



~~Virtio and friends~~ GENIVI AMM Munich  
Matti Möll, OpenSynergy GmbH

- **Platform Standardization**
- **Looking at Virtualized platforms**
- **Comparing techniques**
- **Virtio**
  - **What it does**
  - **How it works**
- **Summary**

- **Defacto standard for personal computers „IBM PC“ and „PC Compatible“**
  - Hardware and software interface
  - BIOS
  - Peripheral standards (ISA, ATA)
  - Platform design became the x86 Architecture
- **PXE**
  - Preboot Execution Environment
  - Early execution environment for Extension ROMs and Netboot
- **UEFI**
  - PXE + BIOS + Bootloader
  - Extensible preboot and runtime environment
  - Provides services and HALs

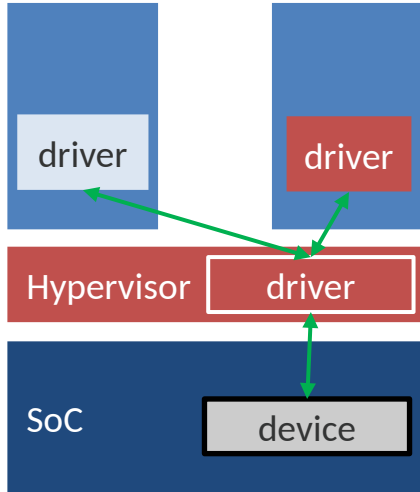
- **Desktop Virtualization (early days)**
  - Almost all x86
  - Replicate desktop system
  - Emulated IDE controller, Intel Network card, VGA graphics, etc.
  - Drivers exist, interfaces already standardized
  - Vendor specific device models
- **Server Virtualization**
  - x86 (and a couple of mainframes)
  - Vendor Specific device models
  - Mostly network and storage
- **Cloud Virtualization**
  - Vendor specific
  - Virtio based models
  - Almost only network and storage

- **Virtualized IO devices are available for desktop and cloud applications because everyone uses standardized interfaces (virtio, xen, vmware)**
  - Disk
  - Network
- **Embedded devices lack the ecosystem that cloud providers build upon**
- **Challenges for virtualized IO devices in automotive**
  - High effort of SoC specific device virtualization
  - Multimedia device virtualization
  - Low amount of reusable virtual devices

- Mechanisms
- Where do we cut?
- COTS or domain specific?
- Strict requirements or rough consensus?

Focus on I/O, virtual/shared devices and drivers

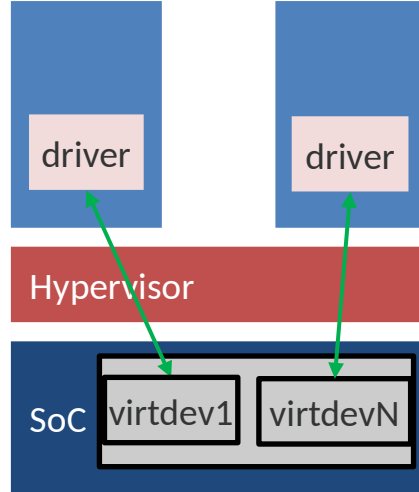
in Hypervisor



- Only used for UART (optionally)
- not recommended for other devices as the Hypervisor is minimalistic.

Example: UART

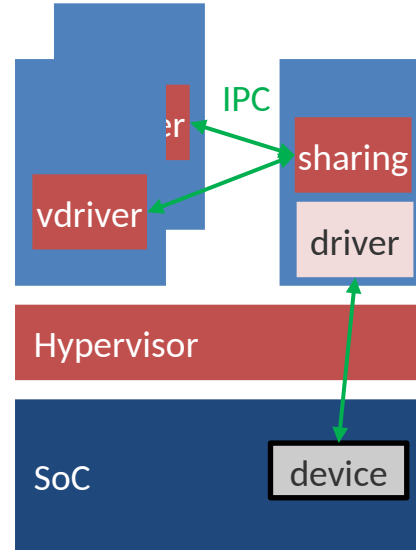
device with virtualization support



- COQOS supports this when the SoC hardware supports virtualized devices
- Recommended wherever the hardware supports it, as it tends to give the best performance and separation

