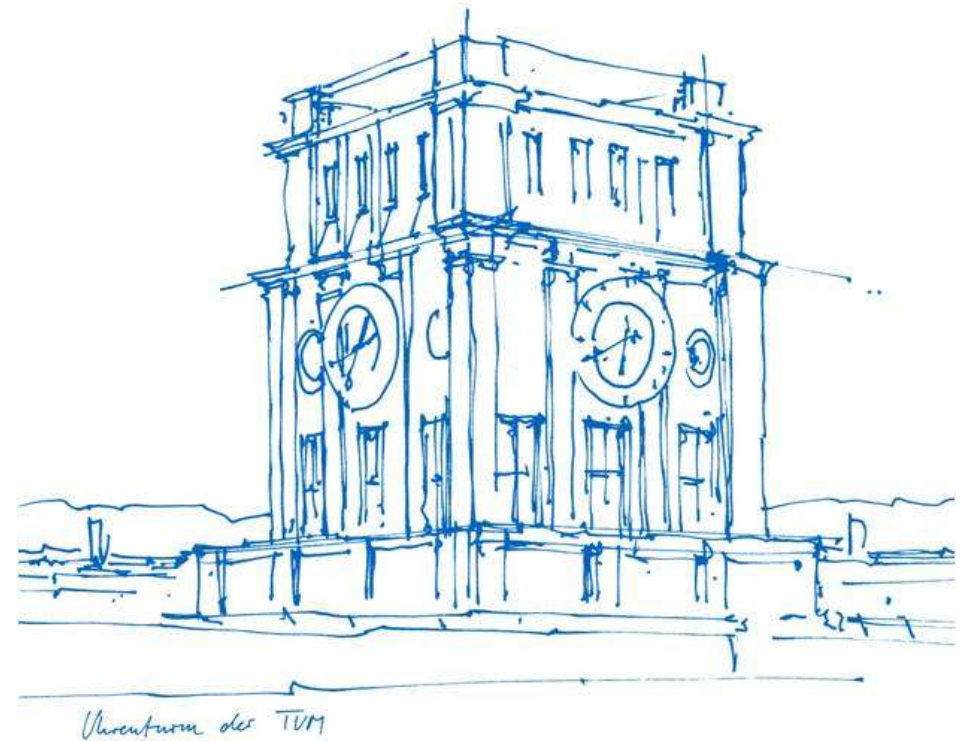


# Jailhouse Porting and Debug

Advisor            Dr. Andrea Bastoni  
Co-advisor        Dr. Alexander Züpke  
Presented by      Haozheng Huang  
Munich, 26. June 2024

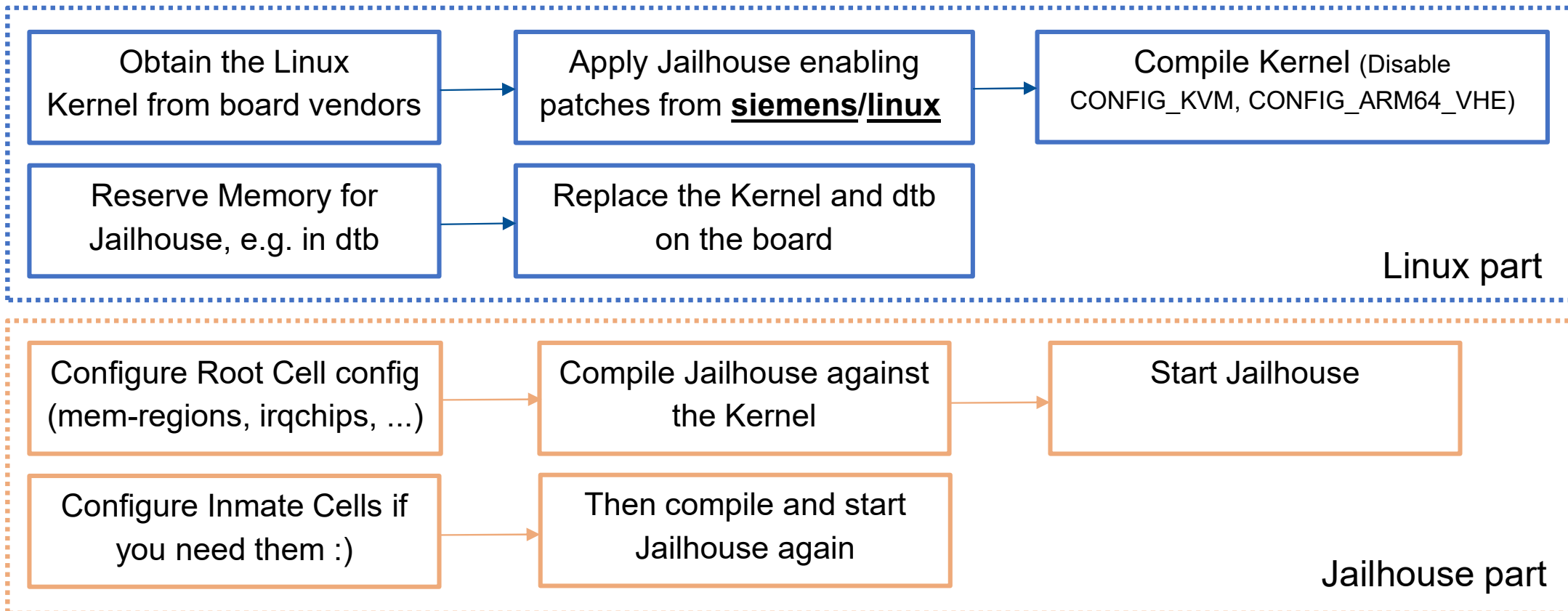


# Jailhouse Porting

# Example: Porting Jailhouse to RK3588

- Patches to the Linux Kernel 5.10
  1. Jailhouse enabling patches on ARM64
  2. Disable CONFIG\_KVM
  3. Disable CONFIG\_ARM64\_VHE
  4. Disable CONFIG\_ARM\_GIC\_V3 ITS
- Patches to the Jailhouse
  1. Patch to allow SMC calls to reach the firmware
  2. Skip FPEXC32\_EL2 (It does not exist in A76 core!)
  3. Skip Software Delegated Exception Interface (SDEI)
- Memory reservation and address mapping

# General Procedure to Port Jailhouse



## Obtain the Linux Kernel from board vendors

Vendors usually have applied some patches to adapt the mainline Linux Kernel to the board.  
These can be found according to the guide from vendors.

