EVIDENCE®

EMBEDDING TECHNOLOGY

all rights reserved

Linux in ambito industriale, real-time, hypervisors e boot veloce

Paolo Gai, pj@evidence.eu.com

Bruno Morelli, bruno@evidence.eu.com

Applicazioni industriali e real-time su dispositivi eterogenei multi/many-core

T3Lab, Bologna, 18/10/2017

all rights reserved

The company

Founded in 2002 as spin-off company of the
Real-Time Systems Lab at Scuola Superiore S.Anna

- ~22 qualified people with an average age of 38 years
- 15+ years of experience in academic and industrial projects
  - One third of the company has a PhD degree

Our Mission:
We design and develop great software for small electronic devices

all rights reserved

products and services

RTOS , Firmware, Embedded Linux

- AUTOSAR, OSEK/VDX, device drivers
- Embedded Linux: 12 Yrs experience BSPs, GCC, U-Boot, Kernel drivers
- Initial developers of the SCHED_DEADLINE patch
- Hypervisors, Android, Ubuntu Core, QEMU and emulators

Application Development

Model-based design

- Matlab/Simulink/Stateflow
- National Instruments LabView
- E4Coder toolset for code generation
- UML/SYSML/Ecore/ Eclipse/Acceleo

all rights reserved

Linux and Real-Time

Why is a Real-Time Linux interesting?

- Merge together the real-time part and the HMI on the same machine
- Availability of Linux API and device drivers with real-time properties

What is the current status of Real-Time Linux?

It depends a lot on…

- the CPU architecture chosen for the product
- the peripherals (mainly the device drivers!)
- the needed quality of service

all rights reserved

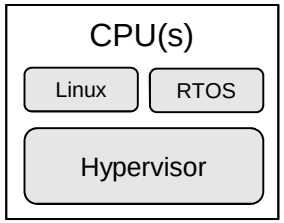
Linux, Real-Time and QoS

- Hard Real-Time < 1ms
  - Multi-OS Approach (with/without hypervisor)
  - Dual-OS Approach (RTAI/Xenomai)
- Soft Real-Time 1-10ms
  - Preemption Patch
  - Real-Time priorities
  - SCHED_DEADLINE/GRUB
- Soft Real-Time > 10ms
  - Unpatched kernel
  - Real-Time priorities
  - SCHED_DEADLINE/GRUB

all rights reserved

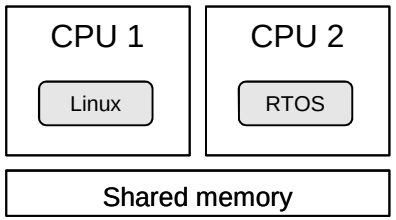
Multi-OS configurations

Main Idea: Linux on one CPU, RTOS on another CPU



1. Hypervisor:

- One/more CPU running both Linux and a certified RTOS
- Open-Source hypervisors (KVM, Xen, Jailhouse)



2. Multi-core platform:

- Example: i.mx6, i.mx8, soloX
- One core: Linux
- Other core: RTOS
- The «other core» can be a «small» core

…we'll see some examples later on in the presentation

all rights reserved

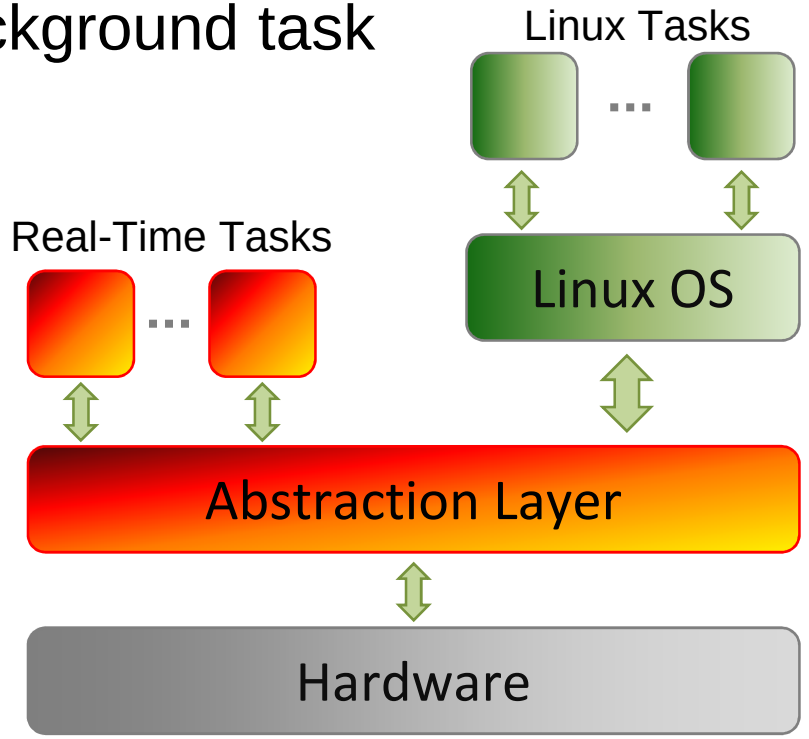
Dual-OS configurations

- Linux is patched to route the “important” IRQs to a small microkernel
  - **RTAI** or **Xenomai**
    - Abstraction Layer with full control of interrupts and system timers
    - Linux executed as a background task
    - Web: http://www.rtai.org and http://www.xenomai.org

all rights reserved

Real-Time Linux: RTAI / Xenomai

- Basic idea: Hardware Abstraction Layer (HAL)
  - Full control of interrupts and system timers
  - Linux executed as a background task
- Various projects
  - RTAI / Xenomai
  - Rt-Linux / Partikle

