rtla timerlat

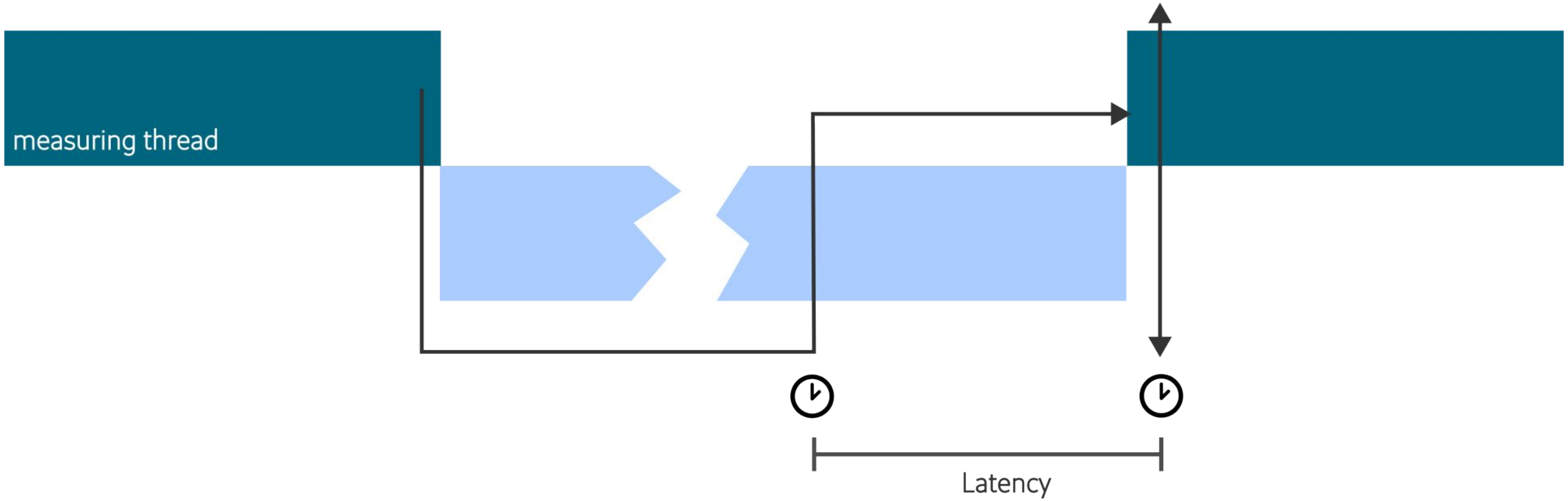
Debugging Real-time Linux Scheduling Latency

Daniel Bristot de Oliveira, Red Hat
@bristot

- Linux has been used as an RTOS - it is a fact!
- There are multiple reasons for people to use it
  - Software stack and availability
  - Man-power
- But also because Linux achieves the desired timing behavior
- Some key features to help with that are:
  - The fully preemptive mode
  - Real-time scheduling
    - SCHED_DEADLINE

- One of the problems, however, is the way that we show the timing properties of Linux
- Linux has been tested using **blackbox tools** that mimic typical workload:
  - Event-driven application: cyclictest
- The "latency" report is important for many use-cases. For example:
  - The kernel-rt has to deliver < 150 us *cyclictest latency* under stress
    - cyclictest latency of 10~20 us on isolated & tuned systems