Example: GPU on RCAR-H3

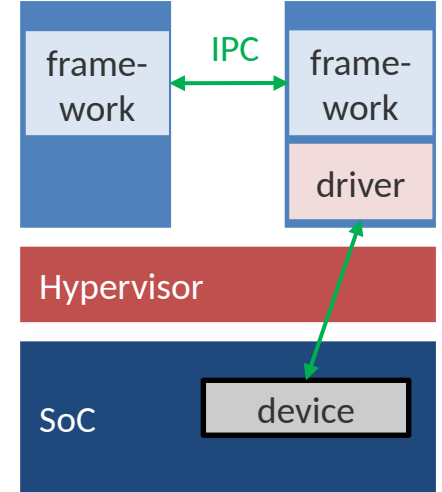
low-level client-server



- Single driver in VM that acts as "server"
- Driver-specific sharing logic is needed.
- Other VMs use "virtual driver".
- Compromise between performance and flexibility

Example: shared block device

distributed frameworks over Guest IPC



- Allows reuse of existing frameworks for distributed applications in a virtualized environment over VNET.
- Supports complex sharing semantics at the cost of more overhead

Example: NFS

Technology	Description	Reusability	Platform independence
Standard library (or layer) virtualization (OpenGL, DRM, Android HAL ...)	Implement hypervisor specific standard libraries	As long as the same hypervisor is used	As good as vendor interface
Virtio	Implement virtio based devices that follow either existing standards or specify new ones	Virtio support is available in Linux, Android and many other operating systems	Builds upon the kernel-userspace interface of Linux and allows large flexibility because the devices themselves make no assumption about the hardware
HV vendor custom	Develop virtual devices optimized to be used with a particular hypervisor	As long as the same hypervisor is used	Implementation specific

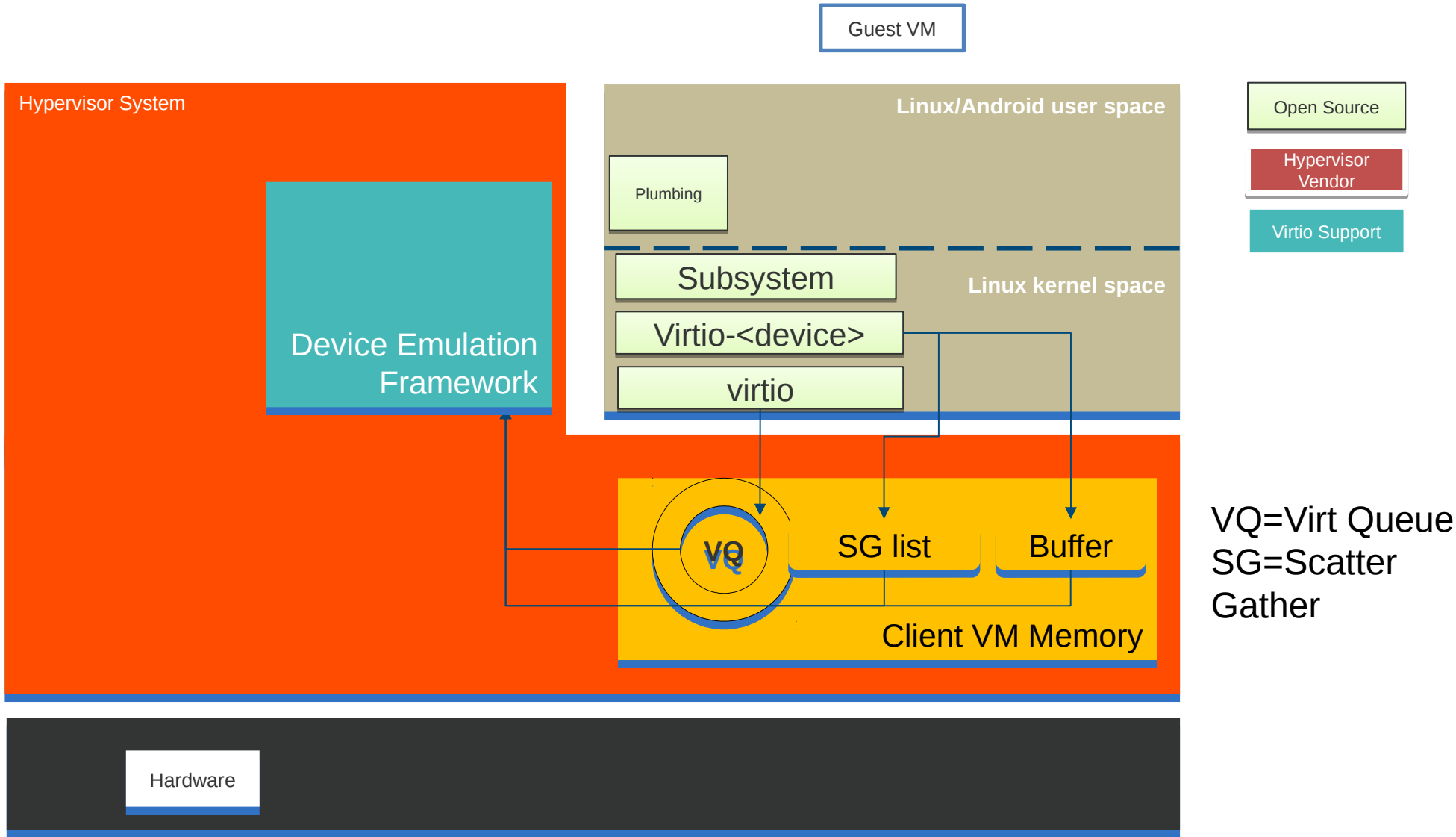
Trade-off between development effort, reusability, platform independence, availability and maturity