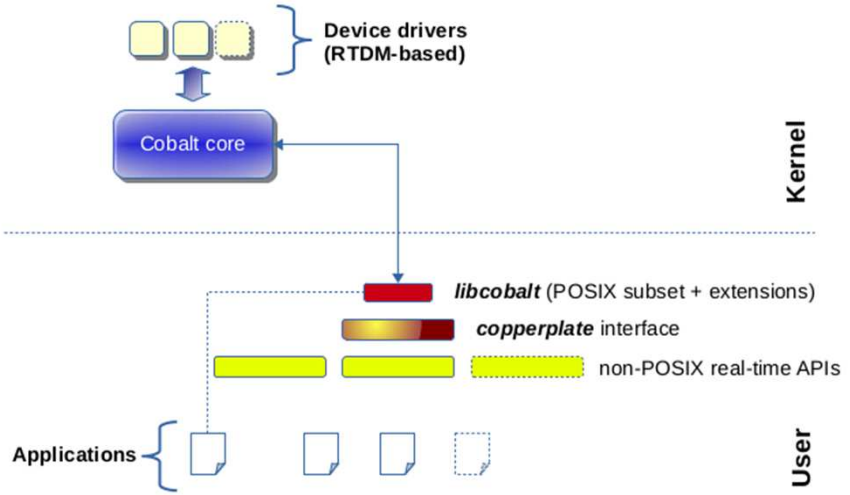


all rights reserved

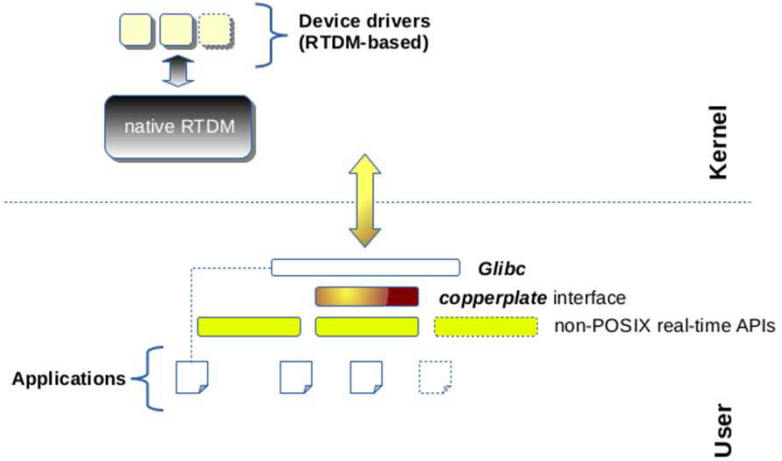
Xenomai 3

- Started in 2001 as an emulation of traditional RTOS API
- Skins implementing various APIs of traditional RTOS such as VxWorks, as well as the POSIX API, and a ``native'' API.

Dual OS configuration



As a user library



all rights reserved

Soft Real-Time: PREEMPT_RT

- It reduces maximum latencies through threaded interrupts and preemptible spinlocks
- A Linux Foundation project:
  https://wiki.linuxfoundation.org/realtime
  Main contributors: I. Molnar, S. Rostedt, T. Gleixner
- Available for the most recent Linux kernel releases
- Supports all the major embedded architectures
- Makes all but the most critical kernel code involuntarily preemptible

all rights reserved

PREEMPT_RT (2)

- Preemptible spinlocks:
  - Replaces most spinlocks with preemptible priority-inheritance mutexes
  - Benefit: reduce maximum latency
- Threaded interrupts:
  - Converting interrupt handlers into preemptible kernel threads
  - Reduces latency by running soft IRQs and selected or all hard interrupts in a dedicated kernel thread (managed in process context by the regular Linux scheduler).
  - Thanks to this, real-time tasks can have priority over some or all IRQs. Previously, even soft IRQs were run before any task managed by the scheduler.
- Priority Inheritance
  - Merged inside vanilla kernel since Linux 2.6.18

all rights reserved

Myths about PREEMPT_RT

- It will improve throughput and overall performance
  - Wrong: it will degrade overall performance.
- It will reduce average latency
  - Often wrong. The maximum latency will be reduced.
- The primary goal is to make the system predictable and deterministic!

all rights reserved

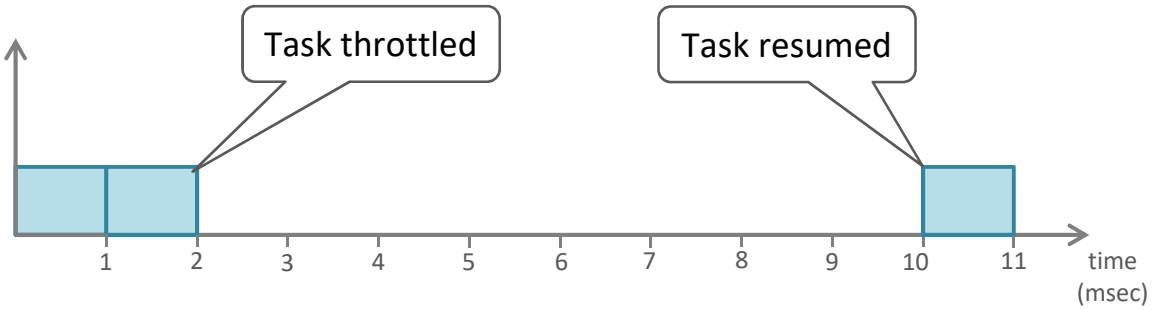
SCHED_DEADLINE

- Real-time CPU scheduler created by Evidence Srl
  - Made in collaboration with ReTiS Lab of Scuola Sant'Anna
  - It allows real-time isolation between running tasks
  - http://www.evidence.eu.com/sched_deadline.html
  - http://en.wikipedia.org/wiki/SCHED_DEADLINE
- Integrated into the official Linux kernel
  - Available since kernel 3.14
  - Reclaiming added in kernel 4.13

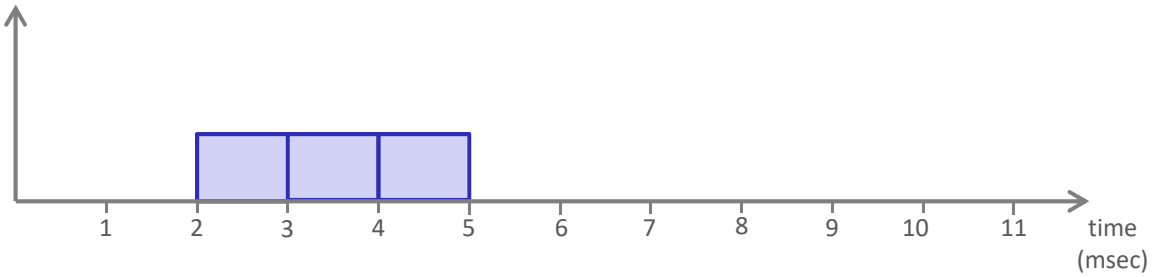
all rights reserved

SCHED_DEADLINE

SCHED_DEADLINE task
runtime = 2 msec
period = 10 msec



SCHED_DEADLINE task
runtime = 3 msec
period = 20 msec



- "Hard reservation" rule to avoid starvation of lower-priority tasks
- Non work-conserving: RT tasks blocked even if system is idle
- http://en.wikipedia.org/wiki/SCHED_DEADLINE

