SIEMENS

Siemens Corporate Technology | May 2013

Virtualizing Real-Time Systems with Linux

Image: Marcus Quigmire, licensed under CC BY 2.0

Virtualizing Real-Time Systems with Linux

Agenda

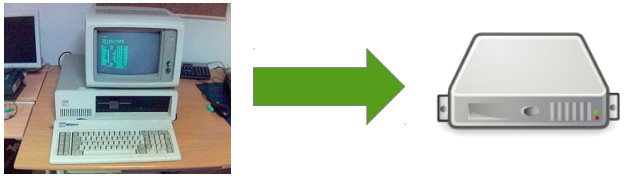
4 Main Reasons to Use Virtualization

Consolidation

- Energy saving
- Lower bill of material
- Maintenance reduction
- Integrate multiple levels of safety & security

Legacy system migration

- Stable hardware interfaces
- Multicore exploitation



High availability

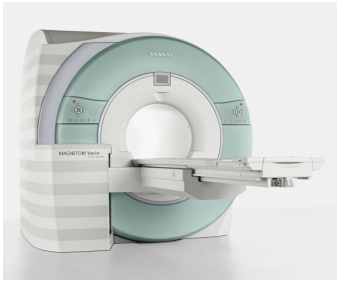
Development & test environment

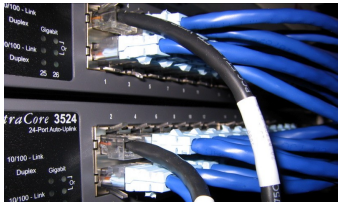
Images: RRZE icon set, licensed under CC BY-SA 3.0, XT PC by Sherool, licensed under CC BY-SA 3.0

Real-Time Systems Can Benefit from Virtualization

Virtualizable real-time systems

- **Possible scenarios**
  - Control systems (industry, healthcare, etc.)
  - Communication systems
    (media streaming & switching, etc.)
  - Trading systems (stocks, goods, etc.)
  - …
- **Primary drivers**
  - Consolidation
  - Legacy system migration
  - [Development & test]





Images: Ethernet switch by Ben Stanfield, licensed under CC BY-SA 2.0, stock market by Katrina.Tuliao, licensed under CC BY 2.0

Virtualizing Real-Time Systems with Linux

Agenda

- Motivation & Background
- **Real-Time Virtualization Basics**
- Linux/KVM-based Setup
- A New Approach for Lowest Latency
- Summary

RT Guests – From Very Simple to Ordinary OS

Characteristics of Real-Time Guests

Guest types

- Bare-metal (no OS)
- Classic real-time operating system (RTOS)
- Asymmetric multi-processing setup (RTOS + GPOS)
- General purpose OS with RT requirements

Hard RT guest resources

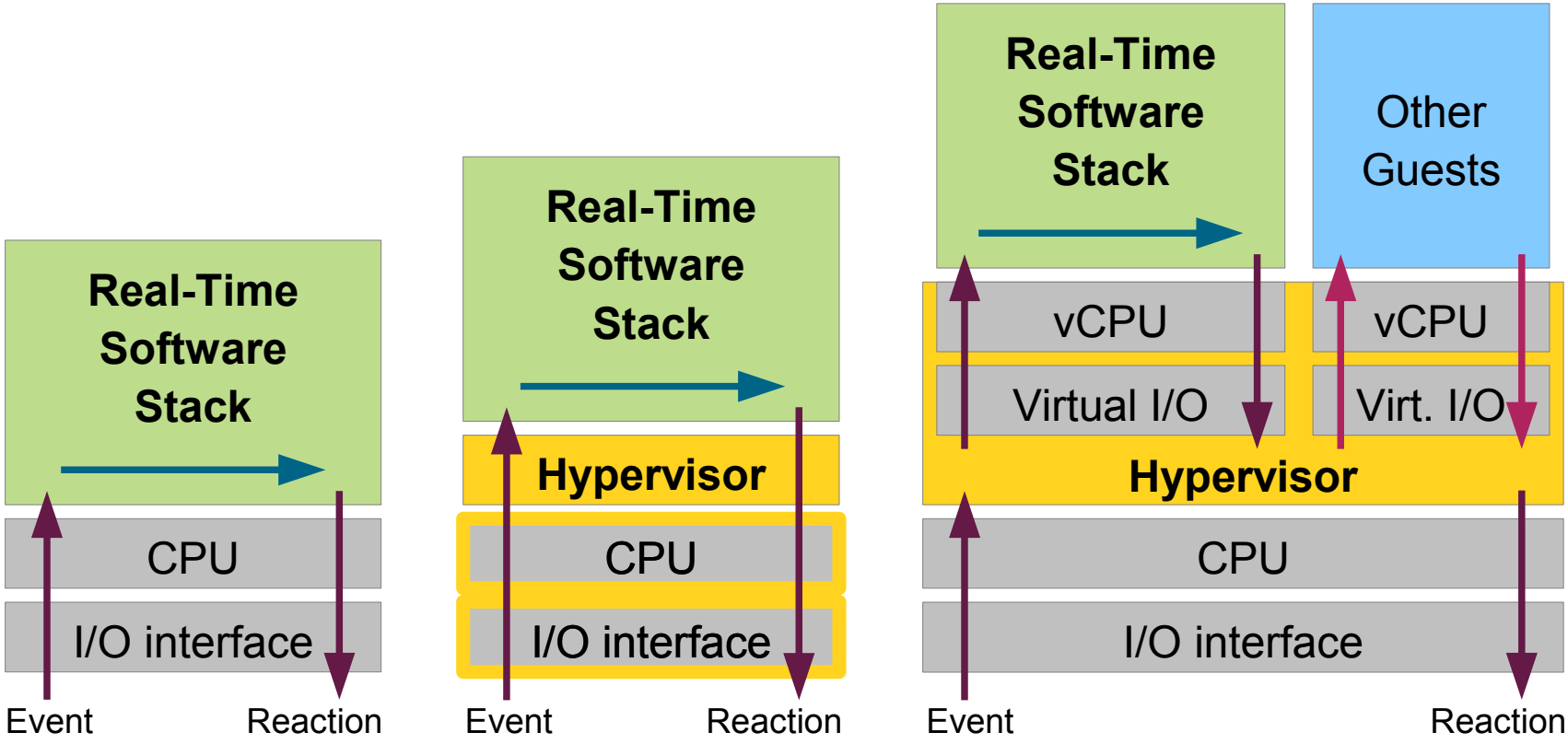
- CPU (bandwidth, event latency)
- Timers & clocks
- RT networks (RT Ethernet, fieldbuses, ...)
- Digital/analogue I/O interfaces
- Custom hardware

