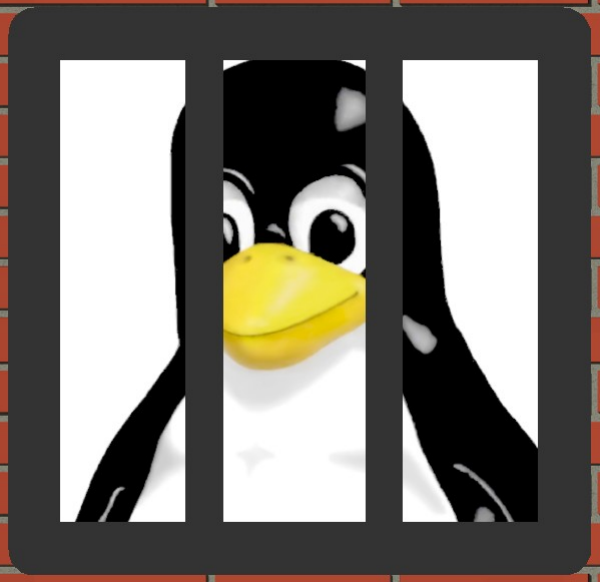
SIEMENS



Siemens Corporate Technology | October 2014

Real Safe Times in the Jailhouse Hypervisor

Real Safe Times in the Jailhouse Hypervisor

Agenda

- **Jailhouse introduction**
- Safe isolation
- Architecture support
- Jailhouse application development
- Summary
- [Demo]

SIEMENS

What is Jailhouse?

A tool to run

… real-time and/or safety tasks

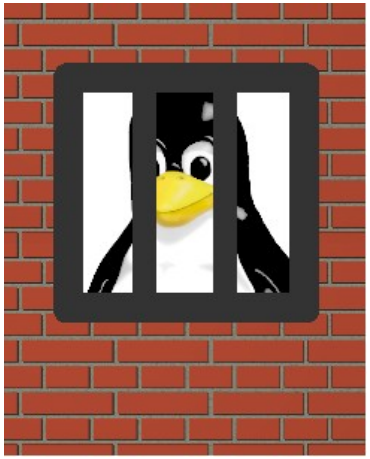
… on multicore platforms (AMP)

… aside Linux

It provides

- **strong & clean isolation**
- **bare-metal-like performance & latencies**
- ***no* reason to modify Linux**

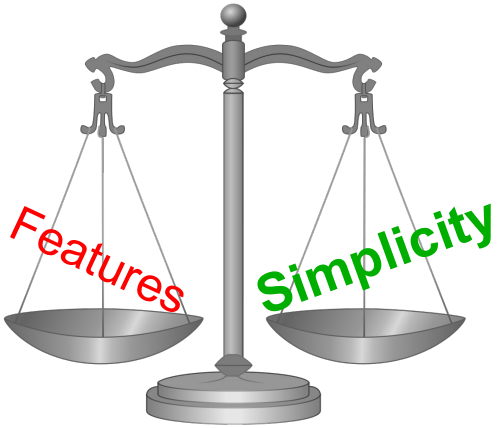
… and it's open source (GPLv2)