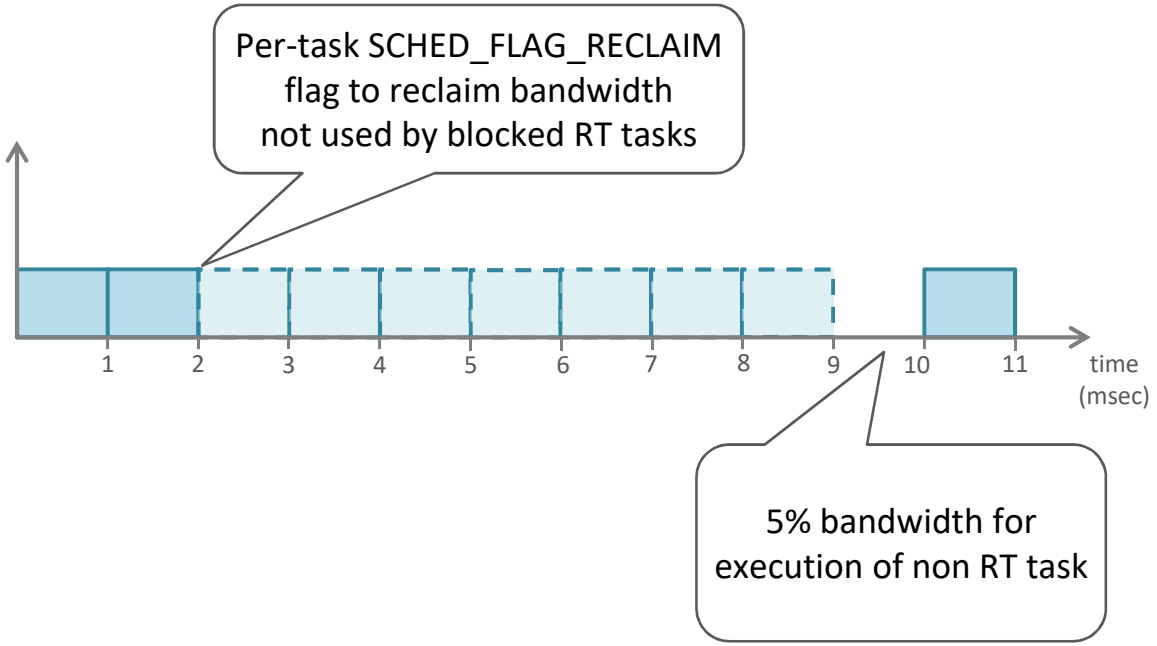
Merged into the official Linux kernel since 3.14 (2014)

all rights reserved

GRUB

- Greedy Reclamation of Unused Bandwidth (GRUB) algorithm:

SCHED_DEADLINE task
runtime = 2 msec
period = 10 msec



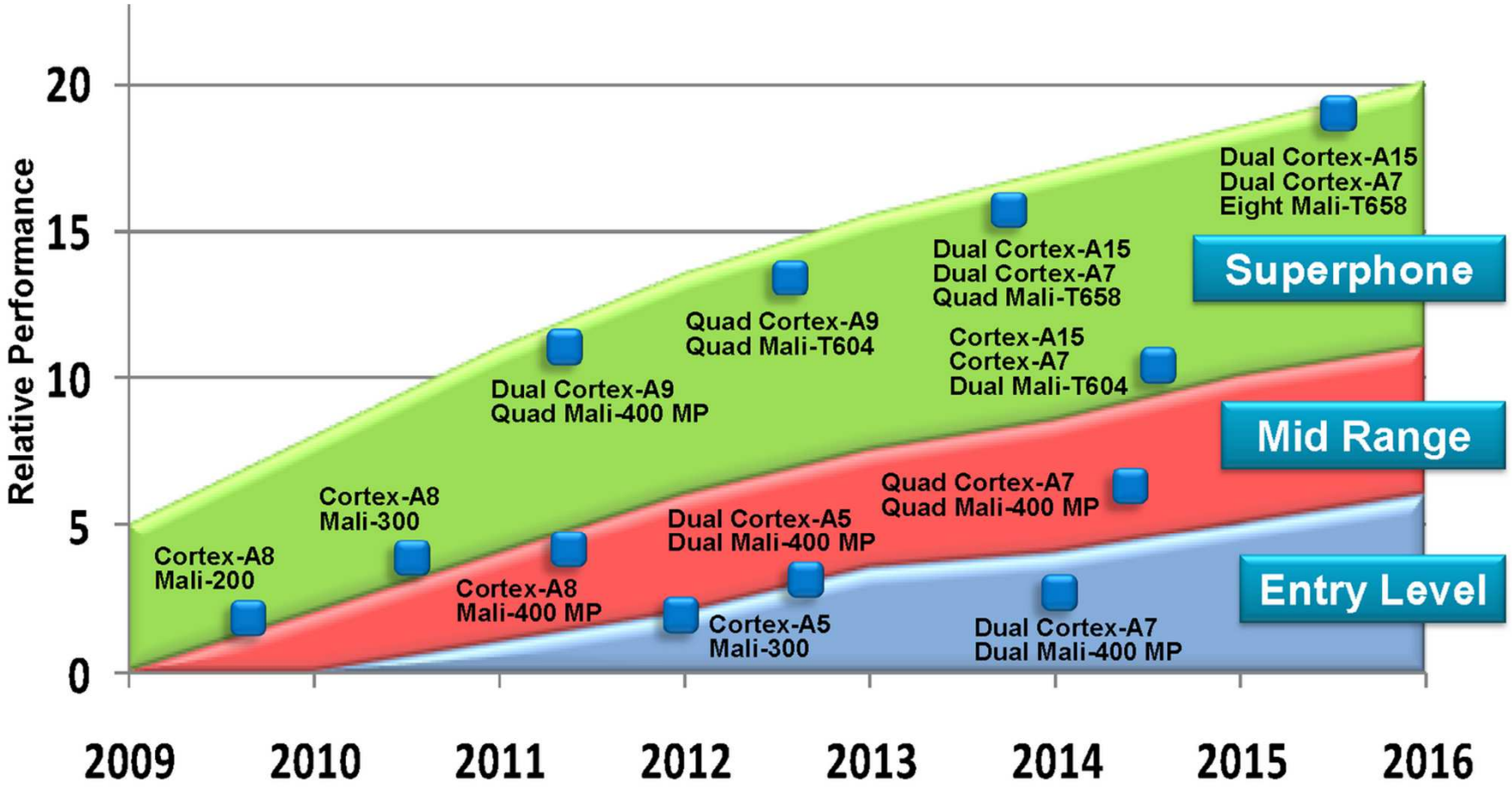
- G. Lipari, S. Baruah, *Greedy reclamation of unused bandwidth in constant-bandwidth servers*, 12th IEEE Euromicro Conference on Real-Time Systems, 2000.
- L. Abeni, J. Lelli, **C. Scordino**, L. Palopoli, *Greedy CPU reclaiming for SCHED_DEADLINE*, Real-Time Linux Workshop (RTLWS), Dusseldorf, Germany, 2014.

