emtrion
embedded systems

Virtualisierung im industriellen Umfeld

Frank Erdrich, frank.erdrich@emtrion.de
ESE-Kongress 2016, Sindelfingen

emtrion
embedded systems

Wer virtualisiert?

emtrion
embedded systems

Grundlagen Virtualisierung

Begriffe I

emtrion
embedded systems

Virtualisierung

Virtualisierung bezeichnet in der Informatik die Nachbildung eines Hard- oder Software-„Objekts“ durch ein ähnliches Objekt vom selben Typ mit Hilfe einer Software-Schicht. Dadurch lassen sich virtuelle (d. h. nicht-physische) Dinge wie emulierte Hardware, Betriebssysteme, Datenspeicher oder Netzwerkressourcen erzeugen. **Dies erlaubt es etwa, […] ein Betriebssystem innerhalb eines anderen auszuführen.**

Quelle: https://de.wikipedia.org/wiki/Virtualisierung_(Informatik)

Warum Virtualisierung?

emtrion
embedded systems

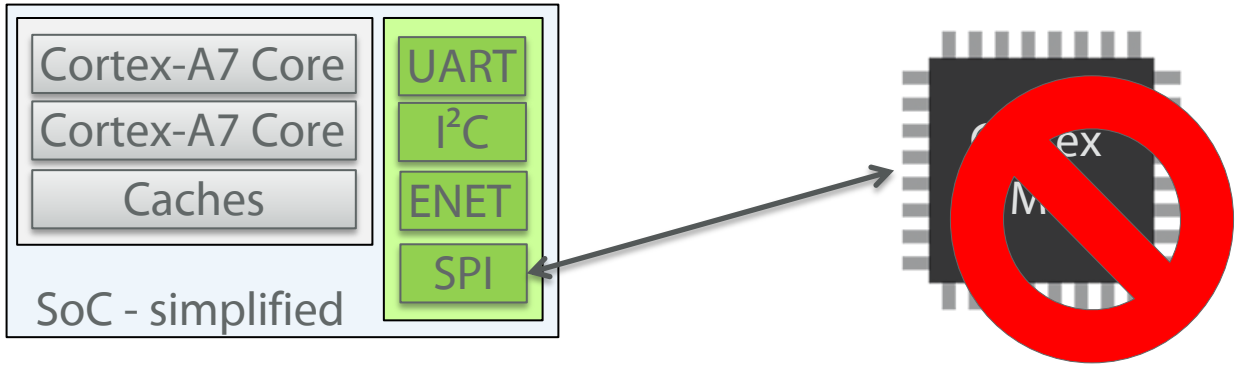
- Verschiedene Betriebssysteme parallel
  - Linux
  - FreeRTOS
  - …

- Safety/Security
  - Fehler in einem Teilsystem führen nicht zu komplettem Systemversagen
  - Neustart des fehlerhaften Teilsystems

Warum Virtualisierung?

emtrion
embedded systems

- Kostenersparnis
  - Weniger Hardware-Komponenten
  - kleinere Leiterplatte



Warum Virtualisierung?

emtrion
embedded systems

- Sicherheitsaspekte
  - Sicherheitskritische Software in getrennter Umgebung
  - Kommunikation mit GUI über dedizierten Kanal
  - Latenzen in GUI behindern Ausführung der kritischen Software nicht
- Updatesicherheit
  - Update in Test-VM
  - Ausrollen nach erfolgreichem Test

Warum nicht virtualisieren?

emtrion
embedded systems

- Performanzverlust
  - Overhead durch Hypervisor
  - Hardwarzugriffe u. U. geteilt oder emuliert
  - Ziel: Hypervisoreingriffe vermeiden
- Systemkomplexität
- Zusätzliche Kommunikationsschnittstelle

