
NATIONAL
INSTRUMENTS™

RT Troubles

Lessons Learned & Open Questions

Grațian Crișan

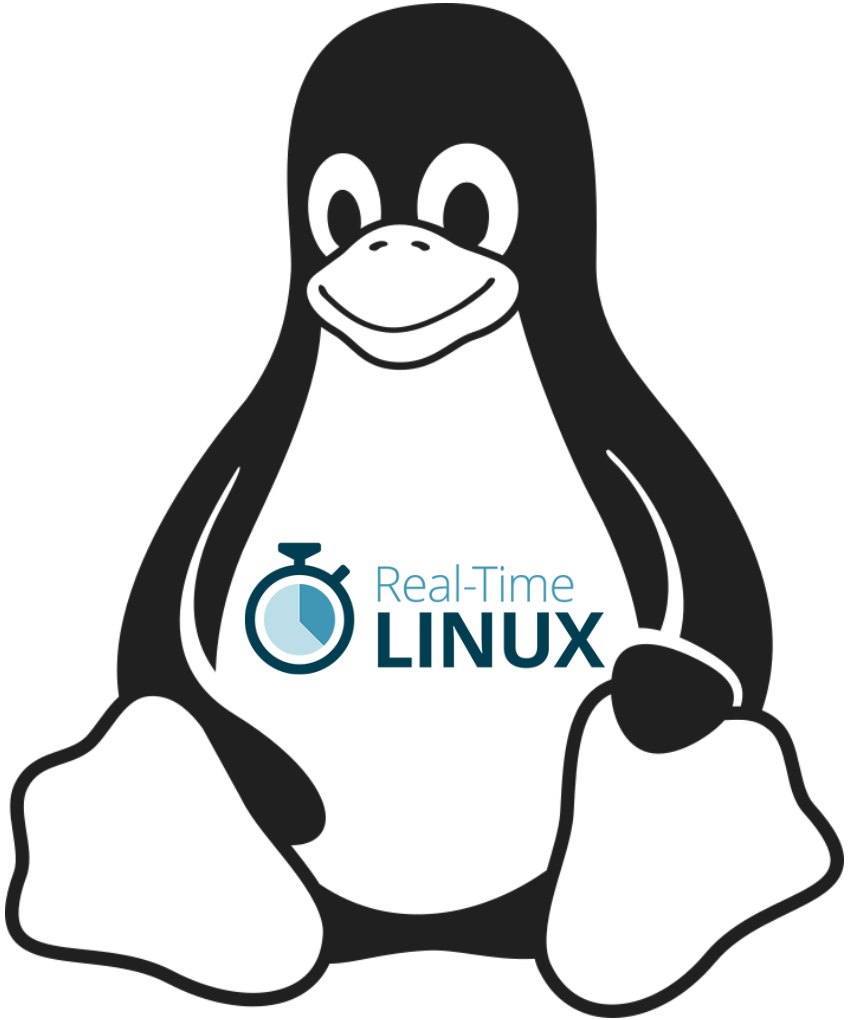
RT-Summit 2017

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

NATIONAL INSTRUMENTS

Context

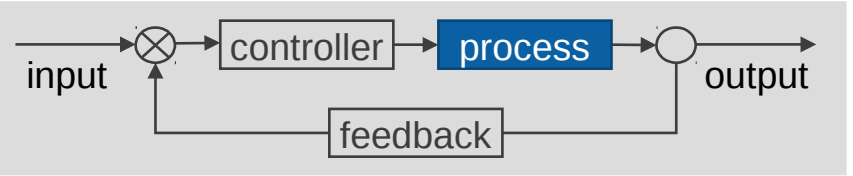
- National Instruments
  - Makes hardware & software for test, measurement and control
- Real-Time OS group
  - Using PREEMPT_RT for 6+ years
  - ARM and Intel x86_64 architectures
  - Embedded CPU + FPGA products
  - OpenEmbedded / Yocto – based distribution
- Disclaimers
  - Work by multiple people
  - Intentionally picked problems with ugly hacks
  - Some data might be stale by now

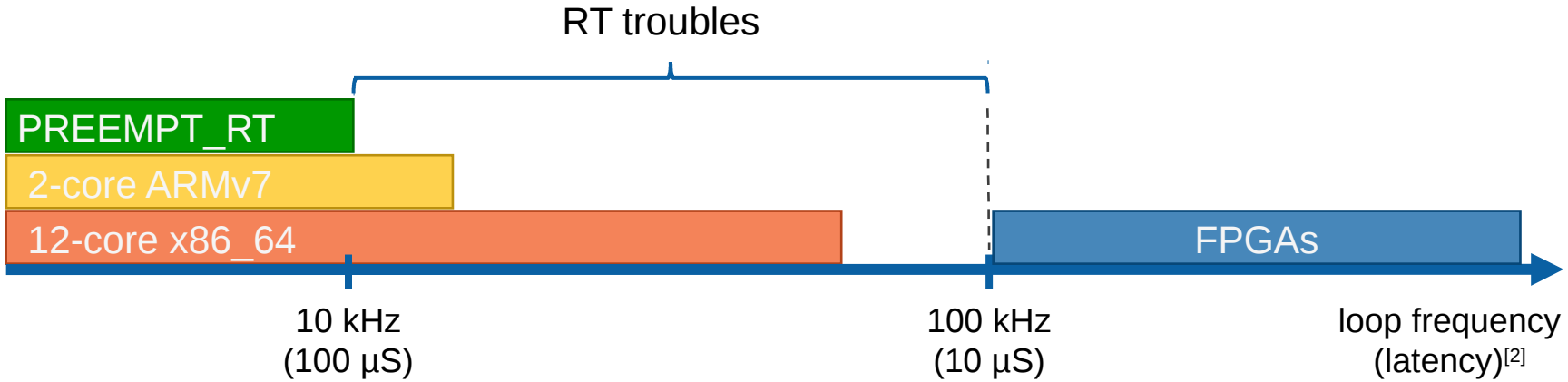


Agenda

- problem_space();
- do {
  - rt_trouble_area();
  - discussion();
- } while (topics && time);

Problem space[1]





[1] our current use cases
[2] in general wake-up latency accounts for majority of control loop latency

NATIONAL INSTRUMENTS

RT Trouble #1

How bumping an Ethernet cable can ruin ~~your day~~ RT and some TPM troubles

Symptoms

- CPU appears stalled in a MMIO read instruction for hundreds of µS
- Timer interrupts get delivered late even though the interrupts are enabled
- Initially discovered in e1000 / e1000e network drivers[1]
  - by (accidentally) bumping into an Ethernet cable during a cyclictest run
- Recently found in TPM driver[2]
  - by (intentionally) accessing the TPM chip while running cyclictest

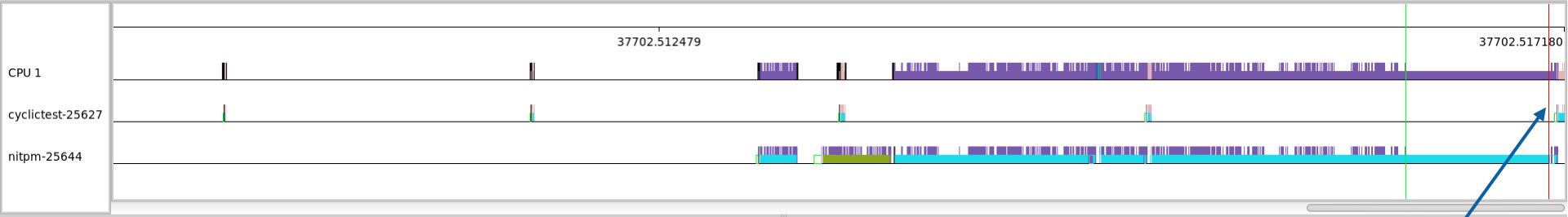
[1] drivers/net/ethernet/intel/e1000e/*
[2] drivers/char/tpm/tpm_tis.c

NATIONAL INSTRUMENTS

kernelshark(trace.dat)

File Filter Plots Capture Help

Pointer: 37702.514296 Cursor: 37702.517099 MarkerA: 37702.516354 MarkerB: 37702.517099 A,B Delta: 0.000745