all rights reserved

Multi-OS on Linux

all rights reserved

Rationale

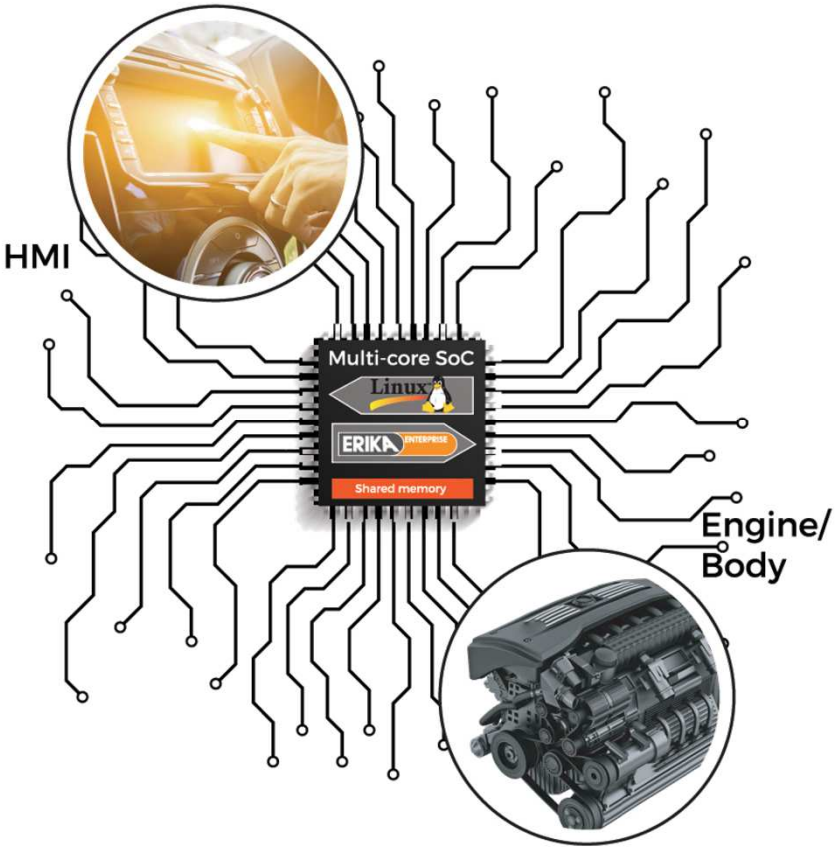


all rights reserved

AMP configuration: with/without Hyp.

Real-time and quality of service for IVI systems with/without hypervisor!

- ERIKA running on one core
- Linux on the other cores
- No Hypervisor
- …or JailHouse Hypervisor



all rights reserved

ERIKA Enterprise (v2 and v3)

http://erika.tuxfamily.org

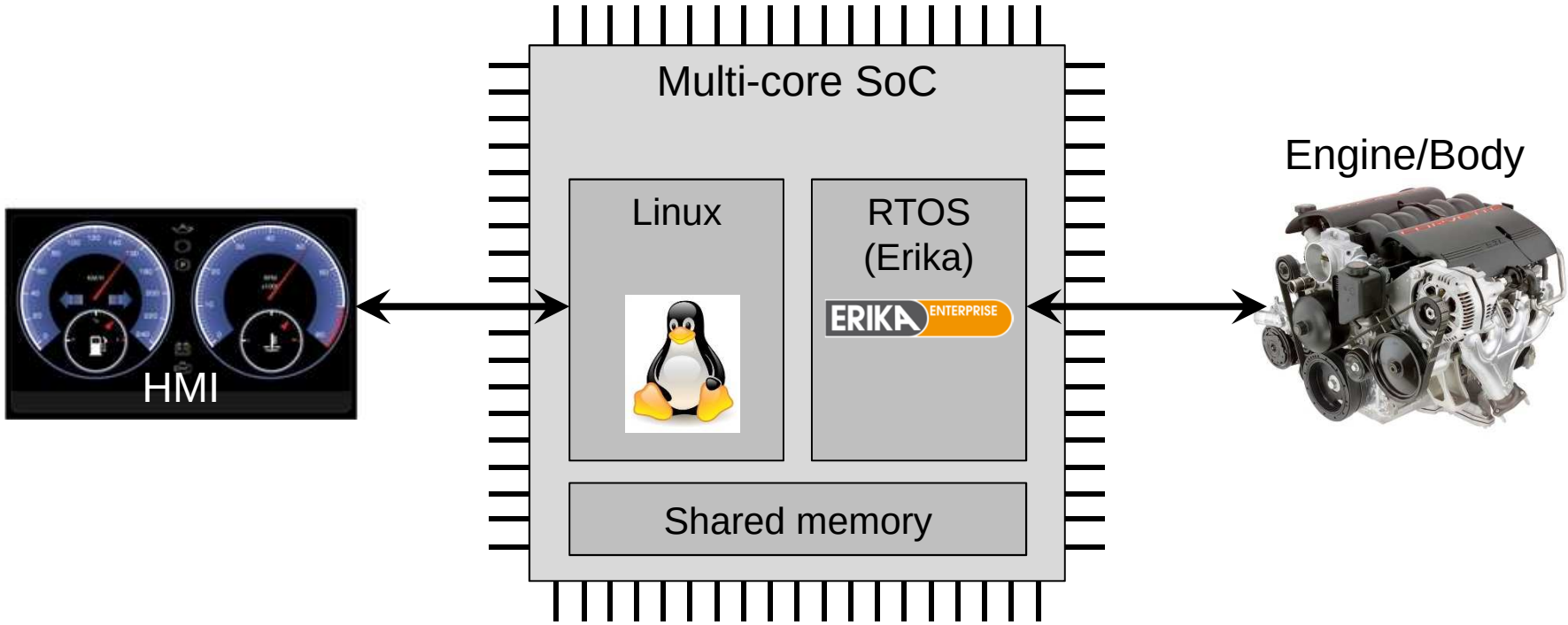
http://www.erika-enterprise.com

- ERIKA Enterprise is an RTOS OSEK/VDX certified
- ERIKA Enterprise implements the AUTOSAR OS API
- open-source license and double licensing options allowing static linking of closed src code
- Used by various industries and research projects

all rights reserved

Linux + ERIKA without hypervisor

We envision the possibility to exploit multi-cores to run Linux and Erika Enterprise complementing each other!



all rights reserved

Demo presented at AGLW 2013

Based on a Freescale iMX6

We let U-Boot handle the multicore boot

- ERIKA starts almost immediately
- Linux can start afterwards

No hypervisor

- could be useful in some cases to protect the behavior of misbehaving applications
- limited need because we statically allocate a CPU to each OS

Can be used with other CPUs

- Example – SoloX (Cortex A + Cortex M)

all rights reserved

AMP: Interaction model

Linux → ERIKA

- Linux can trigger the following actions:
  - activate a task
  - set an event
  - start an Alarm
  - increment a counter