SIEMENS

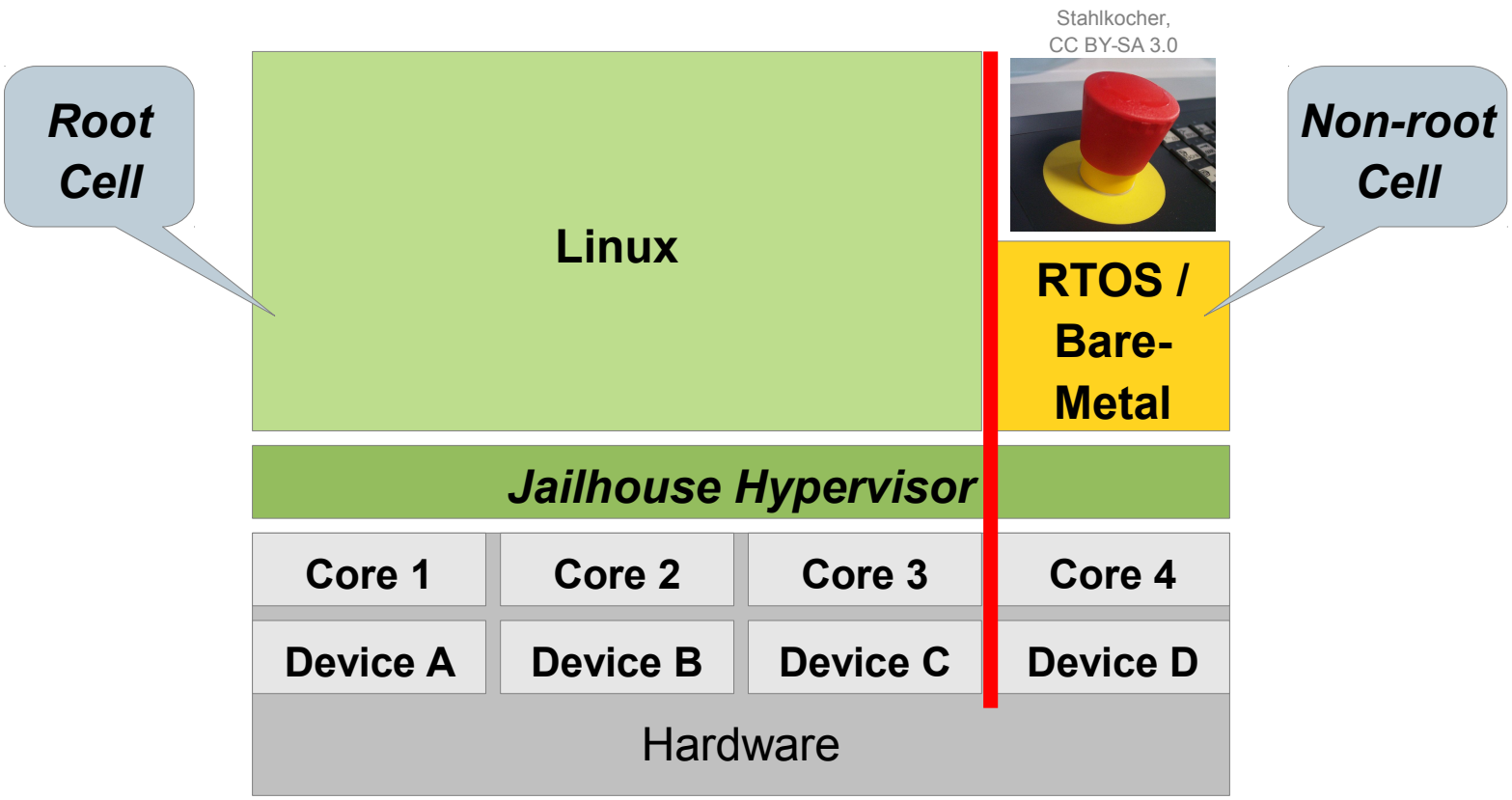
What makes Jailhouse different?

- **Use virtualization for isolation – *ok, nothing new***
- **Prefer simplicity over features**
  - Resource access control
    instead of resource virtualization
  - 1:1 resource assignment
    instead of scheduling
  - Partition booted system
    instead of booting Linux
  - Do not hide existence of Jailhouse
- **Offload work to Linux**
  - System boot
  - Jailhouse and partition (“cell”) loading & starting
  - Control and monitoring



SIEMENS

Asymmetric Multi-Processing with Jailhouse



Real Safe Times in the Jailhouse Hypervisor

Agenda

- Jailhouse introduction
- **Safe isolation**
- Architecture support
- Jailhouse application development
- Summary
- [Demo]