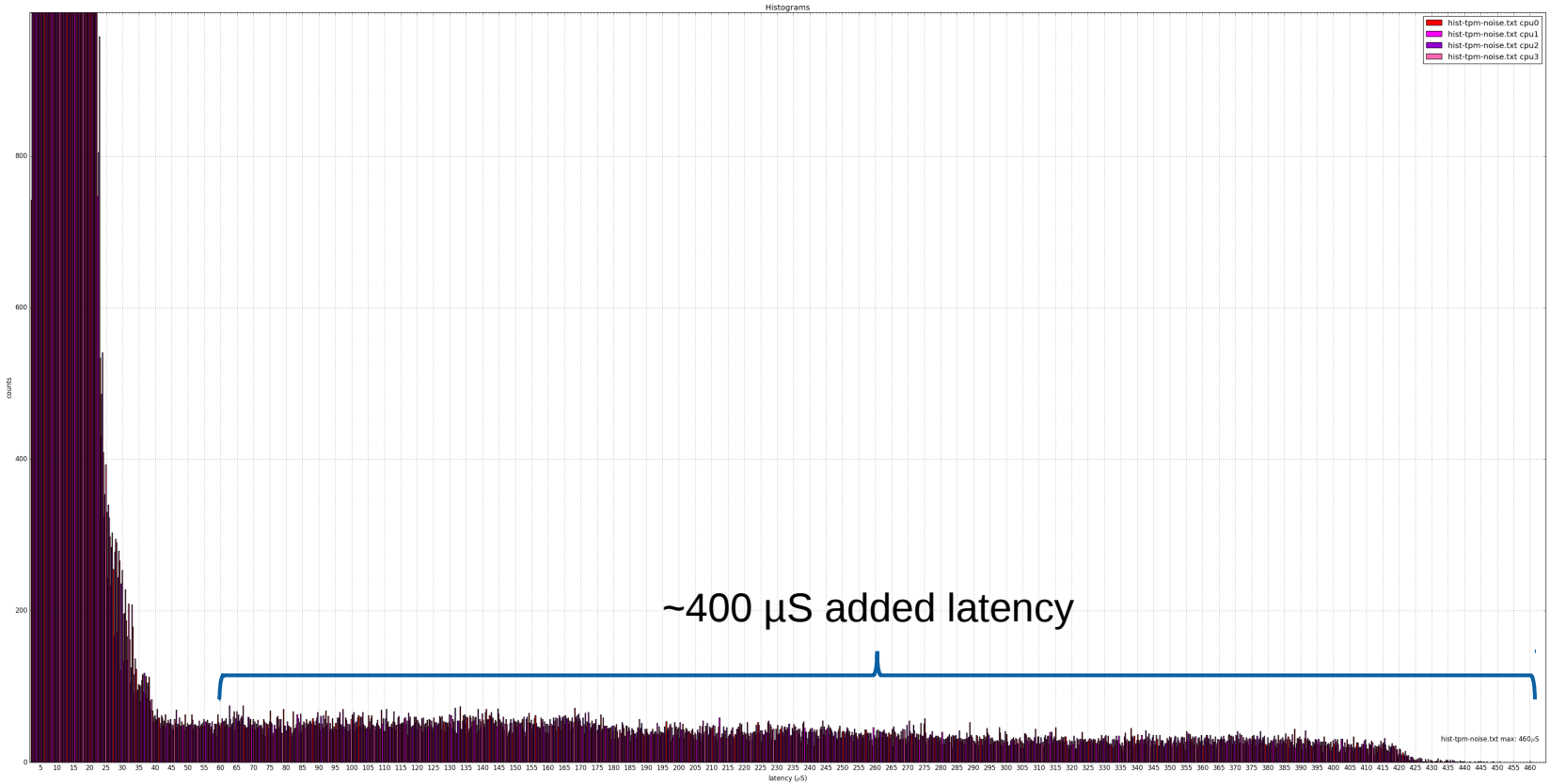


Page 1 Search: Column: # contains graph follows

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
6081	1	37702.516307	nitpm	25644	...10	sys_exit_open	0x5
6082	1	37702.516312	nitpm	25644	d...0	page_fault_user	address=rt6_uncached_list ip=rt6_uncached_list error_code=0x6
6083	1	37702.516315	nitpm	25644	0	mm_page_alloc	page=0xffffea000015102d pfn=1380397 order=0 migratetype=1 gfp_flags=GFP_HIGHUSER_MOVABLE|__GFP_ZERO
6084	1	37702.516352	nitpm	25644	0	sys_enter	NR 1 (5, 244d1a8, 3f, 86f, 244d19c, 1000)
6085	1	37702.516354	nitpm	25644	...10	sys_enter_write	fd: 0x00000005, buf: 0x0244d1a8, count: 0x0000003f
6086	1	37702.517099	nitpm	25644	d.h.0	local_timer_entry	vector=239
6087	1	37702.517109	nitpm	25644	d.h10	hrtimer_cancel	hrtimer=0xffffc9000600fe58
6088	1	37702.517113	nitpm	25644	d.h.0	hrtimer_expire_entry	hrtimer=0xffffc9000600fe58 now=23647271504381 function=hrtimer_wakeup/0x0
6089	1	37702.517116	nitpm	25644	d.h10	sched_waking	comm=cyclictest pid=25627 prio=1 target_cpu=001
6090	1	37702.517125	nitpm	25644	dNh20	sched_wakeup	cyclictest:25627 [1] success=1 CPU:001
6091	1	37702.517126	nitpm	25644	dNh.0	hrtimer_expire_exit	hrtimer=0xffffc9000600fe58
6092	1	37702.517127	nitpm	25644	dNh.0	write_msr	6e0, value 224a4d617da3
6093	1	37702.517128	nitpm	25644	dNh.0	local_timer_exit	vector=239
6094	1	37702.517130	nitpm	25644	dN.10	rcu_utilization	Start context switch
6095	1	37702.517131	nitpm	25644	dN.10	rcu_utilization	End context switch
6096	1	37702.517134	nitpm	25644	dN.20	sched_stat_runtime	comm=nitpm pid=25644 runtime=1310302 [ns] vruntime=949236367768 [ns]
6097	1	37702.517139	nitpm	25644	d..20	sched_switch	nitpm:25644 [120] R ==> cyclictest:25627 [1]
6098	1	37702.517140	nitpm	25644	d..20	tlb_flush	pages=-1 reason=flush on task switch (0)
6099	1	37702.517142	nitpm	25644	d..20	x86_fpu_regs_deactivated	x86/fpu: 0xffff88017906a280 fpregs_active: 0 fpstate_active: 1 counter: 2 xfeatures: 1b xcomp_bv: 800000000000001b
6100	1	37702.517143	nitpm	25644	d..20	x86_fpu_regs_activated	x86/fpu: 0xffff88016dfa2f00 fpregs_active: 1 fpstate_active: 1 counter: 7 xfeatures: 1b xcomp_bv: 800000000000001b
6101	1	37702.517144	nitpm	25644	d..20	x86_fpu_regs_activated	x86/fpu: 0xffff88016dfa2f00 fpregs_active: 1 fpstate_active: 1 counter: 7 xfeatures: 1b xcomp_bv: 800000000000001b
6102	1	37702.517144	nitpm	25644	d..20	write_msr	c0000100, value 7fd681efa700
6103	1	37702.517147	cyclictest	25627	0	sys_exit	NR 230 = 0
6104	1	37702.517148	cyclictest	25627	...10	sys_exit_clock_nanosleep	0x0
6105	1	37702.517169	cyclictest	25627	0	sys_enter	NR 1 (5, 7fd681efa300, 21, 0, 0, 21)
6106	1	37702.517170	cyclictest	25627	...10	sys_enter_write	fd: 0x00000005, buf: 0x7fd681efa300, count: 0x00000021
6107	1	37702.517176	cyclictest	25627	1	print	tracing_mark_write: hit latency threshold (352 > 100)
6108	1	37702.517178	cyclictest	25627	0	sys_exit	NR 1 = 33