Soft RT guest resources

- Anything on the left
- Standard networks
- Mass storage
- Human-machine interfaces
- ...

Virtualization ≠ Acceleration

The Critical Data Path with and without Virtualization

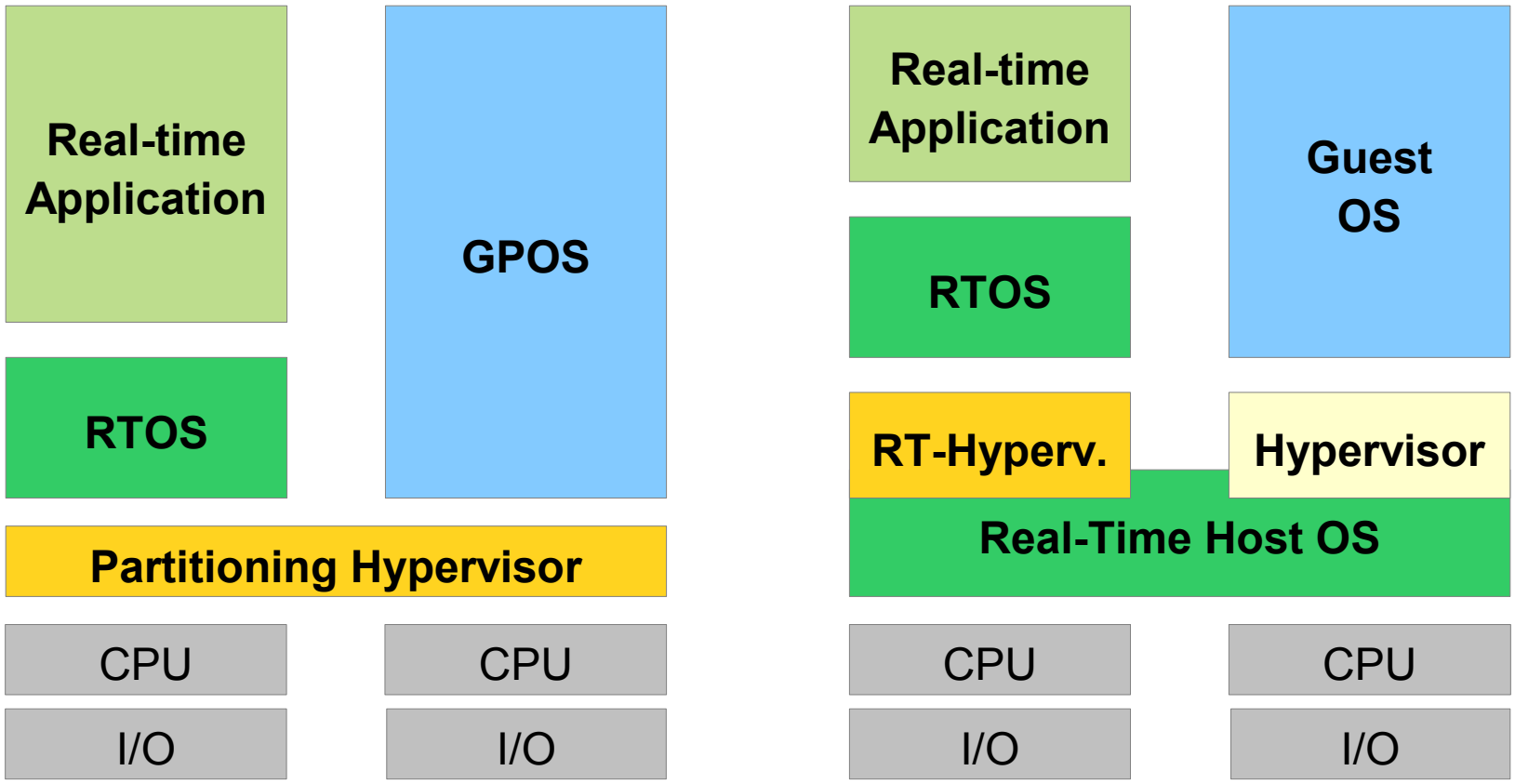


Native real-time setup

Hardware-assisted virtualization

Resource sharing & abstraction via emulation

RT Virtualization – Two Architectural Options



SIEMENS

No “One Size Fits it All”

Comparison of RT-Virtualization Approaches

Partitioning hypervisor

- **Reduced hypervisor complexity**
  - Simplifies safety/security validation
  - Can provide better real-time properties
- **Mostly static assignments**
- **No or limited overcommitment**
- **Duplication of system boot-up logic**
- **Often requires paravirtualization**
- **Non-standard management interfaces**
- **No industry-grade free solutions available**

RT-enhanced hosted hypervisor

- **Reuse of exisiting solutions possible**
  - Works well with common guests
  - Up-to-date virtualization features
  - Large user base
  - Free Linux-based solution available
- **Features vs. complexity**
- **Adding RT support can be non-trivial task**

Virtualizing Real-Time Systems with Linux

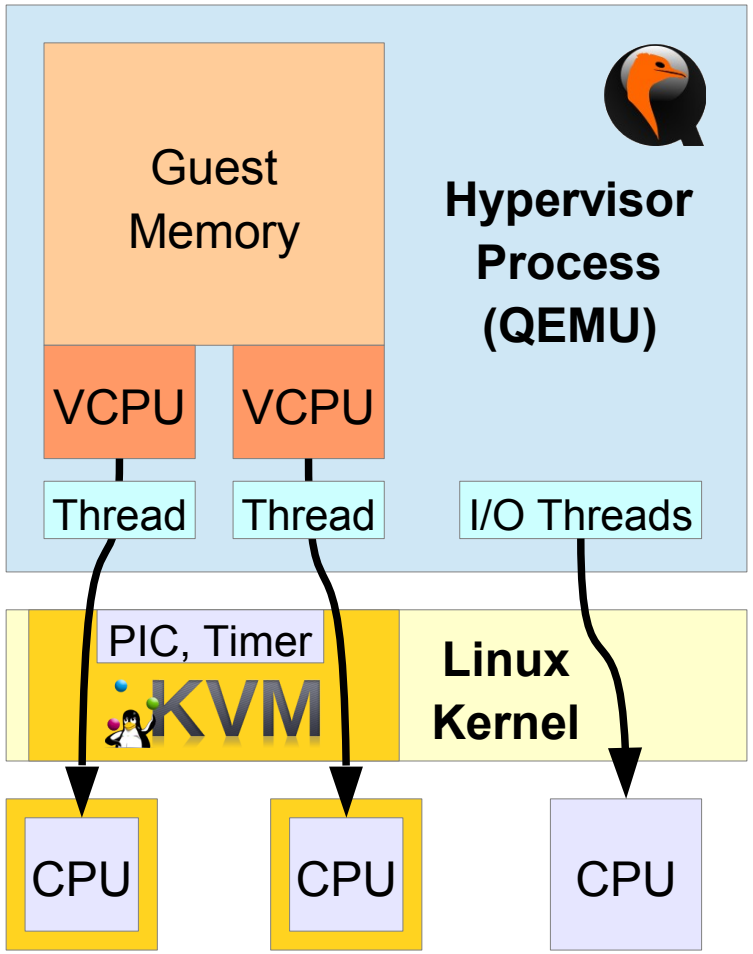
Agenda

- Motivation & Background
- Real-Time Virtualization Basics
- **Linux/KVM-based Setup**
- A New Approach for Lowest Latency
- Summary

KVM – the Swiss Knife among Hypervisors

Main Components of the QEM/KVM Hypervisor Stack

- **KVM subsystem**
  - Gatekeeper for HW- and kernel-assisted virtualization
  - Virtual CPUs
  - Fast device models
- **Use Linux for other hypervisor tasks**
  - Scheduling
  - Memory management
  - I/O stacks
  - Device pass-through (VFIO)
  - ...
- **User-space hypervisor QEMU**
  - Multi-architecture machine emulator
  - Can use KVM for virtual CPUs
  - Large set of device models