Begriffe II

emtrion
embedded systems

Hypervisor/Virtual Machine Monitor

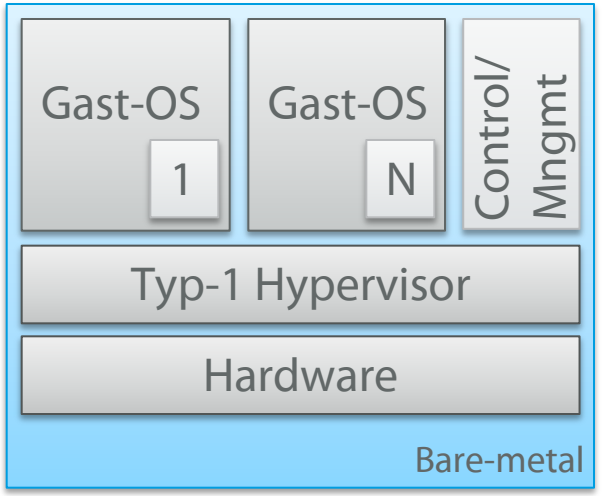
Hypervisor, auch **Virtual-Machine-Monitor** (aus englisch *virtual machine monitor*, kurz **VMM**) genannt, ist die Bezeichnung für eine Klasse von Systemen […], die als **abstrahierende Schicht** zwischen tatsächlich vorhandener Hardware […] und weiteren zu installierenden Betriebssystemen dient. Solche Systeme erlauben es, eine virtuelle Umgebung […] zu definieren, die unabhängig von der tatsächlich vorhandenen Hardware als Basis für die Installation von (Gast-) Betriebssystemen dient.

Quelle: https://de.wikipedia.org/wiki/Hypervisor

emtrion
embedded systems

Typ-1 Hypervisor

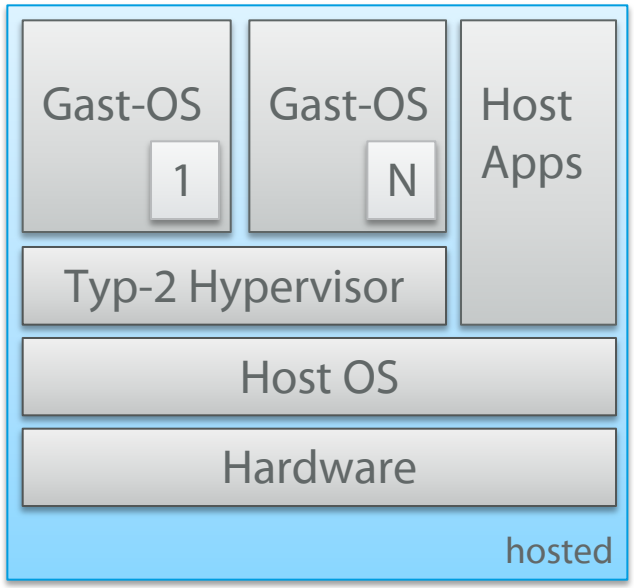
- Bare-Metal Hypervisor
  - Wird direkt auf der Hardware ausgeführt
  - Benötigt Treiber für die Hardware