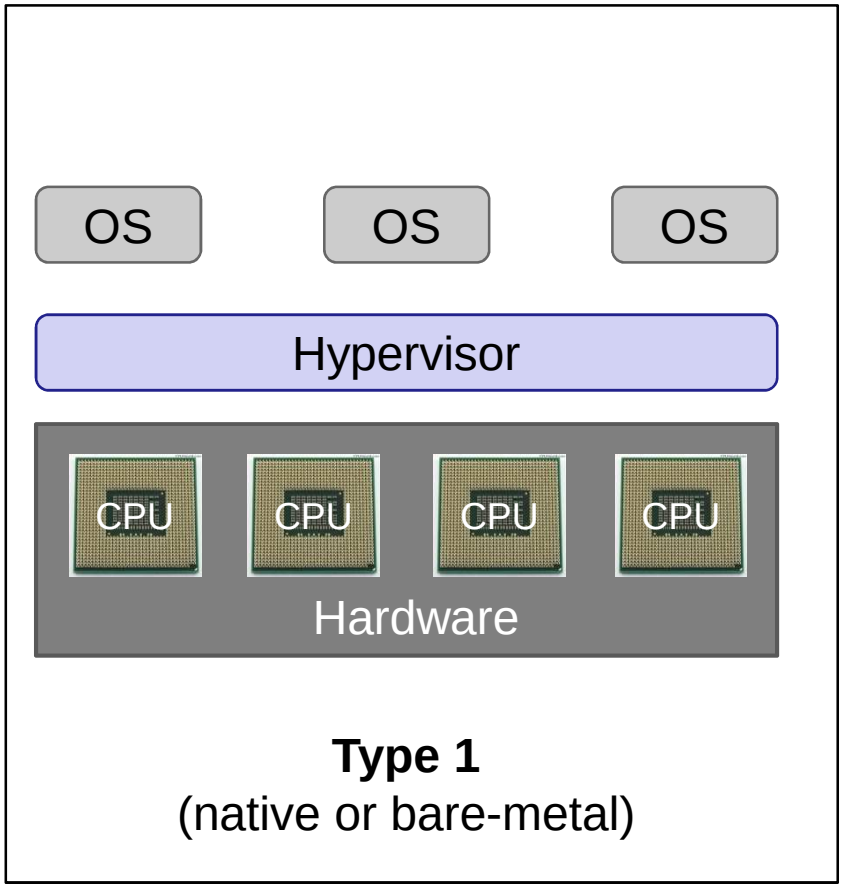
  (similar to those doable on a remote core of an AUTOSAR OS)
- Linux can stop and reload the ERIKA application

Linux ←→ ERIKA

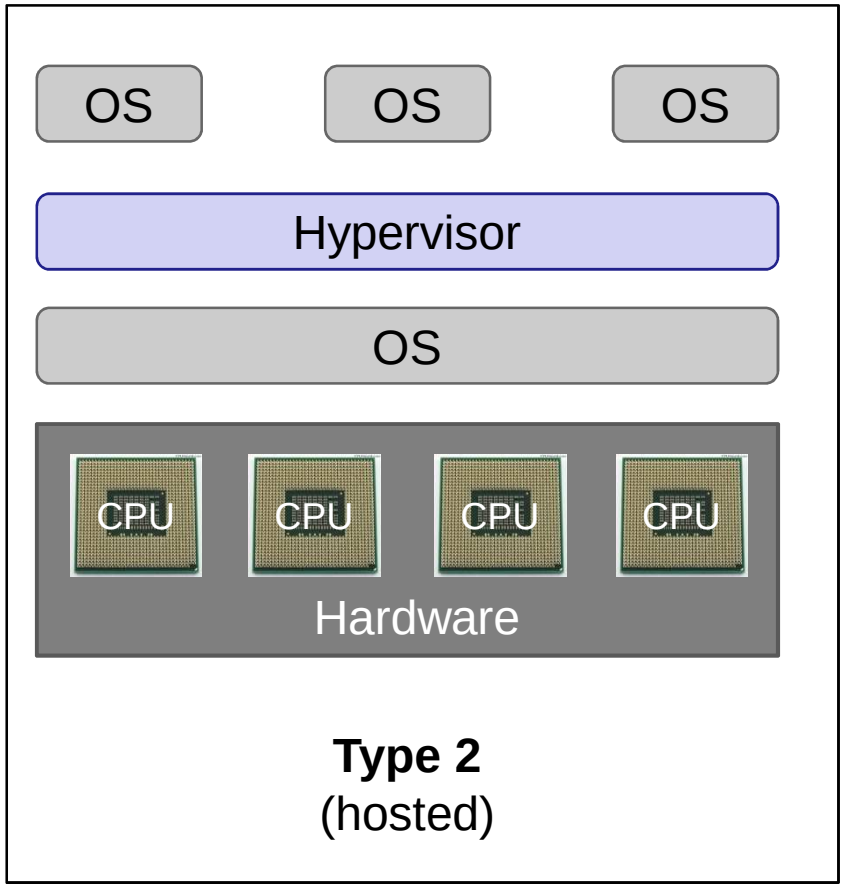
- Simple asynchronous message passing allowing asynchronous read/write of variable length buffers on predefined channels

all rights reserved

Classification of hypervisors



Type 1
(native or bare-metal)



Type 2
(hosted)

all rights reserved

AMP Configuration: ERIKA with XEN

+

http://xenproject.org/about/events/viewevent/154-linuxcon-eu-integrating-linux-and-the-real-time-erika-os-through-the-xen-hypervisor.html

In collaboration with the University of Modena, Arianna Avanzini integrated ERIKA Enterprise as domU under XEN
on a CubieBoard (Allwinner A20)

Presented at CloudCon Europe October 15th 2014, Dusseldorf

all rights reserved

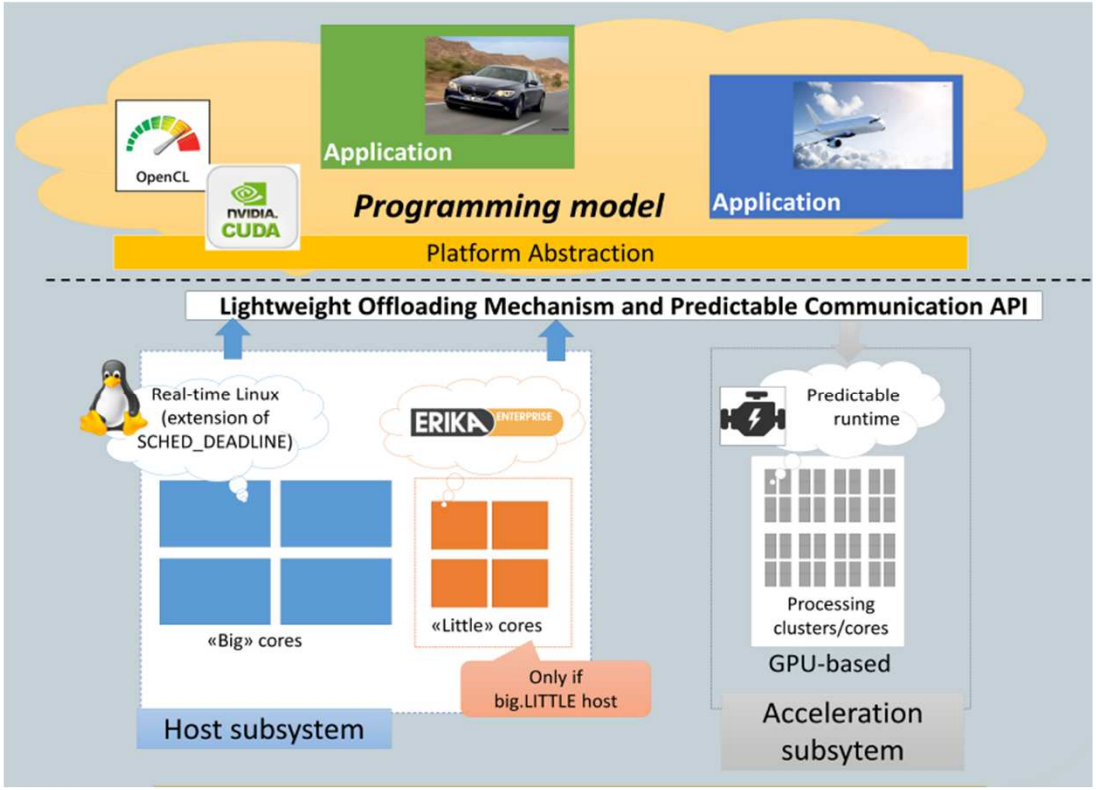
Hypervisor configuration – H2020 Hercules

