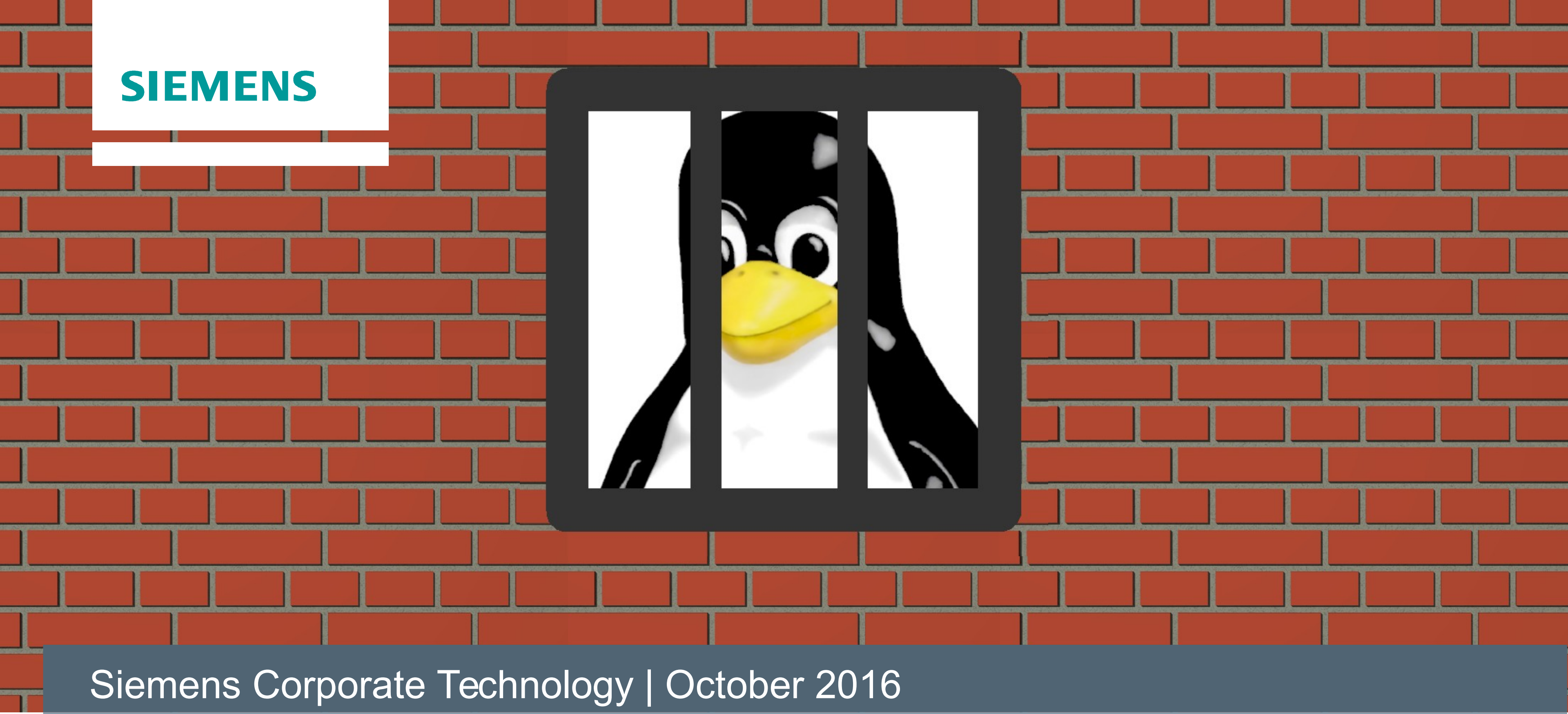
SIEMENS



Siemens Corporate Technology | October 2016

Bootstrapping the Partitioning Hypervisor Jailhouse

SIEMENS

Bootstrapping the Partitioning Hypervisor Jailhouse

Agenda

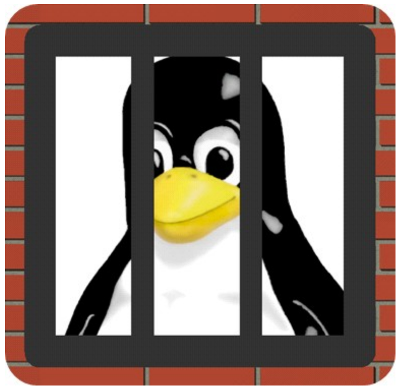
What is Jailhouse?

A tool to run

… real-time and/or safety tasks

… on multicore platforms (AMP)

… aside Linux



It provides

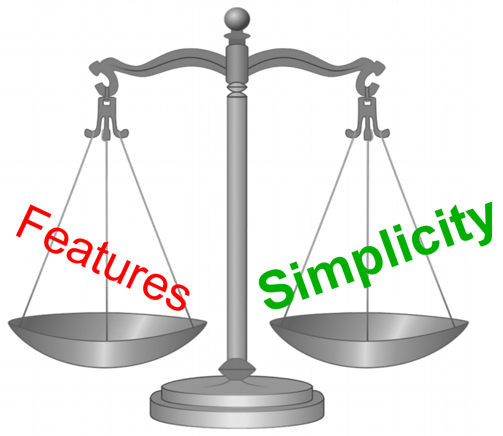
- **strong & clean isolation**
- **bare-metal-like performance & latencies**
- **no reason to modify Linux** (well, almost)

… and it's open source (GPLv2)