## Apply Jailhouse enabling patches from siemens/linux

These patches enable Jailhouse to boot.

They are provided by siemens/linux, with different Kernel versions on different branches.

In the Linux Kernel 5.10 on the RK3588, these commits are applied by `git cherry-pick` onto the Kernel provided by the vendor.

File	Edit	View	Help
<ul style="list-style-type: none"><li>selinux: fix error initialization in inode_doinit_with_dentry()</li><li>RDMA/bnxt_re: Fix entry size during SRQ create</li><li>rtc: pcf2127: fix pcf2127_nvmm_read/write() returns</li><li>RDMA/bnxt_re: Set queue pair state when being queried</li><li>Revert "i2c: i2c-qcom-geni: Fix DMA transfer race"</li><li>soc: qcom: geni: More properly switch to DMA mode</li><li>arm64: dts: qcom: sc7180: Fix one forgotten interconnect reference</li><li>arm64: dts: ipq6018: update the reserved-memory node</li><li>arm64: dts: mediatek: mt8183: fix gce incorrect mbox-cells value</li><li>soc: mediatek: Check if power domains can be powered on at boot time</li><li>soc: renesas: rmobile-sysc: Fix some leaks in rmobile_init_pm_domains()</li><li>arm64: dts: renesas: cat875: Remove rxc-skew-ps from ethernet-phy node</li><li>arm64: dts: renesas: hihope-rzg2-ex: Drop rxc-skew-ps from ethernet-phy node</li><li>drm/tve200: Fix handling of platform_get_irq() error</li><li>drm/mcdc: Fix handling of platform_get_irq() error</li><li>drm/aspeed: Fix Kconfig warning &amp; subsequent build errors</li><li>iiio: adc: at91_adc: add Kconfig dep on the OF symbol and remove of_match_ptr()</li><li>drm/gma500: fix double free of gma_connector</li><li>hwmon: (k10temp) Remove support for displaying voltage and current on Zen CPUs</li><li>ivshmem-net: virtual network device for jailhouse</li><li>arm64: dts: marvell: armada-8030-mcbn: Set pci-domain</li><li>arm64: dts: marvell: armada-37xx: Set pci-domain</li><li>x86: Export lapic_timer_period</li><li>arm, arm64: export __hyp_stub_vectors</li><li>mm: vmalloc: Export __get_vm_area_caller</li><li>mm: Re-export ioremap_page_range</li><li>arm: Export __boot_cpu_mode for use in Jailhouse driver module</li><li>Revert "arm: Remove the ability to set HYP vectors outside of the decompressor"</li><li>jailhouse: Add simple debug console via the hypervisor</li><li>uiio: Add driver for inter-VM shared memory device</li><li>ivshmem: Add header file</li><li>uiio: Enable read-only mappings</li><li>Linux 5.10.3</li><li>md: fix a warning caused by a race between concurrent md_ioctl()s</li><li>nl80211: validate key indexes for cfg80211_registered_device</li><li>crypto: af_alg - avoid undefined behavior accessing salg_name</li><li>media: msi2500: assign SPI bus number dynamically</li><li>fs: quota: fix array-index-out-of-bounds bug by passing correct argument to vfs_cleanup_quota_inode()</li><li>quota: Sanitv-check quota file headers on load</li></ul>	<ul style="list-style-type: none"><li>Tianyue Ren &lt;rentianyue@kylinos.cn&gt;</li><li>Selvin Xavier &lt;selvin.xavier@broadcom.com&gt;</li><li>Dan Carpenter &lt;dan.carpenter@oracle.com&gt;</li><li>Kamal Heib &lt;kamalheib1@gmail.com&gt;</li><li>Douglas Anderson &lt;dlanders@chromium.org&gt;</li><li>Douglas Anderson &lt;dlanders@chromium.org&gt;</li><li>Douglas Anderson &lt;dlanders@chromium.org&gt;</li><li>Kathiravan T &lt;kathirav@codeaurora.org&gt;</li><li>Fabien Parent &lt;fparent@baylibre.com&gt;</li><li>Nicolas Boichat &lt;drinckat@chromium.org&gt;</li><li>Dan Carpenter &lt;dan.carpenter@oracle.com&gt;</li><li>Biju Das &lt;biju.das.jz@bp.renesas.com&gt;</li><li>Biju Das &lt;biju.das.jz@bp.renesas.com&gt;</li><li>Krzysztof Kozlowski &lt;krzk@kernel.org&gt;</li><li>Krzysztof Kozlowski &lt;krzk@kernel.org&gt;</li><li>Randy Dunlap &lt;rdunlap@infradead.org&gt;</li><li>Alexandru Ardelean &lt;alexandru.ardelean@analog.com&gt;</li><li>Tom Rix &lt;trix@redhat.com&gt;</li><li>Guenter Roeck &lt;linux@roeck-us.net&gt;</li><li>Mans Rullgard &lt;mans@mansr.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Ralf Ramsauer &lt;ralf.ramsauer@oth-regensburg.de&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Jan Kiszka &lt;jan.kiszka@siemens.com&gt;</li><li>Greg Kroah-Hartman &lt;gregkh@linuxfoundation.org&gt;</li><li>Dae R. Jeong &lt;dae.r.jeong@kaist.ac.kr&gt;</li><li>Anant Thazhemadam &lt;anant.thazhemadam@gmail.com&gt;</li><li>Eric Biggers &lt;ebiggers@google.com&gt;</li><li>Antti Palosaari &lt;crope@iki.fi&gt;</li><li>Anant Thazhemadam &lt;anant.thazhemadam@gmail.com&gt;</li><li>Ian Kara &lt;jack@suse.cz&gt;</li></ul>	<ul style="list-style-type: none"><li>2020-10-09 03:36:30</li><li>2020-10-13 08:15:52</li><li>2020-10-22 09:04:51</li><li>2020-10-21 13:49:52</li><li>2020-10-13 23:25:29</li><li>2020-10-13 23:25:28</li><li>2020-10-01 23:18:55</li><li>2020-10-14 17:46:17</li><li>2020-10-18 21:42:25</li><li>2020-09-28 05:31:35</li><li>2020-09-23 13:31:42</li><li>2020-10-15 15:23:50</li><li>2020-10-15 15:23:49</li><li>2020-08-27 09:11:07</li><li>2020-08-27 09:11:06</li><li>2020-10-12 01:01:31</li><li>2020-09-30 15:50:47</li><li>2020-10-03 21:39:28</li><li>2020-12-14 19:26:22</li><li>2016-05-26 17:04:02</li><li>2018-09-30 21:22:32</li><li>2018-09-17 08:08:08</li><li>2017-11-23 07:12:57</li><li>2017-06-07 15:48:43</li><li>2020-09-06 17:30:55</li><li>2017-02-07 17:52:00</li><li>2016-07-03 10:02:40</li><li>2020-12-27 15:04:31</li><li>2016-09-11 23:30:04</li><li>2019-06-04 18:40:25</li><li>2019-10-01 12:33:25</li><li>2019-06-04 14:40:09</li><li>2020-12-26 16:02:46</li><li>2020-10-22 03:21:28</li><li>2020-12-04 22:58:25</li><li>2020-10-26 21:07:15</li><li>2019-08-17 03:12:10</li><li>2020-12-08 20:43:38</li><li>2020-11-02 16:16:29</li></ul>	