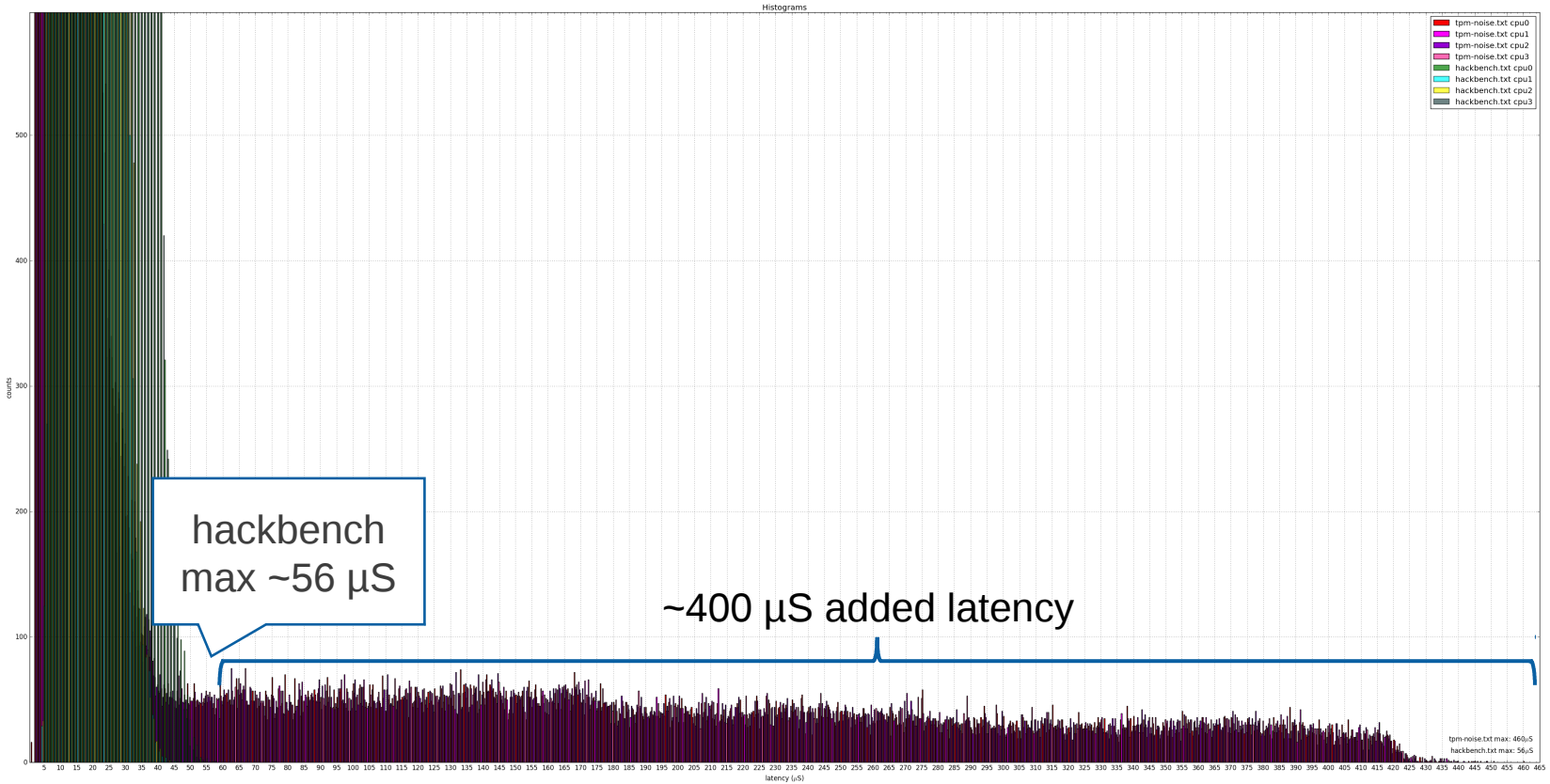
late timer irq

Cyclictest histogram with TPM load



NATIONAL INSTRUMENTS

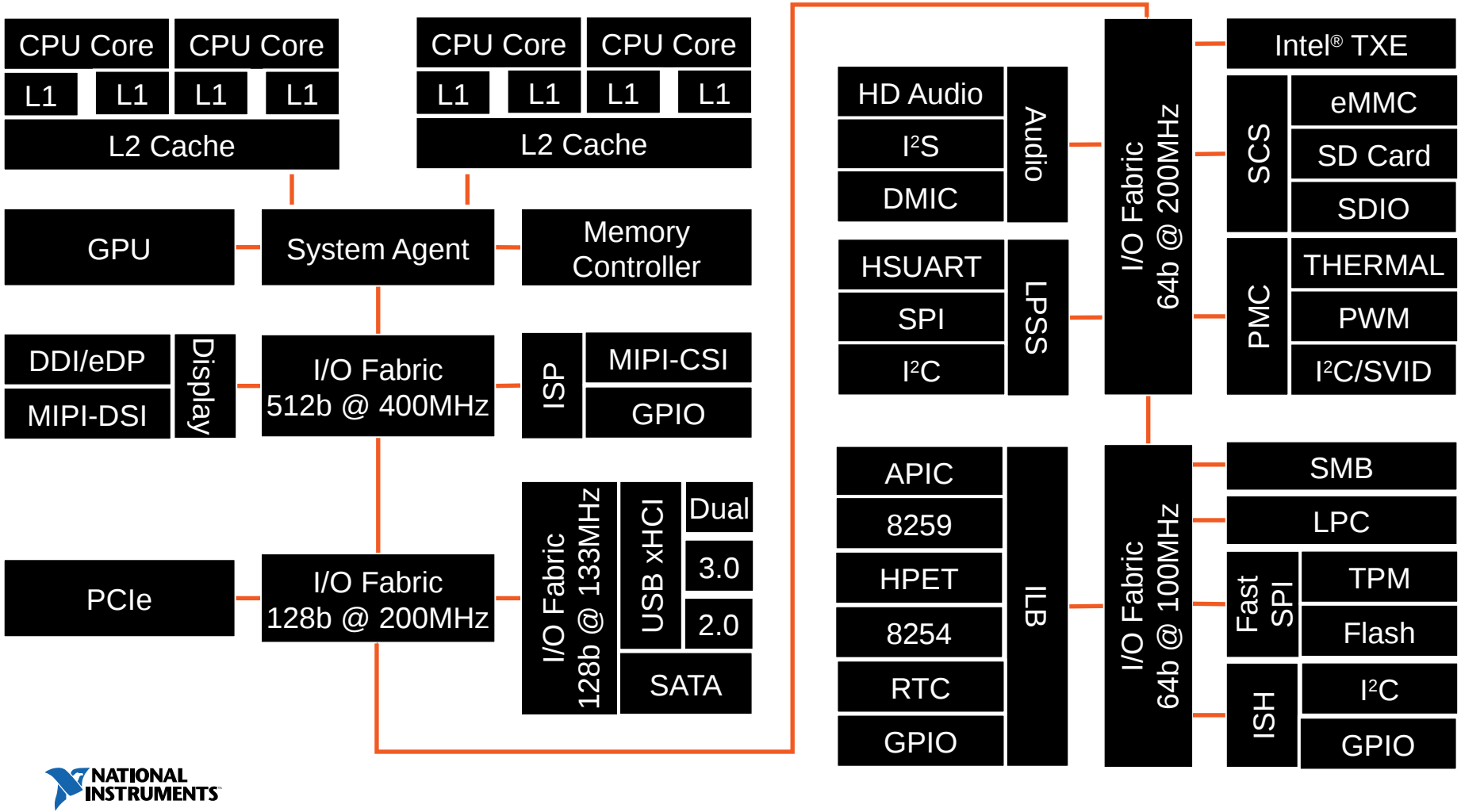
Cyclictest histogram with TPM load



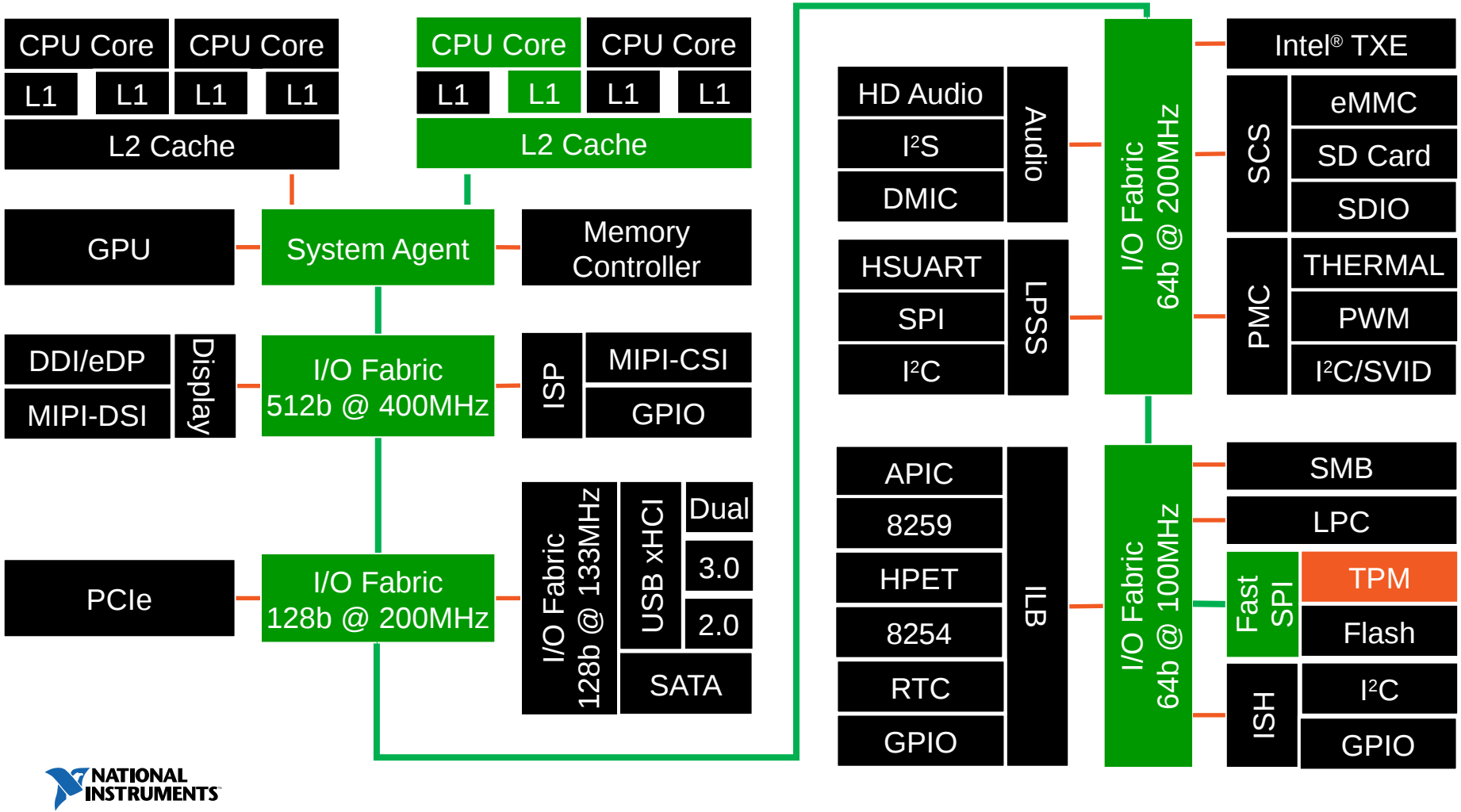
NATIONAL INSTRUMENTS

Why this happens

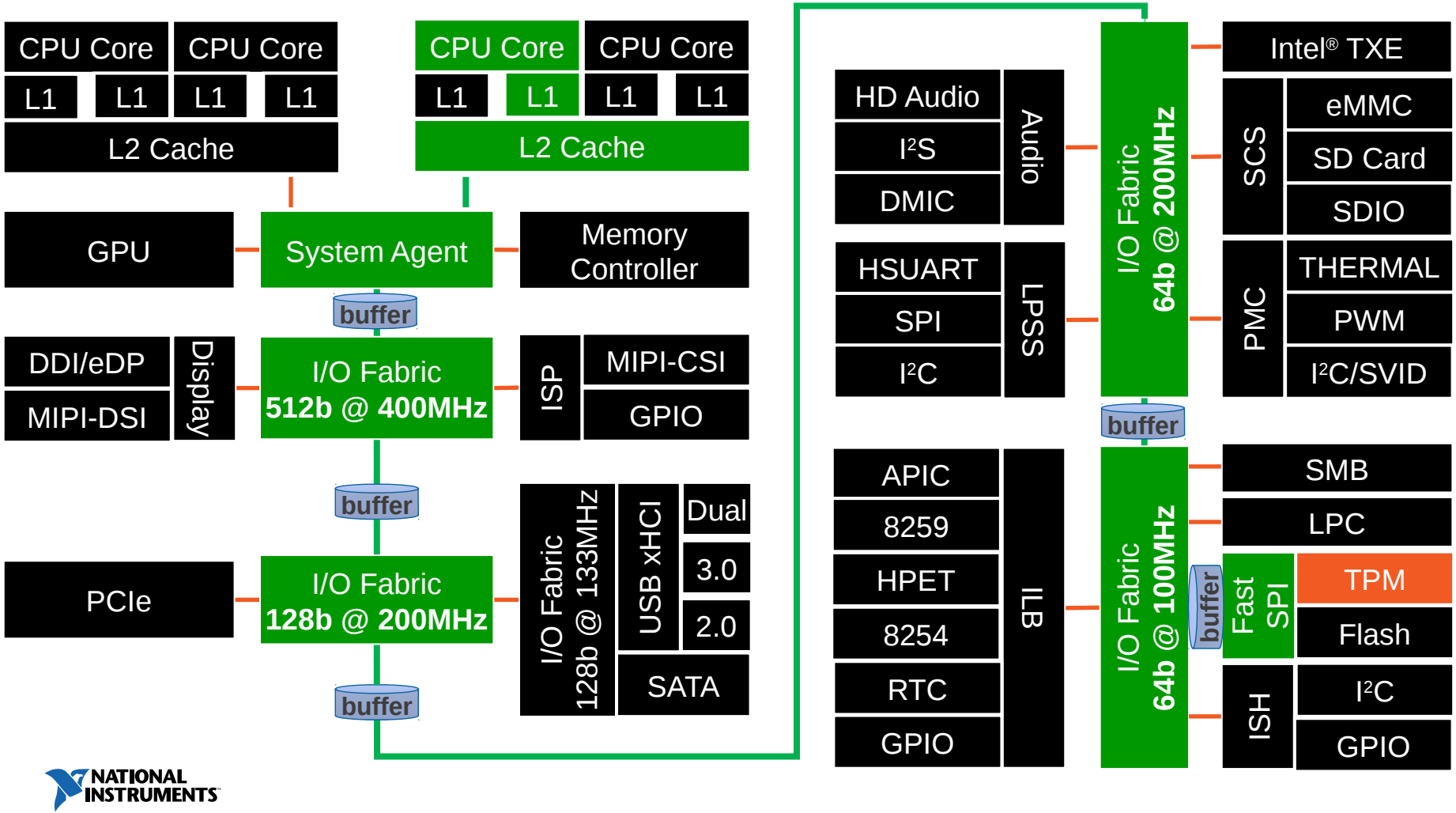
- CPU can write exponentially faster than the I/O device can sink
- Writes are buffered between the CPU and I/O
- Generally not an issue if the number of writes is small
- A MMIO read has to wait for every write to drain resulting in hundreds of µS latency spikes



NATIONAL INSTRUMENTS



NATIONAL INSTRUMENTS



NATIONAL INSTRUMENTS

e1000/e1000e hack[1]

```
--- a/drivers/net/ethernet/intel/e1000e/mac.c
+++ b/drivers/net/ethernet/intel/e1000e/mac.c
@@ -353,6 +353,7 @@ void e1000e_update_mc_addr_list_generic(struct e1000_hw
    /* replace the entire MTA table */
    for (i = hw->mac.mta_reg_count - 1; i >= 0; i--)
        E1000_WRITE_REG_ARRAY(hw, E1000_MTA, i,
                              hw->mac.mta_shadow[i]);
+   E1000_WR_DELAY();
    e1e_flush();
```

```
+#ifdef CONFIG_E1000_DELAY
+#define E1000_WR_DELAY() usleep_range(50, 100)
+#else …
```

[1] https://www.spinics.net/lists/linux-rt-users/msg14077.html

tpm_tis patch[1]

```
--- a/drivers/char/tpm/tpm_tis.c
+++ b/drivers/char/tpm/tpm_tis.c
@@ -103,7 +128,7 @@ static int tpm_tcg_write_bytes(struct tpm_tis_data
*data
    struct tpm_tis_tcg_phy *phy = to_tpm_tis_tcg_phy(data);
    while (len--)
-       iowrite8(*value++, phy->iobase + addr);
+       tpm_tis_iowrite8(*value++, phy->iobase, addr);
    return 0;
```

```
+static inline void tpm_tis_iowrite8(u8 b, void __iomem *iobase,…)
+{
+   iowrite8(b, iobase + addr);
+   tpm_tis_flush(iobase);
+}
```

#ifdef CONFIG_PREEMPT_RT_FULL
ioread8(iobase + TPM_ACCESS(0));

[1] https://lkml.org/lkml/2017/8/15/663

Discussion

- Preface: we know this is a hardware problem that might not have a good software solution.
- Is it possible / likely on other archs?
- Other drivers you know of that have this problem?

Discussion

- Can we detect this access pattern (at runtime)?
- Any way to track I/O buffer states?

Discussion

- Other ways to throttle MMIO stores?
- Is there a more general solution possible?
  - Adding a load (with exceptions) in iowriteN()/writeX() macros for PREEMPT_RT?[1]

[1] https://lkml.org/lkml/2017/8/7/550

NATIONAL INSTRUMENTS

RT Trouble #2

Concurrent hrtimer expirations from low priority threads

~~RT Trouble #2~~ – Thank You!

Concurrent hrtimer expirations from low priority threads

Symptoms

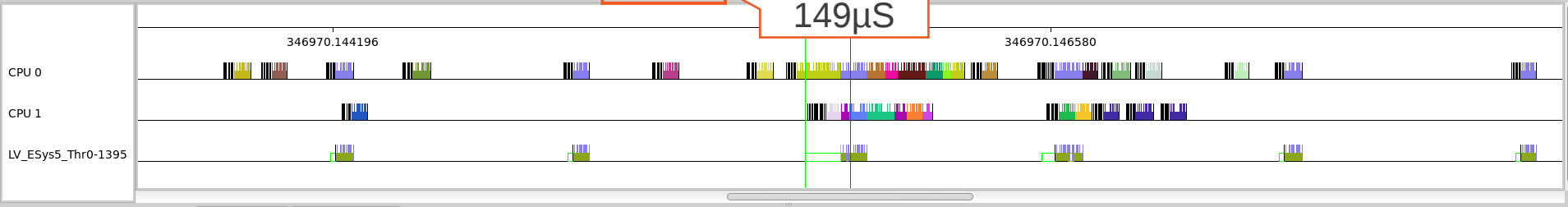
- Multiple timed sleeps or timeouts coming from SCHED_OTHER threads can stack up to large latencies for RT threads
- It is not just clock_nanosleep()
  - lots of other things that use hrtimers e.g. futexes (with timeouts)

kernelshark(trace3.dat.1)

File Filter Plots Capture Help

Pointer: 346970.143556 Cursor: 346970.145917 MarkerA: 346970.145767 MarkerB: 346970.145917 A,B Delta: 0.000149

2-core ARMv7@667MHz with event tracing