SIEMENS

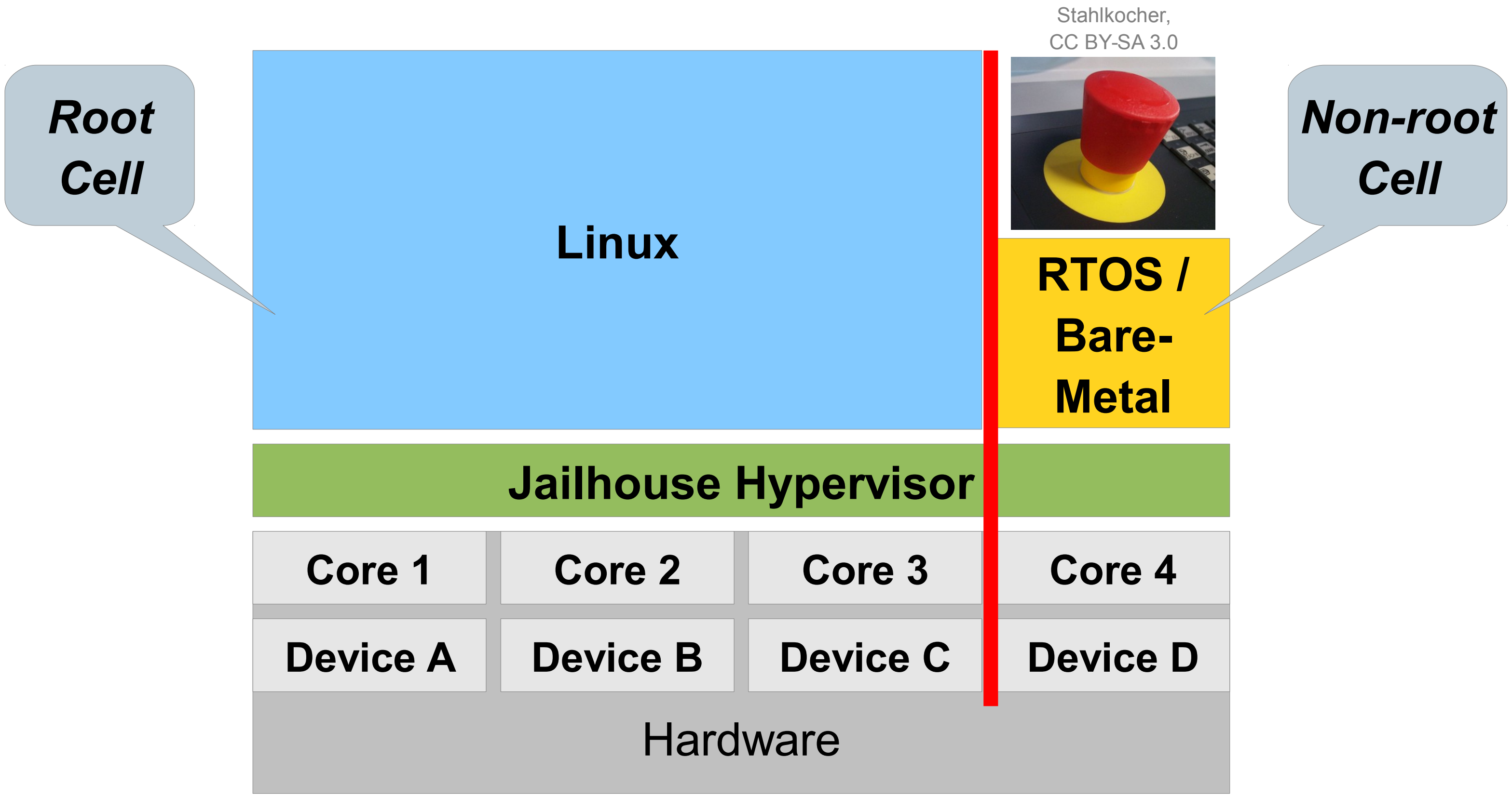
What makes Jailhouse different?

- **Use hardware-assisted virtualization for isolation**
- **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