emtrion
embedded systems

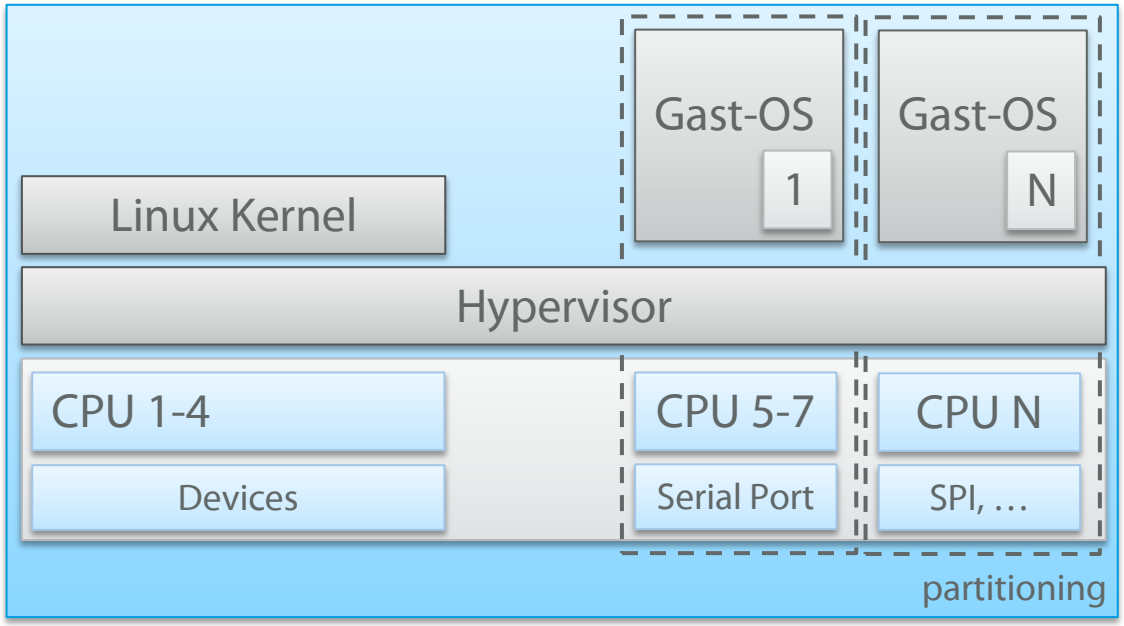
Typ-2 Hypervisor

- Hypervisor läuft auf einem Host-System
- Verwendet Peripherietreiber des Host-Systems



emtrion
embedded systems

Partitionierender Hypervisor



Was wird benötigt?

emtrion
embedded systems

- ARM Virtualisierungs-Erweiterungen
  - ab Cortex-A7/A15
- ARM Security-Erweiterungen
  - Seit ARM9???