- scheduling latency **black box** approach



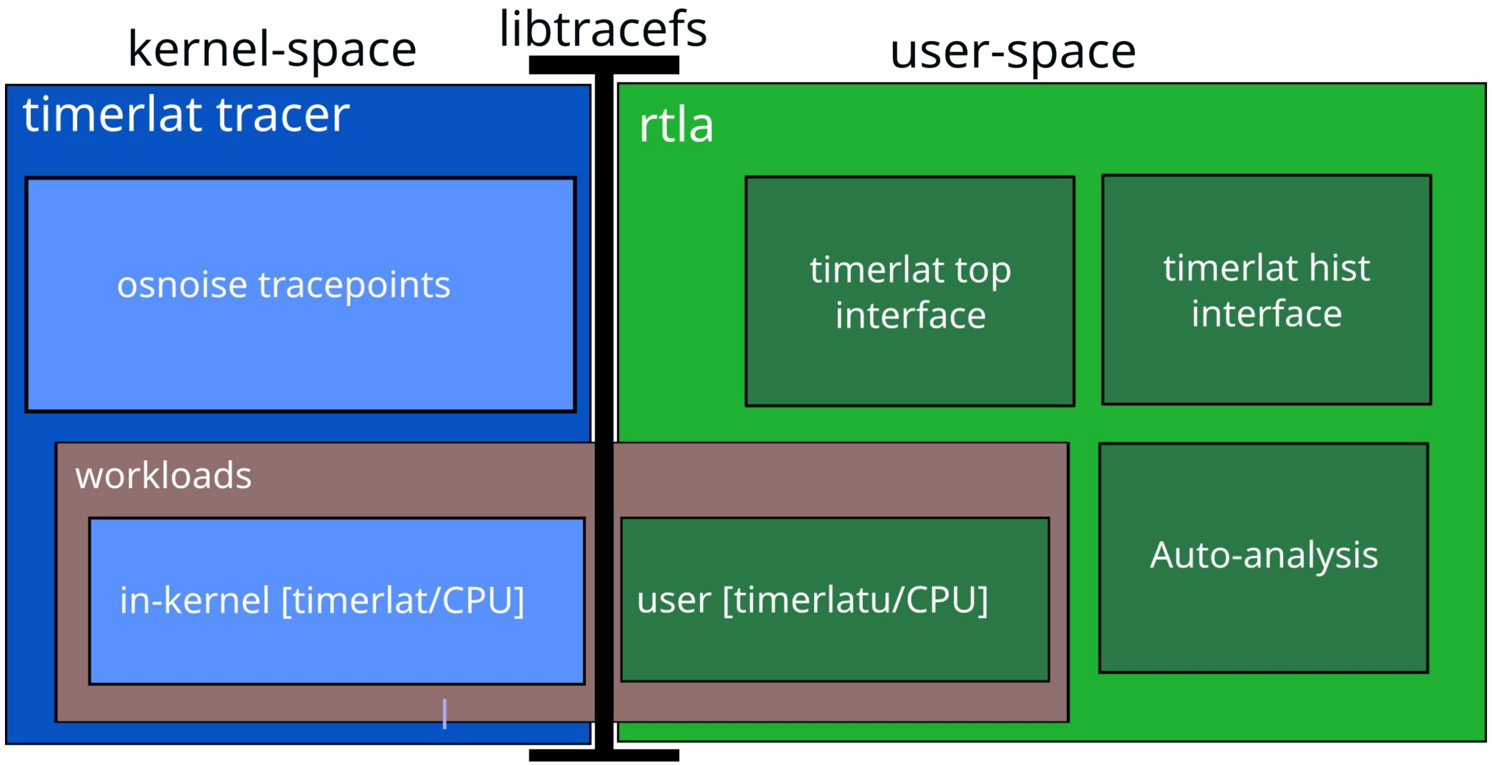
- The **blackbox** approach works, but it has some drawbacks
  - It gives no root cause analysis
- **The root cause analysis is generally done using tracing**
  - But tracing is not that accessible for non-experts
- **Independent thighs are glued by human**
- After 10+ years, one gets annoyed of repeating the same ritual

- **Who cares?**
  - other than the poor dude doing debugging
- **Real-time to the masses**
  - All kernel developers will have to run RT testing/analysis
  - But not them all are interested in learning all the details
- **Projects where numbers need a why**
  - Automotive
  - Automation