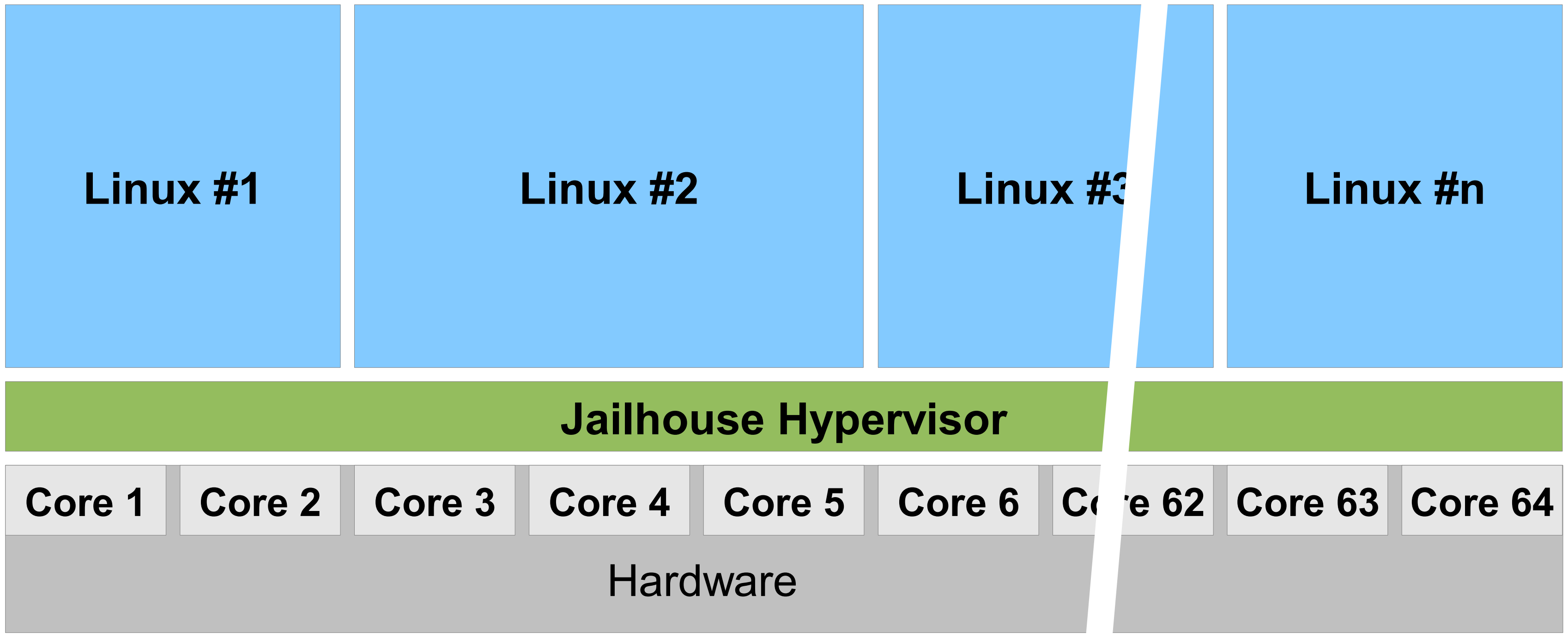


SIEMENS

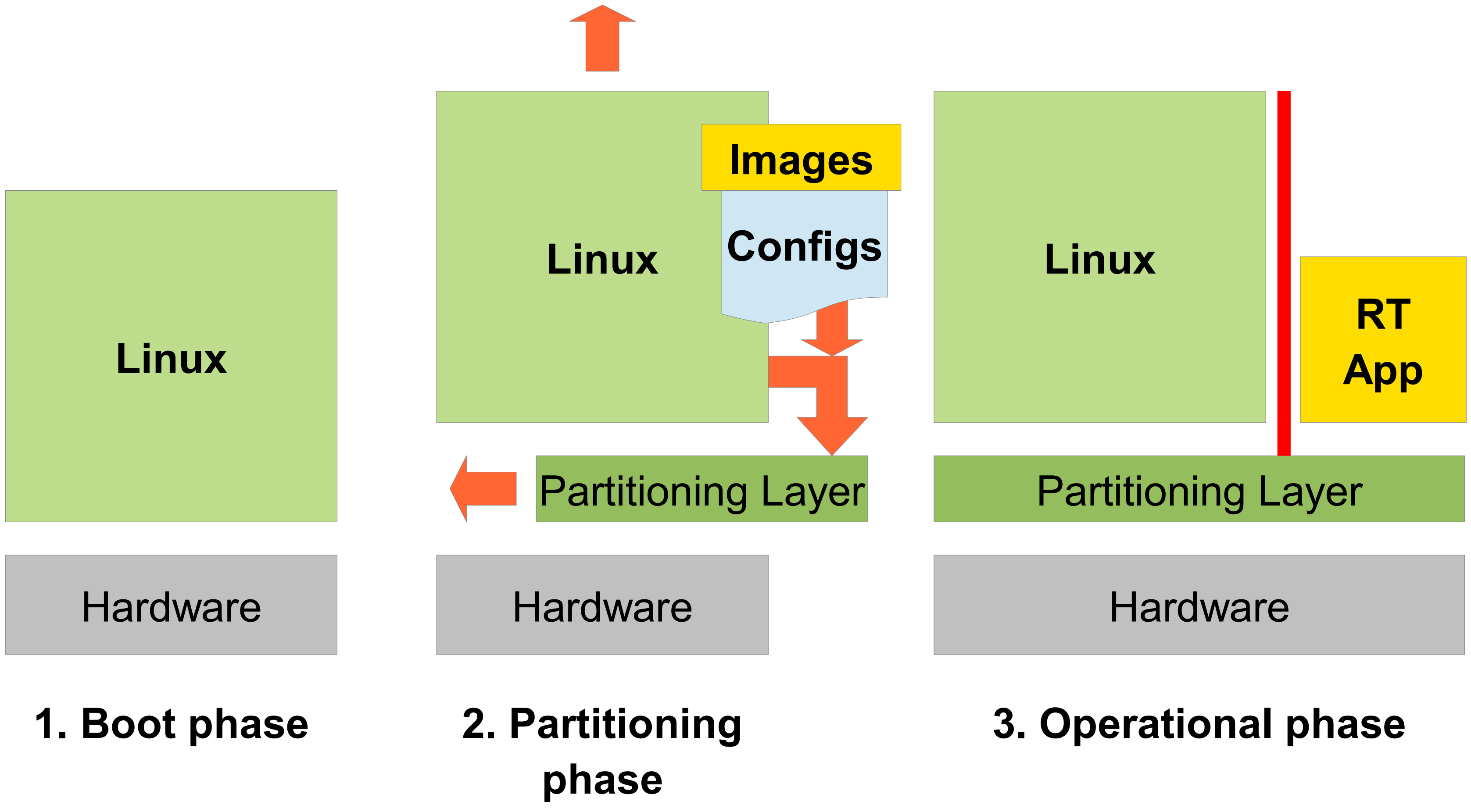
Asymmetric Multiprocessing with Jailhouse