SIEMENS

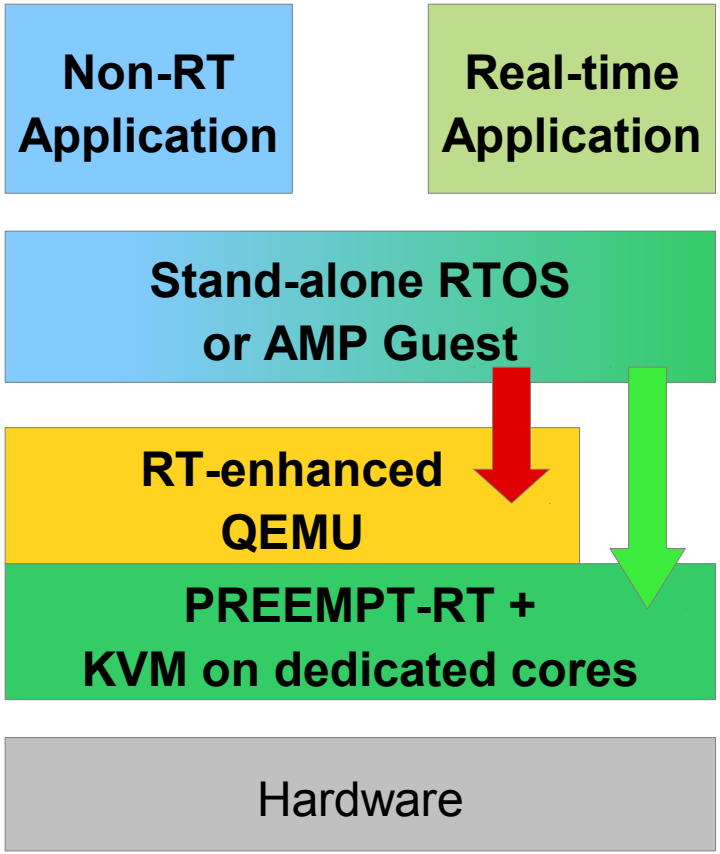
PREEMPT-RT enables RT-Virtualization

Role of Linux extension PREEMPT-RT

- **PREEMPT-RT: fully preemptible Linux kernel**
- **Typical worst-case event delivery latencies: few 10 microseconds**
- **Integrates KVM support**
  - Original use-case: virtualization + native RT applications
- **Allows to prioritize virtualization workload over uncritical tasks**
- **Can be combined with CPU isolation**
  - 1:1 assignment: host CPU – RT guest CPU
  - Off-load all non-RT tasks
    (including low-priority QEMU threads)
  - Warning: No 100% guest CPU load feasible!
    - NO_HZ_FULL extensions work toward enabling this

SIEMENS

Architecture of a KVM-based RT-Hypervisor

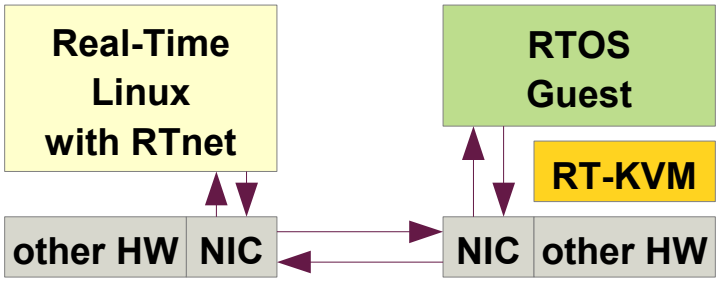


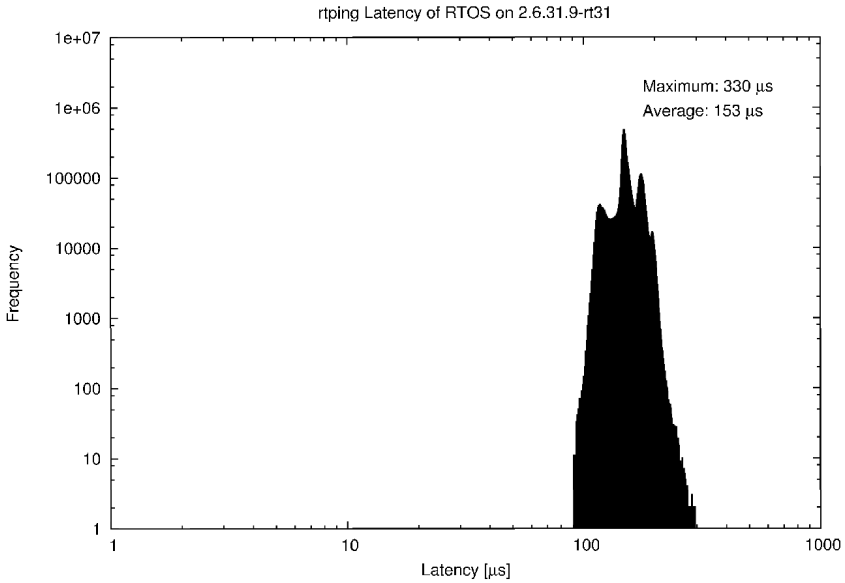
Decent Latencies Achievable in KVM-only Setups