Architecture and platform for customized low-power heterogeneous computing systems, for Automotive and Avionics

Jan 2016-Dec 2018

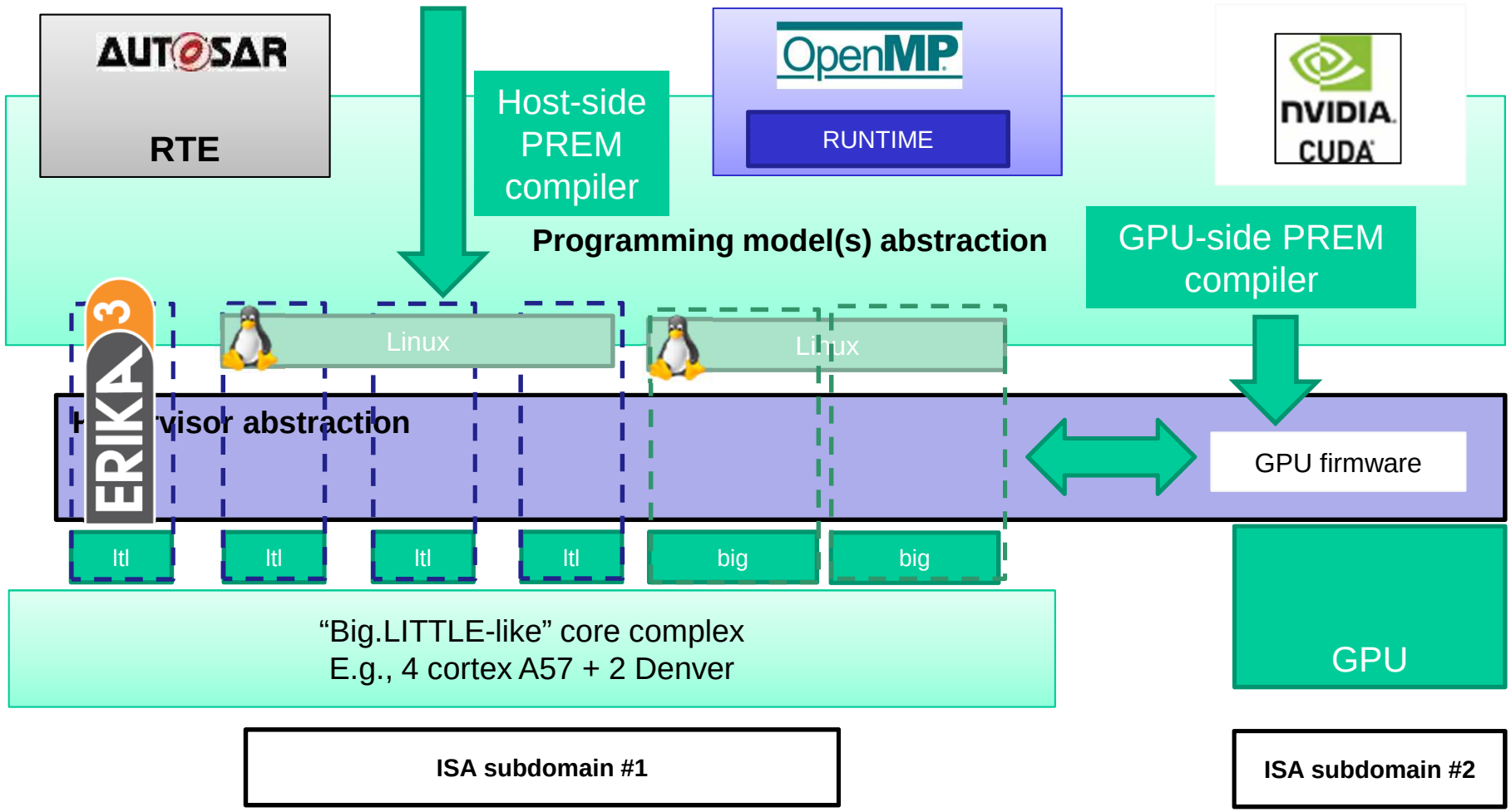
Partners:

- UNIMORE
- Magneti Marelli
- Airbus
- Evidence
- Pitom
- CTU
- ETH Zurich



all rights reserved

Slide architettura hercules GTC



all rights reserved

Jailhouse

- Small, lightweight hypervisor
- Young project (2013) by Siemens
- License: GPLv2
- Code hosted on GitHub
- Goals: safety-critical & certification
  (goal: 10.000 lines of code per architecture)

all rights reserved

Jailhouse (2)

A tool to run

... real-time and/or safety-critical tasks
... on multi-core platforms
... aside Linux

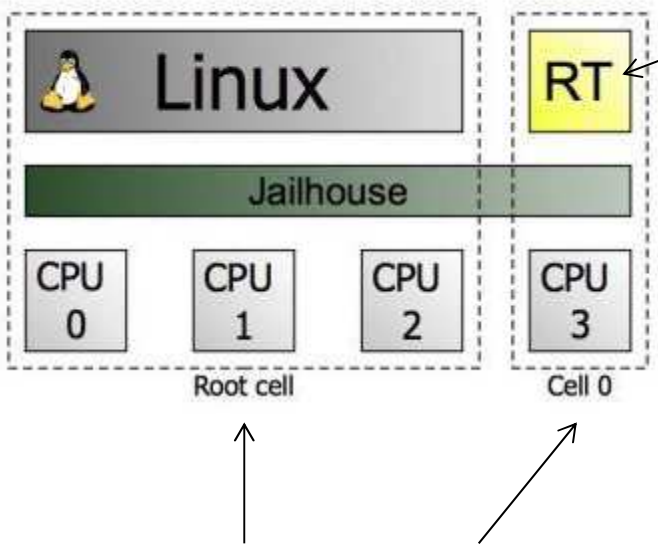
It provides

... strong & clean isolation
... bare-metal like performance & latencies

all rights reserved

Jailhouse (3)