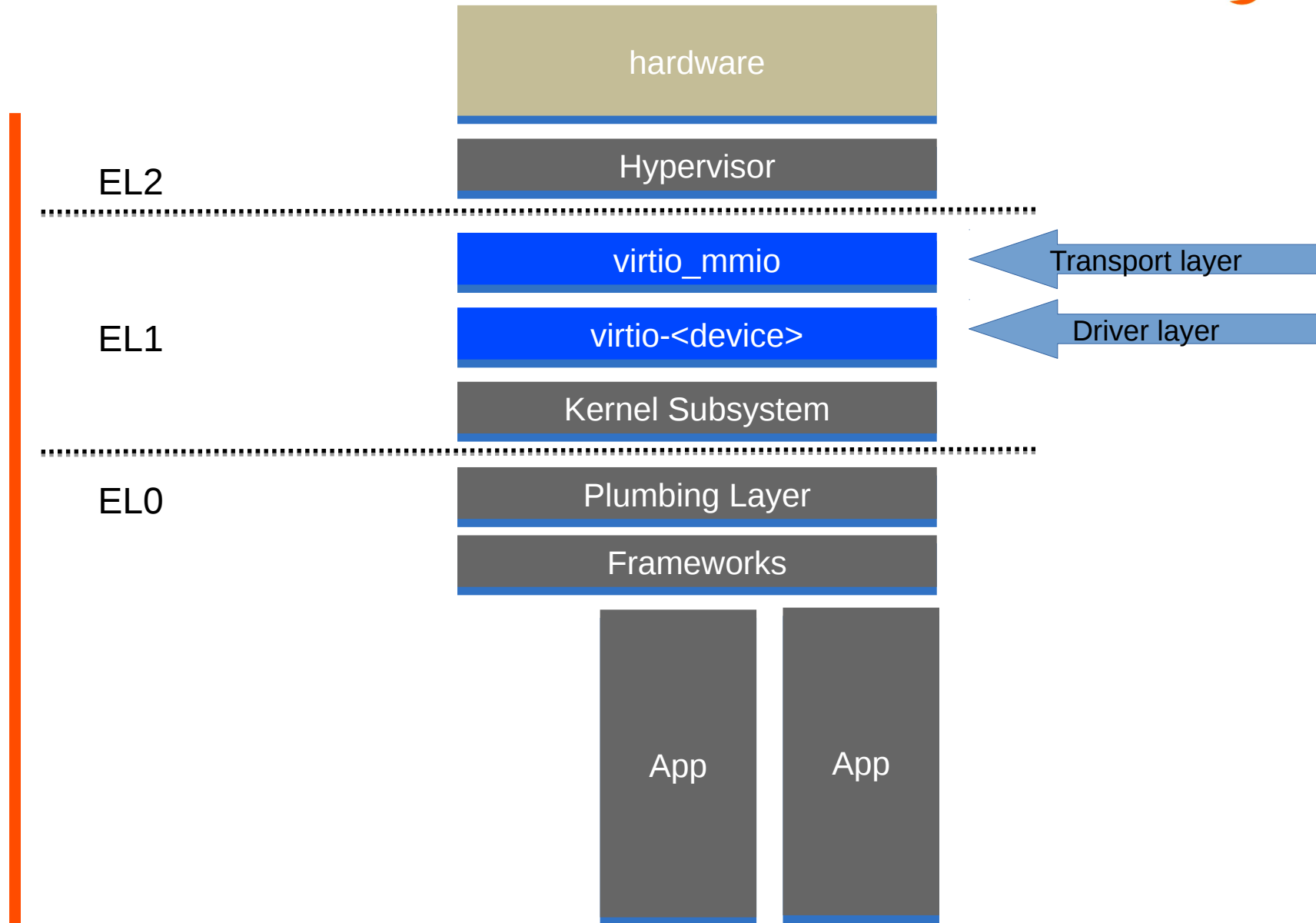


Let's assume virtio

- **Virtio “De-Facto Standard For Virtual I/O Devices” [Russel 2008]**
- **Formally standardized since March 2016 (OASIS VIRTIO-v1.0)**
- **Virtio provides interfaces for many devices**
  - Block Storage
  - Network
  - Console
  - GPU
  - Input (hid)
  - Crypto device
  - vSock
  - File Server (9pfs)
  - Many more in development (vIOMMU, etc.)
- **For the complete Android experience there are still missing pieces**
  - Audio
  - Sensors
  - Media Acceleration Offload (VPU)

- **Device** refers to the implementation of the virtual/para-virtual device, also known as Backend or Server
- **Driver** refers to the guest driver, also known as Frontend or Client
- **Device Host** is the guest that provides the Device to other guests
- **Device Guest** is the consumer of a Device
- **Guest** is a partition or virtual machine





- **The driver decides where memory is allocated**
  - Drivers define allocation policy
  - Pooling possible but not needed
  
  - No Bounce buffers, no wasted memories if system requirements allow this

# Why Virtio?

- **Standardized**
- **Proven in Use**
  - Well tested device models
- **Established community**
  - IBM, Red Hat, Siemens, Huawei, Oracle, ARM, Intel
- **Efficient and performant**
- **Diverse operating system support**
  - Linux, BSD, Windows, UEFI
  - Driver maintenance done upstream
- **Supported by many VMMs and Clouds**
  - Qemu, kvm-tool, Foundation model
  - ARM Foundation model/ Fast model
  - Google Compute Cloud, DigitalOcean, OHV

