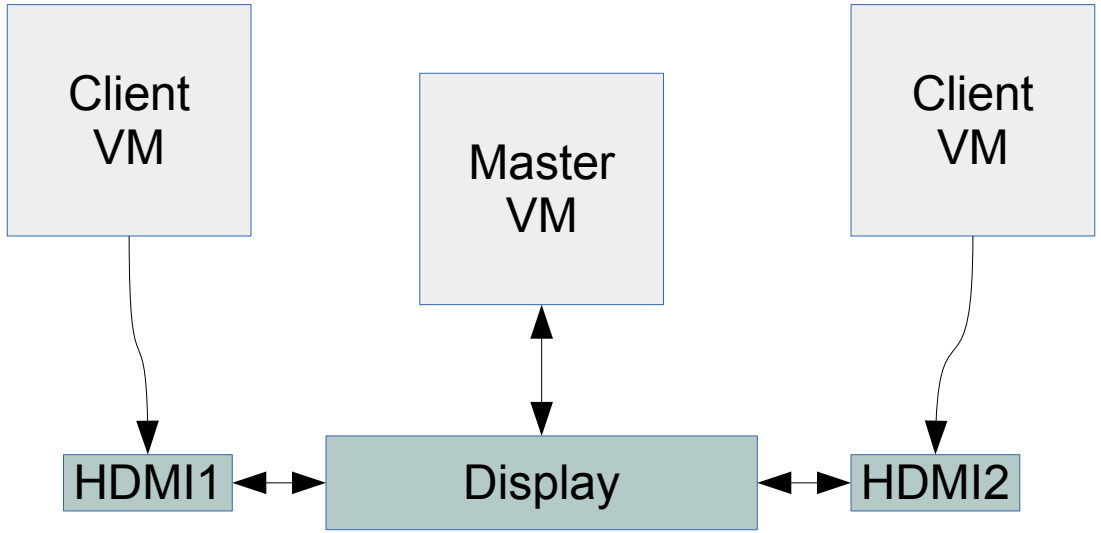
Difficult/tricky to arrange in virtual setups

- Needs careful selection of memory regions to assign to VMs
- Multiple regions per VM required

Memory below 4GiB (32bit) is scarce, especially with more VMs

Configuring the Displays

- Global initial setup required for display setup
- Then: Take-over by client VMs
- Requires arbitration between master and client VMs



Take-away: Virtulization-aware SoCs

Need SoCs and vendors that actively support virtualization

- Hardware should support multiple clients/VMs for performance
  - Network, storage, graphics, audio, …
- DMA-capable devices must use 64-bit addressing
- Solid IOMMU support
  - Per device and/or device function, e.g. PCI devices, DMA channels, etc.
- Vendors should provide drivers for guests and hypervisors/hosts (neutral)
  - Generic interface between guest and host?
- No hard-coded physical memory addresses
- Separation of devices per 4k page (as demanded by MMU protection granularity)
  - No co-location on the same page of multiple devices or facilities, e.g. multiple DMA channels

www.kernkonzept.com

l4re.org

MICROKERNEL MADE IN GERMANY