Hard Partitioning of Linux



Late Partitioning Concept

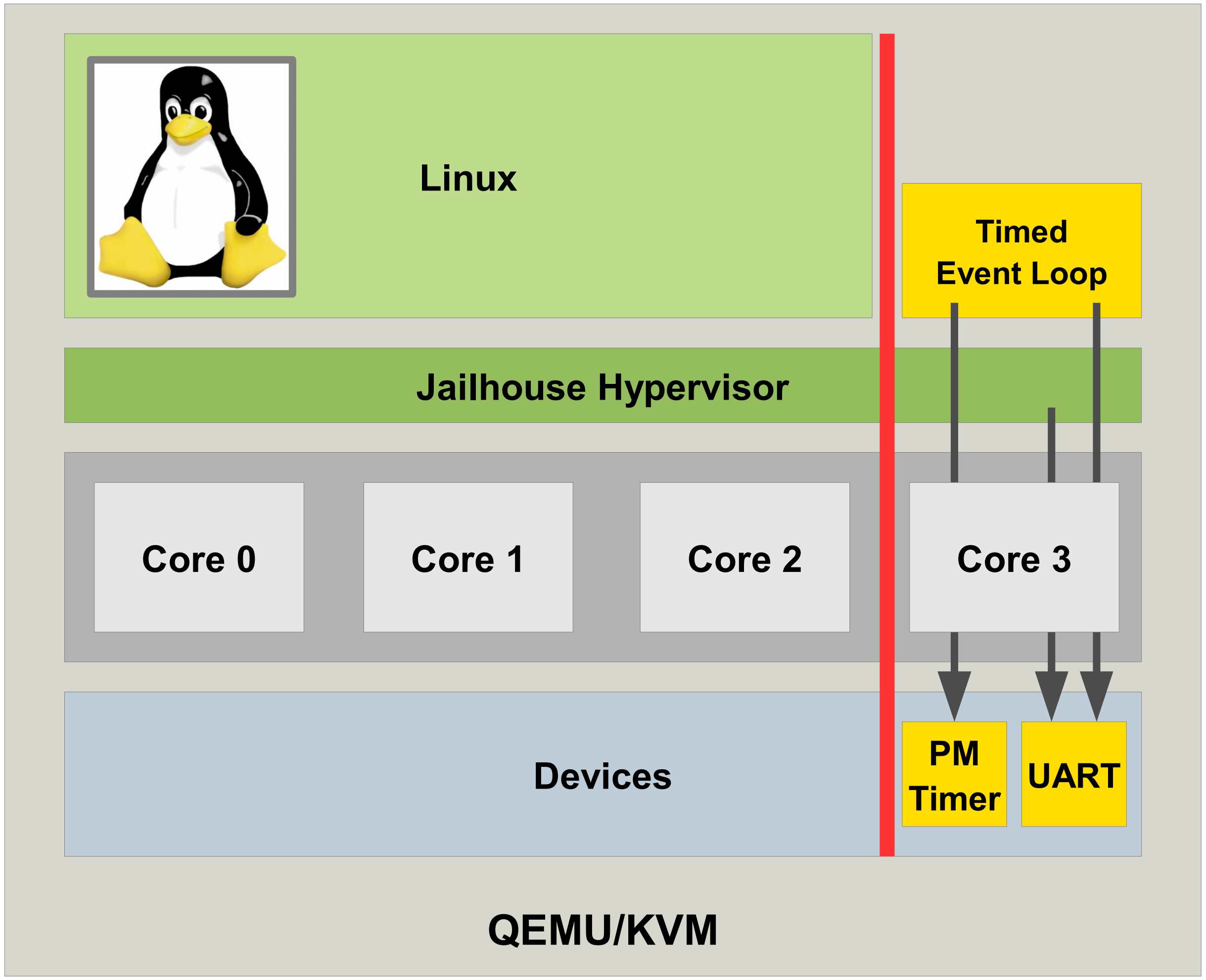


SIEMENS

Jailhouse in QEMU/KVM

- **All steps documented in README.md**
- **Host-side requirements**
  - Intel host with VT-x (**no** VT-d required)
  - Recent kernel (>= 4.4)
  - Very recent QEMU (>= 2.7)
- **Linux guest image**
  - Not too old kernel
  - Tools and sources to build modules **for the guest kernel**
  - Jailhouse source (master)
- **I've prepared something...**

QEMU/KVM Setup: Running a Bare-Metal Workload (apic-demo)