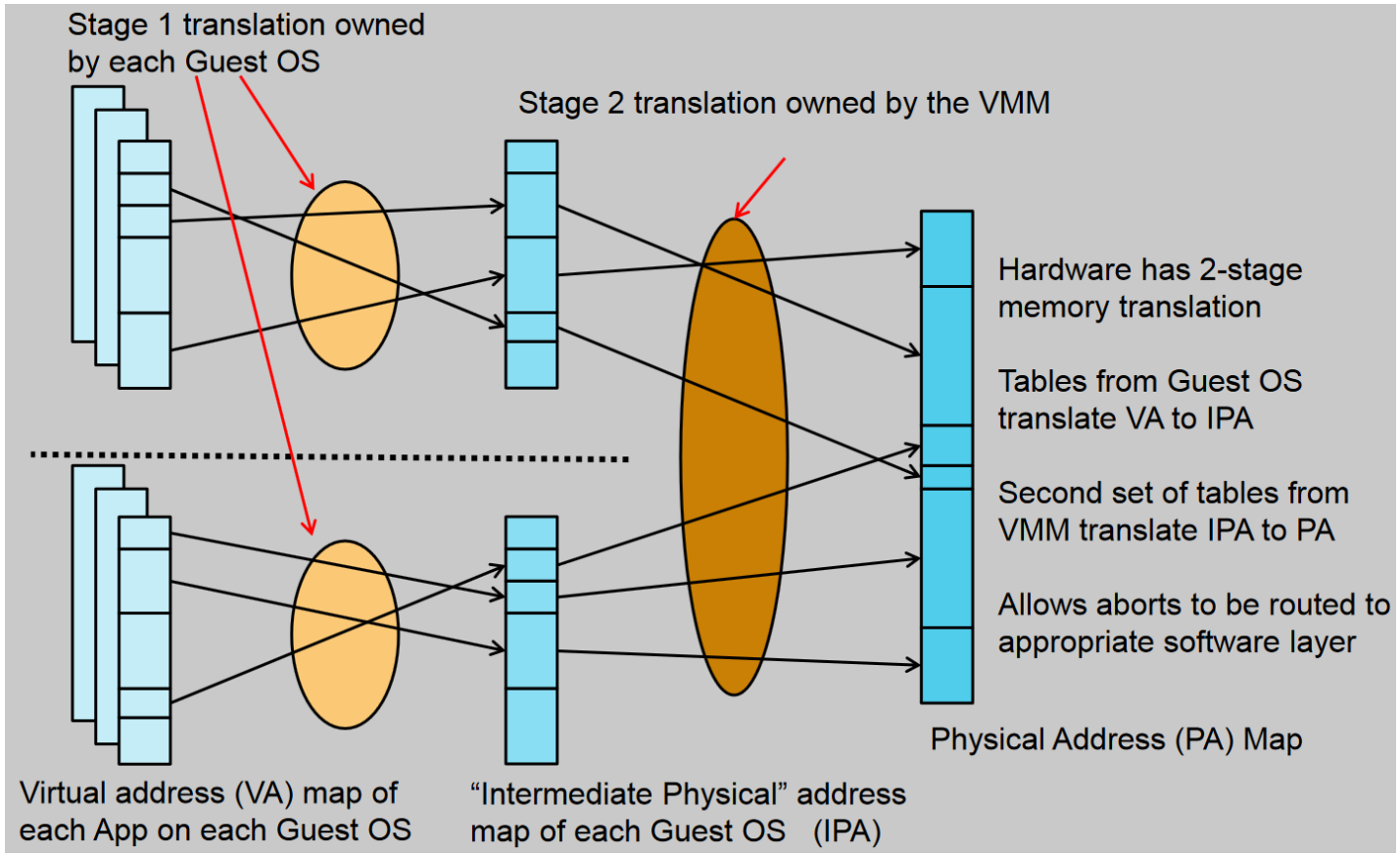
ARM Virtualisierungs-Erweiterung

emtrion
embedded systems

- Hyp Modus zur Ausführung des Hypervisor (PL2)
  - Supervisor Mode für OS (PL1)
  - User Mode für Applikationen (PL0)
- Hardwareerweiterung, um Hypervisoreingriffe zu verringern
  - Page Table Management
  - Speicherrelokation für Device Treiber (System MMU)
  - Virtual Generic Interrupt Controller (GIC)

Virtueller Speicher

emtrion
embedded systems

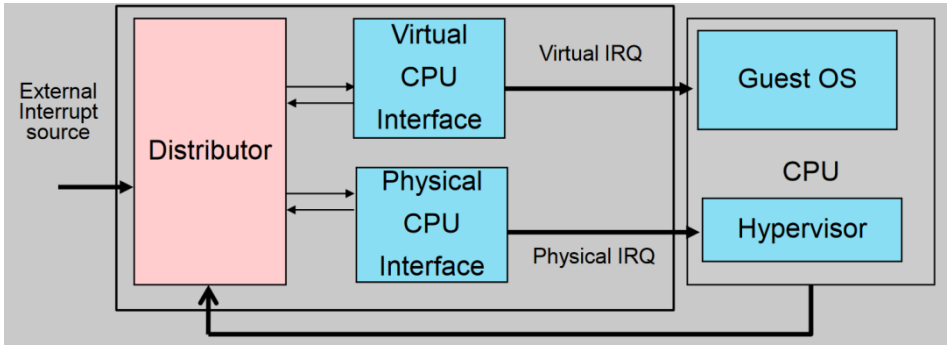


Quelle: Hardware accelerated Virtualization in the ARM Cortex™ Processors – John Goodacre

ARM Interrupt Virtualisierung

emtrion
embedded systems

- Virtueller Interrupt Controller – virtuelles GIC Interface
  - Physische und virtuelle Register
  - Virtualisierte Systeme greifen auf die virtuellen Register zu
- Gast-OS interagiert mit virtuellem Interrupt Controller
  - Dadurch sind keine Hypercalls notwendig, beispielsweise um einen IRQ zu bestätigen oder die IRQ Priorität zu verändern
- IRQ löst Hypervisor Trap aus
  - Hypervisor „verteilt“ IRQ weiter
  - Virtueller IRQ triggert Gast-OS



