Building Mixed Criticality Linux Systems with the Jailhouse Hypervisor

Ralf Ramsauer¹, Jan Kiszka², Wolfgang Mauerer¹,²

¹ Digitalisation Laboratory, Technical University of Applied Sciences Regensburg (OTH)
² Siemens AG, Corporate Technology
1. Jailhouse introduction & current status

- Jailhouse introduction & current status
- Mixed Criticality Systems with Jailhouse
- Jailhouse Performance
- Requirements on Partitioning Hardware
- Conclusion
Motivation

- SMP is everywhere
  - Enables consolidation of formerly separate devices
- Linux is almost everywhere, but
  - Legacy software stacks require bare-metal
  - Safety-critical software stacks
  - DSP-like real-time workloads
Jailhouse Architecture

- Build static partitions on SMP systems
- Use hardware-assisted virtualization
- Do not schedule
  - No CPU core sharing
  - 1:1 device assignment
- Split up running Linux system
- Simplicity over Features

1. Boot phase
2. Partitioning phase
3. Operational phase
Version 0.6 released in January

- Merged ARMv8 support
- Reworked ARMv7
- Shared memory device, enables virtual networks
- Support for multiple Linux instances (cells)
- Support for Intel Cache Allocation Technology
- AMD IOMMU support
- Many new boards

Nvidia Jetson TX1, ZynqMP ZC102, Xunlong Orange Pi Zero

Images © Nvidia, Zynq, Xunlong
Upcoming Developments

- Enhanced shared memory device
  - Unidirectional channels
    (supports safety scenarios)
  - Performance improvements

- Jailhouse is (likely) participating in GSoC

- Safety certification of Jailhouse
  - Code-wise feasible
  - Heavily depends on hardware support
  - Stay tuned!
2. Mixed Criticality Systems with Jailhouse

Agenda

• Jailhouse introduction & current status
• Mixed Criticality Systems with Jailhouse
• Jailhouse Performance
• Requirements on Partitioning Hardware
• Conclusion
Motivation

Mixed Criticality Systems

- Systems executing **critical** and **uncritical** payloads
- **Currently:** separate physical systems
- **Future:** multi-core systems enable consolidation to **single hardware units**
Motivation

Mixed Criticality Systems

- Systems executing critical and uncritical payloads
- Currently: separate physical systems
- Future: multi-core systems enable consolidation to single hardware units

Demonstration Platform

- typical real-time / safety environment
- Reliability, robustness, ...
- Port existing critical payload
- Jailhouse-Multicopter (JAPTER)
2. Mixed Criticality Systems with Jailhouse

Classic Approach: Separation of Systems
Classic Approach: Separation of Systems
2. Mixed Criticality Systems with Jailhouse

Classic Approach: Separation of Systems

- **Critical**
  - Drones
  - ArduPilot

- **Uncritical**
  - Camera
  - OpenCV
  - Linux

R. Ramsauer, J. Kiszka

Mixed Criticality Systems with Jailhouse

February 21, 2017
Classic Approach: Separation of Systems

Critical

Uncritical

Communication

R. Ramsauer, J. Kiszka

Mixed Criticality Systems with Jailhouse

February 21, 2017
Jailhouse: From Separation to Isolated Consolidation

Mixed Criticality Approach

- ARDUPILOT
- OpenCV
- Jailhouse

SIEMENS

R. Ramsauer, J. Kiszka

Mixed Criticality Systems with Jailhouse

February 21, 2017
Architectural Decisions

Octa-Frame

Mikrokopter OktoXL

Image © HiSystems GmbH
2. Mixed Criticality Systems with Jailhouse

Architectural Decisions

Octa-Frame

Mikrokopter OktoXL
Image © HiSystems GmbH

Control Unit

Emlid Navio2
Image © Emlid Ltd.