QEMU/KVM Setup: Steps Shown

- `insmod jailhouse.ko`
- `jailhouse enable qemu-vm.cell`

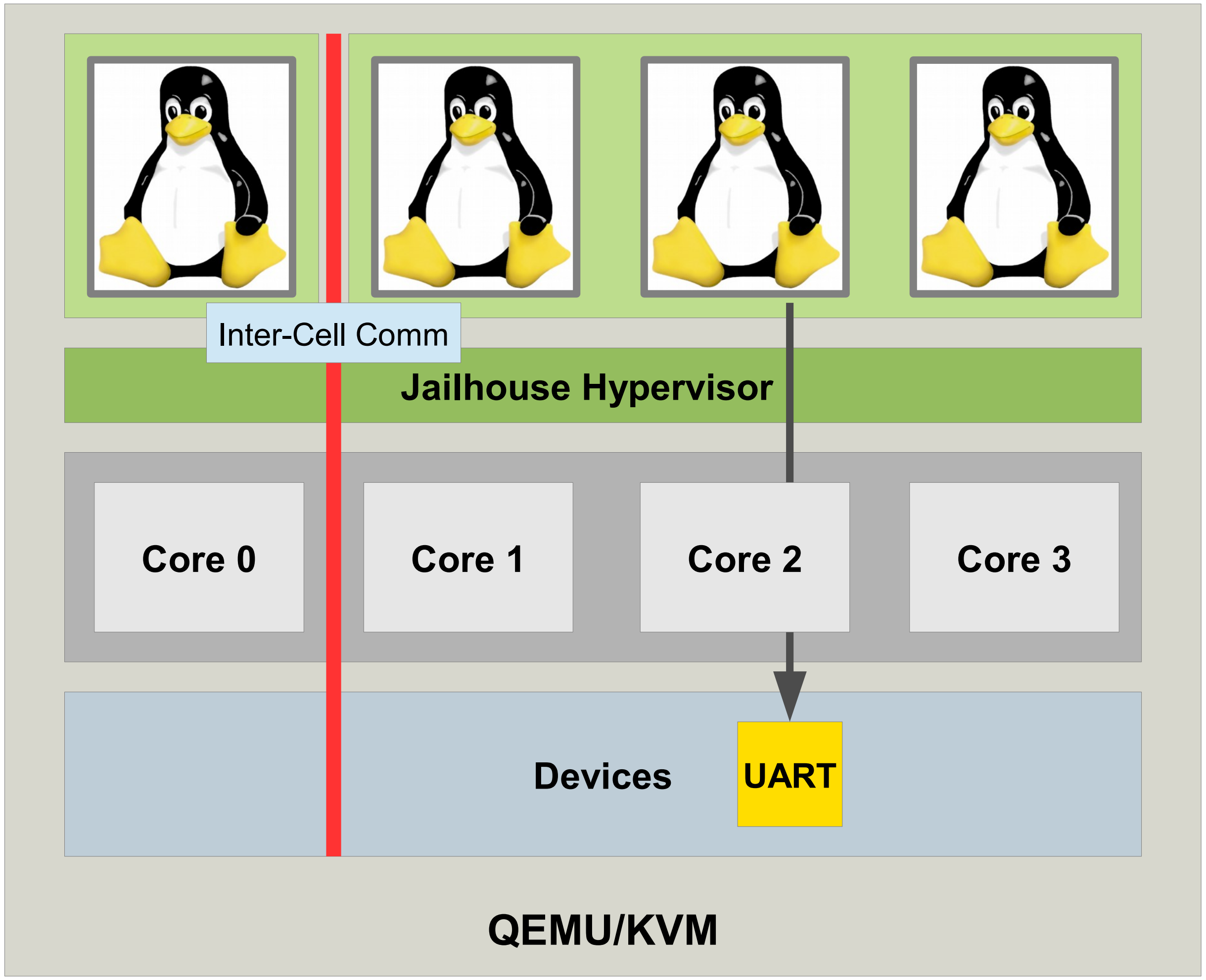
- `jailhouse cell create apic-demo.cell`
- `jailhouse cell load apic-demo apic-demo.bin -a 0xf0000`
- `jailhouse cell start apic-demo`

- `jailhouse cell list`
- `jailhouse cell stats apic-demo`

- `jailhouse cell destroy apic-demo`
- `jailhouse disable`

QEMU/KVM Setup #2: Linux as Non-Root Cell

SIEMENS



QEMU/KVM Setup #2:
Steps Shown

SIEMENS

- `insmod jailhouse.ko`
- `jailhouse enable qemu-vm.cell`

-
  ```
  jailhouse cell linux linux-x86-demo.cell \
      bzImage -i initrd -c "console=ttyS0"
  ```
- `ssh other-cell`

Let's use real hardware!

- **Supermicro Mini-ITX board X10SDV-TLN4F**
  - Xeon D-1540
    - 8 cores, 2 threads each
    - up to 2 GHz
    - No on-chip GPU (lower latencies...!)
  - 32 GB RAM
  - 2x   1 GBit/s Ethernet
  - 2x 10 GBit/s Ethernet
  - Some slowly rotating rust

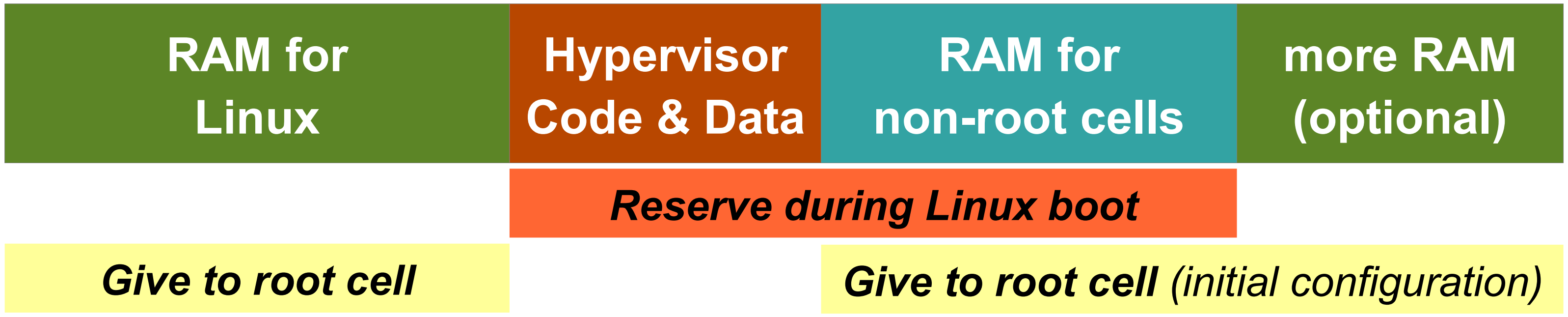
SIEMENS

Wanted: System Configuration

- **Preparation**
  - Connect a UART (or step in the dark...)
  - Enable VT-x and VT-d in the BIOS,
    keep TXT (Trusted Execution Technology) off
  - Kernel? >= 4.4, vanilla preferred, no patching for first steps
  - intel_iommu=off, intremap=on
- `jailhouse config create configs/xeon-d.c \`
  `    --mem-hv 6M --mem-inmates 59M`
  - Enables everything Linux considers valid
  - Rather permissive configuration
  - Defaults to first UART
- `make KDIR=/patch/to/kernel/build`
- `jailhouse hardware check confis/xeon-d.cell`

RAM Reservation

- **Typical RAM layout** (generated and used for QEMU)



- **Reserve physical memory**
  - `memmap=SIZE$ADDRESS`
  - `mem=PHYSICAL_SIZE_MINUS_RESERVATION`
  - Device tree (ARM/ARM64 only)
- **grub2 is ~~your friend~~ evil**
  - Use proper escaping in `/etc/default/grub`

```
GRUB_CMDLINE_LINUX_DEFAULT="memmap=66M\\\$0x3b000000"
```

First Try – First Crash...?