SIEMENS

Isolation Properties of Jailhouse

- **Prevents access to unassigned resources – enforced for both CPUs and devices**
  - Memory
  - I/O
  - Interrupt channels
- **Prevents cell interferences**
  - System reset / shutdown
  - Inappropriate power settings [WiP]
- **Hypervisor is protected against all cells**
- **Cell creation/destruction and hypervisor disabling are privileged operations**
  - Can only be issued by root cell
  - Non-root cells may lock system configuration
  - Hypervisor supports non-root cells in validating system setup [WiP]

Limits of Hypervisor-based Isolation

- **No magic to avoid hardware errors**
  - Sporadic hardware faults can bring down the system
  - Or worse: produce wrong output!
- **Jailhouse catches and forwards hardware error reports [WiP]**
  - Reaction configurable, usually application-specific
- **Don't forget potential hardware mistakes**
  - Hidden design errors
  - Undocumented side effects

=> System design has to account for this!

TÜV-approved Hypervisor Safety Concept

- **Hypervisor safety concept completed**
  - Safety features
  - Architecture
  - Hardware requirements
  - Software measures
  - Safety-related application conditions
- **TÜV Rheinland confirmed**
  - No deficiencies
  - Concept feasible

Real Safe Times in the Jailhouse Hypervisor

Agenda

- Jailhouse introduction
- Safe isolation
- **Architecture support**
- Jailhouse application development
- Summary
- [Demo]

Jailhouse Status – x86

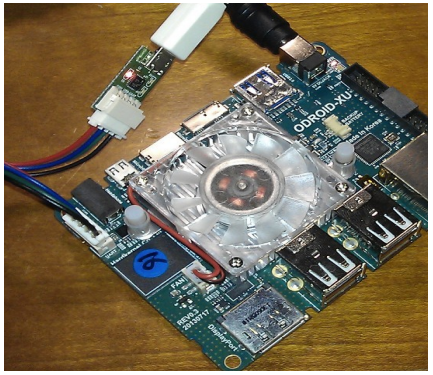
- **Initial focus on Intel x86**
  - Requirements
    - VT-x (~Sandy Bridge)
    - VT-d (IOMMU with interrupt remapping)
    - ≥ 2 cores
  - Currently: 7300 lines of code
  - Recent Linux kernel (3.1x)
- **Supports direct interrupt delivery**

  => Zero VM exits, minimal latencies feasible
- **AMD64 ready for merge**
  - Supported by AMD, performed by Valentine Sinitsyn
  - IOMMU on to-do list

SIEMENS

Jailhouse Status – ARM

- **ARMv7**
  - Initial port sponsored by ARM, performed by Jean-Philippe Brucker
  - (Almost) no changes to Jailhouse core
- **Status**
  - Preparing for merge
  - Works fine in Fast Model
  - Rough support for ODROID-XU
  - Arndale and TI Keystone II board support planned
- **To-Do**
  - SMMU / System MMU
  - Improve board support (device tree?)
  - ARMv8