Compile Kernel (Disable  
CONFIG\_KVM, CONFIG\_ARM64\_VHE)

Compile the patched Kernel with the following configs disabled:

- CONFIG\_KVM
- CONFIG\_ARM64\_VHE
- CONFIG\_ARM\_GIC\_V3\_ITS

Depending on the board, there may be other configs that need to be turned on/off.

## Reserve Memory for Jailhouse, e.g. in dtb

Reserve memory for Jailhouse Hypervisor  
and inmate cells in Device Tree (.dts file).

Then compile the Device Tree:

```
dtc -I dts -O dtb -o output.dtb input.dts
```

```
reserved-memory {  
    #address-cells = <0x02>;  
    #size-cells = <0x02>;  
    ranges;  
  
    reserved-mem-jailhouses@80000000 {  
        reg = <0x00 0x80000000 0x00  
0x10000000>;  
        no-map;  
    };  
  
    reserved-mem-inmate@90000000 {  
        reg = <0x00 0x90000000 0x00  
0x25000000>;  
        no-map;  
    };  
  
    reserved-mem-bomb-ctrl@b5000000 {  
        reg = <0x00 0xb5000000 0x00  
0x10000>;  
        no-map;  
    };  
}
```

Note: this is also addressed in Zhihang's Jailhouse Workshop slides.



Replace the Kernel and dtb  
on the board

On ARM64, the Linux Kernel is typically placed as ``/boot/Image``.

The dtb file is also stored under ``/boot/``.

## Configure Root Cell config (mem-regions, irqchips, ...)

We can refer to existing configs for other boards and adapt them for our board.

Most important configurations are:

- Jailhouse hypervisor address
- Debug console
- CPUs
- Memory regions
- Generic Interrupt Controller (GIC, Arm platforms)

## Compile Jailhouse against the Kernel

This is covered by Zhihang's Jailhouse Workshop slides :).

## Start Jailhouse

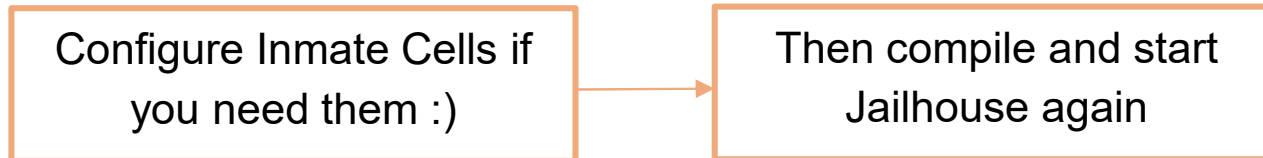
Starting the root cell and Linux inmate cell is covered by Zhihang's Jailhouse Workshop slides :).

Starting a bare-metal inmate cell:

```
jailhouse cell create inmate-config.cell
```

```
jailhouse cell load 1 demo.bin
```

```
jailhouse cell start 1
```



Like configuring the root cell, we can refer to existing configs for other boards and adapt them for our board.

Most important configurations are:

- RAM address reserved for the inmate (virtual address starts from the bottom of inmate memory, i.e. 0x0)
- Debug console
- Communication region: 

```
/* communication region */ {  
    .virt_start = 0x80000000,  
    .size = 0x00001000,  
    .flags = JAILHOUSE_MEM_READ | JAILHOUSE_MEM_WRITE |  
            JAILHOUSE_MEM_COMM_REGION,  
},
```

# Jailhouse Debug

Porting Jailhouse is not as straight-forward as described above.

There are always some errors/bugs on the way.

Debugging Jailhouse is somehow tricky.

There are some debug possibilities, still.