Nvidia Jetson TK1
Image © Nvidia
Core of the System

Nvidia Jetson TK1

- Quad-Core ARMv7 A15 SoC (@2.32GHz)
- 2GiB main memory
- Feature-rich expansion headers (SPI, I^2C, UART, GPIOs, ...)
- ARM-VE: boot in HYP-mode
- ⇒ Jailhouse Enabled
- Mainline Linux support (4.10-rc6)
Octocopter Platform Overview

Critical Hardware Devices:
- **I²C**: Motors, Barometer, RC-Decoder
- **SPI**: Gyroscope(s), Accelerometer(s), GPS, Compass(es)
- **GPIO**: Status LEDs

Critical Software:
- **Flight Stack**: Ardupilot
- **Linux** with **RT patch** (4.9.6-rt4)
- Jailhouse Hypervisor

Jailhouse Diagram:
- GPIO
- SPI-0
- I2C-0
- I2C-1
- SPI-1
- CPU 0
- CPU 1
- CPU 2
- CPU 3
Engineering Approach

- Develop and test safety critical application on bare-metal without Jailhouse
  - Kernel: Apply Preempt_RT patch, modify Tegra device drivers, ...
  - Ardupilot: Implement motor driver, battery sensing, ...
- Enable Jailhouse: move critical devices and software to isolated cell
- Add uncritical payload
Linux as Jailhouse guest

- Jailhouse supports unmodified mainline Linux as Jailhouse guest on ARM
- Preempt_RT patched kernel
- Tiny, tailored device-trees
- Userland as initrd in memory
- IVHSMEM inter-cell network driver

*Credits go to Måns Rullgård*
Devices

- Jailhouse remaps memory
  - No interception if PAGE_SIZE-aligned
  - Otherwise dispatch access
- Jailhouse reinjects interrupts
  - Jailhouse receives interrupts and reinjects them to guests
  - Minimum overhead
  - (No overhead on x86 with intremap)

Original Jetson TK1 DT:

```c
i2c@7000c000 {
    compatible = "nvidia,tegra114-i2c";
    reg = <0x0 0x7000c000 0x0 0x100>;
    interrupts = <GIC_SPI 38 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&tegra_car TEGRA124_CLK_I2C1>;
    clock-names = "div-clk";
    resets = <&tegra_car 12>;
    reset-names = "i2c";
    dmas = <&apbdma 21>, <&apbdma 21>;
    dma-names = "rx", "tx";
};
```
Devices

- Jailhouse remaps memory
  - No interception if PAGE_SIZE-aligned
  - Otherwise dispatch access
- Jailhouse reinjects interrupts
  - Jailhouse receives interrupts and reinjects them to guests
  - Minimum overhead
  - (No overhead on x86 with intremap)

Original Jetson TK1 DT:

```c
i2c@7000c000 {
    compatible = "nvidia,tegra114-i2c";
    reg = <0x0 0x7000c000 0x0 0x100>;
    interrupts = <GIC_SPI 38 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&tegra_car TEGRA124_CLK_I2C1>;
    clock-names = "div-clk";
    resets = <&tegra_car 12>;
    reset-names = "i2c";
    dmas = <&apbdma 21>, <&apbdma 21>;
    dma-names = "rx", "tx";
};
```
2. Mixed Criticality Systems with Jailhouse

Original Jetson TK1 DT:

```c
i2c@7000c000 {
    compatible = "nvidia,tegra114-i2c";
    reg = <0x0 0x7000c000 0x0 0x100>;
    interrupts = <GIC_SPI 38 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&tegra_car TEGRA124_CLK_I2C1>;
    clock-names = "div-clk";
    resets = <&tegra_car 12>;
    reset-names = "i2c";
    dmas = <&apbdma 21>, <&apbdma 21>;
    dma-names = "rx", "tx";
};
```
Devices

- Jailhouse remaps memory
  - No interception if PAGE_SIZE-aligned
  - Otherwise dispatch access
- Jailhouse reinjects interrupts
  - Jailhouse receives interrupts and reinjects them to guests
  - Minimum overhead
  - (No overhead on x86 with intremap)