Measuring I/O latency of an RT Guest

- **Host setup**
  - KVM on x86 PREEMPT-RT Linux
  - Virtual machine on dedicated core
  - Intel NIC (E1000 family) as I/O device, directly assigned to guest
  - Permanent disk I/O load
- **Guest setup**
  - Proprietary RTOS
  - Real-time network stack
- **Measurement setup**
  - Linux/Xenomai (native installation)
  - Real-time network stack RTnet
  - Periodic ICMP ping messages sent to target
  - Recorded round-trip latency (error <50 µs)

=> Worst-case latency after 16h: 330 µs

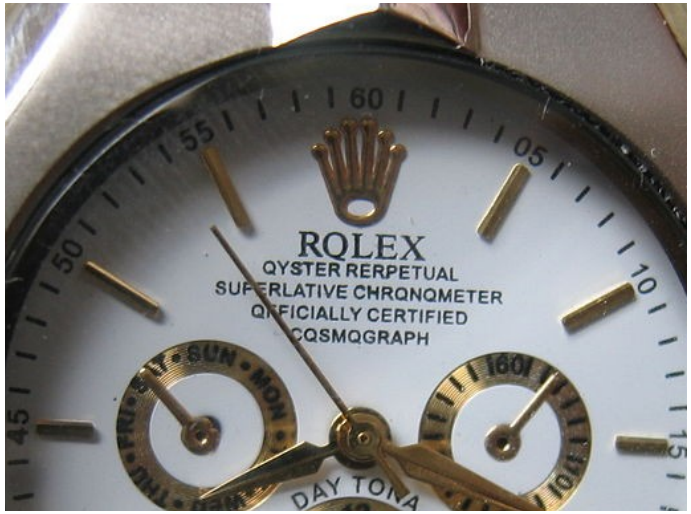




RT-QEMU is Required for Emulating in Real-Time

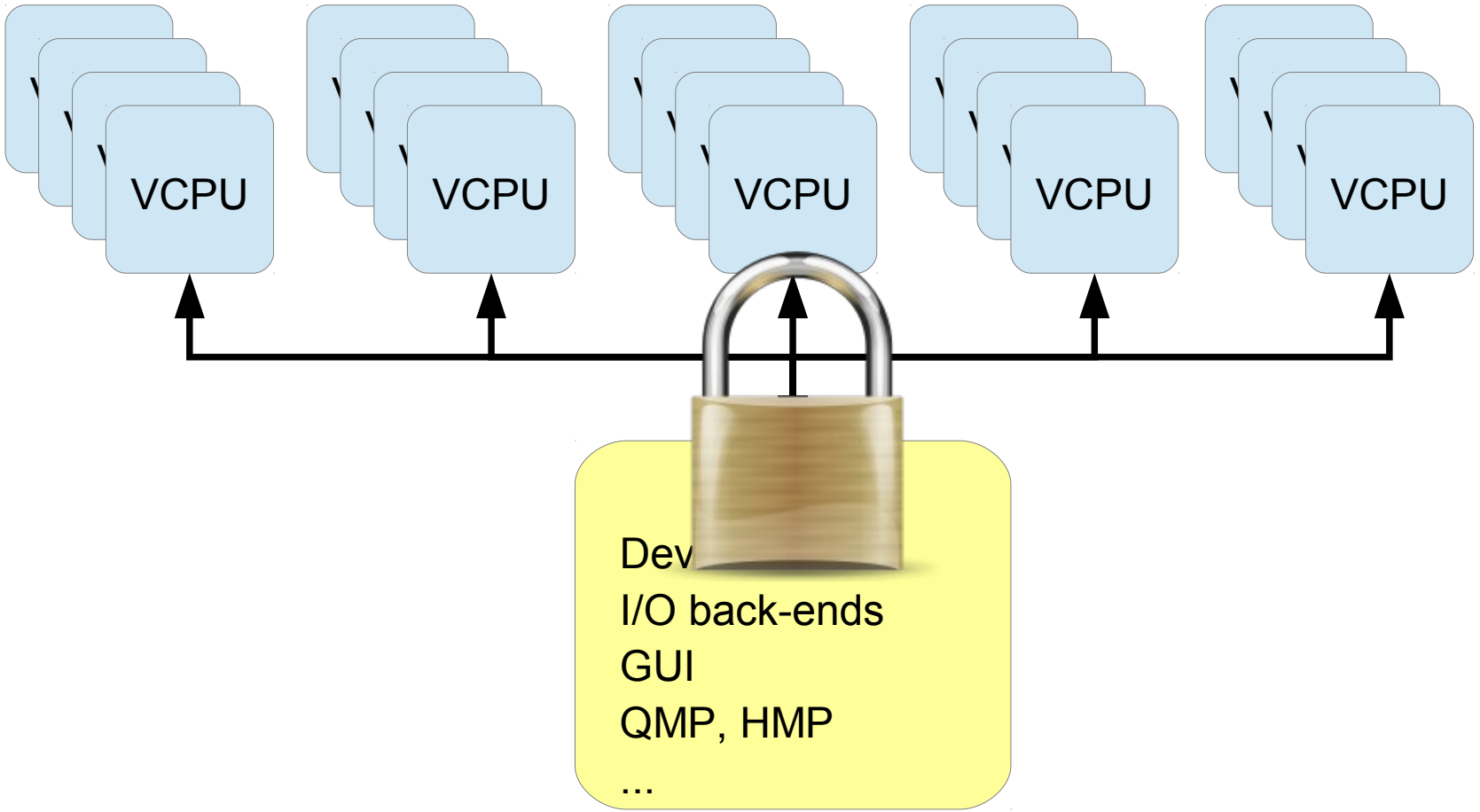
QEMU as a Real-Time Device Emulator

- **Scenarios**
  - Guest uses NIC A, host has NIC B attached
  - Legacy devices are no longer available on a modern host
  - Multiple guests share one I/O interface for talking to different devices (e.g. on a CAN bus)
- **QEMU can handle such scenarios via emulation**
- **Requirements on emulation**
  - Equivalent functional behavior
  - Devices models need to react in time on guest requests
  - Devices models need to deliver external events to the guest timely

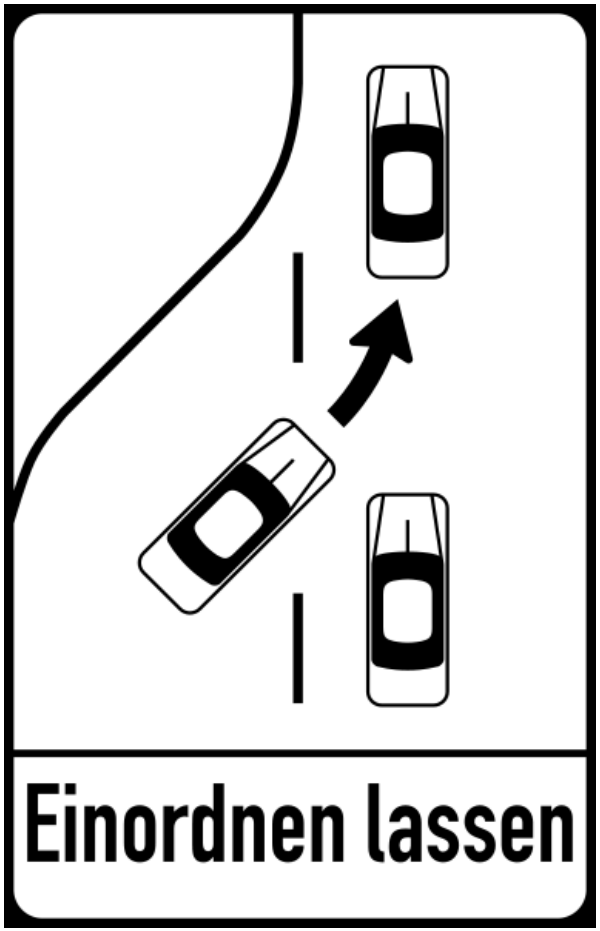


Concurrency in QEMU/KVM – The Big QEMU Lock (BQL)

SIEMENS



BQL – One after the other



SIEMENS

Breaking the BQL is Challenging but Feasible

Challenges in Reworking QEMU's Concurrency Scheme

- Establish lock ordering rules