# Common Debug Possibilities

- Linux kernel log dmesg
- Hypervisor dump
  - Especially pc and esr

```
[ 1990.451647] rcu: INFO: rcu_sched self-detected stall on CPU
[ 1990.451662] rcu: 7-....: (125950 ticks this GP) idle=1a2/1/0x4000000000000002
softirq=3018/3018 fqs=37790
[ 1990.451671] (t=126006 jiffies g=4517 q=39582)
[ 1990.451677] Task dump for CPU 7:
[ 1990.451682] task:kworker/7:1 state:R running task stack: 0 pid: 1994
ppid: 2 flags:0x0000000a
[ 1990.451702] Workqueue: events dbs_work_handler
[ 1990.451710] Call trace:
[ 1990.451719] dump_backtrace+0x0/0x1b0
[ 1990.451727] show_stack+0x20/0x2c
[ 1990.451735] sched_show_task+0x118/0x150
[ 1990.451746] dump_cpu_task+0x4c/0x5c
[ 1990.451754] rcu_dump_cpu_stacks+0xf0/0x100
[ 1990.451763] rcu_sched_clock_irq+0x250/0x630
[ 1990.451773] update_process_times+0x6c/0x98
[ 1990.451781] tick_sched_handle+0x48/0x5c
[ 1990.451790] tick_sched_timer+0x54/0x98
[ 1990.451800] __hrtimer_run_queues+0x164/0x234
[ 1990.451811] hrtimer_interrupt+0xb8/0x1c0
[ 1990.451820] arch_timer_handler_phys+0x34/0x4c
[ 1990.451829] handle_percpu_devid_irq+0x70/0x12c
[ 1990.451840] generic_handle_irq_desc+0x14/0x20
[ 1990.451849] __handle_domain_irq+0xc0/0xc8
[ 1990.451857] gic_handle_irq+0x2b8/0x308
[ 1990.451864] el1_irq+0xccc/0x100
[ 1990.451874] ioread32+0xc/0x20
[ 1990.451884] smc_send_message+0x3c/0x100
[ 1990.451892] do_xfer+0xdc/0x2a0
[ 1990.451900] scmi_clock_rate_get+0x78/0xe0
[ 1990.451911] scmi_clk_recalc_rate+0x40/0x74
[ 1990.451918] clk_recalc+0x48/0x68
[ 1990.451926] __clk_recalc_rates+0x3c/0xa0
[ 1990.451934] clk_core_get_rate_recalc+0x30/0x48
[ 1990.451941] clk_get_rate+0x2c/0x50
[ 1990.451949] dev_pm_opp_set_rate+0x12c/0x4e0
[ 1990.451958] rockchip_cpufreq_opp_set_rate+0x58/0xe0
[ 1990.451966] set_target+0x38/0x44
[ 1990.451976] __cpufreq_driver_target+0x248/0x2f4
[ 1990.451987] od_dbs_update+0xf0/0x174
[ 1990.451995] dbs_work_handler+0x48/0x80
[ 1990.452006] process_one_work+0x1e0/0x298
[ 1990.452016] worker_thread+0x1e0/0x278
[ 1990.452026] kthread+0xf4/0x104
[ 1990.452035] ret_from_fork+0x10/0x30
```

Kernel log

```
FATAL: forbidden access (exception class 0x20)
Cell state before exception:
pc: ffffffff0010f43bc lr: ffffffff0010f43bc spsr: 20400085 EL1
sp: ffffffff00a17be90 elr: ffffffff0010f43bc esr: 20 1 0000085
x0: 0000000000000000 x1: 0000000000000000 x2: 0000000000000000
x3: 0000000000000000 x4: 0000000000000000 x5: 0000000000000000
x6: 0000000000000000 x7: 0000000000000000 x8: 0000000000000000
x9: 0000000000000000 x10: 0000000000000000 x11: 0000000000000000
x12: 0000000000000000 x13: 0000000000000000 x14: 0000000000000000
x15: 0000000000000000 x16: 0000000000000000 x17: 0000000000000000
x18: 0000000000000000 x19: ffffffff0010f8e90 x20: 0000000000000000
x21: 0000000000000000 x22: 0000000000000000 x23: ffffffff009f0a3b0
x24: 0000000000000000 x25: ffffffff00945659f x26: ffffffff00a17c000
x27: 0000000000000000 x28: ffffffff80030e8000 x29: ffffffff00a17be90

Parking CPU 4 (Cell: "orangeip5p")

FATAL: Unhandled HYP exception: synchronous abort from EL2
pc: 0000ffffc0203e60 lr: 0000ffffc0203e60 spsr: 604003c9 EL2
sp: 0000ffff0000010e60 elr: 0000ffffc0203e60 esr: 00 1 0000000
x0: 0000000000000000 x1: 0000000000000000 x2: 0000000000000000
x3: 0000000000000000 x4: 000000000000003f x5: 0000ffffc021e000
x6: 0000ffffc021c0c0 x7: 000000000000000a x8: 0000000000000000
x9: 0000000000000000 x10: 0000000000000020 x11: 0000000000000000
x12: 0000000000000000 x13: 0000000000000000 x14: 0000000000000000
x15: 0000000000000000 x16: 0000ffffc020dc5c x17: 0000000000000000
x18: 0000000000000000 x19: 0000000000000000 x20: 0000000000000000
x21: 0000000000000000 x22: 0000000000000000 x23: ffffffff009f0a3b0
x24: 0000000000000000 x25: ffffffff00945659f x26: ffffffff00a17c000
x27: 0000000000000000 x28: ffffffff80030e8000 x29: 0000ffff0000010e60

Hypervisor stack before exception (0x0000ffff0000010e60 - 0x0000ffff0000011000):
0e60: 00010e90 0000ffff c0208888 0000ffff c021b000 0000ffff 00012000 0000ffff
0e80: 00000000 00000000 00000001 00000000 00010e90 0000ffff c020b798 0000ffff
0ea0: 00010e90 0000ffff c020891c 0000ffff 010f8e90 ffffffff 00000000 00000000
0ec0: 00000000 00000000 c0204f88 0000ffff 00000000 00000000 00010f08 0000ffff
0ee0: 010f43bc ffffffff 82000085 00000000 20400085 00000000 0a17be90 ffffffff
0f00: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f20: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f40: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f60: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f80: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0fa0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0fc0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Hypervisor dump

# Debug Jailhouse Example 1

Failing due to Improper Memory Translation Setup

esr\_el1: 0x20 0x1 0x0000085

```
FATAL: forbidden access (exception class 0x20)
Cell state before exception:
pc: ffffffff0010f43bc   lr: ffffffff0010f43bc   spsr: 20400085   EL1
sp: ffffffff00a17be90   elr: ffffffff0010f43bc   esr: 20 1 0000085
x0: 0000000000000000   x1: 0000000000000000   x2: 0000000000000000
x3: 0000000000000000   x4: 0000000000000000   x5: 0000000000000000
x6: 0000000000000000   x7: 0000000000000000   x8: 0000000000000000
x9: 0000000000000000   x10: 0000000000000000   x11: 0000000000000000
x12: 0000000000000000   x13: 0000000000000000   x14: 0000000000000000
x15: 0000000000000000   x16: 0000000000000000   x17: 0000000000000000
x18: 0000000000000000   x19: ffffffff0010f8e90   x20: 0000000000000000
x21: 0000000000000000   x22: 0000000000000000   x23: ffffffff009f0a3b0
x24: 0000000000000000   x25: ffffffff00945659f   x26: ffffffff00a17c000
x27: 0000000000000000   x28: ffffffff80030e8000   x29: ffffffff00a17be90

