Look Mum, No VM Exits! (Almost)
Static Hardware Partitioning with the Jailhouse Hypervisor

Ralf Ramsauer*, Jan Kiszka†, Daniel Lohmann‡, Wolfgang Mauerer*†

* Technical University of Applied Sciences Regensburg
† Siemens AG, Corporate Research and Technology
‡ University of Hanover

OSPERT 2017, Dubrovnik, Croatia

June 27, 2017
Motivation

- Driven by consolidation of physical hardware units [1]
- Reduction of physical control units
- **System of systems** on a chip
  - Increasing complexity
  - Scalability
  - Maintainability
- Consolidation of multiple software stacks requires safe isolation
1. Introduction

Embedded hypervisors

- Consolidation of services
- Mixed-criticality systems
- Maintain RT capabilities
- Minimal impact
- Certifiability

Mixed-criticality system

Image © RTC Group, Inc
Related work (incomplete!)

- **Quest-V** [2]
  - Allows direct I/O access
  - Rich set of device drivers (OS + VMM)
  - Virtualisation only for isolation
  - Communication: Shared memory + IPI
  - Only trap on violations
  - Traditional boot sequence
- **PikeOS** [3]
- **XtratuM** [4]
2. Related Work

Related work (incomplete!)

- **Quest-V** [2]
- **PikeOS** [3]
  - Allows direct I/O access
  - Paravirtualisation, hardware-assisted virtualisation
  - Time or Event triggered scheduling
- **XtratuM** [4]

PikeOS architecture

Images © SYSGO AG
2. Related Work

Related work (incomplete!)

- **Quest-V** [2]
- **PikeOS** [3]
- **XtratuM** [4]
  - ARINC 653
  - Feature-rich hypercall interface
  - Paravirtualisation
  - Schedules partitions
  - Fully-fledged hypervisor

XtratuM architecture

Images © fentISS S.L., Universitat Politècnica de València
Jailhouse
Yet another hypervisor?
3. Jailhouse

What makes Jailhouse different?

Jailhouse, an **Exohypervisor**

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition »cell« management
  - Control and monitoring
  - Deferred VMM activation
Jailhouse, an *Exohypervisor*

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncrritical work to Linux**
  - System boot and initialisation
  - Partition »cell« management
  - Control and monitoring
  - Deferred VMM activation
Jailhouse, an *Exohypervisor*

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition »cell« management
  - Control and monitoring
  - Deferred VMM activation

1. Boot phase
Jailhouse, an Exohypervisor

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition `cell` management
  - Control and monitoring
  - Deferred VMM activation

Linux (root cell)

Jailhouse

<table>
<thead>
<tr>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device A</td>
<td>Device B</td>
<td>Device C</td>
<td>Device D</td>
</tr>
</tbody>
</table>

1. Boot phase
Jailhouse, an Exohypervisor

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition »cell« management
  - Control and monitoring
  - Deferred VMM activation

2. Partitioning phase
Jailhouse, an Exohypervisor

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition »cell« management
  - Control and monitoring
  - Deferred VMM activation

2. Partitioning phase

<table>
<thead>
<tr>
<th>Linux (root cell)</th>
<th>non-root cell</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Jailhouse</strong></td>
<td></td>
</tr>
<tr>
<td>Core 1</td>
<td>Core 2</td>
</tr>
<tr>
<td>Core 3</td>
<td>Core 4</td>
</tr>
<tr>
<td>Device A</td>
<td>Device B</td>
</tr>
<tr>
<td>Device C</td>
<td>Device D</td>
</tr>
</tbody>
</table>
Jailhouse, an Exohypervisor

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition «cell» management
  - Control and monitoring
  - Deferred VMM activation

3. Operational phase
Jailhouse, an Exohypervisor

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition »cell« management
  - Control and monitoring
  - Deferred VMM activation

Linux (root cell)

Jailhouse

<table>
<thead>
<tr>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device A</td>
<td>Device B</td>
<td>Device C</td>
<td>Device D</td>
</tr>
</tbody>
</table>

*(Dev: Destroy cells)*
Jailhouse, an *Exohypervisor*

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition »cell« management
  - Control and monitoring
  - Deferred VMM activation

(Dev: Disable hypervisor)
Jailhouse, an *Exohypervisor*

- Minimalist hypervisor skeleton (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition »cell« management
  - Control and monitoring
  - Deferred VMM activation

---

1. **Boot phase**

```
Core 1 | Core 2 | Core 3 | Core 4
------|-------|-------|-------
| Hardware
```

Linux
3. Jailhouse

What makes Jailhouse different?