- **Creating platforms with multiple independent providers has fueled innovation in the past**
- **PC, Servers, Cloud**
- **Proposal: Standardize on virtio for I/O devices**
- **What already exists and how does virtio work**
- **Virtio is standardized and designed to allow interoperable and independent implementations**



# Discussion



- Can virtio provide the I/O interfaces we need?
- What's missing?
- Level of standardization?
- What's next?



## OpenSynergy GmbH

Rotherstraße 20  
D-10245 Berlin  
Germany

Phone +49 30 60 98 540-0  
E-Mail [info@opensynergy.com](mailto:info@opensynergy.com)  
Web [www.opensynergy.com](http://www.opensynergy.com)

## OpenSynergy GmbH

Starnberger Str. 22  
D-82131 Gauting / Munich  
Germany

Phone +49 89 89 34 13-33  
E-Mail [bluetooth@opensynergy.com](mailto:bluetooth@opensynergy.com)

## OpenSynergy, Inc. (USA)

765 East 340 South  
Suite 106  
American Fork, Utah 84003

Phone +1 619 962 1725  
E-Mail [bluetooth@opensynergy.com](mailto:bluetooth@opensynergy.com)

OpenSynergy, COQOS SDK, Blue SDK, IrDA SDK, Voice SDK, Update SDK, Qonformat, and other OpenSynergy products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of OpenSynergy GmbH in Germany and in other countries around the world. All other product and service names in this document are the trademarks of their respective companies. These materials are subject to change without notice. These materials are provided by OpenSynergy GmbH for informational purposes only, without representation or warranty of any kind and OpenSynergy GmbH shall not be liable for errors or omissions with respect to the materials. © OpenSynergy GmbH 2016