Parking CPU 4 (Cell: "orangeipi5p")

FATAL: Unhandled HYP exception: synchronous abort from EL2
pc: 0000ffffc0203e60   lr: 0000ffffc0203e0c   spsr: 604003c9   EL2
sp: 0000ff00000010e60   elr: 0000ffffc0203e60   esr: 00 1 0000000
x0: 00000000000300000   x1: 00000000000000000   x2: 00000000000000100
x3: 000000000000000100   x4: 0000000000000003f   x5: 0000ffffc021e000
x6: 0000ffffc021c0c0   x7: 0000000000000000a   x8: 00000000000000000
x9: 00000000000000000   x10: 00000000000000020   x11: 00000000000000000
x12: 00000000000000000   x13: 00000000000000000   x14: 00000000000000000
x15: 00000000000000000   x16: 0000ffffc020dc5c   x17: 00000000000000000
x18: 00000000000000000   x19: 00000000000000000   x20: 00000000000000000
x21: 00000000000000000   x22: 00000000000000000   x23: ffffffff009f0a3b0
x24: 00000000000000000   x25: ffffffff00945659f   x26: ffffffff00a17c000
x27: 00000000000000000   x28: ffffffff80030e8000   x29: 0000ff00000010e60

Hypervisor stack before exception (0x0000ff00000010e60 - 0x0000ff00000011000):
0e60: 00010e90 0000ff00 c0200888 0000ffff c021b000 0000ffff 00012000 0000ff00
0e80: 00000000 00000000 00000001 00000000 00010ea0 0000ff00 c020b798 0000ffff
0ea0: 00010ec0 0000ff00 c020891c 0000ffff 010f8e90 ffffffff00 00000000 00000000
0ec0: 00000000 00000000 c0204fc8 0000ffff 00000000 00000000 00010f08 0000ff00
0ee0: 010f43bc ffffffff00 82000085 00000000 20400085 00000000 0a17be90 ffffffff00
0f00: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f20: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f40: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f60: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f80: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0fa0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0fc0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0fe0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Stopping CPU 4 (Cell: "orangeipi5p")
```

Hypervisor dump



# Debug Jailhouse Example 1

Failing due to Improper Memory Translation Setup

esr\_el1: 0x20 0x1 0x0000085

Check the ESR\_EL1 in the manual.

-> EC (Exception Class) = 0x20 = 0b100000

## D19.2.37 ESR\_EL1, Exception Syndrome Register (EL1)

The ESR\_EL1 characteristics are:

### Purpose

Holds syndrome information for an exception taken to EL1.

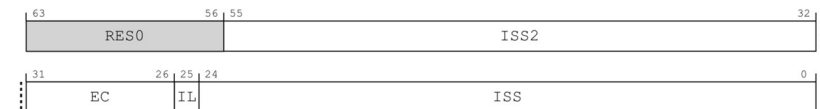
### Configurations

AArch64 System register ESR\_EL1 bits [31:0] are architecturally mapped to AArch32 System register DFSR[31:0].

### Attributes

ESR\_EL1 is a 64-bit register.

### Field descriptions



ESR\_EL1 is made UNKNOWN as a result of an exception return from EL1.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL1, the value of ESR\_EL1 is UNKNOWN. The value written to ESR\_EL1 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

Description of ESR\_EL1 from Arm Architecture Reference Manual for A-profile architecture

# Debug Jailhouse Example 1

Failing due to Improper Memory Translation Setup

esr\_el1: 0x20 0x1 0x0000085

Check the ESR\_EL1 in the manual.

-> EC (Exception Class) = 0x20 = 0b100000

-> MMU faults

-> This indicates that probably the memory translation in Jailhouse is not properly set up.

**EC == 0b100000**

Instruction Abort from a lower Exception level.

Used for **MMU faults** generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug-related exceptions.

See [ISS encoding for an exception from an Instruction Abort](#).

Description of EC from Arm Architecture Reference Manual for A-profile architecture

## Debug Jailhouse Example 2

Failing at ITS (Interrupt Translation Service)

pc: 0xffffffc0086590f4

This address seems to be in the Linux Kernel,  
because it is so high (starting with 0xffff...).

```
FATAL: unhandled trap (exception class 0x24)
Cell state before exception:
pc: ffffffc0086590f4  lr: ffffffc00865b324  spsr: a0c00085      EL1
sp: ffffffc00a37bb10  elr: ffffffc0086590f4  esr: 24 1 1800006
x0: ffffffc00a140090  x1: 0000000000000000  x2: ffffff81000f0000
x3: 0000000000000000  x4: 0000000000000000  x5: 0000000000000007
x6: ffffffc00a37bb70  x7: 0000000000000000  x8: 0000000000000000
x9: ffffffc00865b324  x10: 000000000000000a2  x11: ffffffc009dcf7b0
x12: ffffffc3f4794000  x13: ffffff8100394c40  x14: ffffff8100293f10
x15: ffffff810040d7f0  x16: 0000000000000000  x17: 0000000000000000
x18: 0000000000000000  x19: ffffff81000f4a00  x20: ffffff83fc50e800
x21: 000000000000f4240  x22: 0000000000000080  x23: ffffffc00a37bbd8
x24: 0000000000000001  x25: ffffff8106a35800  x26: 0000000000000000
x27: ffffffc009cd4000  x28: ffffff8106a35818  x29: ffffffc00a37bb10

Parking CPU 1 (Cell: "orangeipi5p")
```

Hypervisor dump

# Debug Jailhouse Example 2

Failing at ITS (Interrupt Translation Service)

pc: 0xffffffc0086590f4

Check the disassembly of the Linux Kernel.

`aarch64-none-linux-gnu-objdump -D vmlinux >`

`vmlinux_disassembly.txt`

This address is within the function `its_allocate_entry`.

However, ITS is not supported in Jailhouse.

-> Disable it in the Kernel config