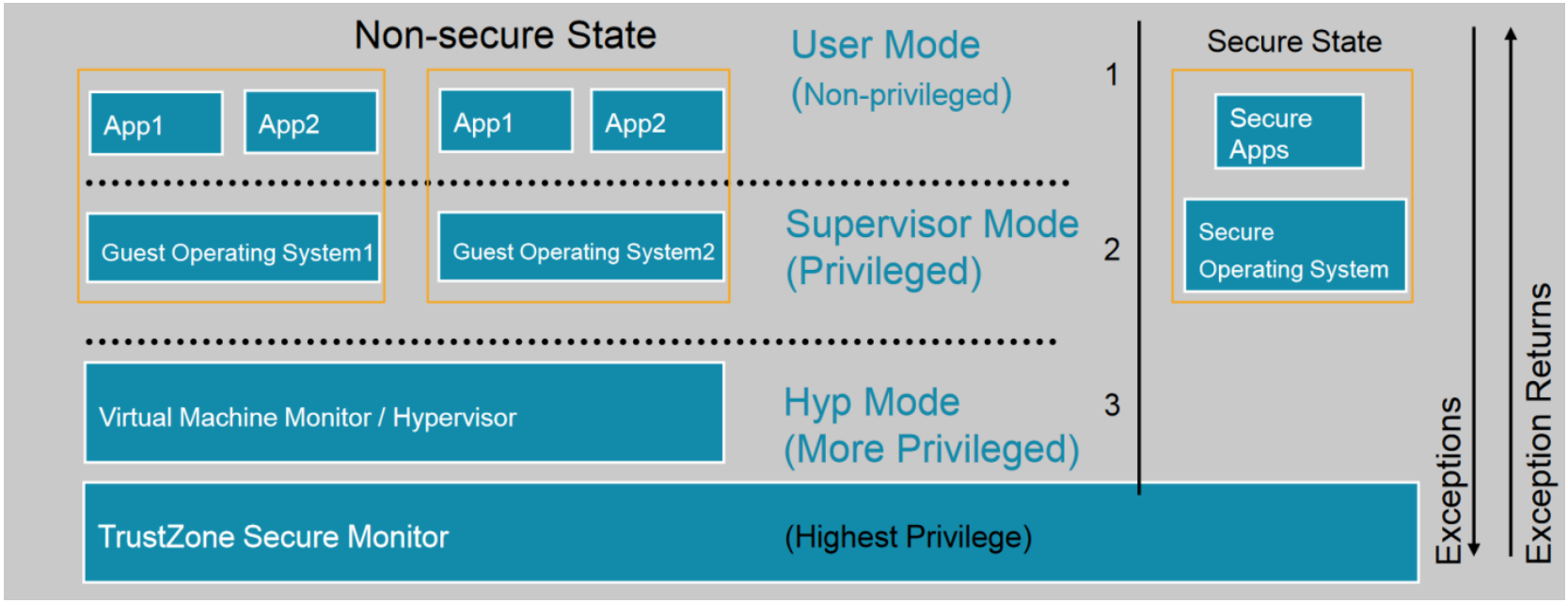
ARM Security Erweiterungen

emtrion
embedded systems

- TrustZone-Erweiterung
- Weiterer Privilegierungslevel
- Kann als zusätzliche Virtualisierung gesehen werden
- Getrennt laufende Sicherheitsapplikationen

ARM Privilegierungsebenen

emtrion
embedded systems



Quelle: Hardware accelerated Virtualization in the ARM Cortex™ Processors – John Goodacre

emtrion
embedded systems

Jailhouse

Jailhouse – Partitioning Hypervisor

emtrion
embedded systems

- Entworfen für
  - Mehrkern Plattformen (ab 2 Kernen sinnvoll nutzbar)
  - Echtzeit Aufgaben
  - Sicherheitskritische Aufgaben
- Keine virtuellen CPUs
- Keine virtuellen Devices

Jailhouse – Features und Merkmale

emtrion
embedded systems

- Zugriffskontrolle anstelle von Virtualisierung von Ressourcen
- 1:1 Ressourcenzuteilung, kein Scheduling
- Existenz von Jailhouse wird nicht versteckt
- Benötigt laufendes Linux, kein Bare-Metal Hypervisor

Quelle: http://events.linuxfoundation.org/sites/events/files/slides/LinuxConNA-2015-Jailhouse_0.pdf – Jan Kiszka