Original Jetson TK1 DT:

```c
i2c@7000c000 {
  compatible = "nvidia,tegra114-i2c";
  reg = <0x0 0x7000c000 0x0 0x100>;
  interrupts = <GIC_SPI 38 IRQ_TYPE_LEVEL_HIGH>;
  clocks = <&tegra_car TEGRA124_CLK_I2C1>;
  clock-names = "div-clk";
  resets = <&tegra_car 12>;
  reset-names = "i2c";
  dmas = <&apbdma 21>, <&apbdma 21>;
  dma-names = "rx", "tx";
};
```

R. Ramsauer, J. Kiszka
Mixed Criticality Systems with Jailhouse
February 21, 2017
Clocks and Resets

- **Clocks**
  - (Peripheral) devices are driven by clocks
  - Ungating idling devices saves power
  - Select different speeds or baudrates

- **Resets**
  - (De)asserts reset-lines of devices
  - Reset to initial state

**Driver usage:**

```c
clk_enable(dev->clk);

[...]
reset_control_assert(dev->rst);
udelay(2);
reset_control_deassert(dev->rst);

[...]
clk_disable(dev->clk),
```
Clocks and Resets

- **Clocks**
  - (Peripheral) devices are driven by clocks
  - Ungating idling devices saves power
  - Select different speeds or baudrates

- **Resets**
  - (De)asserts reset-lines of devices
  - Reset to initial state

- **Organized as Clock & Reset controllers**
  - Contiguous MMIO region
  - Controls **whole** platform
  - Hard to partition

Device tree definition:

```c
tegra_car: clock@60006000 {
  compatible = "nvidia,tegra124-car";
  reg = <0x0 0x60006000 0x0 0x1000>;
  #clock-cells = <1>;
  #reset-cells = <1>;
};
```
Clocks and Resets

- Jailhouse Context
  - Real-time and no low power consumption requirements
  - No dynamic change of baudrate or speed
  - *Idea: Enable Clocks and Deassert resets before starting guest*
Clocks and Resets

- Jailhouse Context
  - Real-time and no low power consumption requirements
  - No dynamic change of baudrate or speed
  - Idea: Enable Clocks and Deassert resets before starting guest 🚫
  - Don’t ignore clocks!!

drivers/i2c/busses/i2c-tegra.c:

div_clk = devm_clk_get(&pdev->dev, "div-clk");
if (IS_ERR(div_clk)) {
    dev_err(&pdev->dev, "missing controller clock\n");
    return PTR_ERR(div_clk);
}

R. Ramsauer, J. Kiszka

Mixed Criticality Systems with Jailhouse

February 21, 2017
Clocks and Resets

- Jailhouse Context
  - Real-time and no low power consumption requirements
  - No dynamic change of baudrate or speed
  - *Idea: Enable Clocks and Deassert resets before starting guest* 🚧
  - *Don’t ignore clocks!!*

```c
drivers/i2c/busses/i2c-tegra.c:

i2c_dev->rst =
    devm_reset_control_get(&pdev->dev, "i2c");
if (IS_ERR(i2c_dev->rst)) {
    dev_err(&pdev->dev, "missing controller reset");
    return PTR_ERR(i2c_dev->rst);
}
```
Paravirtualise C&R

- Minimalistic paravirtual Clock and Reset controller
- Root-Linux: Trap on MMIO access, and dispatch access on a bit-granular level (slow)
- **Future: Use hypercalls (faster)**
- Access bitmaps must be created manually

Jailhouse Clock and Reset Controller:

```c
jailhouse_car: clock@60006000 {
    compatible = "jailhouse,jailhouse-car";
    reg = <0x60006000 0x1000>;
};

i2c@7000c000 {
    compatible = "nvidia,tegra114-i2c";
    reg = <0x7000c000 0x100>;
    interrupts = <GIC_SPI 38 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&jailhouse_car 0>;
    clock-names = "div-clk";
    resets = <&jailhouse_car 0>;
    reset-names = "i2c";
    dmas = <&apbdma 21>, <&apbdma 21>;
    dma-names = "rx", "tx";
};
```
Paravirtualise C&R