```
ffffffc0086590c0 <its_allocate_entry>:
ffffffc0086590c0: d503201f      nop
ffffffc0086590c4: d503201f      nop
ffffffc0086590c8: d503233f      paciasp
ffffffc0086590cc: a9bd7bfd      stp     x29, x30, [sp, #-48]!
ffffffc0086590d0: 910003fd      mov     x29, sp
ffffffc0086590d4: a90153f3      stp     x19, x20, [sp, #16]
ffffffc0086590d8: aa0003f4      mov     x20, x0
ffffffc0086590dc: f90013f5      str     x21, [sp, #32]
ffffffc0086590e0: 52884815      mov     w21, #0x4240
ffffffc0086590e4: 72a001f5      movk    w21, #0xf, lsl #16
ffffffc0086590e8: f9403280      ldr     x0, [x20, #96]
ffffffc0086590ec: a947ce82      ldp     x2, x19, [x20, #120]
ffffffc0086590f0: 91024000      add     x0, x0, #0x90
ffffffc0086590f4: b9400000      ldr     w0, [x0]
ffffffc0086590f8: cb020261      sub     x1, x19, x2
ffffffc0086590fc: 9345fc21      asr     x1, x1, #5
ffffffc008659100: 11000421      add     w1, w1, #0x1
ffffffc008659104: 12002821      and     w1, w1, #0x7ff
ffffffc008659108: 6b40143f      cmp     w1, w0, lsr #5
ffffffc00865910c: 540001a0      b.eq    ffffffc008659140 <its_allocate
ffffffc008659110: 91008260      add     x0, x19, #0x20
ffffffc008659114: 91404041      add     x1, x2, #0x10, lsl #12
ffffffc008659118: eb01001f      cmp     x0, x1
ffffffc00865911c: 9a821000      csel    x0, x0, x2, ne // ne = any
ffffffc008659120: f9004280      str     x0, [x20, #128]
ffffffc008659124: a9007e7f      stp     xzr, xzr, [x19]
ffffffc008659128: a9017e7f      stp     xzr, xzr, [x19, #16]
ffffffc00865912c: 14000012      b       ffffffc008659174 <its_allocate
ffffffc008659130: d503203f      yield
ffffffc008659134: d28218e0      mov     x0, #0x10c7
```

Linux Kernel Disassembly Snippet

# Debug Jailhouse Example 3

Failing at accessing FPEXC32\_EL2

pc: 0x0000ffffc020460c

This address is not that high, so it does not seem to be within the Kernel. It could be within the Jailhouse.

```
FATAL: Unhandled HYP exception: synchronous abort from EL2
pc: 0000ffffc020460c  lr: 0000ffffc02045b8  spsr: 604003c9  EL2
sp: 0000ff00000010e40  elr: 0000ffffc020460c  esr: 00 1 00000000
x0: 000000000000300000  x1: 000000000000000000  x2: 000000000000001000
x3: 00000000000000100  x4: 0000ffffc02039a0  x5: 000000000000000000
x6: 000000000000000000  x7: 000000000000000000  x8: 000000000000000000
x9: ffffffff008c55054  x10: 00000000000000ae0  x11: 000000000000000008
x12: ffffffff3f485e000  x13: ffffffff81002f2dc0  x14: ffffffff81001a3d10
x15: ffffffff81044b1c60  x16: 000000000000000000  x17: 000000000000000000
x18: 000000000000000000  x19: 000000000000000000  x20: 000000000000000000
x21: 000000000000000000  x22: ffffffff009768638  x23: 000000000000000080
x24: ffffffff009cc49f8  x25: 000000000000000000  x26: 000000000000000000
x27: 000000000000000000  x28: 000000000000000000  x29: 0000ff00000010e40

Hypervisor stack before exception (0x0000ff00000010e40 - 0x0000ff00000011000):
0e40: 00010e70 0000ff00 c0200890 0000ffff 00010f08 0000ff00 00010ed8 0000ff00
0e60: 09cc4988 ffffffff c0209bac 0000ffff 00010e80 0000ff00 c02039a4 0000ffff
0e80: 00010ea0 0000ff00 c0203e5c 0000ffff 00010f08 0000ff00 00010ed8 0000ff00
0ea0: 00010ec0 0000ff00 c0209bac 0000ffff 08fe28e8 ffffffff 00000007 00000000
0ec0: 00000000 00000000 c02057c8 0000ffff 00000000 00000000 00010f08 0000ff00
0ee0: 0802799c ffffffff 5e000000 00000000 60c003c5 00000000 0a32bea0 ffffffff
0f00: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f20: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f40: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f60: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0f80: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0fa0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0fc0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0fe0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
Stopping CPU 7 (Cell: "orangeip5p")
```

Hypervisor dump



# Debug Jailhouse Example 3

Failing at accessing FPEXC32\_EL2

pc: 0x0000ffffc020460c

Check the disassembly of the Jailhouse.

aarch64-none-linux-gnu-objdump -D hypervisor.o

> hypervisor\_disassembly.txt

This address corresponds to reading fpexc32\_el2.

However, fpexc32\_el2 does not exist on Cortex-A76.

-> Skip reading fpexc32\_el2; we don't need it now.