- Prevent lock recursions
- Design efficient object locking and event dispatching scheme
  - Which objects to track?
  - Reference counting? Mutexes? RCU?
- Enable smooth transition from old to new model
  - Existing code base cannot be converted in a single step
  - Both models must coexists
- Incrementally convert device models and backends
- Teach new model to developers

SIEMENS

Critical BQL Zones

CPUState

- Read/write access
- cpu_single_env

Coalesced MMIO flushing

PIO/MMIO request-to-device dispatching

Back-end access

- TX on network layer
- Write to character device
- Timer setup, etc.

Back-end events (iothread jobs)

- Network RX, read from chardev, timer signals, …

IRQ delivery

- Raising/lowering from device model to IRQ chip
- Injection into VCPU (if user space IRQ chips)

Virtualizing Real-Time Systems with Linux

Agenda

- Motivation & Background
- Real-Time Virtualization Basics
- Linux/KVM-based Setup
- **A New Approach for Lowest Latency**
- Summary

SIEMENS

No “one size fits it all”

Comparison of RT-virtualization approaches

Partitioning hypervisor

- Reduced hypervisor complexity
  - Simplifies safety/security validation
  - Can provide better real-time properties
- Mostly static assignments
- No or limited overcommitment
- Duplication of system boot-up logic
- Often requires paravirtualization
- Non-standard management interfaces
- No industry-grade free solutions available