Page 1 Search: Column: # contains graph follows

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
50990	0	346970.145750	TraceLogger	1169	d.h20	irq_handler_entry	irq=29 name=twd
50991	0	346970.145754	TraceLogger	1169	d.h30	hrtimer_cancel	hrtimer=0xde27dee0
50992	0	346970.145756	TraceLogger	1169	d.h20	hrtimer_expire_entry	hrtimer=0xde27dee0 now=346970139980733 function=hrtimer_wakeup/0x0
50993	0	346970.145759	TraceLogger	1169	d.h30	sched_waking	comm=LV_ESys5_Thr0 pid=1395 prio=5 target_cpu=000
50994	0	346970.145765	TraceLogger	1169	dNh40	sched_wakeup	LV_ESys5_Thr0:1395 [5] success=1 CPU:000
50995	0	346970.145767	TraceLogger	1169	dNh20	hrtimer_expire_exit	hrtimer=0xde27dee0
50996	0	346970.145771	TraceLogger	1169	dNh20	irq_handler_exit	irq=29 ret=handled
50997	0	346970.145774	TraceLogger	1169	dNh20	irq_handler_entry	irq=29 name=twd
50998	0	346970.145777	TraceLogger	1169	dNh30	hrtimer_cancel	hrtimer=0xdfbd0020
50999	0	346970.145780	TraceLogger	1169	dNh20	hrtimer_expire_entry	hrtimer=0xdfbd0020 now=346970140004082 function=tick_sched_timer/0x0
51000	0	346970.145793	TraceLogger	1169	dNh30	sched_stat_runtime	comm=TraceLogger pid=1169 runtime=74874 [ns] vruntime=8265491630797 [ns]
51001	0	346970.145800	TraceLogger	1169	dNh20	softirq_raise	vec=7 [action=SCHED]
51002	0	346970.145803	TraceLogger	1169	dNh20	softirq_raise	vec=1 [action=TIMER]
51003	0	346970.145805	TraceLogger	1169	dNh20	rcu_utilization	Start scheduler-tick
51004	0	346970.145808	TraceLogger	1169	dNh30	sched_waking	comm=rcuc/0 pid=13 prio=98 target_cpu=000
51005	0	346970.145813	TraceLogger	1169	dNh40	sched_wakeup	rcuc/0:13 [98] success=1 CPU:000
51006	0	346970.145816	TraceLogger	1169	dNh20	rcu_utilization	End scheduler-tick
51007	0	346970.145819	TraceLogger	1169	dNh20	hrtimer_expire_exit	hrtimer=0xdfbd0020
51008	0	346970.145821	TraceLogger	1169	dNh30	hrtimer_start	hrtimer=0xdfbd0020 function=tick_sched_timer/0x0 expires=346970150000000 softexpires=346970150000000
51009	0	346970.145826	TraceLogger	1169	dNh30	hrtimer_cancel	hrtimer=0xdccb3f00
51010	0	346970.145829	TraceLogger	1169	dNh20	hrtimer_expire_entry	hrtimer=0xdccb3f00 now=346970140053090 function=hrtimer_wakeup/0x0
51011	0	346970.145831	TraceLogger	1169	dNh30	sched_waking	comm=LV_Countdown pid=1367 prio=93 target_cpu=000
51012	0	346970.145836	TraceLogger	1169	dNh40	sched_wakeup	LV_Countdown:1367 [93] success=1 CPU:000
51013	0	346970.145838	TraceLogger	1169	dNh20	hrtimer_expire_exit	hrtimer=0xdccb3f00
51014	0	346970.145840	TraceLogger	1169	dNh30	hrtimer_cancel	hrtimer=0xdcccde40
51015	0	346970.145842	TraceLogger	1169	dNh20	hrtimer_expire_entry	hrtimer=0xdcccde40 now=346970140053090 function=hrtimer_wakeup/0x0
51016	0	346970.145845	TraceLogger	1169	dNh30	sched_waking	comm=TraceLogger pid=1284 prio=120 target_cpu=000
51017	0	346970.145850	TraceLogger	1169	dNh40	sched_stat_runtime	comm=TraceLogger pid=1169 runtime=57456 [ns] vruntime=8265491688253 [ns]
51018	0	346970.145855	TraceLogger	1169	dNh40	sched_wakeup	TraceLogger:1284 [120] success=1 CPU:000