```
0000ffffc0204570 <arm_cpu_reset>:
ffffc0204570: a9bd7bfd      stp     x29, x30, [sp, #-48]!
ffffc0204574: 910003fd      mov     x29, sp
ffffc0204578: a90153f3      stp     x19, x20, [sp, #16]
ffffc020457c: aa0003f4      mov     x20, x0
ffffc0204580: d2810000      mov     x0, #0x800
ffffc0204584: f2a61a00      movk    x0, #0x30d0, lsl #16
ffffc0204588: f90013f5      str     x21, [sp, #32]
ffffc020458c: 2a0103f5      mov     w21, w1
ffffc0204590: d5181000      msr     sctlr_el1, x0
ffffc0204594: d2800013      mov     x19, #0x0
ffffc0204598: d518e113      msr     cntkctl_el1, x19
ffffc020459c: d51b9c13      msr     pmcr_el0, x19
ffffc02045a0: d281e000      mov     x0, #0xf00
ffffc02045a4: d2802002      mov     x2, #0x100
ffffc02045a8: f2a00020      movk    x0, #0x1, lsl #16
ffffc02045ac: 52800001      mov     w1, #0x0
ffffc02045b0: f2dfe000      movk    x0, #0xff00, lsl #32
ffffc02045b4: 9400217e      bl      fffff020cbac <memset>
ffffc02045b8: d5184113      msr     sp_el0, x19
ffffc02045bc: d51c4113      msr     sp_el1, x19
ffffc02045c0: d5184013      msr     spsr_el1, x19
ffffc02045c4: d5185113      msr     afsr0_el1, x19
ffffc02045c8: d5185133      msr     afsr1_el1, x19
ffffc02045cc: d518a313      msr     amair_el1, x19
ffffc02045d0: d518d033      msr     contextidr_el1, x19
ffffc02045d4: d2a00600      mov     x0, #0x300000
ffffc02045d8: d5181040      msr     cpacr_el1, x0
ffffc02045dc: d51a0013      msr     csselr_el1, x19
ffffc02045e0: d5185213      msr     esr_el1, x19
ffffc02045e4: d5186013      msr     far_el1, x19
ffffc02045e8: d518a213      msr     mair_el1, x19
ffffc02045ec: d5187413      msr     par_el1, x19
ffffc02045f0: d5182053      msr     tcr_el1, x19
ffffc02045f4: d51bd073      msr     tpidrro_el0, x19
ffffc02045f8: d51bd053      msr     tpidr_el0, x19
ffffc02045fc: d518d093      msr     tpidr_el1, x19
ffffc0204600: d5182013      msr     ttbr0_el1, x19
ffffc0204604: d5182033      msr     ttbr1_el1, x19
ffffc0204608: d518c013      msr     vbar_el1, x19
ffffc020460c: d53c5300      mrs     x0, fpexc32_el2
ffffc0204610: b2620000      orr     x0, x0, #0x40000000
ffffc0204614: d51c5300      msr     fpexc32_el2, x0
ffffc0204618: d51be233      msr     cntp_ctl_el0, x19
ffffc020461c: d51be253      msr     cntp_cval_el0, x19
ffffc0204620: d51be213      msr     cntp_tval_el0, x19
ffffc0204624: d51be333      msr     cntv_ctl_el0, x19
```

Jailhouse Disassembly Snippet

# Debug Jailhouse Example 4

CPU stalling at smc calls

The Linux Kernel has detected a CPU stall.

The `smc_send_message` before the interrupt handler  
(that periodically checks whether the CPU is making  
progress) seems suspicious.

```
[ 1990.451647] rcu: INFO: rcu_sched self-detected stall on CPU
[ 1990.451662] rcu: 7-....: (125950 ticks this GP) idle=1a
[ 1990.451671] (t=126006 jiffies g=4517 q=39582)
[ 1990.451677] Task dump for CPU 7:
[ 1990.451682] task:kworker/7:1 state:R running task
0x0000000a
[ 1990.451702] Workqueue: events dbs_work_handler
[ 1990.451710] Call trace:
[ 1990.451719] dump_backtrace+0x0/0x1b0
[ 1990.451727] show_stack+0x20/0x2c
[ 1990.451735] sched_show_task+0x118/0x150
[ 1990.451746] dump_cpu_task+0x4c/0x5c
[ 1990.451754] rcu_dump_cpu_stacks+0xf0/0xfc
[ 1990.451763] rcu_sched_clock_irq+0x250/0x630
[ 1990.451773] update_process_times+0x6c/0x98
[ 1990.451781] tick_sched_handle+0x48/0x5c
[ 1990.451790] tick_sched_timer+0x54/0x98
[ 1990.451800] __hrtimer_run_queues+0x164/0x234
[ 1990.451811] hrtimer_interrupt+0xb8/0x1c0
[ 1990.451820] arch_timer_handler_phys+0x34/0x4c
[ 1990.451829] handle_percpu_devid_irq+0x70/0x12c
[ 1990.451840] generic_handle_irq_desc+0x14/0x20
[ 1990.451849] __handle_domain_irq+0xc0/0xc8
[ 1990.451857] gic_handle_irq+0x2b8/0x308
[ 1990.451864] el1_irq+0xcc/0x180
[ 1990.451874] ioread32+0xc/0x20
[ 1990.451884] smc_send_message+0x3c/0x100
[ 1990.451892] do_xfer+0xdc/0x2a0
[ 1990.451900] scmi_clock_rate_get+0x78/0xe0
[ 1990.451911] scmi_clk_recalc_rate+0x40/0x74
[ 1990.451918] clk_recalc+0x48/0x68
[ 1990.451926] __clk_recalc_rates+0x3c/0xa0
[ 1990.451934] clk_core_get_rate_recalc+0x30/0x48
[ 1990.451941] clk_get_rate+0x2c/0x50
[ 1990.451949] dev_pm_opp_set_rate+0x12c/0x4e0
[ 1990.451958] rockchip_cpufreq_opp_set_rate+0x58/0xe0
[ 1990.451966] set_target+0x38/0x44
[ 1990.451976] __cpufreq_driver_target+0x248/0x2f4
[ 1990.451987] od_dbs_update+0xf0/0x174
[ 1990.451995] dbs_work_handler+0x48/0x80
[ 1990.452006] process_one_work+0x1e0/0x298
[ 1990.452016] worker_thread+0x1e0/0x278
[ 1990.452026] kthread+0xf4/0x104
[ 1990.452035] ret_from_fork+0x10/0x30
```

Kernel log

# Debug Jailhouse Example 4

CPU stalling at smc calls

If we search for `smc_send_message` before in the Linux source code, it is in `drivers/firmware/arm_scm/smc.c`.

Further investigation shows that it is stuck at a spin lock

in `shmem_tx_prepare` of `drivers/firmware/arm_scm/shmem.c`.