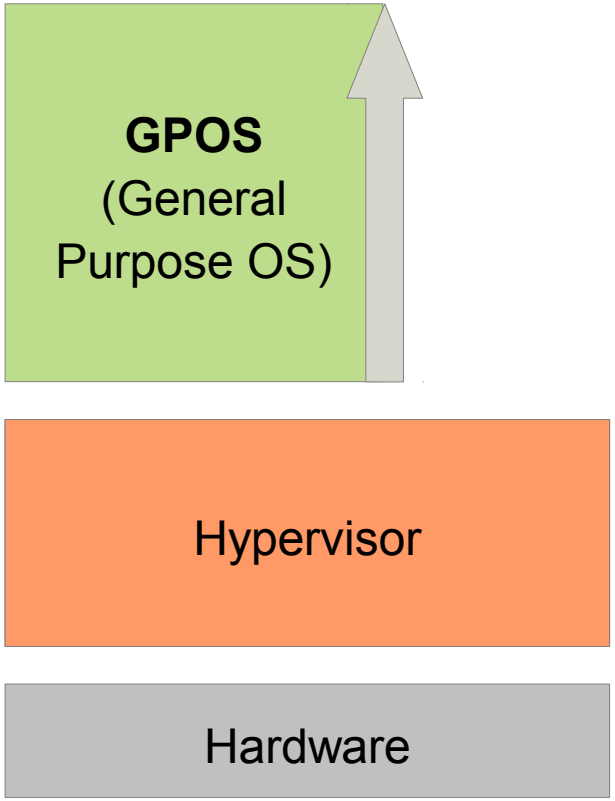
RT-enhanced commodity hypervisor

- Reuse of exisiting solutions
  - Work well with common guests
  - Up-to-date virtualization features
  - Large user base
  - Free Linux-based solution available
- Features vs. complexity
- Adding RT support can be non-trivial task