- `jailhouse enable xeon-d.cell`

```
FATAL: Invalid MMIO/RAM read, addr: 0x00000000798ab018
RIP: 0xffffffff813c60c1 RSP: 0xffff88087f5c3da8 FLAGS: 10246
RAX: 0xffffc900035dc018 RBX: 0xffff88087f5c3e10 RCX: 0xffff88087f5c3da8
RDX: 0x0000000000000000 RSI: 0x00000000798ab020 RDI: 0x00000000798ab018
CS: 10 BASE: 0x0000000000000000 AR-BYTES: a09b EFER.LMA 1
CR0: 0x0000000080050033 CR3: 0x000000085d9bf000 CR4: 0x00000000003426e0
EFER: 0x0000000000000d01
Parking CPU 15 (Cell: "RootCell")
```

- **Don't get confused by “unsupported instruction” message!**
  - Jailhouse only supports few (but enough) MMIO access instructions
- **Check /proc/iomem**

```
79711000-798defff : reserved
  79793018-79793018 : APEI ERST
  ...
```

SIEMENS

System Configuration Format

- **Binary format, generated from C sources (currently)**
- **`hypervisor/include/jailhouse/cell-config.h`**
  - System config header
    - Hypervisor location and size
    - Debug console address, size, type
    - Platform parameters (poor-man's ACPI / device tree)
    - interrupt_limit (x86 only so far: IOAPIC pins + MSI/MSI-X vectors)
  - Root cell header
    - Cell name
    - Flags (0 for root cell)
    - Number / size of resources
  - Root cell resources

SIEMENS

Configurable Cell Resources

- **CPU set (bitmap)**
  - Bit number = Linux logical CPU IDs
- **Memory regions**
  - Physical address, virtual address, size, flags (**`JAILHOUSE_MEM_*`**)
  - **`JAILHOUSE_MEM_READ`**, ...**`WRITE`**, ...**`EXECUTE`**
  - ...**`DMA`** (add to IOMMU page tables)
  - ...**`IO`** (MMIO)
  - ...**`IO_8/16/32/64`** (subpage MMIO: permitted access widths)
  - ...**`IO_UNALIGNED`** (subpage MMIO: permit unaligned access)
  - ...**`COMM_REGION`** (info/signal page hypervisor ↔ *non-root* cell)
  - ...**`LOADABLE`** (*non-root* cell memory loadable by root cell)
  - ...**`ROOTSHARED`** (non-root cell memory shared with root cell)

Fixing the APEI Crash

- Add corresponding memory region

```
/* MemRegion: 79711000-798defff : reserved (APEI) */
{
    .phys_start = 0x79711000,
    .virt_start = 0x79711000,
    .size = 0x1ce000,
    .flags = JAILHOUSE_MEM_READ | JAILHOUSE_MEM_WRITE |
        JAILHOUSE_MEM_EXECUTE | JAILHOUSE_MEM_DMA,
},
```

- Don't forget: extend mem_regions array
- `make`
- `jailhouse enable xeon-d.cell`

SIEMENS

PIO Access Violation

- A made-up violation

```
FATAL: Invalid PIO write, port: 70 size: 1
RIP: 0xffffffff810240b6 RSP: 0xffff88083d1f3cf8 FLAGS: 46
RAX: 0x000000000000000a RBX: 0x0000000000000282 RCX: 0x0000000000000000
RDX: 0x0000000000000001 RSI: 0xffff88083d1f3da4 RDI: 0x000000000000000a
CS: 10 BASE: 0x0000000000000000 AR-BYTES: a09b EFER.LMA 1
CR0: 0x0000000080050033 CR3: 0x000000084639e000 CR4: 0x00000000003426f0
EFER: 0x0000000000000d01
Parking CPU 0 (Cell: "RootCell")
```

- Check /proc/ioports

```
0070-0071 : rtc0
```

SIEMENS

Configurable Cell Resources (2)

- **PIO bitmap (x86 port-based I/O)**
  - Covers full PIO address space 0x0000..0xffff
  - Bit cleared: access permitted
- **IRQ chips (pin-based interrupts)**
  - MMIO address (for correlation), ID (used by IOMMU)
  - PIN bitmap (up to 128 pins) + number of first pin in bitmap
- **Cache regions (warning: to be reworked!)**
  - Start, size, type, flags (unused)
  - Types: L3, L3 code-only, L3 data-only

SIEMENS

Configurable Cell Resources (3)

- **PCI devices**
  - BDF (bits 15..8: bus, 7:3: device, 2:0 function), domain (yet unsupported)
  - Type (device, bridge, shared memory device)
  - Index into capability list, number of entries (shareable with other devices)
  - IOMMU association, MSI/MSI-X parameters, modifiable bits in BARs
  - Shared-mem device: index to associated memory region
  - Many fields... use the config generator where possible!
- **PCI capabilities**
  - ID (or'ed with **`JAILHOUSE_PCI_EXT_CAP`** for PCIe caps)
  - Start and length in config space
  - Flags: so far only **`JAILHOUSE_PCICAPS_WRITE`**
  - Again: use the config generator

SIEMENS

More on Debug Consoles

- **Memory-mapped UARTs**
  - Set `JAILHOUSE_MEM_IO` in flags, define physical address and size
  - x86
    - 8250-compatible
    - `JAILHOUSE_MEM_IO_32`: 32-bit register width
- **VGA text console as alternative (x86-only)**
  - See `jailhouse/Documentation/vga-console.txt`