- Minimalistic paravirtual Clock and Reset controller
- Root-Linux: Trap on MMIO access, and dispatch access on a bit-granular level (slow)
- Future: Use hypercalls (faster)
- Access bitmaps must be created manually

Jailhouse Clock and Reset Controller:

```c
jailhouse_car: clock@600006000 {
    compatible = "jailhouse,jailhouse-car";
    reg = <0x600006000 0x1000>;
}

i2c@7000c000 {
    compatible = "nvidia,tegra114-i2c";
    reg = <0x7000c000 0x100>;
    interrupts = <GIC_SPI 38 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&jailhouse_car 0>;
    clock-names = "div-clk";
    resets = <&jailhouse_car 0>;
    reset-names = "i2c";
    dmas = <&apbdma 21>, <&apbdma 21>;
    dma-names = "rx", "tx";
}
```

R. Ramsauer, J. Kiszka  Mixed Criticality Systems with Jailhouse  February 21, 2017  19 / 32
DMA controllers

▶ Jailhouse Context
   ▶ Low latency matters more than high throughput
   ▶ Disable DMAed access

drivers/spi/spi-tegra114.c:

dma_chan = dma_request_slave_channel_reason(tspi->dev,
   dma_to_memory ? "rx" : "tx");

if (IS_ERR(dma_chan)) {
   ret = PTR_ERR(dma_chan);
   if (ret != -EPROBE_DEFER)
      dev_err(tspi->dev,
           "Dma/uni2423channel/uni2423is/uni2423not/uni2423available:/uni2423%d\n", ret);

   return ret;
}
2. Mixed Criticality Systems with Jailhouse

DMA controllers

- Jailhouse Context
  - Low latency matters more than high throughput
  - *Disable DMAed access*

- Assign DMA device exclusively to critical cell

```c
text
```
```c
i2c@7000c000 {
    compatible = "nvidia,tegra114-i2c";
    reg = <0x7000c000 0x100>;
    interrupts = <GIC_SPI 38 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&jailhouse_car 0>;
    clock-names = "div-clk";
    resets = <&jailhouse_car 0>;
    reset-names = "i2c";
    dmas = <&apbdma 21>, <&apbdma 21>;
    dma-names = "rx", "tx";
};
```
Now we have...

- Real-Time OS running inside Jailhouse
- MMIO-based devices assigned to critical cell
- Virtual Clocks
- Jailhouse-independent environment
- Execute legacy payload application in critical cell
- Uncritical cell under load
- Ready to fly!
3. Jailhouse Performance

Agenda

- Jailhouse introduction & current status
- Mixed Criticality Systems with Jailhouse
- Jailhouse Performance
- Requirements on Partitioning Hardware
- Conclusion
Interrupt Reinjection

- Externally toggle GPIO, wait for response, measure delay.
- Measures platform-dependent minimal IRQ answer time
  - Bare-Metal latency
  - Jailhouse latency
  - Linux latency
  - Preempt_RT latency
Interrupt Reinjection

- Externally toggle GPIO, wait for response, measure delay.
- Measures platform-dependent minimal IRQ answer time
  - Bare-Metal latency
  - Jailhouse latency
  - Linux latency
  - Preempt_RT latency

---

Measurement | VMM | Avg | Max | Load
--- | --- | --- | --- | ---
bare-metal | off | 0.44 | 0.50 | off

in µs, measured at 50Hz, duration: 4h
Interrupt Reinjection

- Externally toggle GPIO, wait for response, measure delay.
- Measures platform-dependent minimal IRQ answer time
  - Bare-Metal latency
  - Jailhouse latency
  - Linux latency
  - Preempt_RT latency
- IRQ dispatch overhead: \( \approx 800\text{ns} \)

