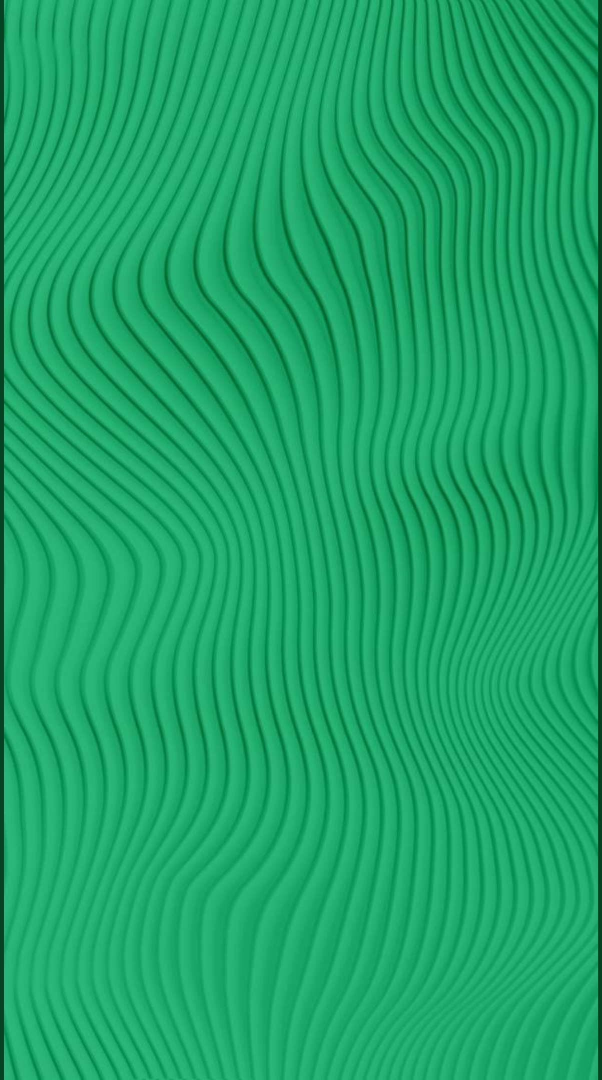


Tips, Tricks, and Gotchas

Linux Real-Time Tuning

Grațian Crișan

gratian.crisan@ni.com,
gratian@gmail.com

About me

- I work for NI (formerly known as National Instruments)
    - Makes hardware & software for test, measurement, and automation
    - Member and supporter of Real-Time Linux Collaborative Project
- Real-Time OS group for the past decade
    - PREEMPT_RT based Linux kernels
    - Embedded 32-bit ARMs and x86_64 systems
    - Distribution based on OpenEmbedded/Yocto
- Maintainer for the Linux kernel shipping on NI’s Real-Time hardware

About this presentation

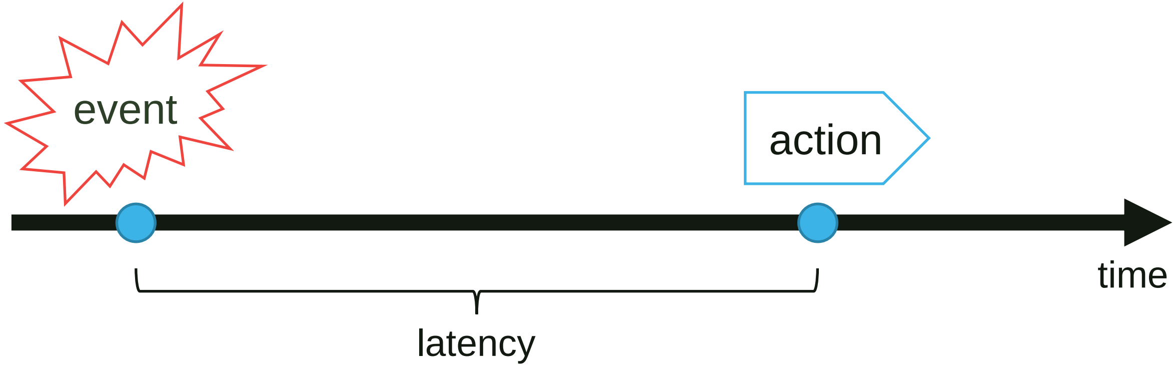
Covered:

- Real-Time
- Tools
- Tuning
- Safety nets
- Gotchas

Not covered:

- Implementation details
- In-depth review of tools
- RT alternatives

Real-Time == Deterministic response to stimulus

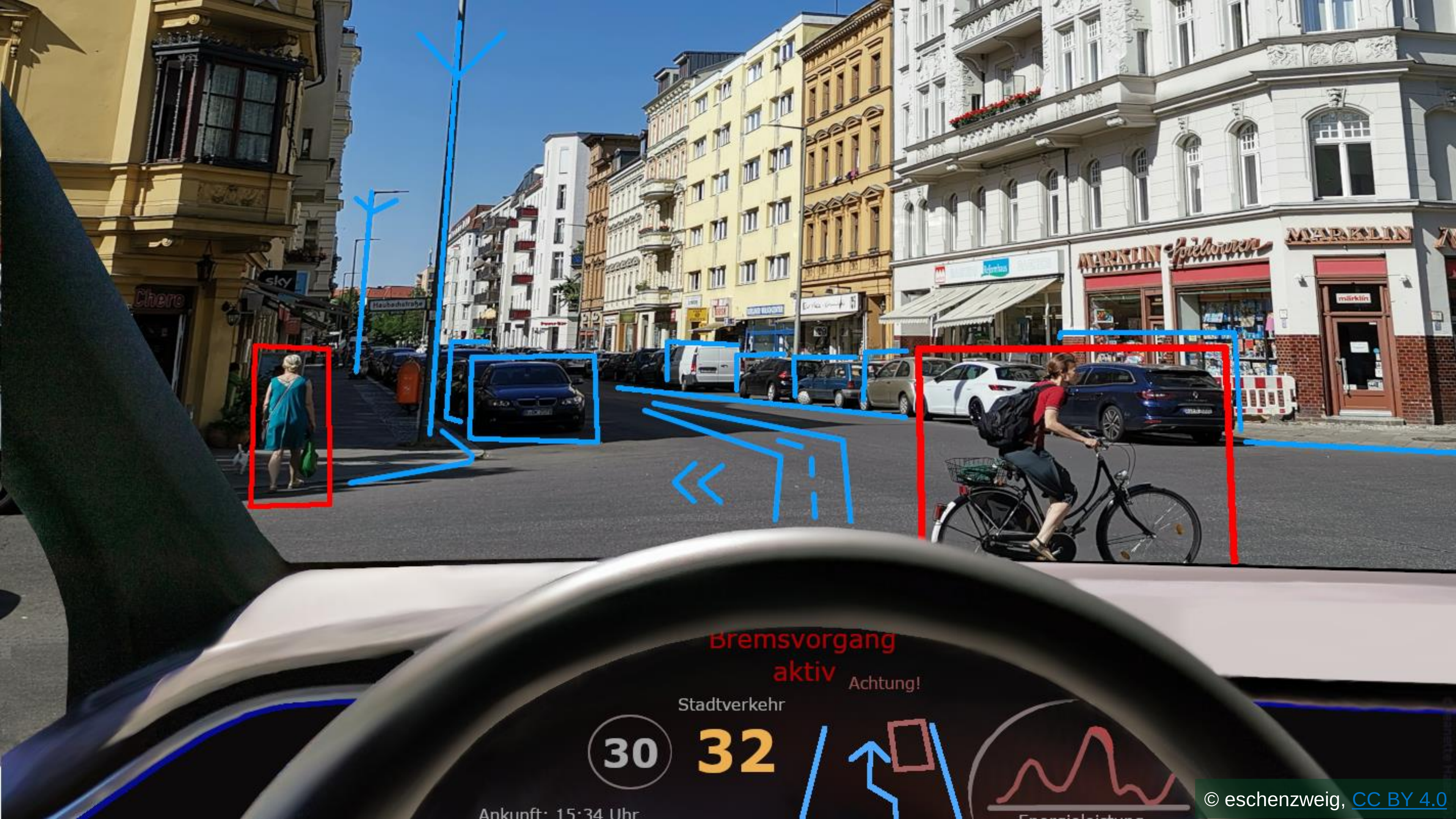


Events can be:

- Asynchronous
- Synchronous (clock driven)

We want the latency to be:

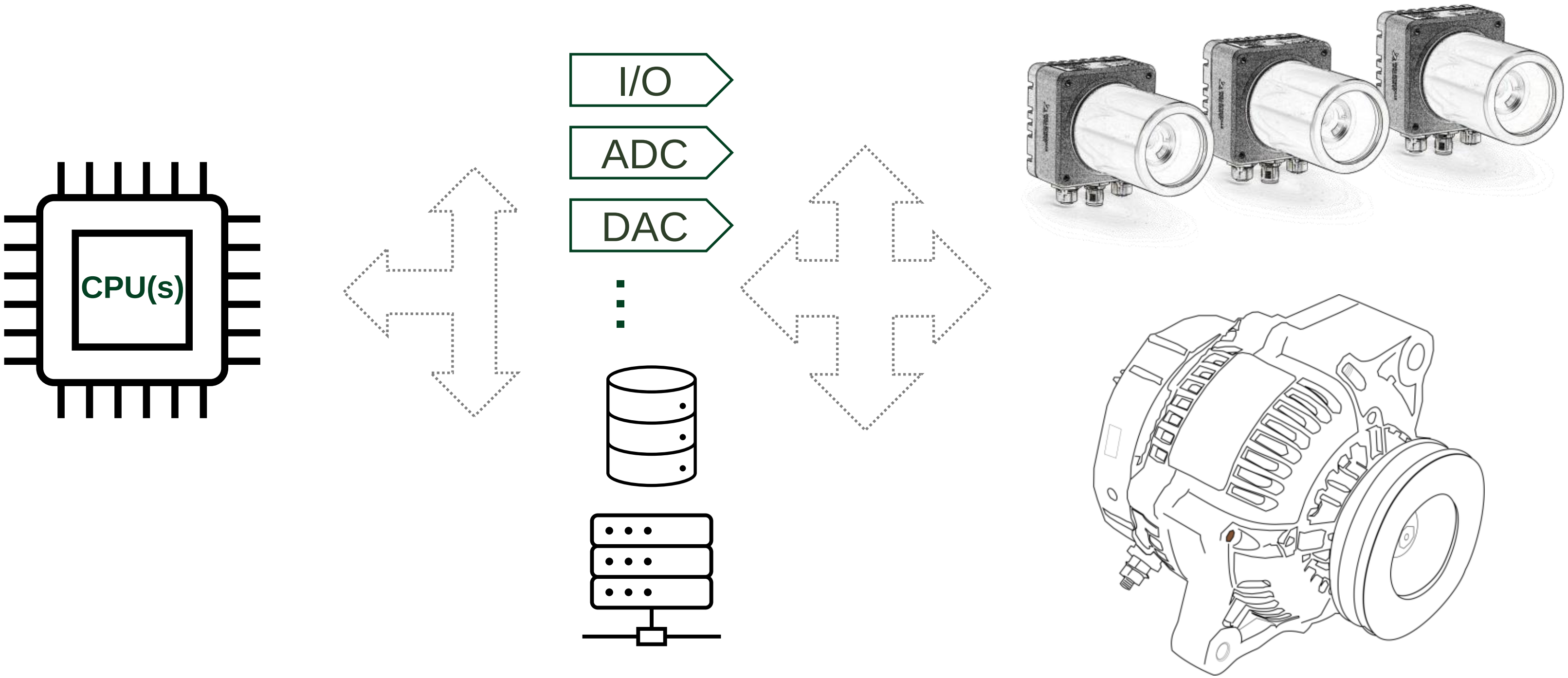
- Predictable
- Bounded



© eschenzweig, CC BY 4.0

Tools

Measuring latency



CPU(s)