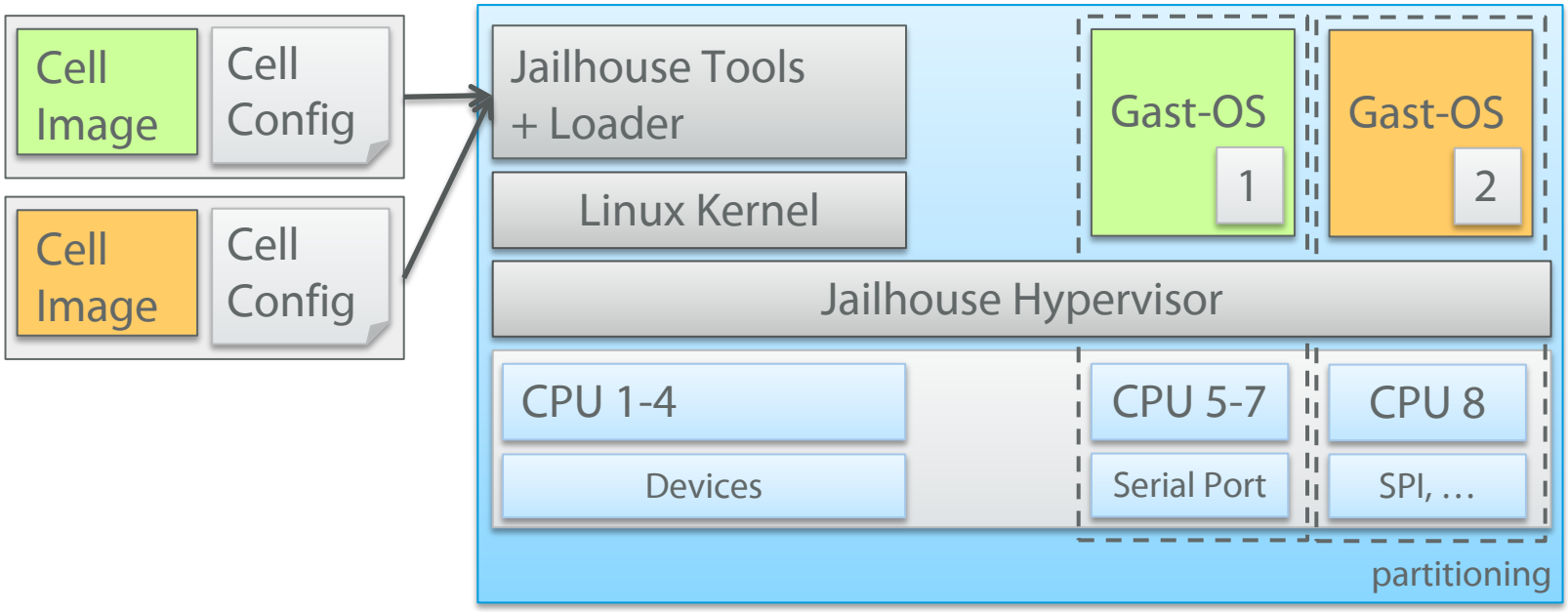
Jailhouse – Features und Merkmale

emtrion
embedded systems

- Zertifizierbarkeit
- Kleine Codebasis
  - Derzeit kleiner 10k LoC
  - Xen > 400k LoC
- Host-Linux übernimmt
  - Start von Jailhouse sowie laden und starten der Zellen
  - Kontrolliert Jailhouse