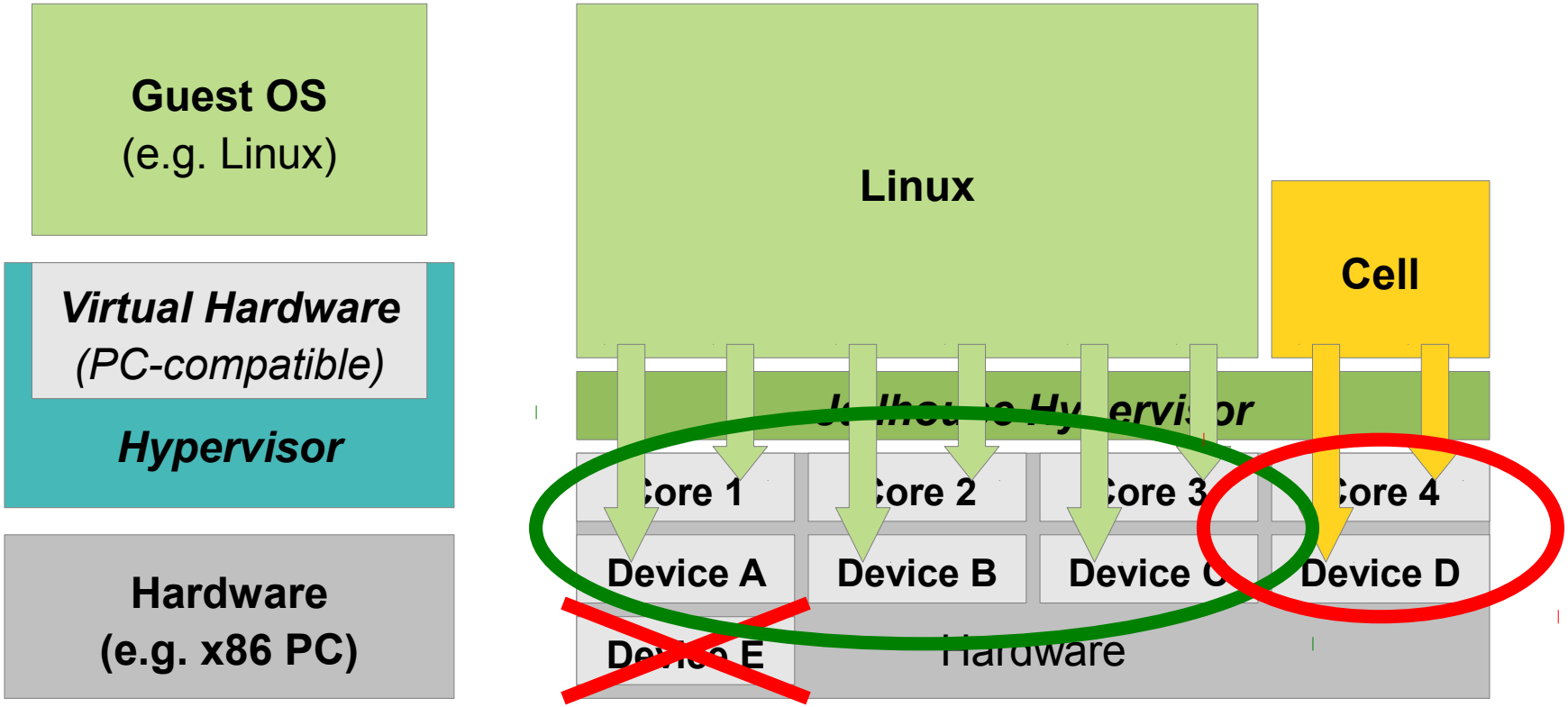


Real Safe Times in the Jailhouse Hypervisor

Agenda

- Jailhouse introduction
- Safe isolation
- Architecture support
- **Jailhouse application development**
- Summary
- [Demo]

Difference to Standard Hypervisors



SIEMENS

Available Resources for Cells

Resource	x86
RAM • Address space customizable	• No restrictions by BIOS, ROMs etc.
CPU cores • Inter-processor communication • Non-virtualized IDs • Modified CPU bootstrap	• Inter-processor interrupts • Different start vector & SMP boot, no boot through BIOS
Clock • At least one reference clock	• ACPI PM timer • CPU-local TSC
Timer	• Local APIC timer
Data exchange with assigned devices	• MMIO & PIO to device • DMA to cell RAM
Interrupts from assigned devices	• Accesses to required IOAPIC slots
Inter-cell communication	• Virtual PCI device [WiP]

SIEMENS

OS-less Jailhouse Application

- **For simple scenarios**
  - Single task
  - Typically single-core
  - Few devices
  - New design or few dependencies
- **Required infrastructure**
  - CPU bootstrap (assembly)
  - I/O initialization and operation
    - Devices
    - Inter-cell [if needed]