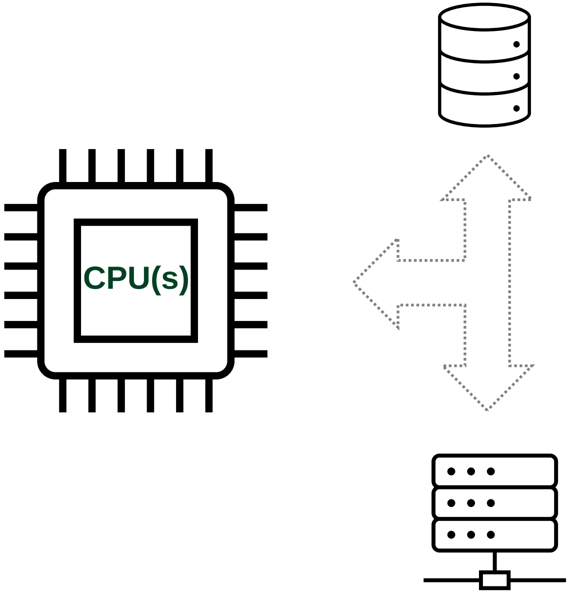
I/O

ADC

DAC

⋮

Cyclictest

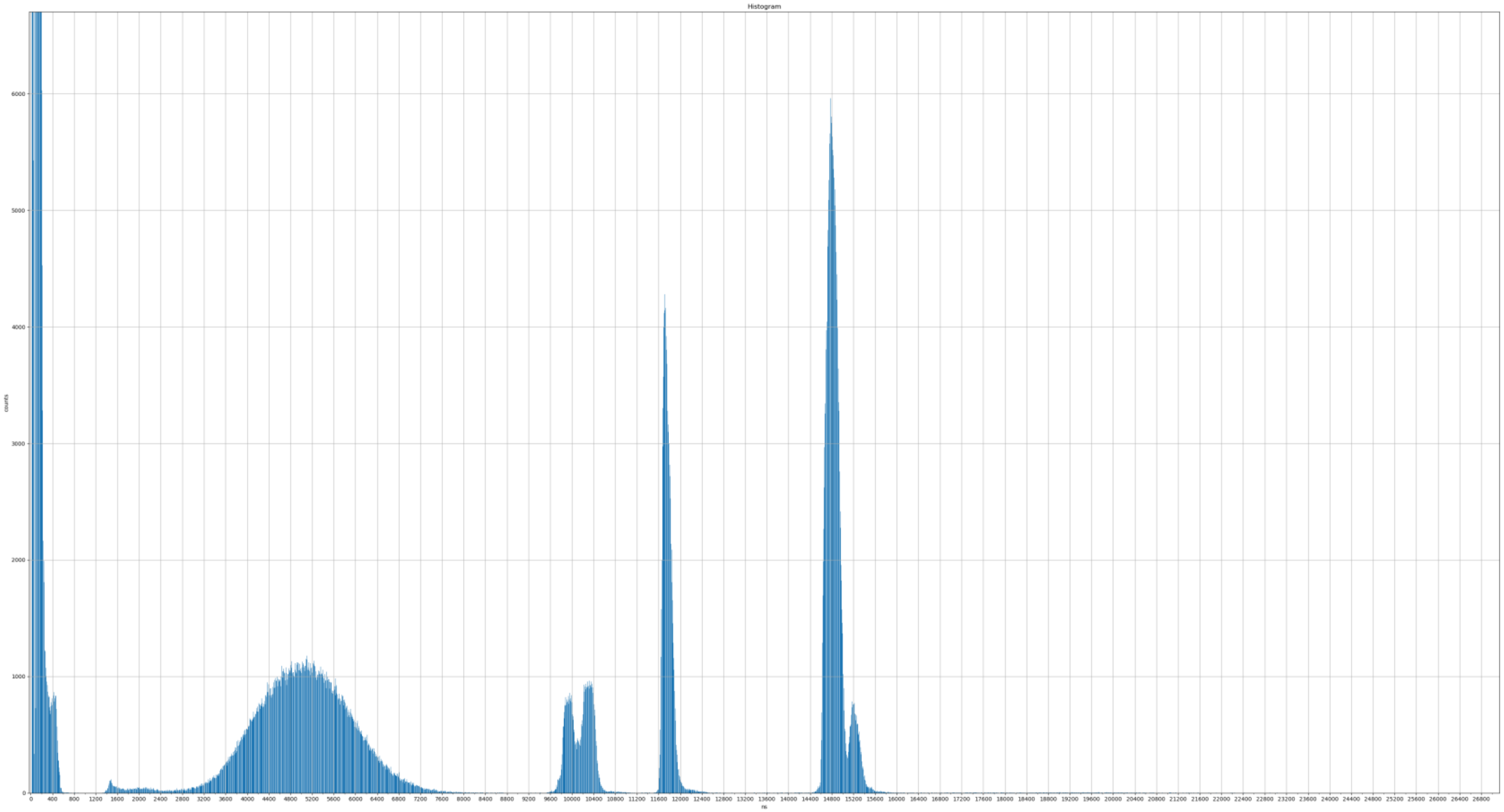


```
static void *timerthread(void *param)
{
    while (!shutdown) {
        clock_gettime(clock, &before);
        clock_nanosleep(clock,…, &interval);
        clock_gettime(clock, &after);

        latency = after - before - interval;
        /* compute statistics/histogram */
    }
}
```

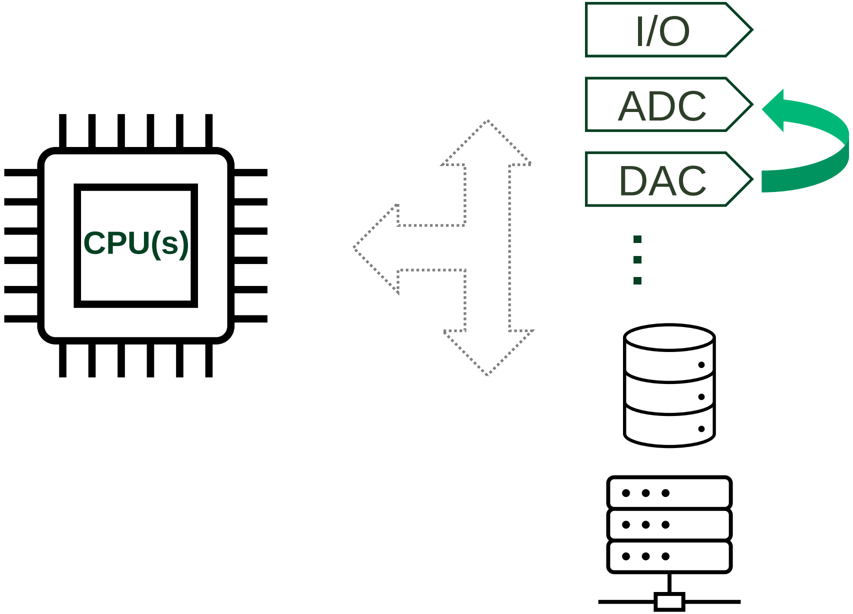
Simulate the load:

- iperf (network)
- fio (disk)
- hackbench (scheduler)

Histograms

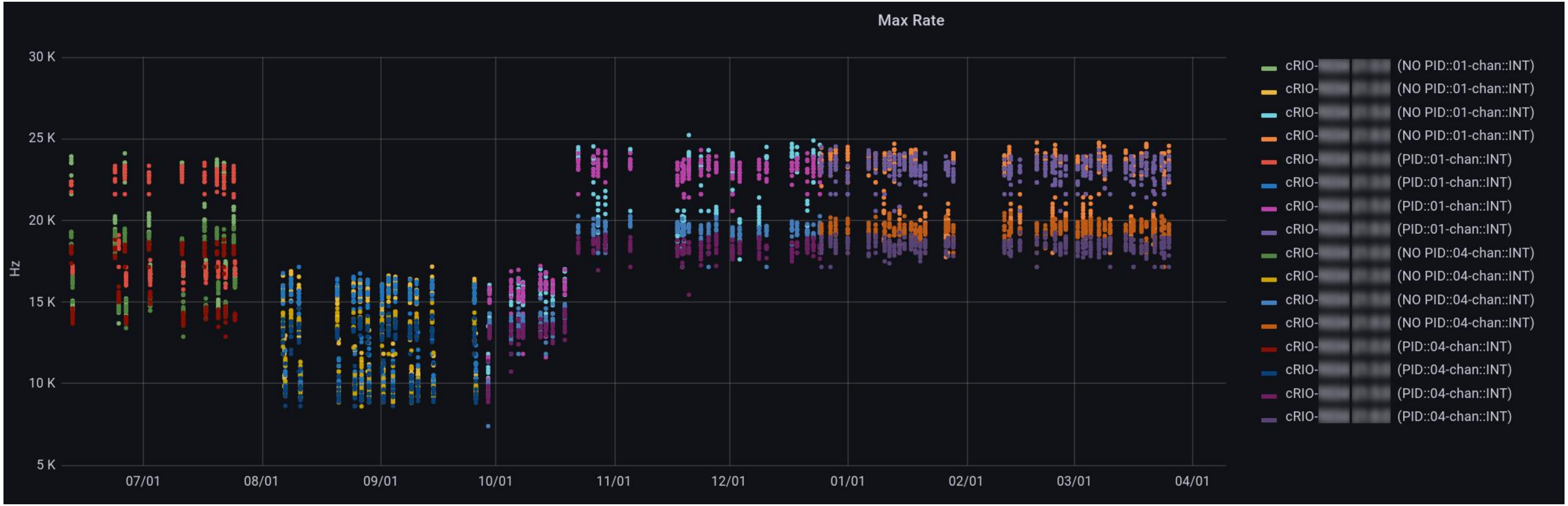

Histogram
counts
0
1000
2000
3000
4000
5000
6000
0 400 800 1200 1600 2000 2400 2800 3200 3600 4000 4400 4800 5200 5600 6000 6400 6800 7200 7600 8000 8400 8800 9200 9600 10000 10400 10800 11200 11600 12000 12400 12800 13200 13600 14000 14400 14800 15200 15600 16000 16400 16800 17200 17600 18000 18400 18800 19200 19600 20000 20400 20800 21200 21600 22000 22400 22800 23200 23600 24000 24400 24800 25200 25600 26000 26400 26800
ns

I/O latency

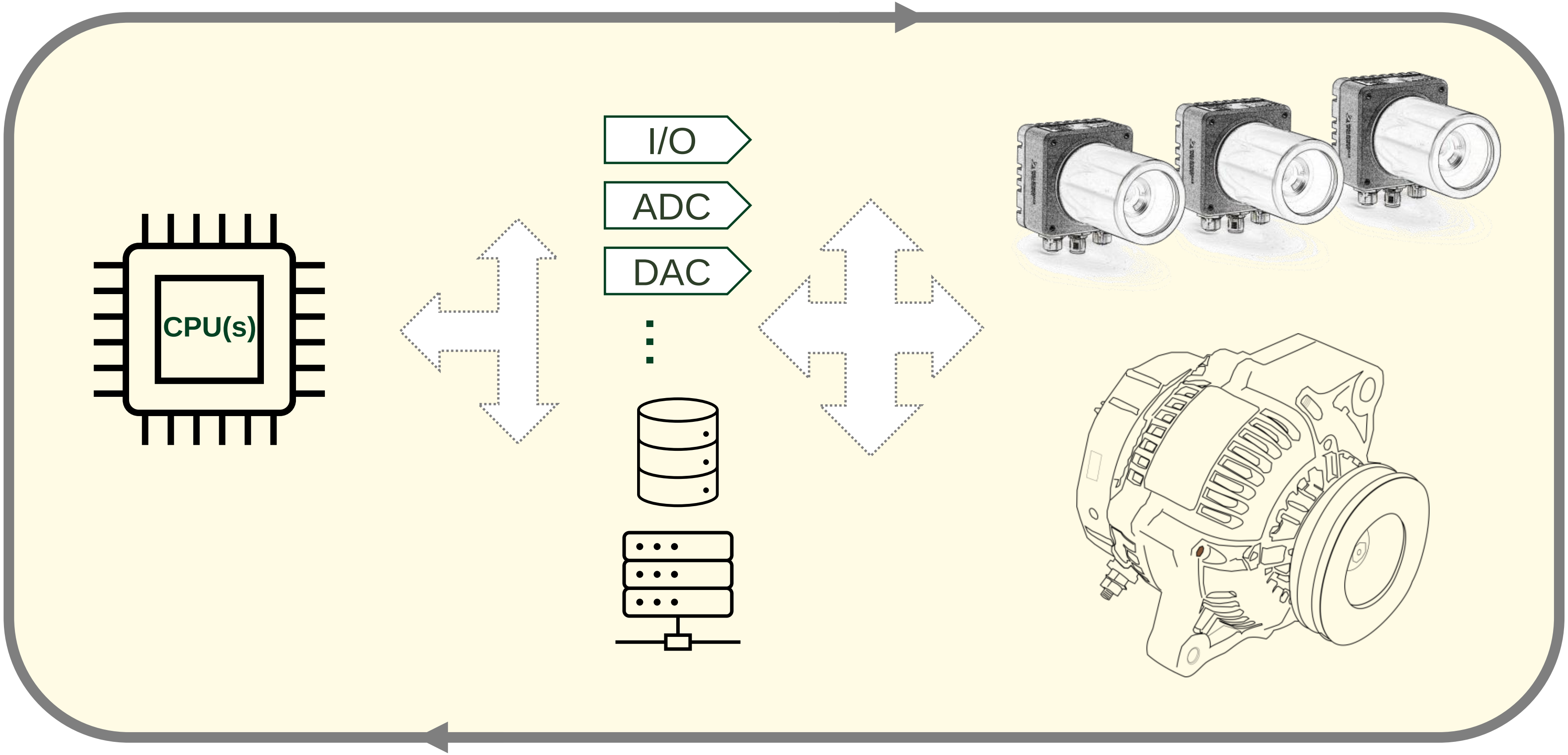


```
static void *timerthread(…)
{
    while (!shutdown) {
        clock_gettime(&before);

        read_inputs(…);
        process_data(…);
        update_outputs(…);

        clock_gettime(&after);

        latency = after-before;
        /* statistics */
    }
}
```