high priority thread wakeup

another timer irq

sched tick ~40 µS

~15 µS

… more timer expirations

Pathological test

- Configurable number of SCHED_OTHER threads doing

```
while (!g_stop) {
    t.tv_sec = 0;
    interval = (rand() * 1000000LL) / RAND_MAX;
    t.tv_nsec = interval;
    clock_nanosleep(CLOCK_MONOTONIC, 0, &t, NULL);
}
```

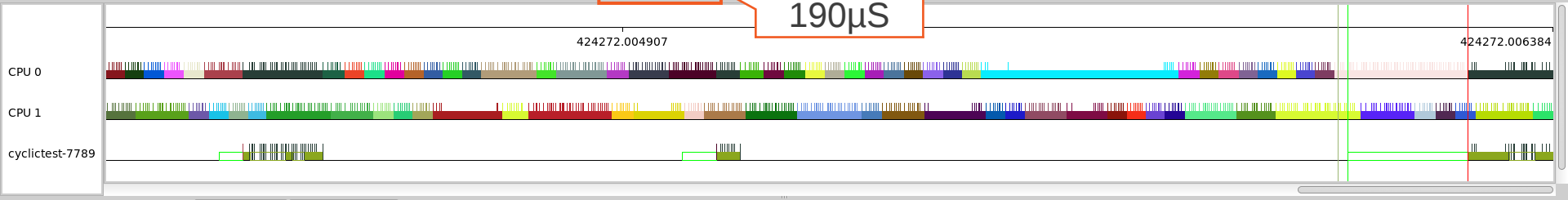
kernelshark(trace.dat)

File Filter Plots Capture Help

Pointer: 424272.004124 Cursor: 424272.006043 MarkerA: 424272.006059 MarkerB: 424272.006249 A,B Delta: 0.000190

190µS

2-core ARMv7@667MHz with event tracing



Page 1 Search: Column: # contains graph follows

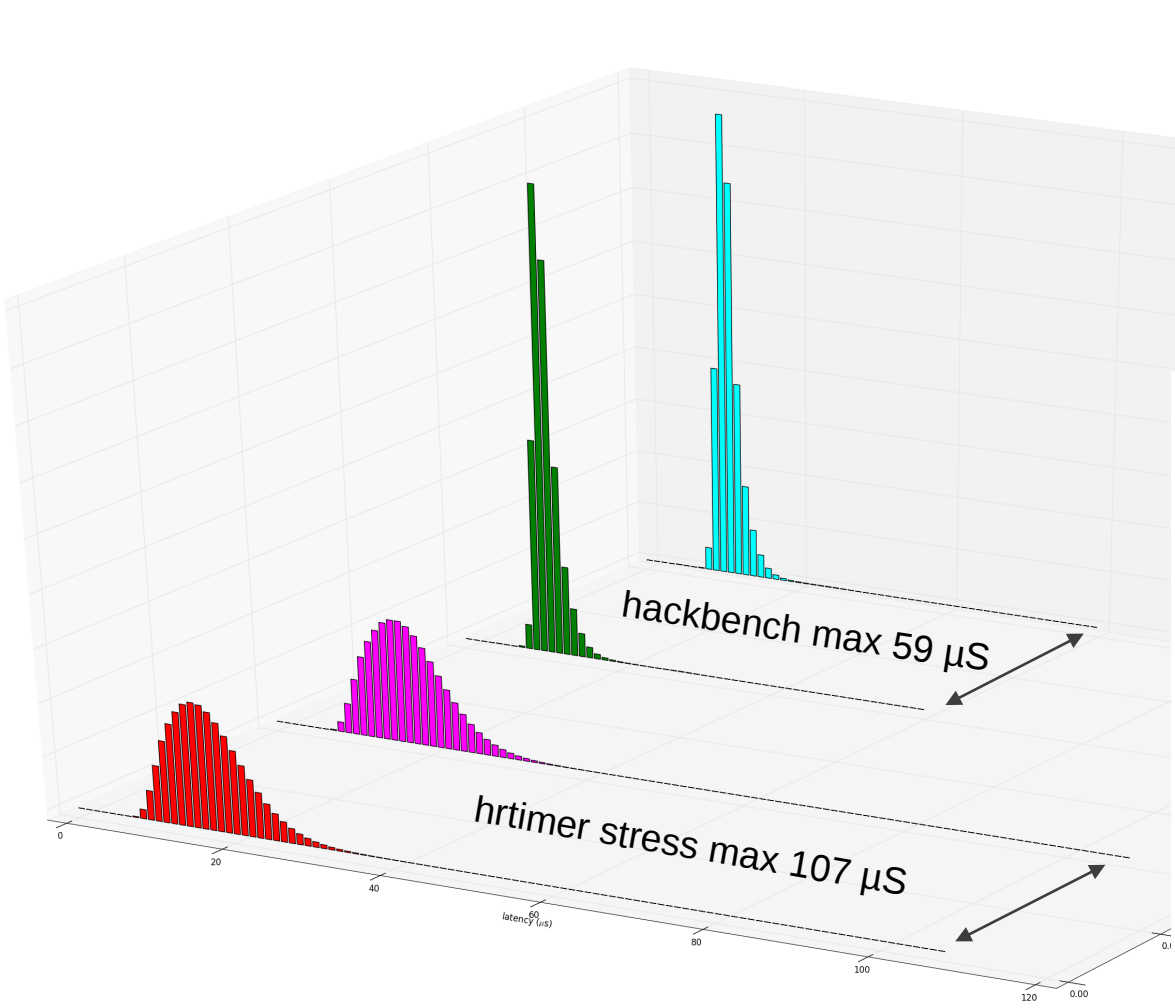
#	CPU	Time Stamp	Task	PID	Latency	Event	Info
34607	0	424272.006043	ttest	7939	d.h20	irq_handler_entry	irq=29 name=twd
34608	0	424272.006047	ttest	7939	d.h30	hrtimer_cancel	hrtimer=0xc7ae3ef0
34609	0	424272.006049	ttest	7939	d.h20	hrtimer_expire_entry	hrtimer=0xc7ae3ef0 now=424272002659534 function=hrtimer_wakeup/0x0
34610	0	424272.006052	ttest	7939	d.h30	sched_waking	comm=cyclictest pid=7789 prio=1 target_cpu=000
34611	0	424272.006058	ttest	7939	dNh40	sched_wakeup	cyclictest:7789 [1] success=1 CPU:000
34612	0	424272.006060	ttest	7939	dNh20	hrtimer_expire_exit	hrtimer=0xc7ae3ef0
34613	0	424272.006062	ttest	7939	dNh30	hrtimer_cancel	hrtimer=0xc820bef0
34614	0	424272.006065	ttest	7939	dNh20	hrtimer_expire_entry	hrtimer=0xc820bef0 now=424272002659534 function=hrtimer_wakeup/0x0
34615	0	424272.006067	ttest	7939	dNh30	sched_waking	comm=ttest pid=8251 prio=120 target_cpu=000
34616	0	424272.006073	ttest	7939	dNh40	sched_stat_runtime	comm=ttest pid=7939 runtime=42276 [ns] vruntime=1478082751184 [ns]
34617	0	424272.006080	ttest	7939	dNh40	sched_wakeup	ttest:8251 [120] success=1 CPU:000
34618	0	424272.006082	ttest	7939	dNh20	hrtimer_expire_exit	hrtimer=0xc820bef0
34619	0	424272.006084	ttest	7939	dNh30	hrtimer_cancel	hrtimer=0xc8a7bef0
34620	0	424272.006086	ttest	7939	dNh20	hrtimer_expire_entry	hrtimer=0xc8a7bef0 now=424272002659534 function=hrtimer_wakeup/0x0
34621	0	424272.006088	ttest	7939	dNh30	sched_waking	comm=ttest pid=7840 prio=120 target_cpu=000
34622	0	424272.006093	ttest	7939	dNh40	sched_stat_runtime	comm=ttest pid=7939 runtime=20955 [ns] vruntime=1478082772139 [ns]
34623	0	424272.006098	ttest	7939	dNh40	sched_wakeup	ttest:7840 [120] success=1 CPU:000
34624	0	424272.006100	ttest	7939	dNh20	hrtimer_expire_exit	hrtimer=0xc8a7bef0
34625	0	424272.006102	ttest	7939	dNh30	hrtimer_cancel	hrtimer=0xc7b85ef0
34626	0	424272.006104	ttest	7939	dNh20	hrtimer_expire_entry	hrtimer=0xc7b85ef0 now=424272002659534 function=hrtimer_wakeup/0x0
34627	0	424272.006106	ttest	7939	dNh30	sched_waking	comm=ttest pid=7833 prio=120 target_cpu=000
34628	0	424272.006111	ttest	7939	dNh40	sched_stat_runtime	comm=ttest pid=7939 runtime=17691 [ns] vruntime=1478082789830 [ns]
34629	0	424272.006115	ttest	7939	dNh40	sched_wakeup	ttest:7833 [120] success=1 CPU:000
34630	0	424272.006117	ttest	7939	dNh20	hrtimer_expire_exit	hrtimer=0xc7b85ef0
34631	0	424272.006118	ttest	7939	dNh30	hrtimer_cancel	hrtimer=0xc7b5fef0
34632	0	424272.006121	ttest	7939	dNh20	hrtimer_expire_entry	hrtimer=0xc7b5fef0 now=424272002659534 function=hrtimer_wakeup/0x0
34633	0	424272.006123	ttest	7939	dNh30	sched_waking	comm=ttest pid=7794 prio=120 target_cpu=000
34634	0	424272.006127	ttest	7939	dNh40	sched_stat_runtime	comm=ttest pid=7939 runtime=16503 [ns] vruntime=1478082806333 [ns]
34635	0	424272.006132	ttest	7939	dNh40	sched_wakeup	ttest:7794 [120] success=1 CPU:000
34636	0	424272.006134	ttest	7939	dNh20	hrtimer_expire_exit	hrtimer=0xc7b5fef0