=> Use Jailhouse “inmate” skeleton

- Enables “main loop” development in C
- Essential I/O library available for x86 and ARM

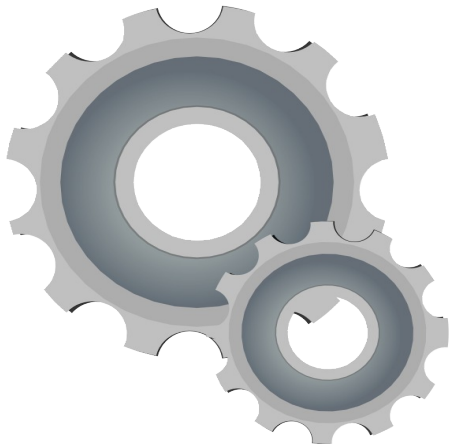
```
void main(void)
{
    init();
    while (1) {
        do_work();
    }
}
```

SIEMENS

RTOS-based Application

- **For advanced scenarios**
  - Multiple tasks
  - SMP
  - Complex device setups
  - Preexisting RTOS stacks
- **Required steps**
  - Remove most platform setup logic
  - Switch to available timers, clocks etc. [if needed]
  - Add inter-cell I/O support [if needed]

=> Reference: Jailhouse support for RTEMS



SIEMENS

RTEMS as Jailhouse “Inmate”

- **Why RTEMS?**
  - Open source, actively developed
  - Reasonable x86 & PCI support
- **Required porting steps**
  - Removed BIOS dependencies, adjusted CPU bootstrap
  - Console only via serial
  - Legacy PIC & PIT → x2APIC & IOAPIC
  - Suitable clock & timer calibrations
- **To be published soon (watch mailing list)**
  - Jailhouse “Board Support Package”
  - Intel e1000-class PCI NIC driver

RTEMS

SIEMENS

Emulation-based Application Debugging

- **Option #1:**
  Hardware debugger
- **Option #2:**
  Fast emulation, virtualization
- **Challenge:**
  Emulate Jailhouse environment (not a “normal” PC)
- **Approach:**
  Extend Linux/KVM hypervisor with Jailhouse awareness
  - QEMU/KVM supports OS-level debugging via gdb
  - We added x86 Jailhouse partition emulation
  - Enables source-level debugging of Jailhouse applications
  - Use (PCI) device pass-through for I/O access
  - Warning: no real-time guarantees!

SIEMENS

Real Safe Times in the Jailhouse Hypervisor

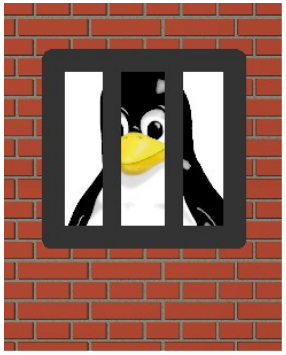
Agenda

- Jailhouse introduction
- Safe isolation
- Architecture support
- Jailhouse application development
- **Summary**
- [Demo]

SIEMENS

Summary

- **Jailhouse provides clean AMP for Linux**
  - Full CPU isolation
  - Minimal latency I/O
  - Reduced to the minimum (goal: <10k LOC/arch)
- **Jailhouse aims at safe segregation**
  - Enable mixed-criticality on multicore
  - TÜV-approved safety concept
- **Jailhouse is a community project**
  - GPLv2, public development for 1 year
  - Significant contributions enabled AMD64 and ARMv7
  - You are invited to join!



Any Questions?

Thank you!

https://github.com/siemens/jailhouse

Jan Kiszka <jan.kiszka@siemens.com>

Real Safe Times in the Jailhouse Hypervisor

Agenda

- Jailhouse introduction
- Safe isolation
- Architecture support
- Jailhouse application development
- Summary
- **Demo!**

Live Demonstration

Running Jailhouse in a virtual machine?!