“Single point” tests



Total system latency

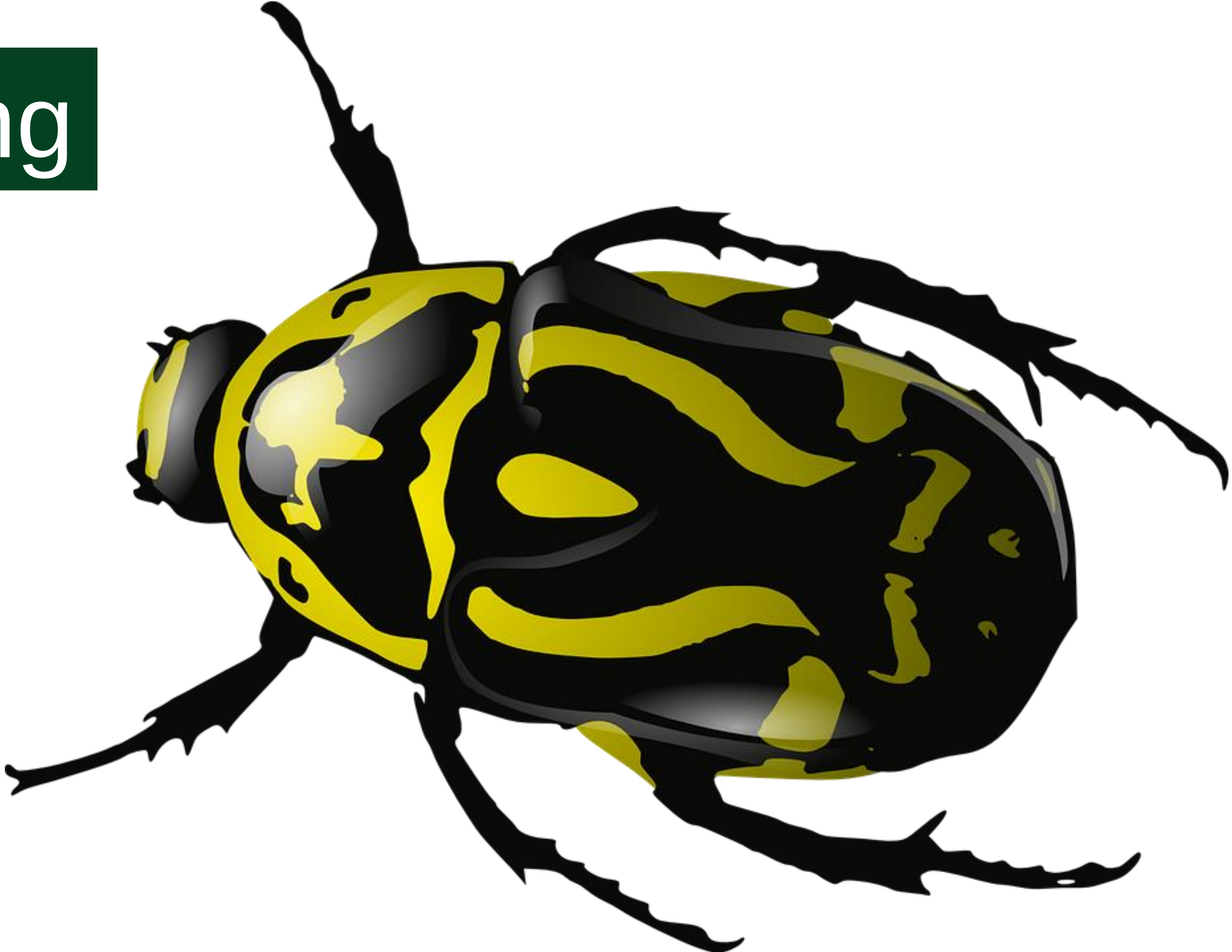


Other tools

- RT-Tests: https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/rt-tests
- RTEval: https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/rteval
- LTP: https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/ltp
- RTLA: https://docs.kernel.org/tools/rtla/index.html

RTLA: Real-time Linux Analysis Toolset - Daniel Bristot De Oliveira, Red Hat
Thursday, Jun 23, 4:55pm; Room 203/204 (Level 2)

Debugging

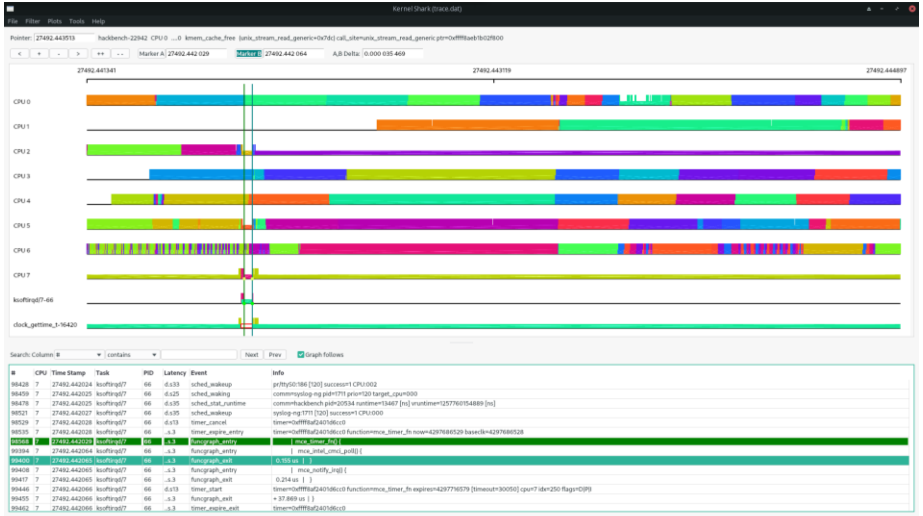


Debugging tools

- ftrace
- trace-cmd
- Kernel Shark
- LTTng, etc.
- perf
- bpftrace, bcc
- GPIO + oscilloscope

```
# cat /sys/kernel/debug/tracing/README
```

```
# trace-cmd --help
```



Pointer: 27492.443513 hackbench-22942 CPU 0 ....0 kmem_cache_free (unix_stream_read_generic+0x7dc) call_site=unix_stream_read_generic ptr=0xffff8aeb1b02f800

Marker A 27492.442 029 Marker B 27492.442 064 A,B Delta: 0.000 035 469

Search: Column # contains

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
98428	7	27492.442024	ksoftirqd/7	66	d.s33	sched_wakeup	pr/ttyS0:186 [120] success=1 CPU:002
98459	7	27492.442025	ksoftirqd/7	66	d.s25	sched_waking	comm=syslog-ng pid=1711 prio=120 target_cpu=000
98478	7	27492.442025	ksoftirqd/7	66	d.s35	sched_stat_runtime	comm=hackbench pid=20534 runtime=13467 [ns] vruntime=1257760154889 [ns]
98521	7	27492.442027	ksoftirqd/7	66	d.s35	sched_wakeup	syslog-ng:1711 [120] success=1 CPU:000
98529	7	27492.442028	ksoftirqd/7	66	d.s13	timer_cancel	timer=0xffff8af2401d6cc0
98535	7	27492.442028	ksoftirqd/7	66	..s.3	timer_expire_entry	timer=0xffff8af2401d6cc0 function=mce_timer_fn now=4297686529 baseclk=4297686528
98568	7	27492.442029	ksoftirqd/7	66	..s.3	funcgraph_entry	| mce_timer_fn() {
99394	7	27492.442064	ksoftirqd/7	66	..s.3	funcgraph_entry	| mce_intel_cmci_poll() {
99400	7	27492.442065	ksoftirqd/7	66	..s.3	funcgraph_exit	0.155 us | }
99408	7	27492.442065	ksoftirqd/7	66	..s.3	funcgraph_entry	| mce_notify_irq() {
99417	7	27492.442065	ksoftirqd/7	66	..s.3	funcgraph_exit	0.214 us | }
99446	7	27492.442066	ksoftirqd/7	66	d.s13	timer_start	timer=0xffff8af2401d6cc0 function=mce_timer_fn expires=4297716579 [timeout=30050] cpu=7 idx=250 flags=D|P|I
99455	7	27492.442066	ksoftirqd/7	66	..s.3	funcgraph_exit	+ 37.869 us | }
99462	7	27492.442066	ksoftirqd/7	66	..s.3	timer_expire_exit	timer=0xffff8af2401d6cc0

Tuning

© Franklin Heijnen, CC BY-SA 2.0, via Wikimedia Commons

Kernel

Patch (for now):

- http://git.kernel.org/cgit/linux/kernel/git/rt/linux-stable-rt.git
- http://git.kernel.org/cgit/linux/kernel/git/rt/linux-rt-devel.git

BoF: Realtime Linux
Steven Rostedt, Google
Friday, Jun 24, 11:10AM
Griffin Hall (Level 2)

CONFIG_PREEMPT_RT = y

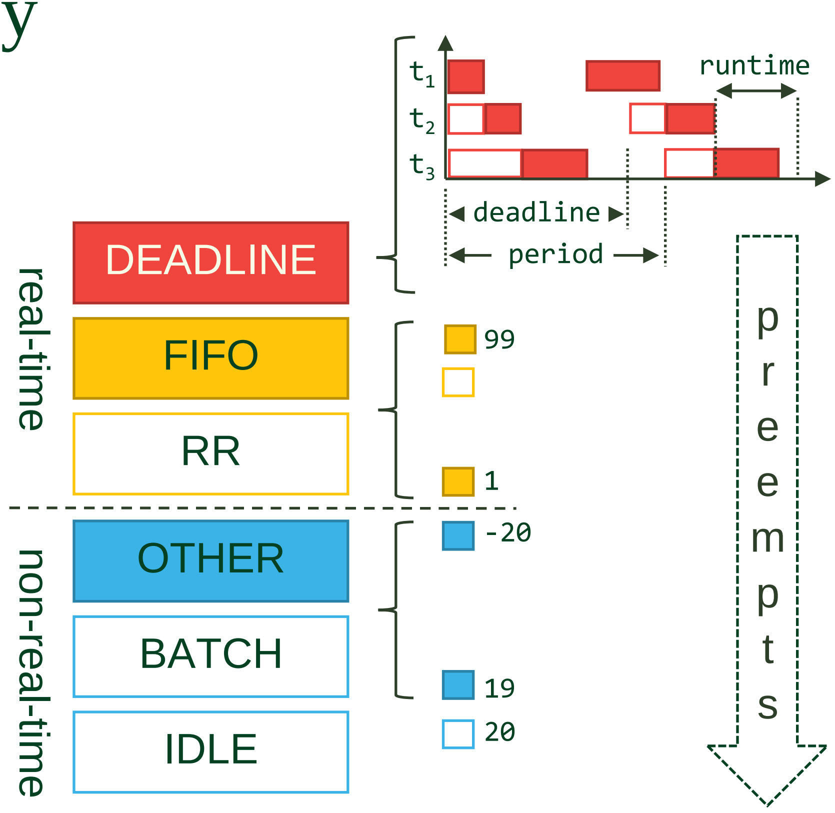
```
General Setup →
        Preemption Model (Fully Preemptible Kernel (Real-Time)) →
                (X) Fully Preemptible Kernel (Real-time)
```

Verify with:

```
# uname -a
Linux NI-PXIe-8880-03096F84 5.15.40-rt43-00095-g915fbd285457 #1 SMP
PREEMPT_RT Tue May 24 16:02:43 CDT 2022 x86_64 x86_64 x86_64 GNU/Linux
```