cyclictest wakeup

~15 µS

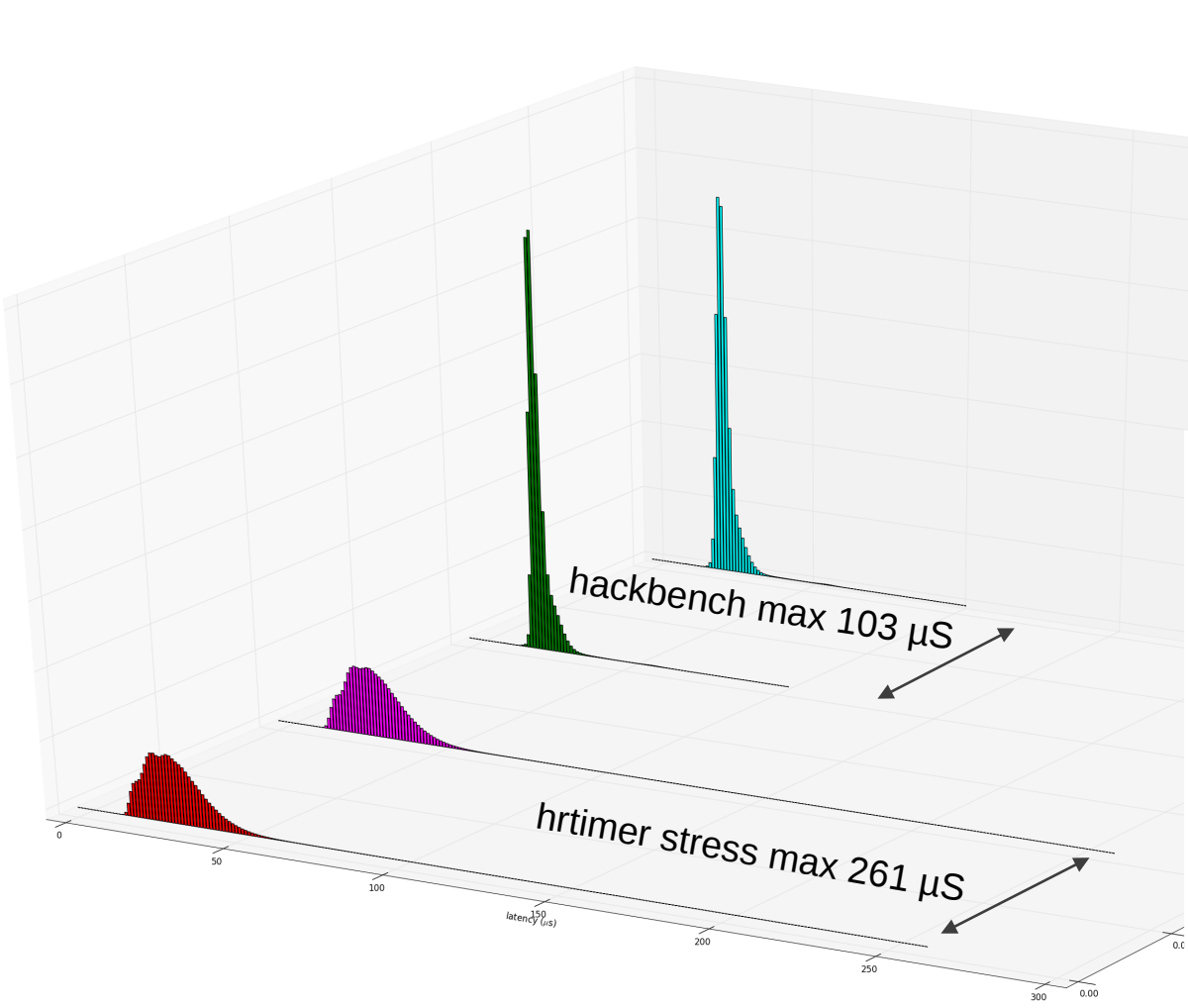
~15 µS

...



- Intel(R) Atom(TM) E3825 @ 1.33GHz dual core
- kernel: 4.11.12-rt14
- cyclictest -m -S -p 98 -i 237 –H 200
- three day run
- two devices:

1. hackbench -g25 -l 1000000000
   - running in process mode with 25 groups using 40 file descriptors each (== 1000 tasks)
2. 1000 threads running timer stress



- ARMv7 (v7l) @ 667MHz dual core
- kernel: 4.11.12-rt14
- cyclictest -m -S -p 98 -i 737 –H 500
- three day run
- two devices:

1. hackbench -l 1000000000 -f 25
   - running in process mode with 10 groups using 50 file descriptors each (== 500 tasks)
2. 500 threads running timer stress

Hack[1] that (kind of) worked before v4.11.12-rt13

```
void hrtimer_init_sleeper(struct hrtimer_sleeper *sl,
            struct task_struct *task)
{
    sl->timer.function = hrtimer_wakeup;
-   sl->timer.irqsafe = 1;
+   sl->timer.irqsafe = rt_task(task);
    sl->task = task;
}
```

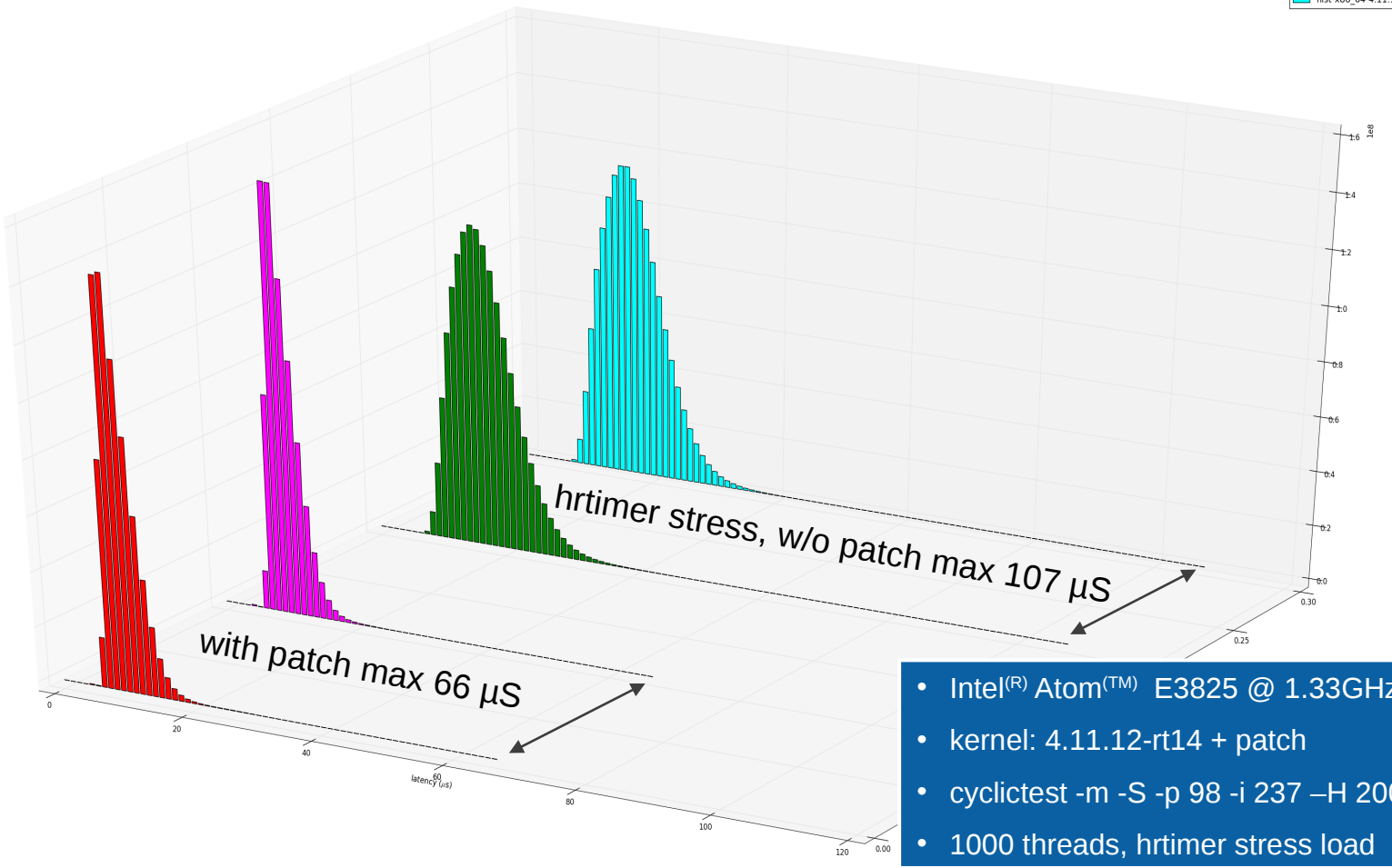
[1] https://marc.info/?l=linux-rt-users&m=148354667204563&w=2