Jailhouse - Struktur

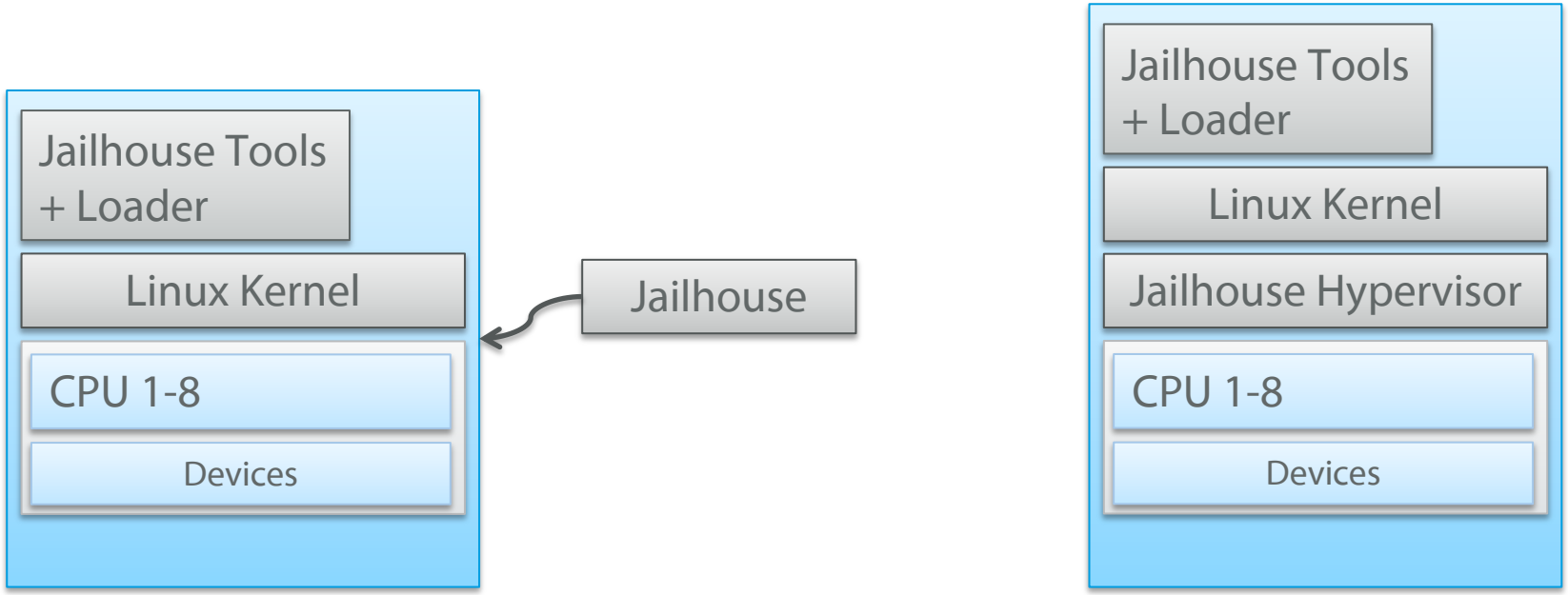
emtrion
embedded systems



Erste Schritte – Cell Start

emtrion
embedded systems

- $ modprobe jailhouse
- $ jailhouse enable ~/jailhouse/configs/bananapi.cell
- $ jailhouse cell create ~/jailhouse/configs/bannapi-freertos-demo.cell
- $ jailhouse cell load FreeRTOS ~/freertos-cell/freertos-demo.bin
- $ jailhouse cell start FreeRTOS



Cell Config

emtrion
embedded systems

```
.cell = {
                .signature = JAILHOUSE_CELL_DESC_SIGNATURE,
                .name = "bananapi-uart-demo",
                .flags = JAILHOUSE_CELL_PASSIVE_COMMREG,
                .cpu_set_size = sizeof(config.cpus),
                .num_memory_regions = ARRAY_SIZE(config.mem_regions),
},
.cpus = {
                0x2,
},
.mem_regions = {
                /* UART 4-7 */ {
                                .phys_start = 0x01c29000,
                                .virt_start = 0x01c29000,
                                .size = 0x1000,
                                .flags = JAILHOUSE_MEM_READ | JAILHOUSE_MEM_WRITE | JAILHOUSE_MEM_IO,
                },
                /* RAM */ {
                                .phys_start = 0x7bff0000,
                                .virt_start = 0,
                                .size = 0x00010000,
                                .flags = JAILHOUSE_MEM_READ | JAILHOUSE_MEM_WRITE |
                                                JAILHOUSE_MEM_EXECUTE | JAILHOUSE_MEM_LOADABLE,
                },
}
```

Erfahrungen mit Virtualisierung

emtrion
embedded systems

- Aktuell nur wenige ARM SoCs außerhalb des Smartphonemarkts mit Virtualisierungsextensions
  - Tendenz steigend (Renesas RZ/G, Omap5, i.MX7, LS10xx, …)
- Unterstützung für Virtualisierung im Kernel nicht zwangsläufig für einen SoC gegeben
- Erfahrung mit Jailhouse größtenteils positiv
- OpenSource -> Was nicht passt, wird passend gemacht

emtrion
embedded systems

Virtualisierung im industriellen Umfeld

Frank Erdrich, frank.erdrich@emtrion.de
ESE-Kongress 2016, Sindelfingen

www.emtrion.de

Referenzen & weiterführende Informationen

emtrion
embedded systems

- Details zur ARM Virtualisierung
  - http://www.slideshare.net/linaroorg/arm-architecture-overview-32539155
  - http://de.slideshare.net/jserv/embedded-hypervisor-for-arm
  - http://de.slideshare.net/jserv/embedded-hypervisor-for-arm
  - http://www-archive.xenproject.org/files/xensummit_seoul11/nov2/2_XSAsia11_JGoodacre_HW_accelerated_virtualization_in_the_ARM_Cortex_processors.pdf
- ARM Infocenter
  - http://infocenter.arm.com/help/index.jsp
- Jailhouse
  - https://docs.google.com/file/d/0B6HTUUWSPdd-Zl93MVhlMnRJRjg/edit
  - http://events.linuxfoundation.org/sites/events/files/slides/LinuxConNA-2015-Jailhouse_0.pdf
  - https://groups.google.com/forum/#!forum/jailhouse-dev