Scheduling policy and priority

- Identify RT workloads
- Assign scheduling policy & priority[1][2]:
  - SCHED_FIFO: 1-98 priority
  - SCHED_DEADLINE: runtime, deadline, period
- Also adjust RT priorities for:
  - IRQ threads, kernel threads, etc.
- Run everything else as:
  - SCHED_OTHER or lower

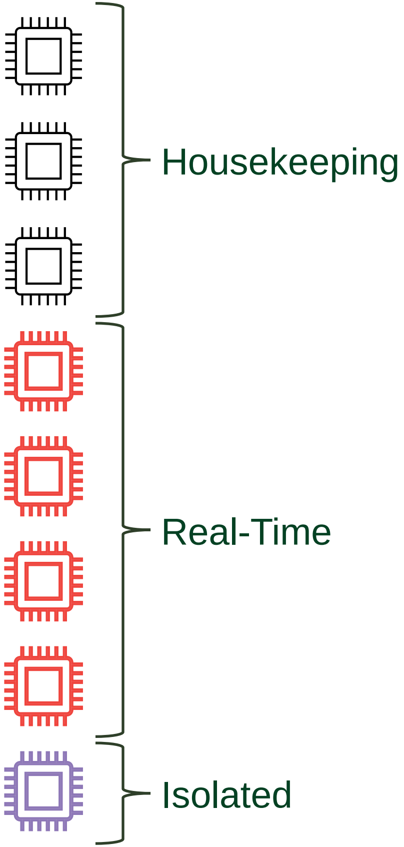


[1] https://man7.org/linux/man-pages/man1/chrt.1.html

[2] https://man7.org/linux/man-pages/man7/sched.7.html

CPU affinity

- Partition CPUs:
  - cpusets (cgroup v1), chrt, sched syscalls
- IRQ affinities:
  - `/proc/irq/default_smp_affinity`
  - `/proc/irq/*/smp_affinity`
- Kernel workqueue threads:
  - `find /sys/devices/virtual/workqueue -name "cpumask"`
- Isolate CPUs for sensitive real-time workloads:
  - kernel parameters: `isolcpus=7 nohz_full=7`
  - `CONFIG_NO_HZ_FULL`



RCU

CONFIG_RCU_NOCB_CPU = y

```
General Setup →
    RCU Subsystem →
        [*] Offload RCU callback processing from boot-selected CPUs
```

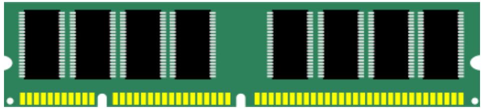
Control at boot via kernel parameters:

```
rcu_nocbs[=cpu-list]
rcu_nocb_poll
```

Verify with:

```
# ps ax | grep rcuop
 15 ?        S      0:21 [rcuop/0]
 28 ?        S      0:00 [rcuop/1]
…
```

Memory



Avoid memory allocations in real-time contexts:

```
~~malloc~~();
```

Consider resolving symbols at start-up:

```
# LD_BIND_NOW=1
# export LD_BIND_NOW
```

Lock pages in memory:

```
#include <sys/mman.h>
int mlockall(MCL_CURRENT | MCL_FUTURE);
```

Delay the vmstat timer far away into the future:

```
sysctl vm.stat_interval=999
```

Clock sources

Check the current clock source:

```
# cat /sys/devices/system/clocksource/clocksource0/current_clocksource
tsc
```

On Intel hardware pick TSC if available:

```
# cat /sys/devices/system/clocksource/clocksource0/available_clocksource
tsc hpet acpi_pm
```

Don’t forget about the trace clock:

```
# cat /sys/kernel/debug/tracing/trace_clock
[local] global counter uptime perf mono mono_raw boot x86-tsc
```

Power management

Disable CPU frequency scaling:

```
CONFIG_CPU_FREQ = N
```

Disable power management at boot via kernel parameters[1]:

```
intel_pstate=           [X86]
intel_idle.max_cstate=  [KNL,HW,ACPI,X86]
processor.max_cstate=   [HW,ACPI]
<GPU power management options>
```

Disable c-states at run-time:

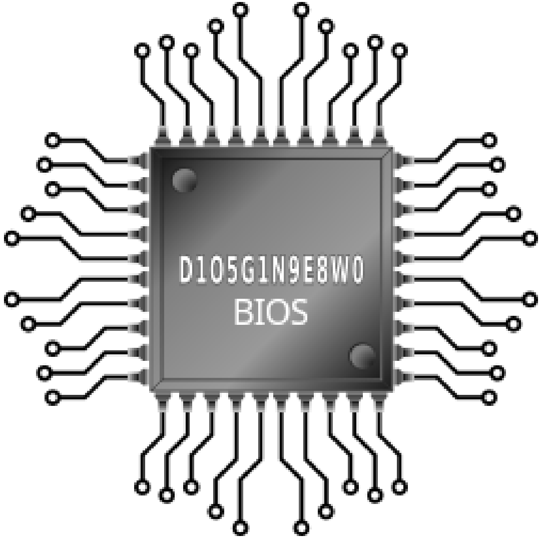
```
for CSTATE in /sys/devices/system/cpu/cpu*/cpuidle/state[^0]/disable;do
        echo 1 > $CSTATE
done
```

[1] https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/plain/Documentation/admin-guide/kernel-parameters.txt

Firmware configuration

Disable:

- Power management: P-states, C-states
- SMT (hyper-threading)[1]
- Intel Turbo Boost
- EDAC or configure to lowest functional level
- Unused peripherals and legacy hardware
- Vendor specific options that affect performance



[1] Core scheduling can be an alternative in kernels >= 5.14 (https://lwn.net/Articles/861251)

Safety Nets

© kevint3141, CC BY 2.0, via Wikimedia Commons

Removing safety nets

Disable RT throttling:

```
# echo -1 > /proc/sys/kernel/sched_rt_runtime_us
```

Disable clocksource watchdog:

```
tsc=nowatchdog
```

Disable soft-lockup detector:

```
nosoftlockup
```

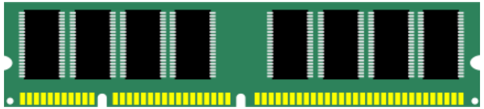
Disable both lockup detectors:

```
nowatchdog
```

Ignore corrected errors:

```
mce=ignore_ce
```

Memory



Don’t overcommit memory:

```
# echo 2 > /proc/sys/vm/overcommit_memory
# sysctl -w vm.overcommit_ratio=<ratio>
```

Prioritize processes to kill:

```
# echo 1000 > /proc/self/oom_score_adj
# echo -17 > /proc/12465/oom_adj
```

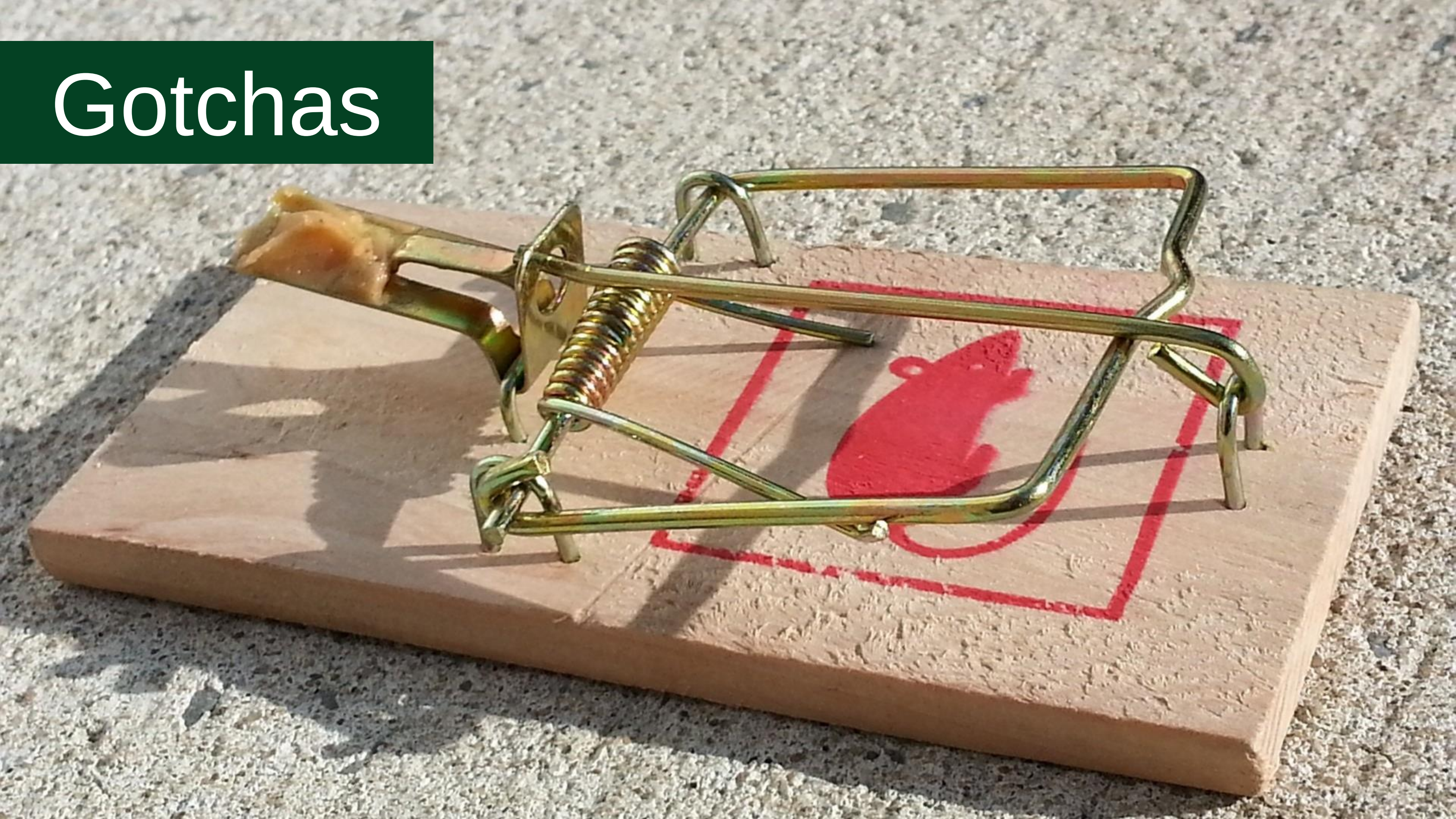
Decide what to do when out of memory:

```
# echo 1 > /proc/sys/vm/panic_on_oom
```

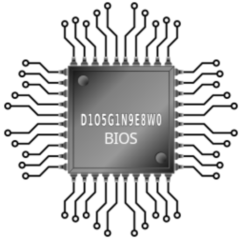
Security mitigations

```
mitigations=
           [X86,PPC,S390,ARM64] Control optional mitigations for
           CPU vulnerabilities.  This is a set of curated,
           arch-independent options, each of which is an
           aggregation of existing arch-specific options.

           off
                      Disable all optional CPU mitigations.  This
                      improves system performance, but it may also
                      expose users to several CPU vulnerabilities.
                      Equivalent to: nopti [X86,PPC]
                             kpti=0 [ARM64]
                             nospectre_v1 [X86,PPC]
                             nobp=0 [S390]
                             nospectre_v2 [X86,PPC,S390,ARM64]
                             spectre_v2_user=off [X86]
                             spec_store_bypass_disable=off [X86,PPC]
                             ssbd=force-off [ARM64]
                             l1tf=off [X86]
                             mds=off [X86]
                             tsx_async_abort=off [X86]
                             kvm.nx_huge_pages=off [X86]
                             no_entry_flush [PPC]
                             no_uaccess_flush [PPC]
```

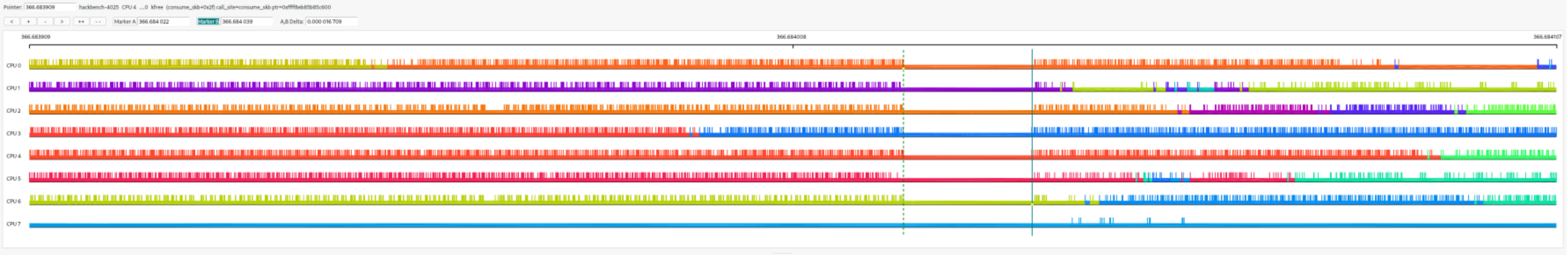
Gotchas



System Management Interrupts (SMI)