The guest software is called **inmate**



The isolated compartments are called **cells:**

- One root cell
- One or more non-root cells

all rights reserved

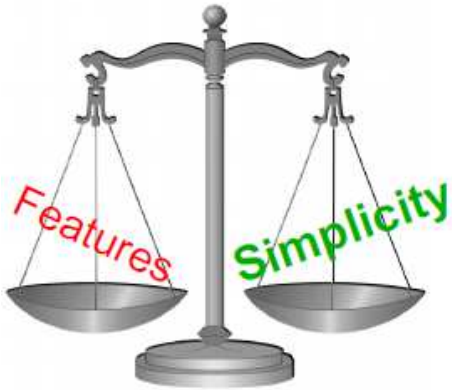
Jailhouse (4)

- Partitioning hypervisor
  - More focused on isolation and resource assignment than on virtualization
- Linux is required ("root" cell)
  - Similar to Xen's dom0 but w/out full control of hw
  - Type-1 (Linux used only for management interface and load/unload guests)
- Can't run unmodified OSs
  - Even Linux on non-root cells need to be patched

all rights reserved

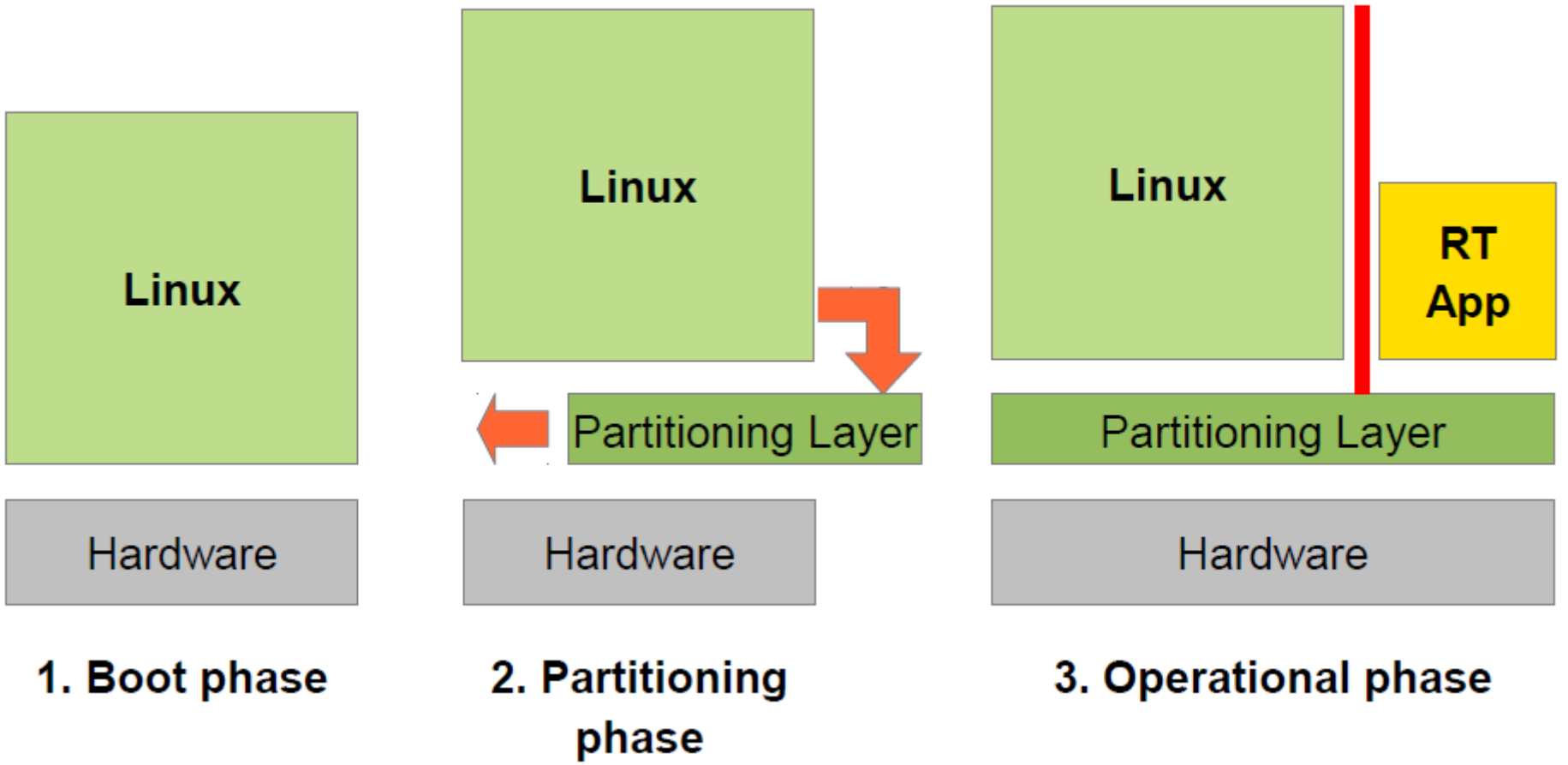
Jailhouse (5)

- Static design:
  - 1:1 resource assignment
  - Guests can't share a core (no scheduling)
  - It doesn't support overcommitment of resources (like CPUs, RAM or devices).
  - No hw emulation
- Real-time guarantees:
  - Must be provided by the guest OS