<table>
<thead>
<tr>
<th>Measurement</th>
<th>VMM</th>
<th>Avg</th>
<th>Max</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>bare-metal</td>
<td>off</td>
<td>0.44</td>
<td>0.50</td>
<td>off</td>
</tr>
<tr>
<td>bare-metal</td>
<td>on</td>
<td>1.20</td>
<td>2.88</td>
<td>off</td>
</tr>
</tbody>
</table>

in µs, measured at 50Hz, duration: 4h
Interrupt Reinjection

- Externally toggle GPIO, wait for response, measure delay.
- Measures platform-dependent minimal IRQ answer time
  - Bare-Metal latency
  - Jailhouse latency
  - Linux latency
  - Preempt_RT latency
- IRQ dispatch overhead: \( \approx 800\text{ns} \)

<table>
<thead>
<tr>
<th>Measurement</th>
<th>VMM</th>
<th>Avg</th>
<th>Max</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>bare-metal</td>
<td>off</td>
<td>0.44</td>
<td>0.50</td>
<td>off</td>
</tr>
<tr>
<td>bare-metal</td>
<td>on</td>
<td>1.20</td>
<td>2.88</td>
<td>off</td>
</tr>
<tr>
<td>bare-metal</td>
<td>on</td>
<td>1.37</td>
<td>7.49</td>
<td>on</td>
</tr>
</tbody>
</table>

in \( \mu s \), measured at 50Hz, duration: 4h
Subpage dispatching

- Subpage dispatch memory access delay
- Measure Single memory access
- Count CPU Cycles (PMCNTR) between accesses

```c
static inline unsigned int ccnt_read(void)
{
    unsigned int value;
    asm volatile ("mrc p15,0,\%0,\%c9,\%c13,0\t\n" : ":r"(value));
    return value;
}

gic_disable_interrupts();
for(;;) {
    ccnt_reset();
    start = ccnt_read();
    *address = 0xdeadbeef;
    end = ccnt_read();
    uart_printf("%d\n", end - start);
    delay();
}
```
### Subpage dispatching

- Subpage dispatch memory access delay
- Measure Single memory access
- Count CPU Cycles (PMCNTR) between accesses

<table>
<thead>
<tr>
<th>Subpaging</th>
<th>VMM</th>
<th>Avg</th>
<th>Max</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>on/off</td>
<td>8</td>
<td>8</td>
<td>off</td>
</tr>
</tbody>
</table>

in CPU cycles
3. Jailhouse Performance

Subpage dispatching

- Subpage dispatch memory access delay
- Measure Single memory access
- Count CPU Cycles (PMCNTR) between accesses

<table>
<thead>
<tr>
<th>Subpaging</th>
<th>VMM</th>
<th>Avg</th>
<th>Max</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>on/off</td>
<td>8</td>
<td>8</td>
<td>off</td>
</tr>
<tr>
<td>no</td>
<td>on</td>
<td>8</td>
<td>217</td>
<td>on</td>
</tr>
</tbody>
</table>

in CPU cycles
### Subpage dispatching

- Subpage dispatch memory access delay
- Measure Single memory access
- Count CPU Cycles (PMCNTR) between accesses

<table>
<thead>
<tr>
<th>Subpaging</th>
<th>VMM</th>
<th>Avg</th>
<th>Max</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>on/off</td>
<td>8</td>
<td>8</td>
<td>off</td>
</tr>
<tr>
<td>no</td>
<td>on</td>
<td>8</td>
<td>217</td>
<td>on</td>
</tr>
<tr>
<td>yes</td>
<td>on</td>
<td>436</td>
<td>980</td>
<td>off</td>
</tr>
</tbody>
</table>

in CPU cycles
### Subpage dispatching

- Subpage dispatch memory access delay
- Measure Single memory access
- Count CPU Cycles (PMCNTR) between accesses

### Subpaging VMM Avg Max Load

<table>
<thead>
<tr>
<th>Subpaging</th>
<th>VMM</th>
<th>Avg</th>
<th>Max</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>on/off</td>
<td>8</td>
<td>8</td>
<td>off</td>
</tr>
<tr>
<td>no</td>
<td>on</td>
<td>8</td>
<td>217</td>
<td>on</td>
</tr>
<tr>
<td>yes</td>
<td>on</td>
<td>436</td>
<td>980</td>
<td>off</td>
</tr>
<tr>
<td>yes</td>
<td>on</td>
<td>529</td>
<td>3018</td>
<td>on</td>
</tr>
</tbody>
</table>

in CPU cycles

---

R. Ramsauer, J. Kiszka
Mixed Criticality Systems with Jailhouse
February 21, 2017
4. Requirements on Partitioning Hardware