- High priority un-maskable hardware interrupts, handled in firmware
- Used for temperature management, legacy hardware emulation, hardware bugs etc.
- The OS is unaware of transitions to/from System Management Mode (SMM)
- x86 specific but other architectures have similar privileged modes:
  - e.g., Secure Monitor Mode on ARM
- https://wiki.linuxfoundation.org/realtime/documentation/howto/debugging/smi-latency/start

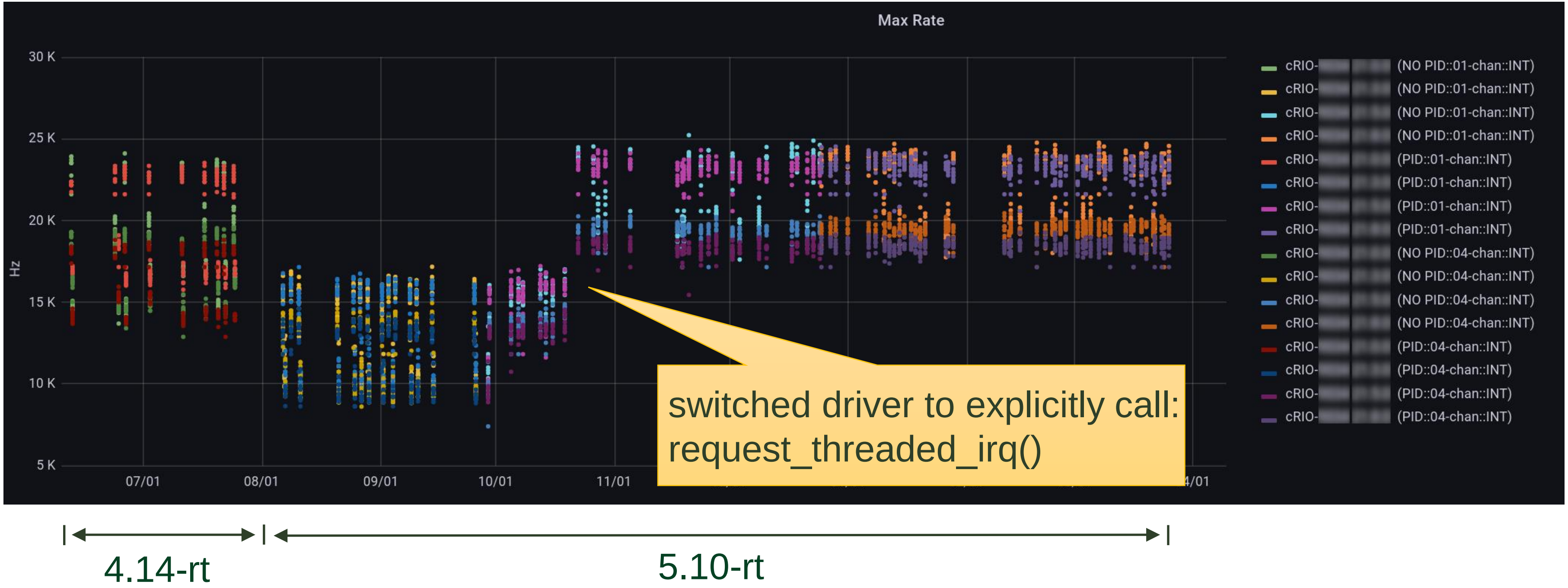


Pointer: 366.683909 hackbench-4025 CPU 4 ...0 kfree (consume_skb+0x2f) call_site=consume_skb ptr=0xffff8eb85b85c600

Marker A 366.684 022 Marker B 366.684 039 A,B Delta: 0.000 016 709 ni.com

Interrupts

Request threaded interrupt handlers:

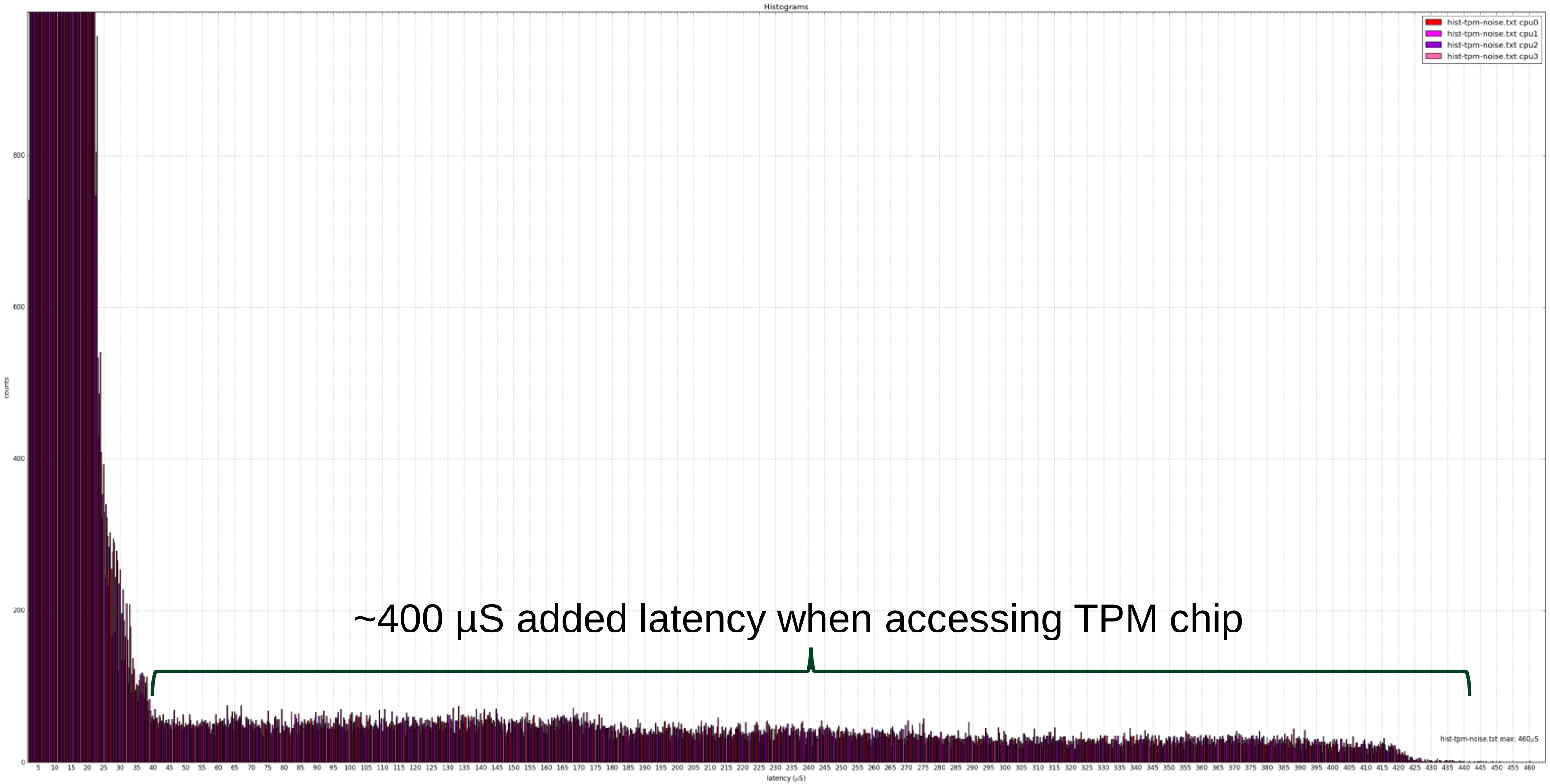


switched driver to explicitly call:
request_threaded_irq()

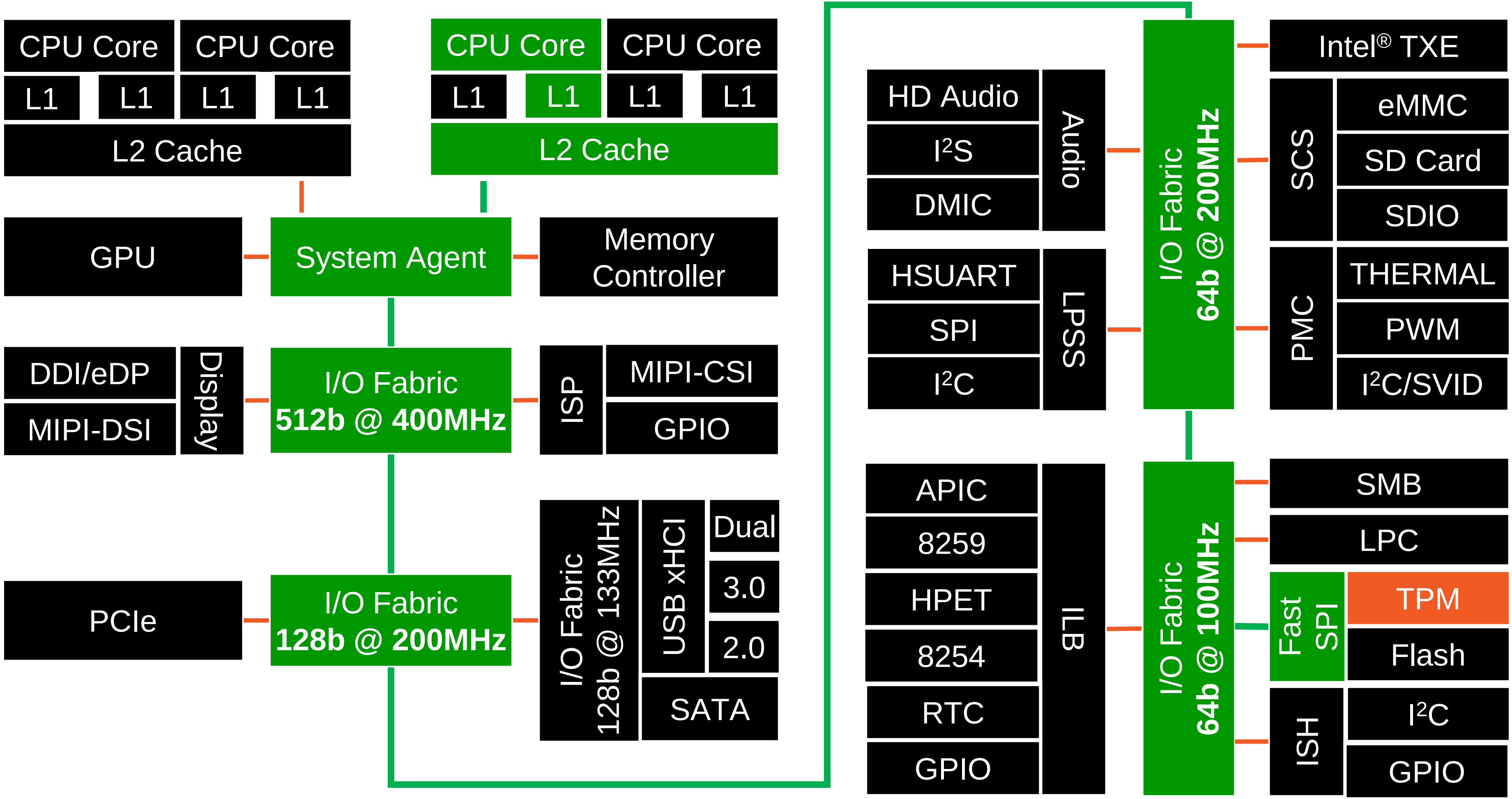
4.14-rt

5.10-rt

MMIO CPU stalls



~400 µS added latency when accessing TPM chip



CPU Core
CPU Core
L1
L1
L1
L1
L2 Cache

CPU Core
CPU Core
L1
L1
L1
L1
L2 Cache

GPU
System Agent
Memory Controller

DDI/eDP
MIPI-DSI
Display
I/O Fabric
512b @ 400MHz
ISP
MIPI-CSI
GPIO

PCIe
I/O Fabric
128b @ 200MHz
I/O Fabric
128b @ 133MHz
USB xHCI
Dual
3.0
2.0
SATA