**Jailhouse, an *Exohypervisor***

- Minimalist hypervisor skeleton
  (cf. Exokernel approach [5])
- **Offload uncritical work to Linux**
  - System boot and initialisation
  - Partition «cell» management
  - Control and monitoring
  - Deferred VMM activation
- **Prefer simplicity over features**
  - Partition booted system
    *instead of booting Linux*
  - Resource access control
    *instead of resource virtualisation*
  - 1:1 static resource assignment
    *instead of scheduling*
Small code-base and tiny impact

- \( \approx 7 \text{kLoC on armv7} \)
  - Simplifies certification efforts
  - Suitable basis for formal verification
- Try to hide (reduce traps), but don’t hide existence
- \#Partitions \( \leq \#\text{CPUs} \), sufficient for many real-world use cases
- \( \Rightarrow \) Maintain real-time capabilities \textit{by design}
Example use cases
No VM Exits! (x86)
3. Jailhouse

Use cases

- Jailhouse Hypervisor
- Core 0
- Core 1
- Core 2
- Core 3
- Devices
3. Jailhouse

Use cases
3. Jailhouse

Use cases
Architectural Challenges

- **Indivisible hardware resources**
  - DMA controllers, Clock and reset controllers
  - No device semantics in the hypervisor for paravirtualisation!

- **Platform dependent limitations**
  - ARM: Interrupt reinjection
    - Jetson TK1 (GICv2): ≈ 800 ns overhead
    - x86: intremap support
    - ARM: upcoming GICv4
  - MMIO device alignment
    - *Subpaging* (trap and dispatch access)
  - Erroneous Hardware behaviour
Architectural Challenges

- **Indivisible hardware resources**
  - DMA controllers, Clock and reset controllers
  - No device semantics in the hypervisor for paravirtualisation!

- **Platform dependent limitations**
  - ARM: Interrupt reinjection
    - Jetson TK1 (GICv2): ≈ 800 ns overhead
    - x86: intremap support
    - ARM: upcoming GICv4
  - MMIO device alignment
    - *Subpaging* (trap and dispatch access)
  - Erroneous Hardware behaviour

---

Look Mum, No VM Exits! (Almost)

June 27, 2017
Architectural Challenges

- Indivisible hardware resources
  - DMA controllers, Clock and reset controllers
  - No device semantics in the hypervisor for paravirtualisation!
- Platform dependent limitations
  - ARM: Interrupt reinjection
    - Jetson TK1 (GICv2): \( \approx 800 \text{ ns overhead} \)
    - x86: intremap support
    - ARM: upcoming GICv4
  - MMIO device alignment
    - Sub paging (trap and dispatch access)
  - Erroneous Hardware behaviour
Architectural Challenges

- Indivisible hardware resources
  - DMA controllers, Clock and reset controllers
  - No device semantics in the hypervisor for paravirtualisation!

- Platform dependent limitations
  - ARM: Interrupt reinjection
    - Jetson TK1 (GICv2): ≈ 800 ns overhead
  - Intel: intremap support
  - ARM: upcoming GICv4
  - MMIO device alignment
    - Subpaging (trap and dispatch access)
  - Erroneous Hardware behaviour
Architectural Challenges

- Indivisible hardware resources
  - DMA controllers, Clock and reset controllers
  - No device semantics in the hypervisor for paravirtualisation!

- Platform dependent limitations
  - ARM: Interrupt reinjection
    - Jetson TK1 (GICv2): \( \approx 800 \text{ ns overhead} \)
    - x86: intremap support
    - ARM: upcoming GICv4
  - MMIO device alignment
    - Subpaging (trap and dispatch access)
  - Erroneous Hardware behaviour
Architectural Challenges

- Indivisible hardware resources ❌
  - DMA controllers, Clock and reset controllers
  - No device semantics in the hypervisor for paravirtualisation!

- Platform dependent limitations ❌
  - ARM: Interrupt reinjection
    - Jetson TK1 (GICv2): ≈ 800 ns overhead
    - x86: intremap support
    - ARM: upcoming GICv4
  - MMIO device alignment
    - Subpaging (trap and dispatch access)
  - Erroneous Hardware behaviour

Jetson TK1: Ext. Stimuli + Response (bare-metal)
Architectural Challenges

- Indivisible hardware resources
  - DMA controllers, Clock and reset controllers
  - No device semantics in the hypervisor for paravirtualisation!
- Platform dependent limitations
  - ARM: Interrupt reinjection
    - Jetson TK1 (GICv2): \( \approx 800 \text{ ns overhead} \)
    - x86: intremap support
  - ARM: upcoming GICv4
- MMIO device alignment
  - Subpaging (trap and dispatch access)
- Erroneous Hardware behaviour

Jetson TK1: Ext. Stimuli + Response (Jailhouse)
Architectural Challenges

- **Indivisible hardware resources**
  - DMA controllers, Clock and reset controllers
  - No device semantics in the hypervisor for paravirtualisation!