```
[ 1990.451647] rcu: INFO: rcu_sched self-detected stall on CPU
[ 1990.451662] rcu: 7-....: (125950 ticks this GP) idle=1a
fqs=37790
[ 1990.451671] (t=126006 jiffies g=4517 q=39582)
[ 1990.451677] Task dump for CPU 7:
[ 1990.451682] task:kworker/7:1 state:R running task
0x0000000a
[ 1990.451702] Workqueue: events dbs_work_handler
[ 1990.451710] Call trace:
[ 1990.451719] dump_backtrace+0x0/0x1b0
[ 1990.451727] show_stack+0x20/0x2c
[ 1990.451735] sched_show_task+0x118/0x150
[ 1990.451746] dump_cpu_task+0x4c/0x5c
[ 1990.451754] rcu_dump_cpu_stacks+0xf0/0xfc
[ 1990.451763] rcu_sched_clock_irq+0x250/0x630
[ 1990.451773] update_process_times+0x6c/0x98
[ 1990.451781] tick_sched_handle+0x48/0x5c
[ 1990.451790] tick_sched_timer+0x54/0x98
[ 1990.451800] __hrtimer_run_queues+0x164/0x234
[ 1990.451811] hrtimer_interrupt+0xb8/0x1c0
[ 1990.451820] arch_timer_handler_phys+0x34/0x4c
[ 1990.451829] handle_percpu_devid_irq+0x70/0x12c
[ 1990.451840] generic_handle_irq_desc+0x14/0x20
[ 1990.451849] __handle_domain_irq+0xc0/0xc8
[ 1990.451857] gic_handle_irq+0x2b8/0x308
[ 1990.451864] el1_irq+0xcc/0x180
[ 1990.451874] ioread32+0xc/0x20
[ 1990.451884] smc_send_message+0x3c/0x100
[ 1990.451892] do_xfer+0xdc/0x2a0
[ 1990.451900] scmi_clock_rate_get+0x78/0xe0
[ 1990.451911] scmi_clk_recalc_rate+0x40/0x74
[ 1990.451918] clk_recalc+0x48/0x68
[ 1990.451926] __clk_recalc_rates+0x3c/0xa0
[ 1990.451934] clk_core_get_rate_recalc+0x30/0x48
[ 1990.451941] clk_get_rate+0x2c/0x50
[ 1990.451949] dev_pm_opp_set_rate+0x12c/0x4e0
[ 1990.451958] rockchip_cpufreq_opp_set_rate+0x58/0xe0
[ 1990.451966] set_target+0x38/0x44
[ 1990.451976] __cpufreq_driver_target+0x248/0x2f4
[ 1990.451987] od_dbs_update+0xf0/0x174
[ 1990.451995] dbs_work_handler+0x48/0x80
[ 1990.452006] process_one_work+0x1e0/0x298
[ 1990.452016] worker_thread+0x1e0/0x278
[ 1990.452026] kthread+0xf4/0x104
[ 1990.452035] ret_from_fork+0x10/0x30
```

Kernel log



# Debug Jailhouse Example 4

CPU stalling at smc calls

The smc call was not passed to the firmware, so the spin lock is never unlocked, and therefore, it gets stuck at the spin lock.

-> A patch that allows the smc call to pass through is applied.

```
[ 1990.451647] rcu: INFO: rcu_sched self-detected stall on CPU
[ 1990.451662] rcu: 7-....: (125950 ticks this GP) idle=1a
fqs=37790
[ 1990.451671] (t=126006 jiffies g=4517 q=39582)
[ 1990.451677] Task dump for CPU 7:
[ 1990.451682] task:kworker/7:1 state:R running task
0x0000000a
[ 1990.451702] Workqueue: events dbs_work_handler
[ 1990.451710] Call trace:
[ 1990.451719] dump_backtrace+0x0/0x1b0
[ 1990.451727] show_stack+0x20/0x2c
[ 1990.451735] sched_show_task+0x118/0x150
[ 1990.451746] dump_cpu_task+0x4c/0x5c
[ 1990.451754] rcu_dump_cpu_stacks+0xf0/0xfc
[ 1990.451763] rcu_sched_clock_irq+0x250/0x630
[ 1990.451773] update_process_times+0x6c/0x98
[ 1990.451781] tick_sched_handle+0x48/0x5c
[ 1990.451790] tick_sched_timer+0x54/0x98
[ 1990.451800] __hrtimer_run_queues+0x164/0x234
[ 1990.451811] hrtimer_interrupt+0xb8/0x1c0
[ 1990.451820] arch_timer_handler_phys+0x34/0x4c
[ 1990.451829] handle_percpu_devid_irq+0x70/0x12c
[ 1990.451840] generic_handle_irq_desc+0x14/0x20
[ 1990.451849] __handle_domain_irq+0xc0/0xc8
[ 1990.451857] gic_handle_irq+0x2b8/0x308
[ 1990.451864] el1_irq+0xcc/0x180
[ 1990.451874] ioread32+0xc/0x20
[ 1990.451884] smc_send_message+0x3c/0x100
[ 1990.451892] do_xfer+0xdc/0x2a0
[ 1990.451900] scmi_clock_rate_get+0x78/0xe0
[ 1990.451911] scmi_clk_recalc_rate+0x40/0x74
[ 1990.451918] clk_recalc+0x48/0x68
[ 1990.451926] __clk_recalc_rates+0x3c/0xa0
[ 1990.451934] clk_core_get_rate_recalc+0x30/0x48
[ 1990.451941] clk_get_rate+0x2c/0x50
[ 1990.451949] dev_pm_opp_set_rate+0x12c/0x4e0
[ 1990.451958] rockchip_cpufreq_opp_set_rate+0x58/0xe0
[ 1990.451966] set_target+0x38/0x44
[ 1990.451976] __cpufreq_driver_target+0x248/0x2f4
[ 1990.451987] od_dbs_update+0xf0/0x174
[ 1990.451995] dbs_work_handler+0x48/0x80
[ 1990.452006] process_one_work+0x1e0/0x298
[ 1990.452016] worker_thread+0x1e0/0x278
[ 1990.452026] kthread+0xf4/0x104
[ 1990.452035] ret_from_fork+0x10/0x30
```

Kernel log