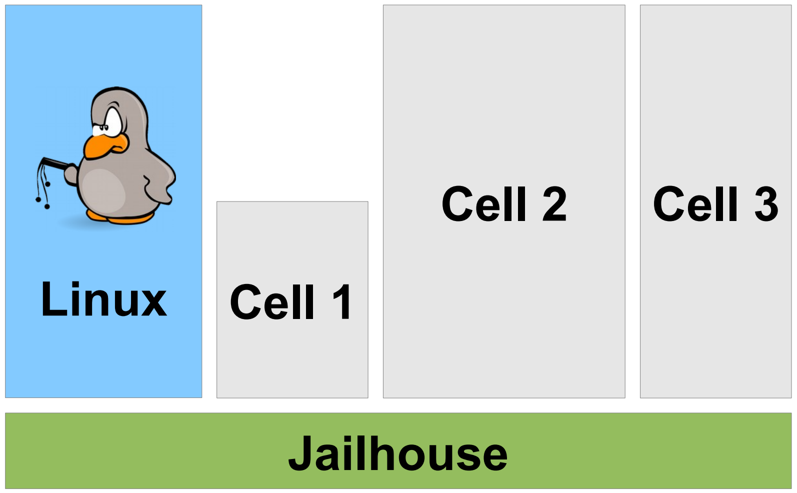
SIEMENS

Non-Root Cell Setup

- **Config structure: Cell header + resources** (see system config)
- **Cell creation removes resources from root cell**
  - Not all resources need to belong to root cell before
  - ...except for CPUs
  - Cell destruction reverts this
- **Linux must stop using non-root cell resource**
  - Memory reservations
  - CPU hotplug (done by driver; CPU 0 requires special care)
  - Device hotplug (only PCI done by driver)
- **Cell flags `JAILHOUSE_CELL_PASSIVE_COMMREG`**:
  No voting over cell shutdown or system reconfiguration

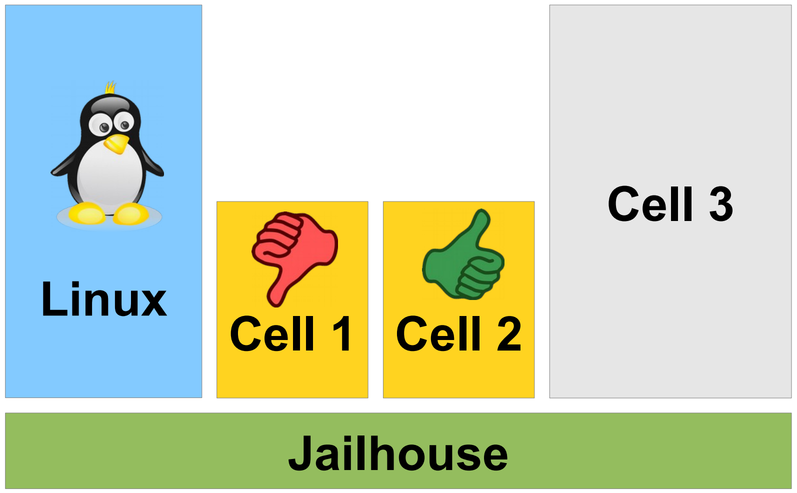
Managing Non-Root Cells: Two Models

Open Model



- Linux (root cell) is in control
- Cells not involved in management decisions
- Sufficient if root cell is trusted

Safety Model



- Linux controls, but...
- Certain cells are configured to vote over management decisions
- Building block for safe operation

Creating Non-Root Cell Configs

- **Sorry, no tool support yet**
- **Use the source, Luke!**
  - Derive from existing examples, on x86: `{tiny,apic,ioapic,e1000,ivshmem,smp,linux-x86}-demo.c`
  - Copy fragments from root cell config
- **Contributions to some “jailhouse config cell” welcome!**

SIEMENS

Non-Root Linux Cell

- `jailhouse cell linux` (Python helper)
  - Create Linux cell
  - Load loader, kernel, initrd, command line
  - Start cell
- **Kernel patches required on x86**
  - `git://git.kiszka.org/linux.git queues/jailhouse`
- **See `Documentation/non-root-linux.txt` for further details**

SIEMENS

Adding a PCI Device to a Non-Root Cell

- **Example: Transfer ownership of second 1-Gbit NIC (0000:06:00.1)**
- **Elements to be copied from root cell config**
  - PCI device
  - PCI capabilities
  - MMIO regions (check /proc/iomem for 06:00.1)
  - PIO ranges (check /proc/ioports for 06:00.1)
- **Adjust**
  - Number of PCI devices, MMIO regions, PCI capabilities
  - PCI capability start index!
  - Do not assign the MSI-X region!!

PCI Capability Access Violation

- **Creating the non-root Linux cell**

```
FATAL: Invalid PCI config write, port: cfc, size 2, address port: 800601a8
RIP: 0xffffffff81576168 RSP: 0xffff88084aba7bf0 FLAGS: 46
[...]
Parking CPU 0 (Cell: "RootCell")
```

- **Dissect address port**
  - Bit 31: enable (always set)
  - Bits 23..8: BDF
  - Bits 7..0: config space offset
- **Check prepared caps entry in config**
  - Start 0xa0 / length 44 → ID 0x10 → PCIe Capability Structure
  - Harmless? Set `JAILHOUSE_PCICAPS_WRITE` in entry flags