HD Audio
I²S
DMIC
Audio

HSUART
SPI
I²C
LPSS

APIC
8259
HPET
8254
RTC
GPIO
ILB

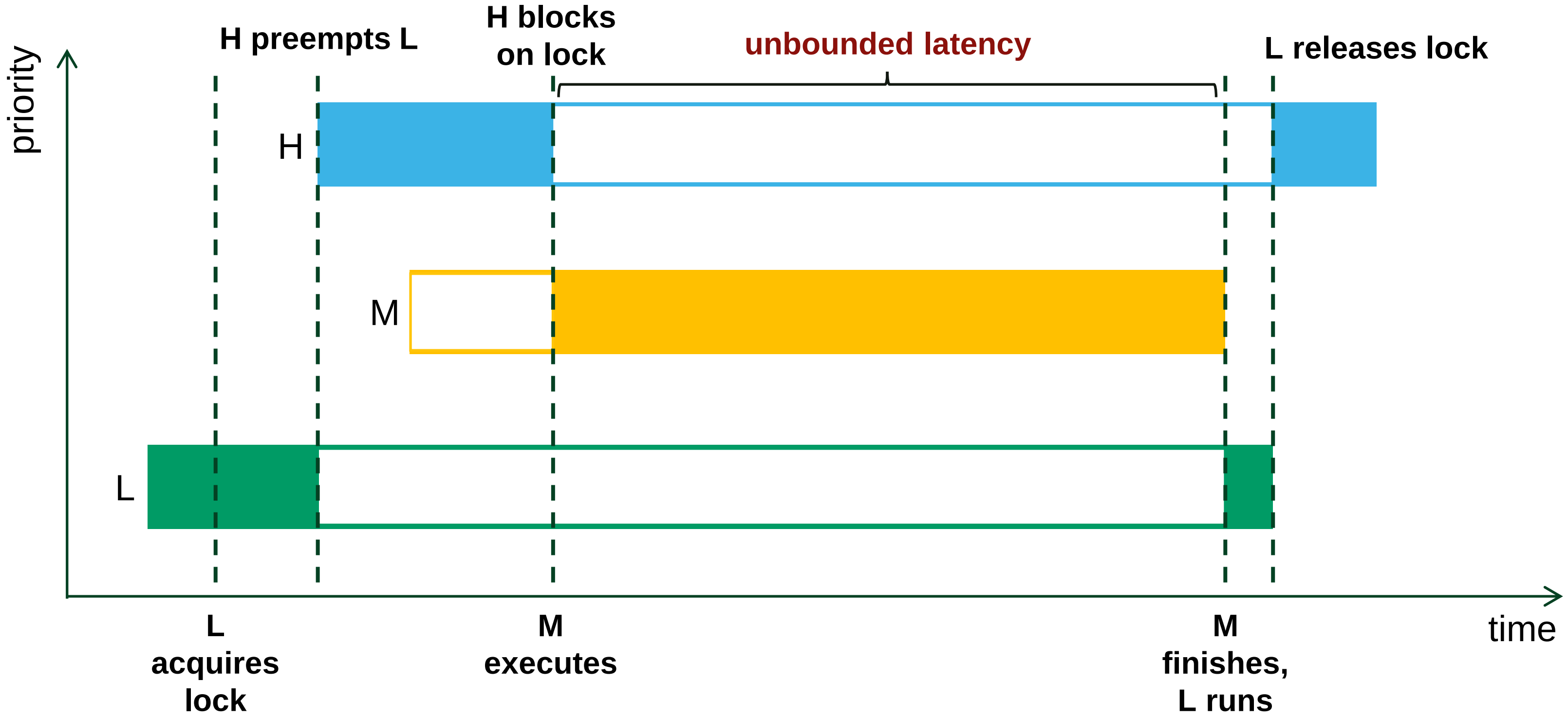
I/O Fabric
64b @ 200MHz

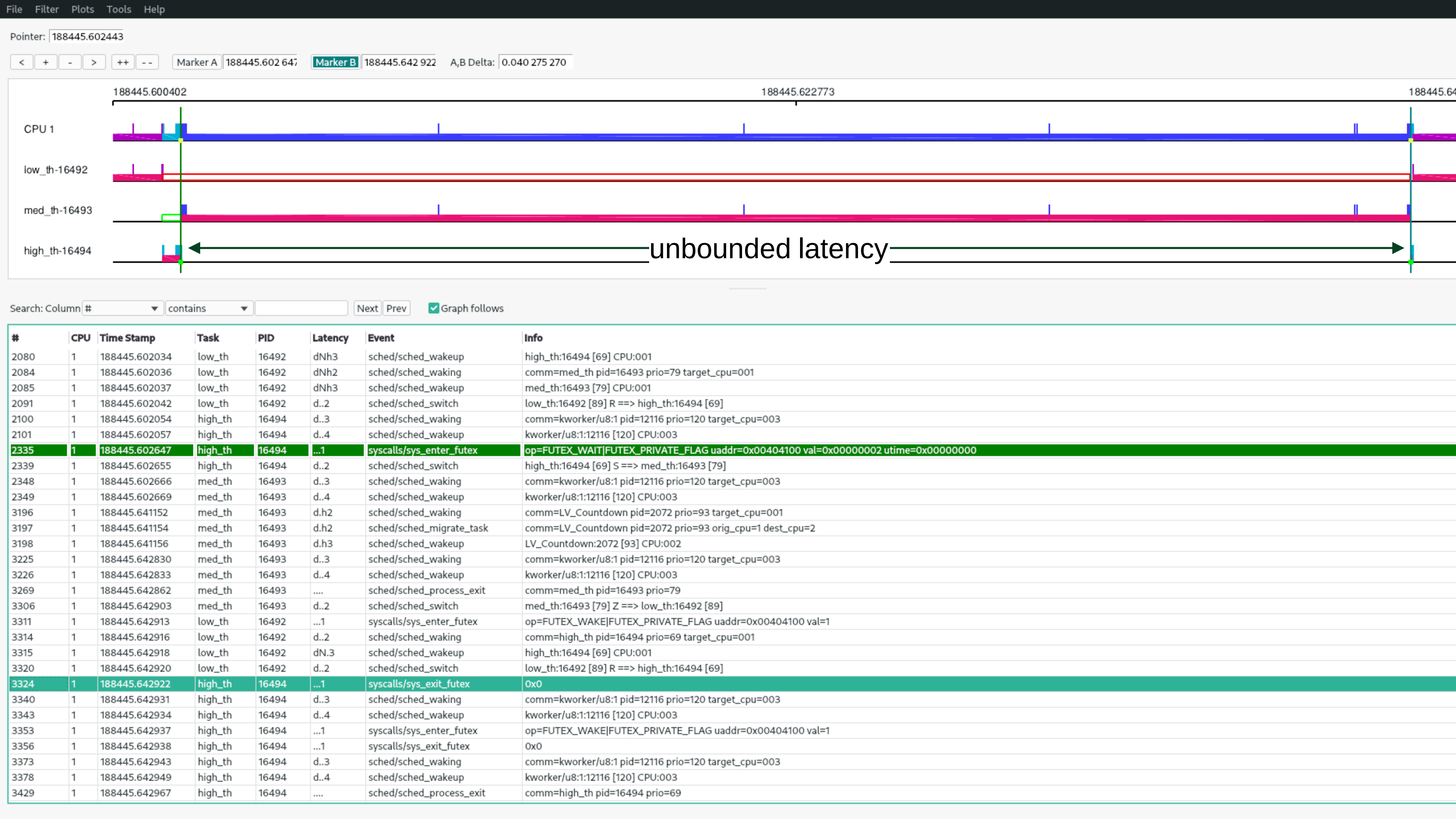
I/O Fabric
64b @ 100MHz

Intel® TXE
SCS
eMMC
SD Card
SDIO
PMC
THERMAL
PWM
I²C/SVID

SMB
LPC
Fast SPI
TPM
Flash
ISH
I²C
GPIO

Priority inversions





Pointer: 188445.602443

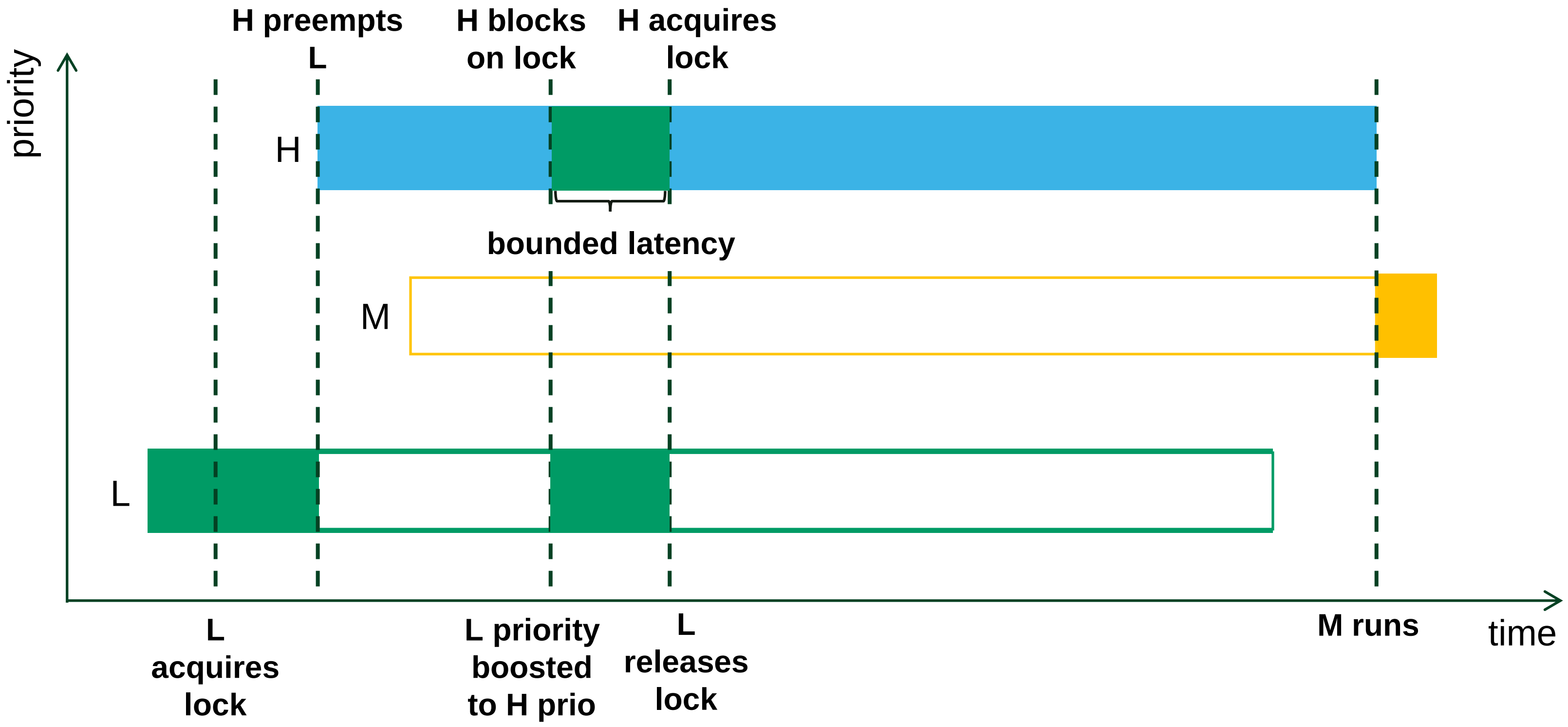
Marker A 188445.602 647 Marker B 188445.642 922 A,B Delta: 0.040 275 270