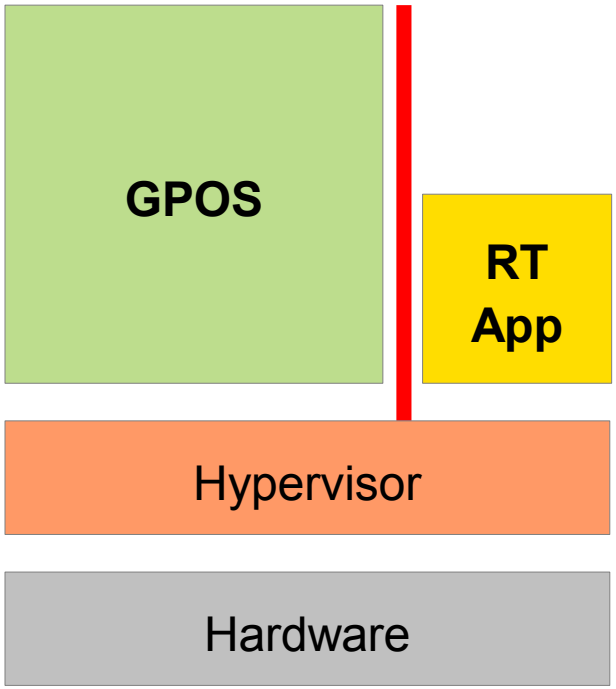
SIEMENS

A bare-metal hypervisor has to boot its guest

Classic type-1 hypervisor boot-up



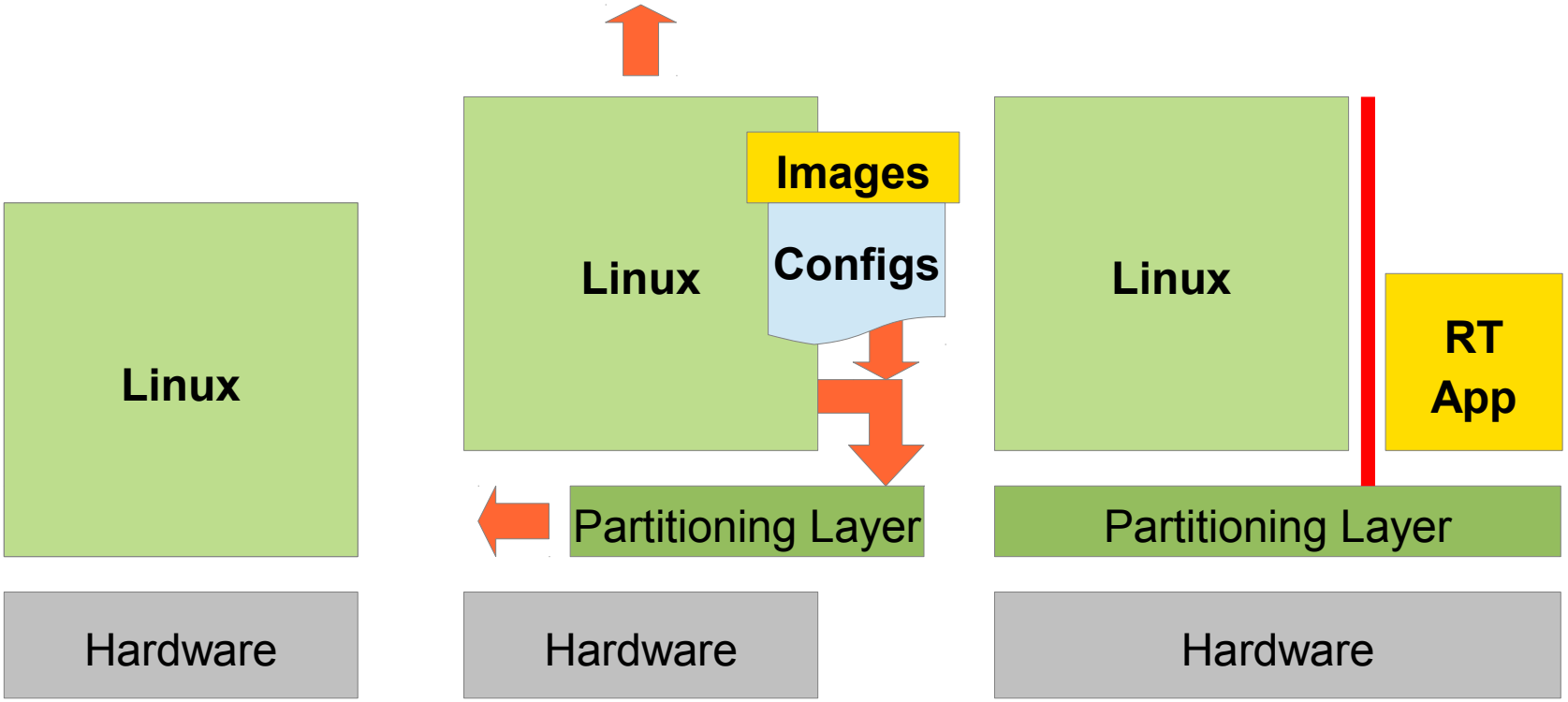
1. Boot phase



2. Operational phase

What about postponing the hypervisor start?

Basic concept of late partitioning

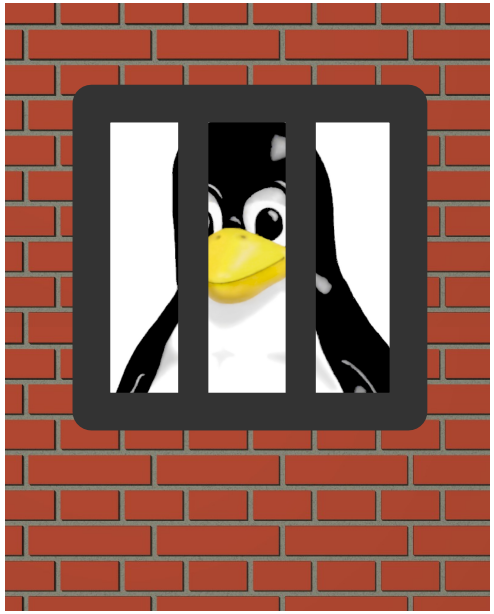


SIEMENS

Jailhouse: Keep it simple, keep it open

The Philosophy of Jailhouse

- **Avoid emulation, focus on hardware assisted isolation**
  - No overcommitment, no scheduler, static partitioning
  - Directly assign physical devices, do not emulate them
  - You need more? Use KVM!
- **Only expose resources that are required for operation**
  - No boot-up phase virtualization
  - Board initialization done by Linux
- **Off-load uncritical tasks to Linux**
  - Initial setup / image loading
  - Reconfigurations while in non-operational mode
  - Monitoring, logging etc.
- **Release as Open Source** (ongoing process – stay tuned)