Much better patch[1]

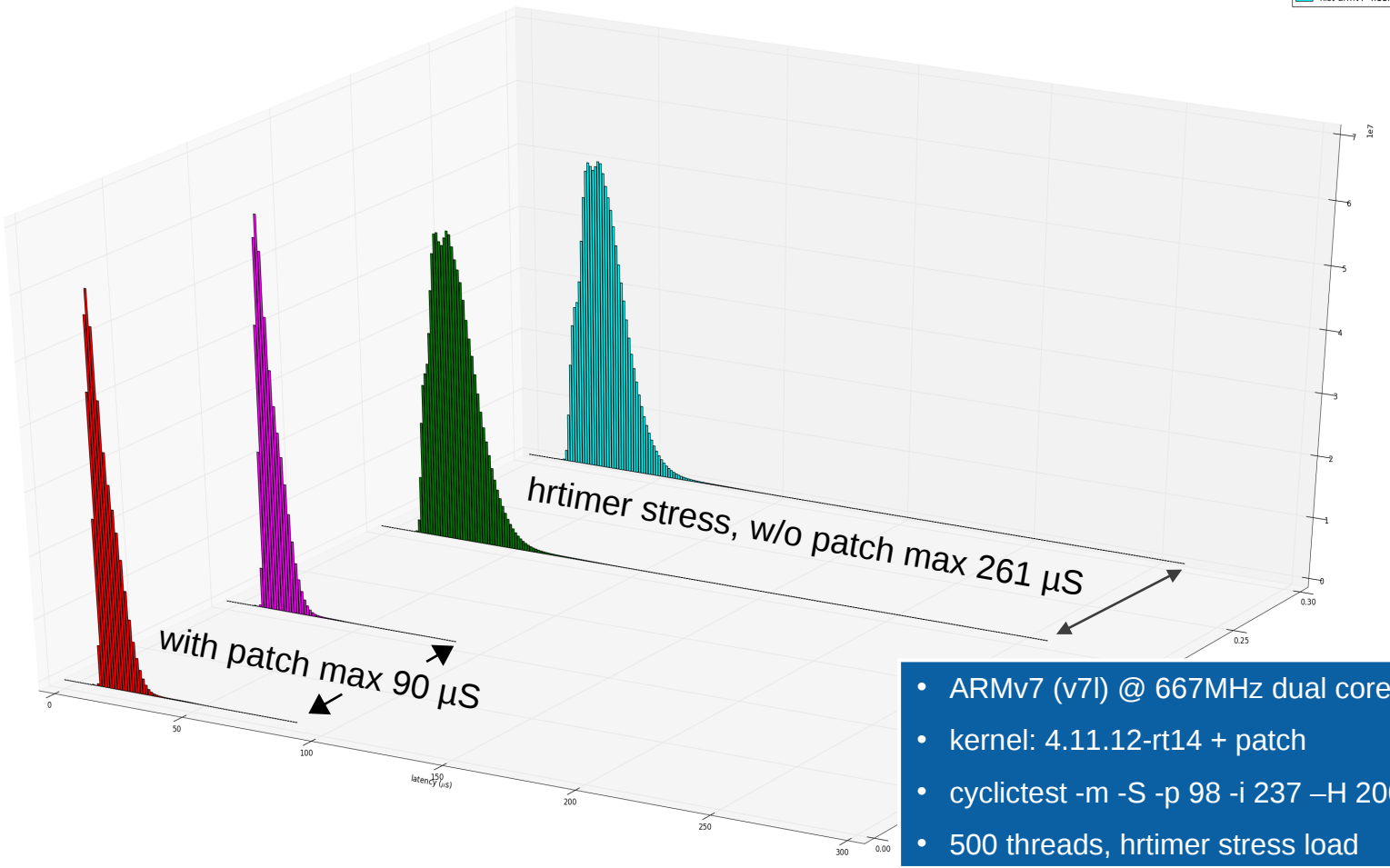
```
time/hrtimer: Use softirq based wakeups for non-RT threads

Normal wake ups (like clock_nanosleep()) which are performed by
normal users can easily lead to 2ms latency spikes if (enough)
hrtimer wakeups are synchronized.
This patch moves all hrtimers wakeups to the softirq queue unless
the caller has a RT priority.

Reported-by: Gratian Crisan <gratian.crisan@ni.com>
Signed-off-by: Sebastian Andrzej Siewior <bigeasy@linutronix.de>
```

[1] https://lkml.org/lkml/2017/10/4/560

NATIONAL INSTRUMENTS



- Intel(R) Atom(TM) E3825 @ 1.33GHz
- kernel: 4.11.12-rt14 + patch
- cyclictest -m -S -p 98 -i 237 –H 200
- 1000 threads, hrtimer stress load



- ARMv7 (v7l) @ 667MHz dual core
- kernel: 4.11.12-rt14 + patch
- cyclictest -m -S -p 98 -i 237 –H 200
- 500 threads, hrtimer stress load

Lessons learned

- Report problems early
- Upgrade to the latest linux-rt-devel branch as soon as possible
- Don’t go on vacation before your RT-Summit presentation

RT Trouble #3

Lack of priority inheritance support in the glibc pthread library

libpthread priority inheritance support

- **With** priority inheritance support:
  - pthread_mutex_* } FUTEX_LOCK_PI/UNLOCK_PI

- **Without** priority inheritance support:
  - pthread_rwlock_* internal lock } FUTEX_WAIT/WAKE
  - sem_* } FUTEX_WAIT_BITSET/WAKE
  - pthread_spin_* } user-space spinning
  - pthread_cond_* internal lock } see next slide

pthread conditional variables

Bug 11588 - pthread condvars are not priority inheritance aware

Status: NEW	**Reported:** 2010-05-11 18:45 UTC by Darren Hart
Alias: None	**Modified:** 2017-08-31 20:44 UTC (History)
	CC List: 16 users (show)
Product: glibc	**See Also:**
Component: nptl (show other bugs)	**Host:**
Version: 2.12	**Target:**
	Build:
Importance: P2 enhancement	**Last reconfirmed:**
Target Milestone: ---	
Assignee: Not yet assigned to anyone	**Flags:** fweimer: security-
URL:	
Keywords:	
Depends on:	
Blocks:	

NATIONAL INSTRUMENTS

Current state glibc bug #11588

Torvald Riegel 2017-01-11 11:50:41 UTC **Comment 56**

The new condition variable implementation is now committed upstream. It should be the base for any improvement suggestions from now on.

How to support PI for condvars has also been discussed at the Linux Real-Time Summit 2016: https://wiki.linuxfoundation.org/realtime/events/rt-summit2016/schedule
So far, there is no known solution for how to achieve PI support given the current constraints we have (eg, available futex operations, POSIX requirements, ...).

Austin Group defect #609

ID	Category	Severity	Type	Date Submitted	Last Update
0000609	[1003.1(2004)/Issue 6] System Interfaces	Editorial	Clarification Requested	2012-09-20 14:18	2016-05-17 22:13

Reporter	mmihaylov	**View Status**	public		
Assigned To	ajosey				
Priority	normal	**Resolution**	Open		
Status	Under Review				

Name	Mihail Mihaylov
Organization	
User Reference	
Section	pthread_cond_broadcast, pthread_cond_signal
Page Number	1043
Line Number	33043 - 33046
Interp Status	---
Final Accepted Text	

Summary	0000609: It is not clear what threads are considered blocked with respect to a call to pthread_cond_signal() or pthread_cond_broadcast()

NATIONAL INSTRUMENTS

Discussion

- Do you know of work underway / progress since last year?
- Alternative libraries out there for RT friendly locking?
- Any RT friendly data structures library (e.g. circular buffers, FIFOs, etc.)

RT Trouble #4

Managing IRQ priorities and IRQ priority inversions

Big disclaimers

- We know the following patches are not appropriate for upstream
- The need for them arises from our inexperience at the time and a usability problem with mapping an IRQ to its corresponding thread PID
- We are looking for best practice ideas

What is the best way to set IRQ priorities?

- Currently carrying[1]:

```
irq: Add priority support to /proc/irq/../

This patch allows configuring priority for different irq threads through
the /proc/irq/ system (much same as the existing mechanism to configure
the core affinity for irqs).
…

Signed-off-by: Sankara S Muthukrishnan <sankara.m@ni.com>
Signed-off-by: Julia Cartwright <julia.cartwright@ni.com>
```

[1] https://github.com/ni/linux/commit/5ff3a76173659863b6c5bda9ecf094d4621ccba7

What is the best way to set priorities on new IRQs?