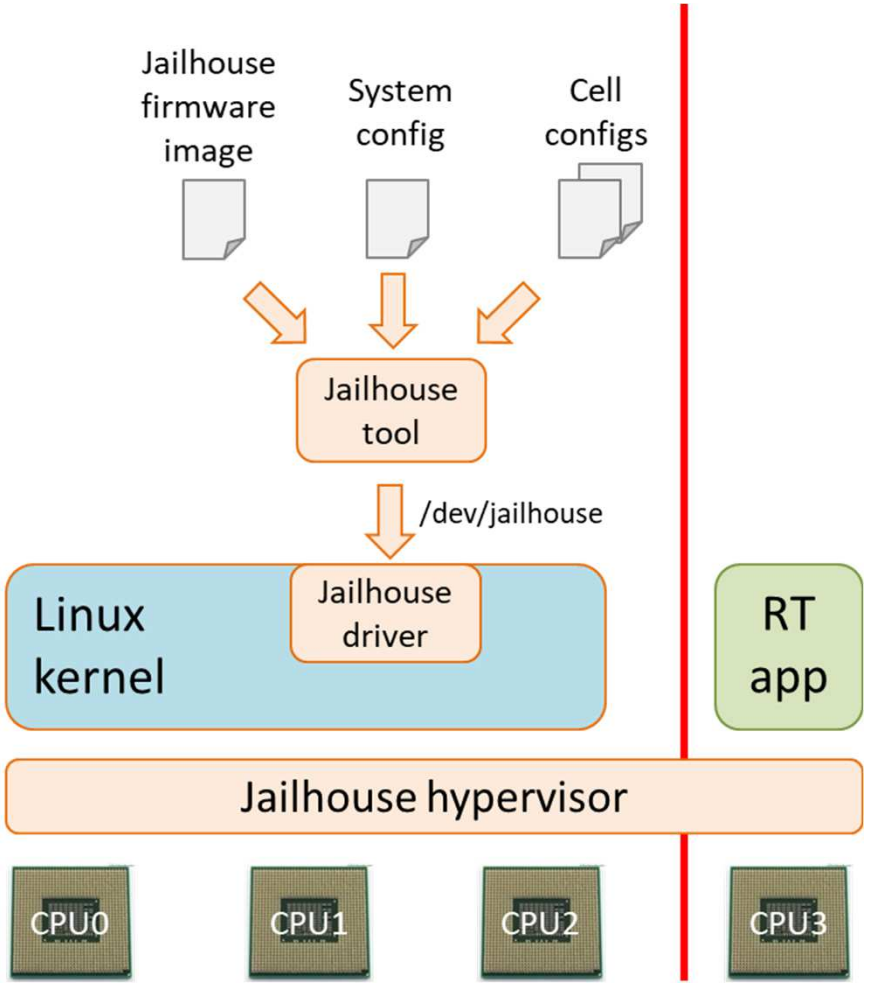
all rights reserved

Late partitioning



all rights reserved

Jailhouse architecture



all rights reserved

Demo on Nvidia TX1

- Demo

- Similar demo available on Drive PX2 / Xilinx Ultrascale+

all rights reserved

Fast boot on Linux

all rights reserved

Fast Boot

- It’s a race over the fastest booting system…
- In reality what is important is the time between power up and the availability of some given services
  - Final user application
  - Splash screen
  - CAN bus active in the first x ms
  - …
- Depending on the requirements, there are various optimizations that can be done
- Most of the boot time can be shrinked with a few main actions. Then the ms tuning require… art!

all rights reserved

Typical tweaks to reduce boot time

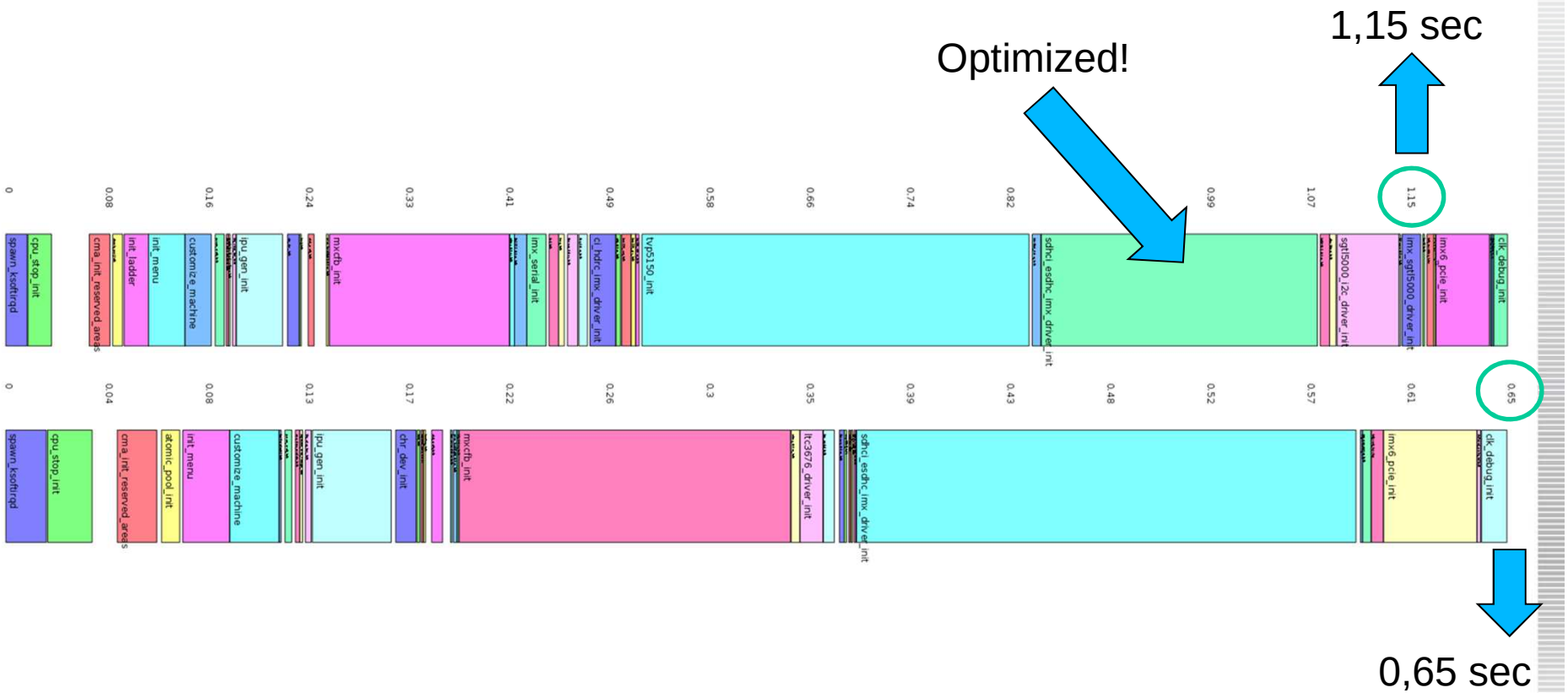
- Splash screen to simulate an early start
- No boot loader
- Peripheral initialization postponed
- Service daemons postponed
- Pre-boot e preload (example when opening the car door)
- Filesystem choice to allow fast boot
  (with/without compression, read-only/write)
- Kernel optimizations

- It is useful to have a tracing tool in HW
  or, whenever not possible, in SW

all rights reserved

Linux Boot Charts

Example of chart taken from the timings of the Linux boot



all rights reserved

Linux Fast Boot

COBOO

- Linux Fast boot on iMX6 Dual
  - U-Boot with USB support
  - Linux kernel with Video support, Touchscreen, USB and CAN
  - Qt application
- Results
  - 1.8s - First CAN message from userspace
  - 3.9s - Qt Application

all rights reserved

Contacts

Evidence Srl

Via Carducci 56

56010 S.Giuliano Terme

Pisa - Italy

Web: http://www.evidence.eu.com

E-mail: info@evidence.eu.com

Phone: +39 050 99 11 224

all rights reserved