=> Minimal-sized hypervisor with full CPU assignment and Linux look-and-feel

Virtualizing Real-Time Systems with Linux

Agenda

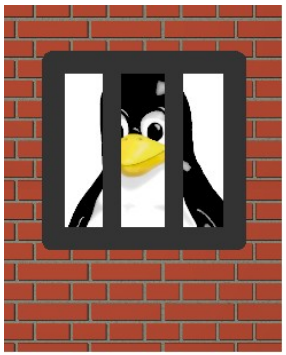
- Motivation & Background
- Real-Time Virtualization Basics
- Linux/KVM-based Setups
- A New Approach for Lowest Latency
- **Summary**

Real-Time – The Next Level for Virtualization

Summary

- **Virtualization provides benefits also for real-time systems**
  - Consolidation
  - Legacy migration (to multicore)
- **Two approaches**
  - Enhance full-featured hypervisor
  - Provide dedicated partitioning hypervisor
- **Open Source based solutions in reach**
  - PREEMPT-RT + KVM + device pass-through works
  - RT device emulation via QEMU is approaching
  - Static partitioning as Linux feature via Jailhouse is WiP

=> Real-time system engineering remains challenging – with or without virtualization



Any Questions?

Thank you!

Jan Kiszka <jan.kiszka@siemens.com>