- **Platform dependent limitations**
  - ARM: Interrupt reinjection
    - Jetson TK1 (GICv2): \(\approx 800\) ns overhead
    - x86: intremap support
    - ARM: upcoming GICv4
  - MMIO device alignment
    - *Subpaging* (trap and dispatch access)

- **Erroneous Hardware behaviour**
Architectural Challenges

- Indivisible hardware resources ✗
  - DMA controllers, Clock and reset controllers ⬤
  - No device semantics in the hypervisor for paravirtualisation!
- Platform dependent limitations ✗
  - ARM: Interrupt reinjection ✔
    - Jetson TK1 (GICv2): \( \approx 800 \text{ ns overhead} \)
  - x86: intremap support
  - ARM: upcoming GICv4
- MMIO device alignment ⬤
  - Subpaging (trap and dispatch access) ⬤
- Erroneous Hardware behaviour ⬤

No Jailhouse specific problems

Shared for all static hardware partitioning approaches!
QEMU, KVM, Xen, …
Look Mum, No VM Exits!
Look Mum, No VM Exits! Almost.
Burn-in test

- Typical mixed-criticality scenario
- Legacy software stack
- Jailhouse support out of the box
- Only **board support**
- Linux/RTOS as common use case
  - critical: flight control (hardware and software)
  - uncritical: computer vision task, video streaming
Burn-in test

- Typical mixed-criticality scenario
- Legacy software stack
- Jailhouse support out of the box
- Only **board support**
- Linux/RTOS as common use case
  - critical: flight control (hardware and software)
  - uncritical: computer vision task, video streaming

Jailhouse

- GPIO
- SPI-0
- I2C-0
- CPU 0
- I2C-1
- CPU 1
- SPI-1
- CPU 2
- CPU 3
Conclusion

- Solid testament for implementing real-time safety critical systems with Jailhouse
- Jailhouse as platform or playground for other academic approaches
- Hardware/Software codesign towards zero traps

Future Work

- Sound quantification of hypervisor influence (there are certain traps)
- Safety certification (Ongoing!)
- Linux mainline integration (Upcoming!)
- Consider extending Jailhouse for heterogeneous architectures
  - +GPU
  - +FPGA
  - +PRU
Conclusion

▶ Solid testament for implementing real-time safety critical systems with Jailhouse
▶ Jailhouse as platform or playground for other academic approaches
▶ Hardware/Software codesign towards zero traps

Future Work

▶ Sound quantification of hypervisor influence (there are certain traps)
▶ Safety certification (Ongoing!)
▶ Linux mainline integration (Upcoming!)
▶ Consider extending jailhouse for heterogeneous architectures
  ▶ +GPU
  ▶ +FPGA
  ▶ +PRU
Thank you!

https://github.com/siemens/jailhouse
<jailhouse-dev@googlegroups.com>

<ralf.ramsauer@othr.de>, <jan.kiszka@siemens.com>


BACKUP
cyclic timer interrupt, measure jitter

- $ modprobe jailhouse
- $ jailhouse enable tk1.cell
- $ jailhouse cell create tk1-demo.cell
- $ jailhouse cell load tk1-demo gic-demo.bin
- $ jailhouse cell start tk1-demo
raw inmate: timed event loop (GIC demo)

Initializing Jailhouse hypervisor v0.7 (26-g918bec06) on CPU 1
Code location: 0xf0000040
Initializing processors:
  CPU 1... OK
  CPU 2... OK
  CPU 0... OK
  CPU 3... OK
Activating hypervisor
Created cell "jetson-tk1-demo"
Page pool usage after cell creation: mem 82/16107, remap 69/131072
Cell "jetson-tk1-demo" can be loaded
Started cell "jetson-tk1-demo"
Initializing the GIC...
Initializing the timer...
  Timer fired, jitter: 3083 ns, min: 3083 ns, max: 3083 ns
  Timer fired, jitter: 2333 ns, min: 2333 ns, max: 3083 ns
  Timer fired, jitter: 2416 ns, min: 2333 ns, max: 3083 ns
  Timer fired, jitter: 3916 ns, min: 2333 ns, max: 3916 ns
  Timer fired, jitter: 3749 ns, min: 2333 ns, max: 3916 ns
  Timer fired, jitter: 3499 ns, min: 2333 ns, max: 3916 ns
  [...]

Ralf Ramsauer
raw inmate: timed event loop (GIC demo)

- $ jailhouse cell destroy tk1-demo
- $ jailhouse disable

[...]

Timer fired, jitter: 3416 ns, min: 2166 ns, max: 3916
Timer fired, jitter: 3499 ns, min: 2166 ns, max: 3916
Timer fired, jitter: 3499 ns, min: 2166 ns, max: 3916

Closing cell "jetson-tk1-demo"

Shutting down hypervisor
  Releasing CPU 2
  Releasing CPU 0
  Releasing CPU 1
  Releasing CPU 3