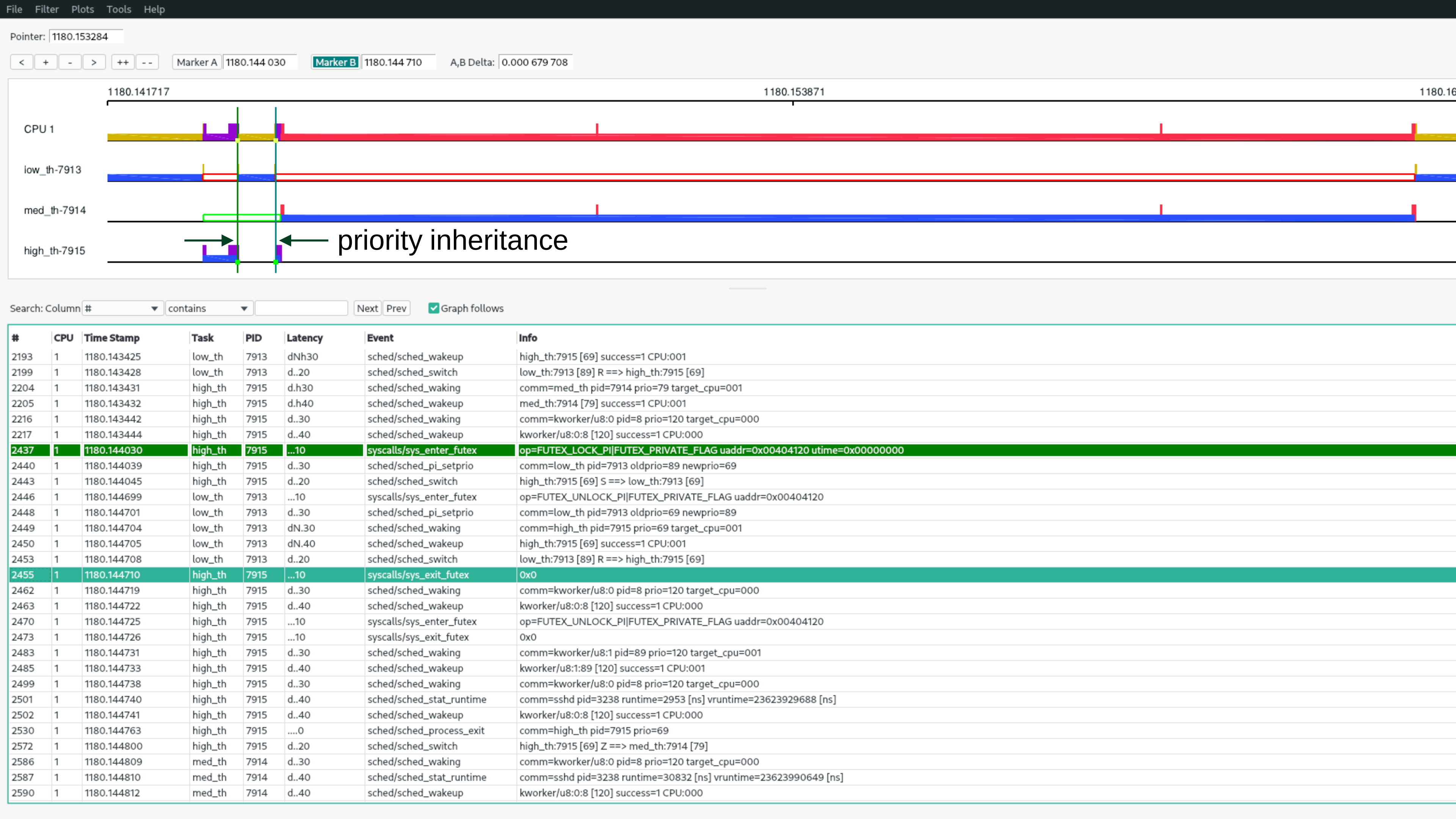
Search: Column # contains

☑ Graph follows

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
2080	1	188445.602034	low_th	16492	dNh3	sched/sched_wakeup	high_th:16494 [69] CPU:001
2084	1	188445.602036	low_th	16492	dNh2	sched/sched_waking	comm=med_th pid=16493 prio=79 target_cpu=001
2085	1	188445.602037	low_th	16492	dNh3	sched/sched_wakeup	med_th:16493 [79] CPU:001
2091	1	188445.602042	low_th	16492	d..2	sched/sched_switch	low_th:16492 [89] R ==> high_th:16494 [69]
2100	1	188445.602054	high_th	16494	d..3	sched/sched_waking	comm=kworker/u8:1 pid=12116 prio=120 target_cpu=003
2101	1	188445.602057	high_th	16494	d..4	sched/sched_wakeup	kworker/u8:1:12116 [120] CPU:003
2335	1	188445.602647	high_th	16494	...1	syscalls/sys_enter_futex	op=FUTEX_WAIT|FUTEX_PRIVATE_FLAG uaddr=0x00404100 val=0x00000002 utime=0x00000000
2339	1	188445.602655	high_th	16494	d..2	sched/sched_switch	high_th:16494 [69] S ==> med_th:16493 [79]
2348	1	188445.602666	med_th	16493	d..3	sched/sched_waking	comm=kworker/u8:1 pid=12116 prio=120 target_cpu=003
2349	1	188445.602669	med_th	16493	d..4	sched/sched_wakeup	kworker/u8:1:12116 [120] CPU:003
3196	1	188445.641152	med_th	16493	d.h2	sched/sched_waking	comm=LV_Countdown pid=2072 prio=93 target_cpu=001
3197	1	188445.641154	med_th	16493	d.h2	sched/sched_migrate_task	comm=LV_Countdown pid=2072 prio=93 orig_cpu=1 dest_cpu=2
3198	1	188445.641156	med_th	16493	d.h3	sched/sched_wakeup	LV_Countdown:2072 [93] CPU:002
3225	1	188445.642830	med_th	16493	d..3	sched/sched_waking	comm=kworker/u8:1 pid=12116 prio=120 target_cpu=003
3226	1	188445.642833	med_th	16493	d..4	sched/sched_wakeup	kworker/u8:1:12116 [120] CPU:003
3269	1	188445.642862	med_th	16493	....	sched/sched_process_exit	comm=med_th pid=16493 prio=79
3306	1	188445.642903	med_th	16493	d..2	sched/sched_switch	med_th:16493 [79] Z ==> low_th:16492 [89]
3311	1	188445.642913	low_th	16492	...1	syscalls/sys_enter_futex	op=FUTEX_WAKE|FUTEX_PRIVATE_FLAG uaddr=0x00404100 val=1
3314	1	188445.642916	low_th	16492	d..2	sched/sched_waking	comm=high_th pid=16494 prio=69 target_cpu=001
3315	1	188445.642918	low_th	16492	dN.3	sched/sched_wakeup	high_th:16494 [69] CPU:001
3320	1	188445.642920	low_th	16492	d..2	sched/sched_switch	low_th:16492 [89] R ==> high_th:16494 [69]
3324	1	188445.642922	high_th	16494	...1	syscalls/sys_exit_futex	0x0
3340	1	188445.642931	high_th	16494	d..3	sched/sched_waking	comm=kworker/u8:1 pid=12116 prio=120 target_cpu=003
3343	1	188445.642934	high_th	16494	d..4	sched/sched_wakeup	kworker/u8:1:12116 [120] CPU:003
3353	1	188445.642937	high_th	16494	...1	syscalls/sys_enter_futex	op=FUTEX_WAKE|FUTEX_PRIVATE_FLAG uaddr=0x00404100 val=1
3356	1	188445.642938	high_th	16494	...1	syscalls/sys_exit_futex	0x0
3373	1	188445.642943	high_th	16494	d..3	sched/sched_waking	comm=kworker/u8:1 pid=12116 prio=120 target_cpu=003
3378	1	188445.642949	high_th	16494	d..4	sched/sched_wakeup	kworker/u8:1:12116 [120] CPU:003
3429	1	188445.642967	high_th	16494	....	sched/sched_process_exit	comm=high_th pid=16494 prio=69

Priority inheritance





Pointer: 1180.153284

Marker A 1180.144 030

Marker B 1180.144 710

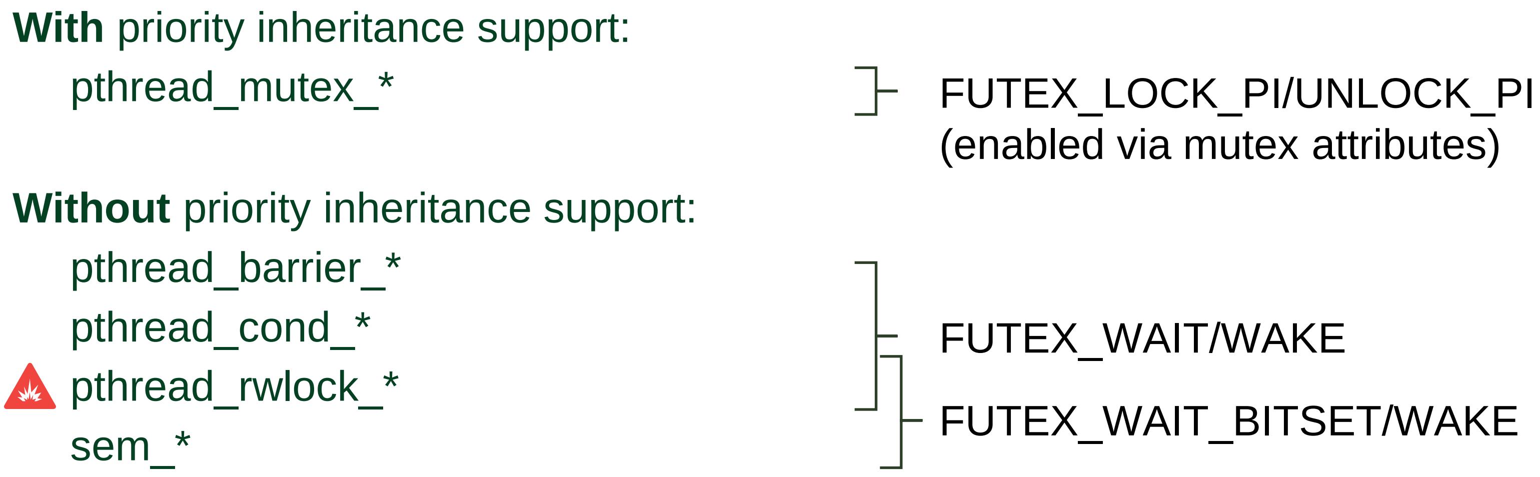
A,B Delta: 0.000 679 708

Search: Column # contains

☑ Graph follows

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
2193	1	1180.143425	low_th	7913	dNh30	sched/sched_wakeup	high_th:7915 [69] success=1 CPU:001
2199	1	1180.143428	low_th	7913	d..20	sched/sched_switch	low_th:7913 [89] R ==> high_th:7915 [69]
2204	1	1180.143431	high_th	7915	d.h30	sched/sched_waking	comm=med_th pid=7914 prio=79 target_cpu=001
2205	1	1180.143432	high_th	7915	d.h40	sched/sched_wakeup	med_th:7914 [79] success=1 CPU:001
2216	1	1180.143442	high_th	7915	d..30	sched/sched_waking	comm=kworker/u8:0 pid=8 prio=120 target_cpu=000
2217	1	1180.143444	high_th	7915	d..40	sched/sched_wakeup	kworker/u8:0:8 [120] success=1 CPU:000
2437	1	1180.144030	high_th	7915	...10	syscalls/sys_enter_futex	op=FUTEX_LOCK_PI|FUTEX_PRIVATE_FLAG uaddr=0x00404120 utime=0x00000000
2440	1	1180.144039	high_th	7915	d..30	sched/sched_pi_setprio	comm=low_th pid=7913 oldprio=89 newprio=69
2443	1	1180.144045	high_th	7915	d..20	sched/sched_switch	high_th:7915 [69] S ==> low_th:7913 [69]
2446	1	1180.144699	low_th	7913	...10	syscalls/sys_enter_futex	op=FUTEX_UNLOCK_PI|FUTEX_PRIVATE_FLAG uaddr=0x00404120
2448	1	1180.144701	low_th	7913	d..30	sched/sched_pi_setprio	comm=low_th pid=7913 oldprio=69 newprio=89
2449	1	1180.144704	low_th	7913	dN.30	sched/sched_waking	comm=high_th pid=7915 prio=69 target_cpu=001
2450	1	1180.144705	low_th	7913	dN.40	sched/sched_wakeup	high_th:7915 [69] success=1 CPU:001
2453	1	1180.144708	low_th	7913	d..20	sched/sched_switch	low_th:7913 [89] R ==> high_th:7915 [69]
2455	1	1180.144710	high_th	7915	...10	syscalls/sys_exit_futex	0x0
2462	1	1180.144719	high_th	7915	d..30	sched/sched_waking	comm=kworker/u8:0 pid=8 prio=120 target_cpu=000
2463	1	1180.144722	high_th	7915	d..40	sched/sched_wakeup	kworker/u8:0:8 [120] success=1 CPU:000
2470	1	1180.144725	high_th	7915	...10	syscalls/sys_enter_futex	op=FUTEX_UNLOCK_PI|FUTEX_PRIVATE_FLAG uaddr=0x00404120
2473	1	1180.144726	high_th	7915	...10	syscalls/sys_exit_futex	0x0
2483	1	1180.144731	high_th	7915	d..30	sched/sched_waking	comm=kworker/u8:1 pid=89 prio=120 target_cpu=001
2485	1	1180.144733	high_th	7915	d..40	sched/sched_wakeup	kworker/u8:1:89 [120] success=1 CPU:001
2499	1	1180.144738	high_th	7915	d..30	sched/sched_waking	comm=kworker/u8:0 pid=8 prio=120 target_cpu=000
2501	1	1180.144740	high_th	7915	d..40	sched/sched_stat_runtime	comm=sshd pid=3238 runtime=2953 [ns] vruntime=23623929688 [ns]
2502	1	1180.144741	high_th	7915	d..40	sched/sched_wakeup	kworker/u8:0:8 [120] success=1 CPU:000
2530	1	1180.144763	high_th	7915	0	sched/sched_process_exit	comm=high_th pid=7915 prio=69
2572	1	1180.144800	high_th	7915	d..20	sched/sched_switch	high_th:7915 [69] Z ==> med_th:7914 [79]
2586	1	1180.144809	med_th	7914	d..30	sched/sched_waking	comm=kworker/u8:0 pid=8 prio=120 target_cpu=000
2587	1	1180.144810	med_th	7914	d..40	sched/sched_stat_runtime	comm=sshd pid=3238 runtime=30832 [ns] vruntime=23623990649 [ns]
2590	1	1180.144812	med_th	7914	d..40	sched/sched_wakeup	kworker/u8:0:8 [120] success=1 CPU:000