- Jailhouse introduction & current status
- Mixed Criticality Systems with Jailhouse
- Jailhouse Performance
- Requirements on Partitioning Hardware
- Conclusion
Memory Mapped I/O

- PAGE_SIZE is finest paging granularity
- Multiple devices per page
- Multiple functionalities per page
- Subpaging leads to performance impacts
- 32 (or more) bits of physical address space
- 2GiB of Ram $\Rightarrow$ 52k devices$^a$

$^a@10$ pages per device

TK1’s /proc/iomem:

[...]
70006300-7000633f : serial
7000c000-7000c0ff : /i2c@7000c000
7000c400-7000c4ff : /i2c@7000c400
7000c500-7000c5ff : /i2c@7000c500
7000c700-7000c7ff : /i2c@7000c700
7000d000-7000d0ff : /i2c@7000d000
7000d400-7000d5ff : /spi@7000d400
7000da00-7000dbff : /spi@7000da00
[...]
Memory Mapped I/O

- PAGE_SIZE is finest paging granularity
- Multiple devices per page
- Multiple functionalities per page
- Subpaging leads to performance impacts
- 32 (or more) bits of physical address space
- 2GiB of Ram $\Rightarrow$ 52k devices$^a$

$^a$@ 10 pages per device

TK1’s /proc/iomem:

[...]  
70006300-7000633f : serial  
7000c000-7000c0ff : /i2c@7000c000  
7000c400-7000c4ff : /i2c@7000c400  
7000c500-7000c5ff : /i2c@7000c500  
7000c700-7000c7ff : /i2c@7000c700  
7000d000-7000d0ff : /i2c@7000d000  
7000d400-7000d5ff : /spi@7000d400  
7000da00-7000dbff : /spi@7000da00  
[...]
Memory Mapped I/O

- PAGE_SIZE is finest paging granularity
- Multiple devices per page
- Multiple functionalities per page
- Subpaging leads to performance impacts
- 32 (or more) bits of physical address space
- 2GiB of Ram ⇒ 52k devices\(^a\)

\(^a\)@ 10 pages per device

TK1’s /proc/iomem:

 [...]  
7000d000-7000d0ff : /i2c@7000d000  
7000d400-7000d5ff : /spi@7000d400  
7000da00-7000dbff : /spi@7000da00  
 [...]
Clock and Reset controllers

- Stick Reset, Clocks and Divider to device memory, if possible
- Make Clock and Reset Controller paritionable
- Otherwise we need (para-)virtualisation
Direct Memory Access

- Latency matters more than Throughput
- Allow absence of DMA channels
- Make DMA controllers partitionable
- Otherwise we need (para-)virtualisation
Erroneous Hardware Behaviour

- Hardware misbehaves
- Tegra Architecture: System freeze on touching ungated device memory
- Errata force to trap

On 12/01/2016 07:15 AM, Ralf Ramsauer wrote:
> Hi,
> I observed that touching MMIO regions of Tegra devices with its corresponding clock gate deactivated immediately freezes the whole system. No kernel panic, nothing. [...]

[Stephen Warren (Nvidia):]
Unfortunately, this is indeed the way the HW works.

[Mikko Perttunen (Nvidia):]
It does apply to the whole architecture for all revisions pre-Tegra186 (which is newer than Tegra210 despite the number).
5. Conclusion

Agenda

- Jailhouse introduction & current status
- Mixed Criticality Systems with Jailhouse
- Jailhouse Performance
- Requirements on Partitioning Hardware
- Conclusion
Conclusion

- Solid testament for implementing real-time safety critical systems with Jailhouse
- Software based workarounds lead to latency and performance impacts
- Hardware design aspects with focus on mixed criticality systems
- Hardware-Software Co-Design
Thank you!

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

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