SIEMENS

Setting Up a Shared Memory Device

- **Two cells share memory via an ivshmem device**
  - If devices use same BDF
  - If memory region addresses same physical RAM
- **Use pattern found in `qemu-vm.c` and `linux-x86-demo.c`**
  - `.type = JAILHOUSE_PCI_TYPE_IVSHMEM`
  - `.bdf = 0xf << 3         // 00:0f.0`
  - `.bar_mask = …           // invariant for current device`
  - `.shmem_region = …       // index into mem_regions`
  - `.num_msix_vectors = 1 // invariant for current device`
- **Set up memory region**
  - Use >= 1MB for ivshmem-net (network over shared memory)
  - Set virt = phys address
  - Don't forget memory region flag `JAILHOUSE_MEM_ROOTSHARED`

SIEMENS

Traps & Pitfalls – x86 Edition

- **Overlapping memory regions / overlaps with hypervisor**
- **Accidentally allowing direct access to**
  - APIC (0xfee00000)
  - IOAPICs (0xfec00000 and more...)
  - MSI-X regions of PCI devices
    (check all jailhouse_pci_device.msix_address and .msix_region_size)
  - IOMMU unit (check platform_info.iommu_units.base and .size)
  - Memory-mapped PCI config space (platform_info.mmconfig_base)
- **Mismatch in indexes after adding/removing entries**
  - Shared memory device region
  - PCI device capability

=> We need to improve on automatic checks

Let's look at the ARM side of life

- **LeMaker HiKey board**
  - Hi6220 SoC (HiSilicon Kirin)
  - 8 A53 cores (ARMv8), up to 1.2 GHz
  - 2 GB RAM
  - 8 GB eMMC
  - WiFi + Bluetooth
  - 2x USB 2.0
  - ...

SIEMENS

ARM64 Background Information

- **Key differences of ARM & ARM64 to x86**
  - No config generator so far
  - Variety of UARTs for debug console
  - Build-time configuration needed (to be reduced)
- **ARM64 status**
  - Contributed by Huawei, developed over >1 year
  - Recently rebased after ARM rework (IRQ handling, PSCI, ...)
  - Awaiting merge in wip/arm64 branch
  - Support for HiKey still fresh

SIEMENS

Getting started on ARM64

- **Prepare custom kernel**
  - Patch needed for **`EXPORT_SYMBOL_GPL(__boot_cpu_mode)`**
  - Configuration adaptions
    - Disable KVM
    - Disable ARMv8.1 Virtualization Host Extension (upcoming)
- **Prepare system**
  - Attach and test UART console
  - Boot self-built kernel
  - Reserve some memory for hypervisor and cells
    - **`mem=PHYSICAL_SIZE_MINUS_RESERVATION`**
- **Identify UART type and address**

Getting started on ARM64 (2)

- **Prepare Jailhouse sources**
  - Checkout working branch from latest git (currently wip/arm64)
  - Set up `hypervisor/include/jailhouse/config.h`, using AMD Seattle board initially

```
#define CONFIG_ARM_GIC_V2           1
#define CONFIG_MACH_AMD_SEATTLE     1
#define CONFIG_SERIAL_AMBA_PL011    1
#define JAILHOUSE_BASE              0x10000000
```

-
```
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- \
    KDIR=/patch/to/kernel/build
```
- **Check also** `Documentation/setup-on-banana-pi-arm-board.md`

Creating a System Config

- **Use `configs/amd-seattle.c` as template**
- **Adjust hypervisor region**
  - Put hypervisor at top of reserved memory (allows cells to grow downward)
  - Update **`JAILHOUSE_BASE`** in config.h
  - Grant some 8MB
- **Set debug console address and size**
- **Set GICD, GICC, GICH, GICV according to device tree**
- **Degine 2 memory regions at least**
  - RAM
  - MMIO (watch out for GIC*!)
- **Grant IRQs up to highest used pin**
  - Check /proc/interrupts, device tree, SoC manual
  - Granting too much may cause access to undefined registers

SIEMENS

Enable the UART

- **Check available support**
  - PrimeCell PL011 (arm-common)
  - DesignWare 8250 (arm)
  - Tegra (arm)
- **Writing a new one is simple**
  - Only polled output required
  - Check hypervisor/arch/arm/include/arm/uart*.h for implementations (NOTE: refactoring planned!)
- **Compile, cross fingers, and type**

```
jailhouse enable my-new-board.cell
```

What you may get on rainy days

- **Made-up MMIO violation**

```
Unhandled data read at 0xf8008024(4)

FATAL: exception unhandled trap
Cell state before exception:
 pc: ffffff800877bff4   lr: ffffff8008126e68 spsr: 400001c5     EL1
 sp: ffffffc078e7cd00  esr: 24 1 1800006
 x0: ffffff8008012024   x1: ffffffc06e5a4100   x2: 0000000000000001
...
x27: 0000000000000006  x28: ffffffc06e4a0000  x29: ffffffc078e7cd00

Parking CPU 6 (Cell: "HiKey")
```

- **Check /proc/iomem...**

Traps & Pitfalls – ARM Edition

- **Overlapping memory regions / overlaps with hypervisor**
  (e.g. due to forgotten/too small reservation)
- **Accidentally allowing direct access to**
  - GICC (double-check config!)
  - GICD
- **More to come while adding x86 features...**

ARM / ARM64 Status & Outlook

- **ARM64 is ready to be merged**
- **Commonly missing features** (after ARM64 merge)
  - PCI support
  - Virtual PCI bus, precondition for...
  - ...shared memory device (“ivshmem 2.0”)
  - SMMU support
  - Config generator
- **wip/linux-arm-inmate**
  - `jailhouse cell linux` for ARM/ARM64 would be nice
- **More SoC & boards are welcome!**
  - TI Sitara AM572x (ARM) under preparation
  - Increasing need to refactor for this
    (less boilerplate, less duplications, less hard-coded parameters)

SIEMENS

Any Questions?

Thank you!

https://github.com/siemens/jailhouse

jailhouse-dev@googlegroups.com

Jan Kiszka <jan.kiszka@siemens.com>