Lack of priority inheritance support



With priority inheritance support:

- pthread_mutex_*

FUTEX_LOCK_PI/UNLOCK_PI
(enabled via mutex attributes)

Without priority inheritance support:

- pthread_barrier_*
- pthread_cond_*
- ⚠ pthread_rwlock_*
- sem_*

FUTEX_WAIT/WAKE

FUTEX_WAIT_BITSET/WAKE

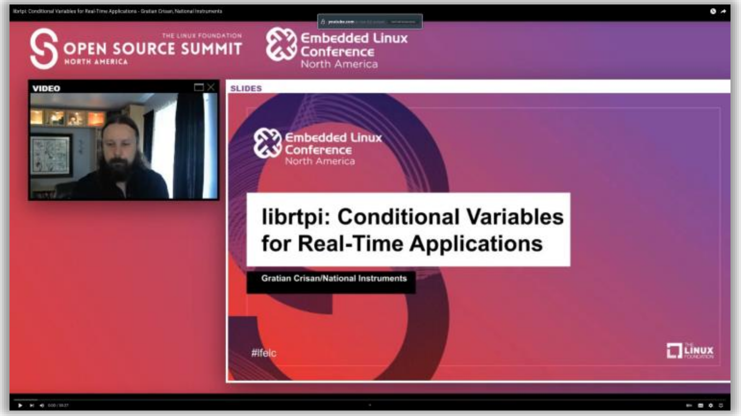
No way of setting priority inheritance attribute on std::mutex()

Partial solution

- librtpi[1][2]

[1] https://github.com/dvhart/librtpi
[2] https://github.com/gratian/librtpi

- PI mutex and condvar



librtpi: Conditional Variables for Real-Time Applications - Gratian Crisan, National Instruments

librtpi: Conditional Variables for Real-Time Applications

Gratian Crisan/National Instruments

- Taking suggestions for RT-aware libraries implementing POSIX locks

Interrupt priority inversions

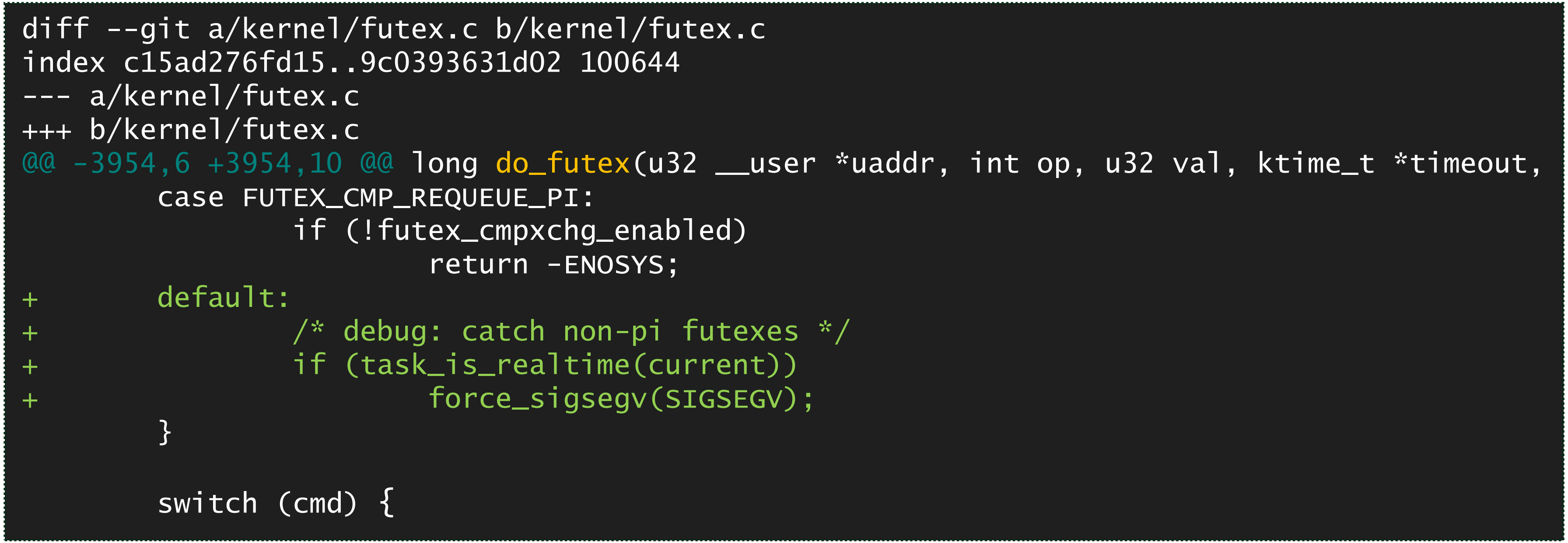
Context:

- Watchdog functionality implemented in a CPLD hanging of a I²C bus
- It can be configured to fire an interrupt (as opposed to a straight reset)

Behavior:

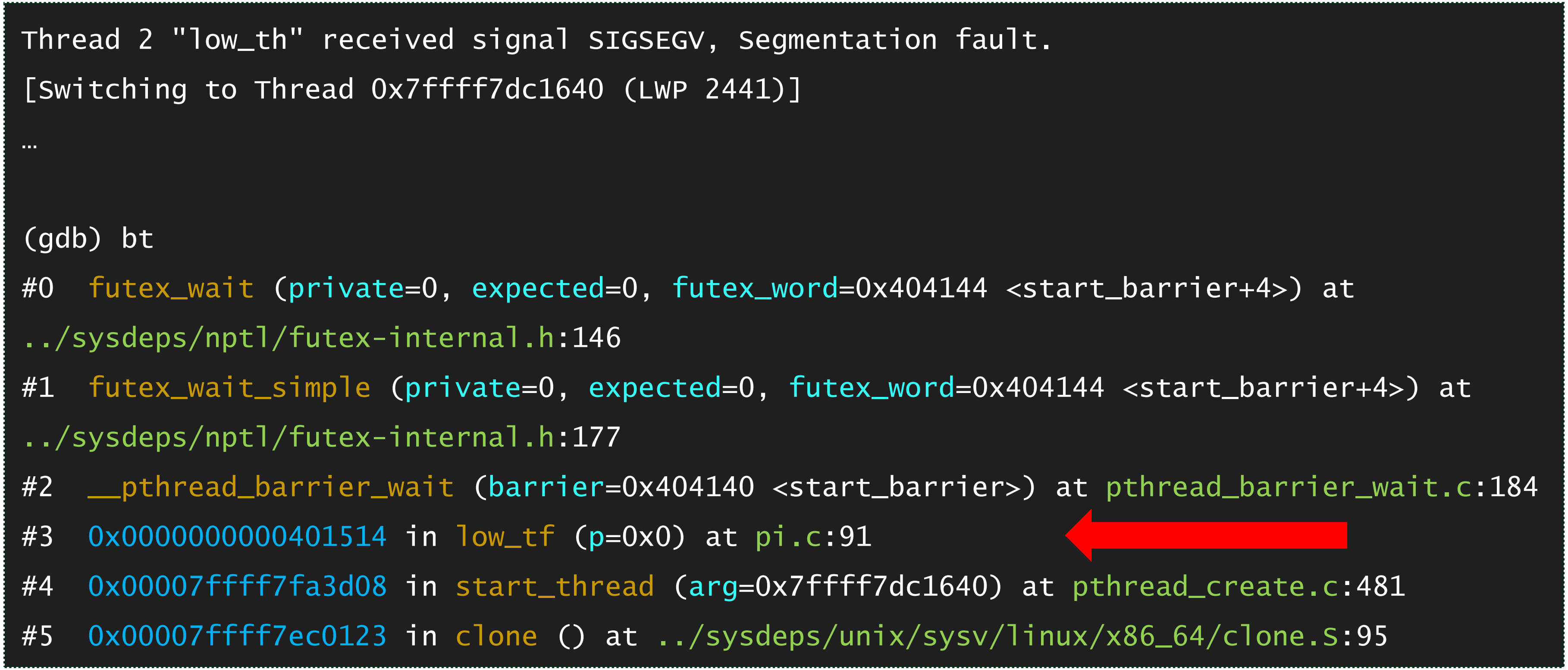
- High priority watchdog interrupt fires
- To acknowledge the interrupt slow I²C transfers need to happen
- I²C interrupt has low priority
- Some unrelated mid-priority irq preempts the I²C interrupt

Futex “trick”



```
diff --git a/kernel/futex.c b/kernel/futex.c
index c15ad276fd15..9c0393631d02 100644
--- a/kernel/futex.c
+++ b/kernel/futex.c
@@ -3954,6 +3954,10 @@ long do_futex(u32 __user *uaddr, int op, u32 val, ktime_t *timeout,
        case FUTEX_CMP_REQUEUE_PI:
                if (!futex_cmpxchg_enabled)
                        return -ENOSYS;
+       default:
+               /* debug: catch non-pi futexes */
+               if (task_is_realtime(current))
+                       force_sigsegv(SIGSEGV);
        }

        switch (cmd) {
```

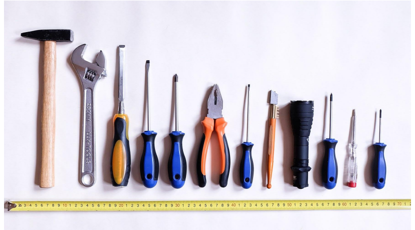
Futex “trick” cont’d



```
Thread 2 "low_th" received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7ffff7dc1640 (LWP 2441)]
…

(gdb) bt
#0  futex_wait (private=0, expected=0, futex_word=0x404144 <start_barrier+4>) at
../sysdeps/nptl/futex-internal.h:146
#1  futex_wait_simple (private=0, expected=0, futex_word=0x404144 <start_barrier+4>) at
../sysdeps/nptl/futex-internal.h:177
#2  __pthread_barrier_wait (barrier=0x404140 <start_barrier>) at pthread_barrier_wait.c:184
#3  0x0000000000401514 in low_tf (p=0x0) at pi.c:91
#4  0x00007ffff7fa3d08 in start_thread (arg=0x7ffff7dc1640) at pthread_create.c:481
#5  0x00007ffff7ec0123 in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:95
```

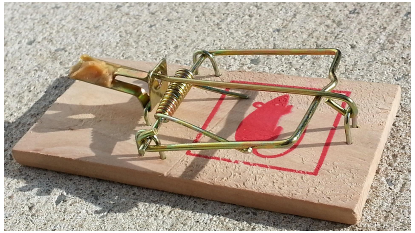
Summary

- Real-Time tools
- Tuning knobs
- Removing safety nets
- Gotchas to avoid



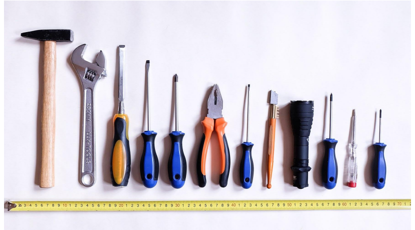
© Franklin Heijnen, CC BY-SA 2.0, via Wikimedia Commons

© kevint3141, CC BY 2.0, via Wikimedia Commons



Summary

- Real-Time tools
- Tuning knobs
- Removing safety nets
- Gotchas to avoid



© Franklin Heijnen, CC BY-SA 2.0, via Wikimedia Commons

© kevint3141, CC BY 2.0, via Wikimedia Commons