- Currently carrying[1][2]:

```
[RFC][PATCH] fs/proc: add poll()ing support to /proc/interrupts

Implement polling on procfs' "interrupts" file which observes changes to
IRQ action handlers. The poll fires each time an action handler is
registered or unregistered.

This change enables daemons to watch for changes and apply certain system
policies relating to IRQ processing. For example, modify execution priority
of dedicated IRQ tasks after they're created.
…
Signed-off-by: Haris Okanovic <haris.okanovic@ni.com>
Signed-off-by: Ovidiu-Adrian Vancea <ovidiu.vancea@ni.com>
Signed-off-by: Brad Mouring <brad.mouring@ni.com>
```

[1] https://github.com/ni/linux/commit/8e2f148e2f4762cc2b6490ec01d5d31bc440bcf8
[2] http://www.spinics.net/lists/linux-rt-users/msg14076.html

NATIONAL INSTRUMENTS

IRQ priority inversions

Example

- Context
  - Watchdog functionality implemented in a CPLD connected to a I^2C 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 I^2C transfers need to happen
  - I^2C interrupt thread has low priority
  - Some unrelated mid-priority thread preempts the I^2C interrupt

Discussion

- Ongoing work on avoiding IRQ priority inversions?
- Is there a more general solution to the priority inversion problem with completion objects?


NATIONAL
INSTRUMENTS™

Extra stuff

Tip #1

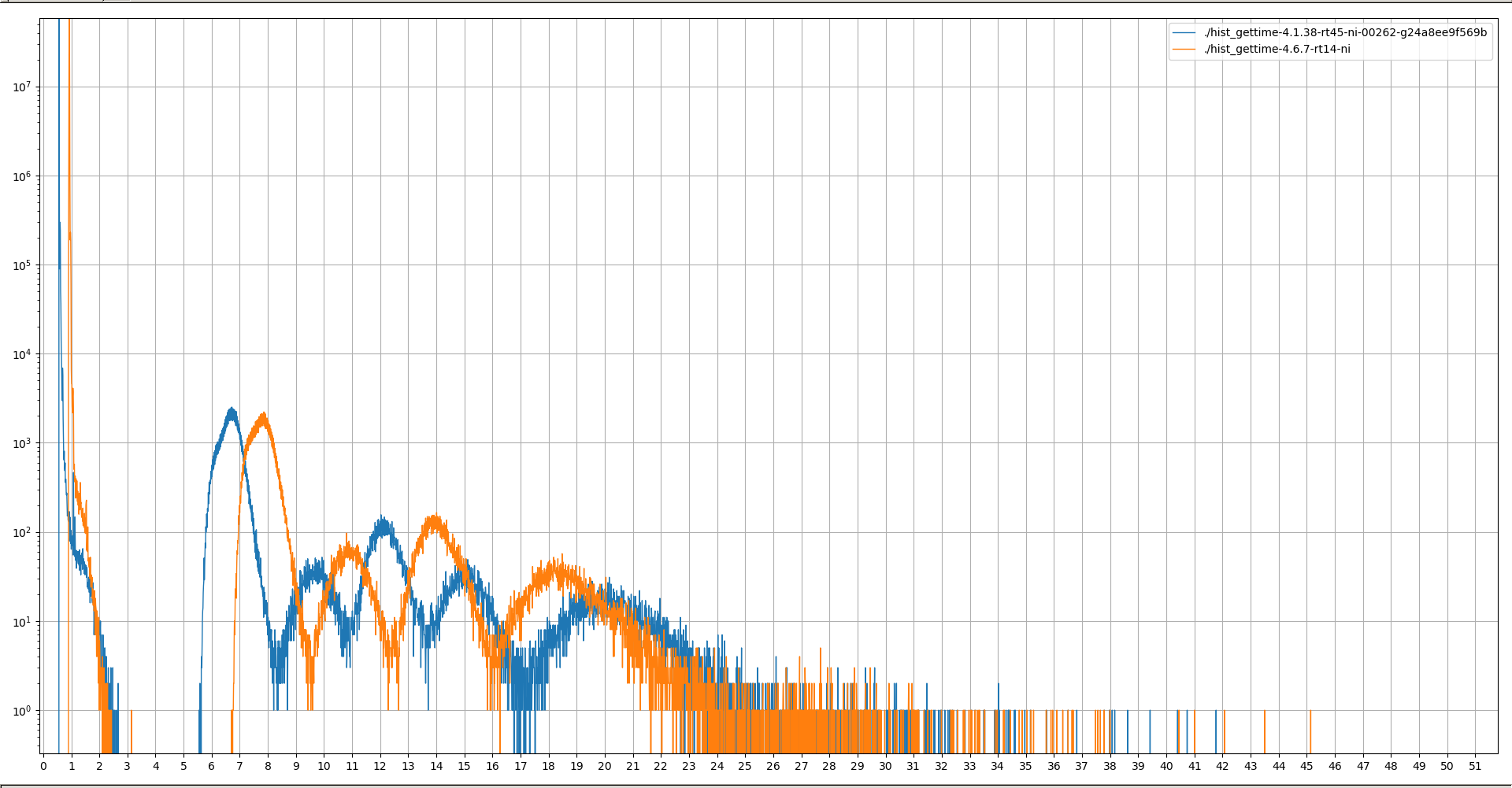
Check config options after a kernel upgrade

```
config CPU_SW_DOMAIN_PAN[1]
 bool "Enable use of CPU domains to implement privileged no-access"
 depends on MMU && !ARM_LPAE
 default y

 help
 Increase kernel security by ensuring that normal kernel accesses
 are unable to access userspace addresses. This can help prevent
 use-after-free bugs becoming an exploitable privilege escalation
 by ensuring that magic values (such as LIST_POISON) will always
 fault when dereferenced.
```

[1] Introduced in v4.3, commit a5e090acbf54 ("ARM: software-based priviledged-no-access support"). Adds code in uaccess_*, save_regs, load_regs macros.

NATIONAL INSTRUMENTS

Figure 1



pan/zoom

Perf Diff

```
# Event 'cycles:ppp'
#
# Baseline    Delta  Shared Object         Symbol
# ........  .......  ....................  ............................
#
    20.83%   -7.06%  [kernel.kallsyms]     [k] __getnstimeofday64
    17.34%   -1.25%  libc-2.23.so          [.] __clock_gettime
    15.93%   +3.05%  [kernel.kallsyms]     [k] vector_swi
     8.88%  +10.38%  [kernel.kallsyms]     [k] sys_clock_gettime
     6.63%   -2.07%  [kernel.kallsyms]     [k] __copy_to_user_std
     5.59%           [kernel.kallsyms]     [k] gt_counter_read
     4.86%   -1.24%  [kernel.kallsyms]     [k] ret_fast_syscall
     3.82%   -0.36%  simple_clock_gettime  [.] main
     3.54%   +1.15%  [kernel.kallsyms]     [k] gt_clocksource_read
     3.29%   -0.12%  [kernel.kallsyms]     [k] getnstimeofday64
     2.99%   +0.19%  [kernel.kallsyms]     [k] posix_clock_realtime_get
     2.94%   +0.44%  [kernel.kallsyms]     [k] clockid_to_kclock
     2.06%           [kernel.kallsyms]     [k] __aeabi_llsr
     0.70%   -0.32%  [kernel.kallsyms]     [k] local_restart
     0.55%   -0.54%  simple_clock_gettime  [.] memset@plt
     0.01%           ld-2.23.so           [.] do_lookup_x
     0.01%           [kernel.kallsyms]     [k] vma_interval_tree_remove
     0.01%   -0.00%  [kernel.kallsyms]     [k] v7_flush_icache_all
     0.01%           [kernel.kallsyms]     [k] _test_and_set_bit
     0.01%           [kernel.kallsyms]     [k] strnlen_user
     0.00%           [kernel.kallsyms]     [k] __alloc_pages_nodemask
     0.00%   -0.00%  [kernel.kallsyms]     [k] perf_event_exec
             +2.99%  [kernel.kallsyms]     [k] _gt_counter_read
             +1.53%  [kernel.kallsyms]     [k] __aeabi_lasr
             +0.68%  simple_clock_gettime  [.] __gmon_start__@plt
             +0.20%  simple_clock_gettime  [.] sched_setaffinity@plt
             +0.01%  [kernel.kallsyms]     [k] __cpuset_node_allowed
             +0.01%  ld-2.23.so           [.] _dl_init_paths
             +0.01%  ld-2.23.so           [.] _dl_lookup_symbol_x
             +0.01%  [kernel.kallsyms]     [k] filemap_map_pages
             +0.01%  [kernel.kallsyms]     [k] do_page_fault
             +0.00%  [kernel.kallsyms]     [k] pfn_valid
             +0.00%  [kernel.kallsyms]     [k] rt_mutex_lock
             +0.00%  [kernel.kallsyms]     [k] finish_task_switch
             +0.00%  [kernel.kallsyms]     [k] _raw_spin_unlock_irq
```

NATIONAL INSTRUMENTS

Tip #2

Check your clock sources

Check your clock sources

- Issues encountered
  - TSC clock source gets disabled by the clock source watchdog due to acpi_pm rollover
  - Boot hang caused by left over test code in BIOS that sets the TSC_ADJUST register on core 0

- On upgrades and new hardware it helps to:
  - Check current clock source

    /sys/devices/system/clocksource/clocksource0/current_clocksource
  - Compare timer expirations against external reference
  - Drive trace off external clock source (e.g. FPGA)

Tip #3

Run reboot tests

Run reboot tests

- Multiple issues discovered by running a simple reboot test
  - Hangs on boot
  - Ext4 data corruptions
  - NAND read disturbs
  - Ethernet link detection issues
  - futex race on exit
  - i915 crash on module load
- Suggestions
  - Simple test - calls reboot once the software stack is up
  - Hard reboots - data loss is OK, data corruption is not
  - (optional) Temperature controlled chamber


NATIONAL
INSTRUMENTS™