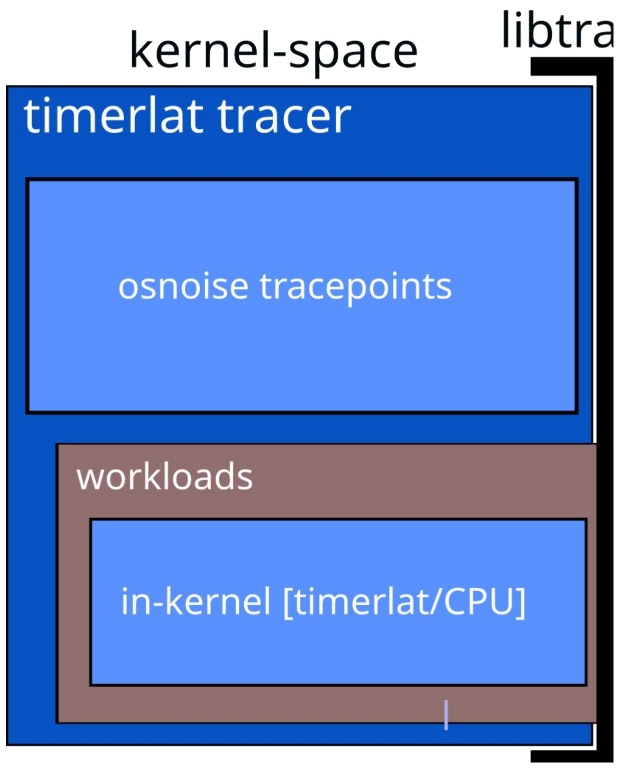
rtla timerlat

a new approach

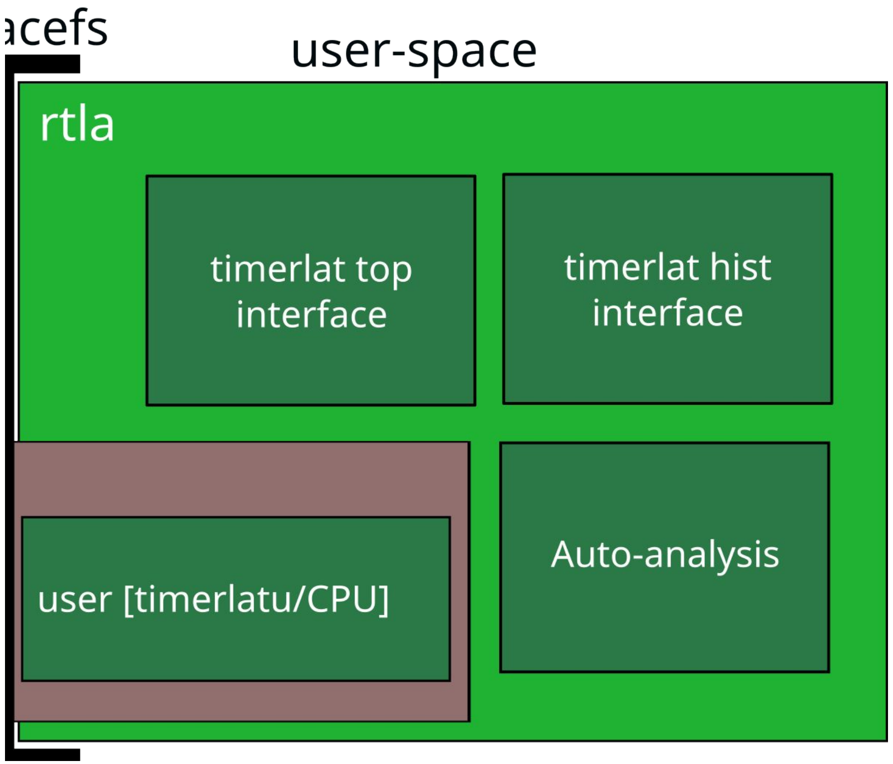
- rtla timerlat is an integrated solution



- **Optimized tracer**
  - In-kernel processing for reduced overhead
    - lockless synchronization
  - It reduces the amount of tracing data
- In kernel workload
- See **Operating System Noise in the Linux Kernel** paper on IEEE Transactions on Computers:

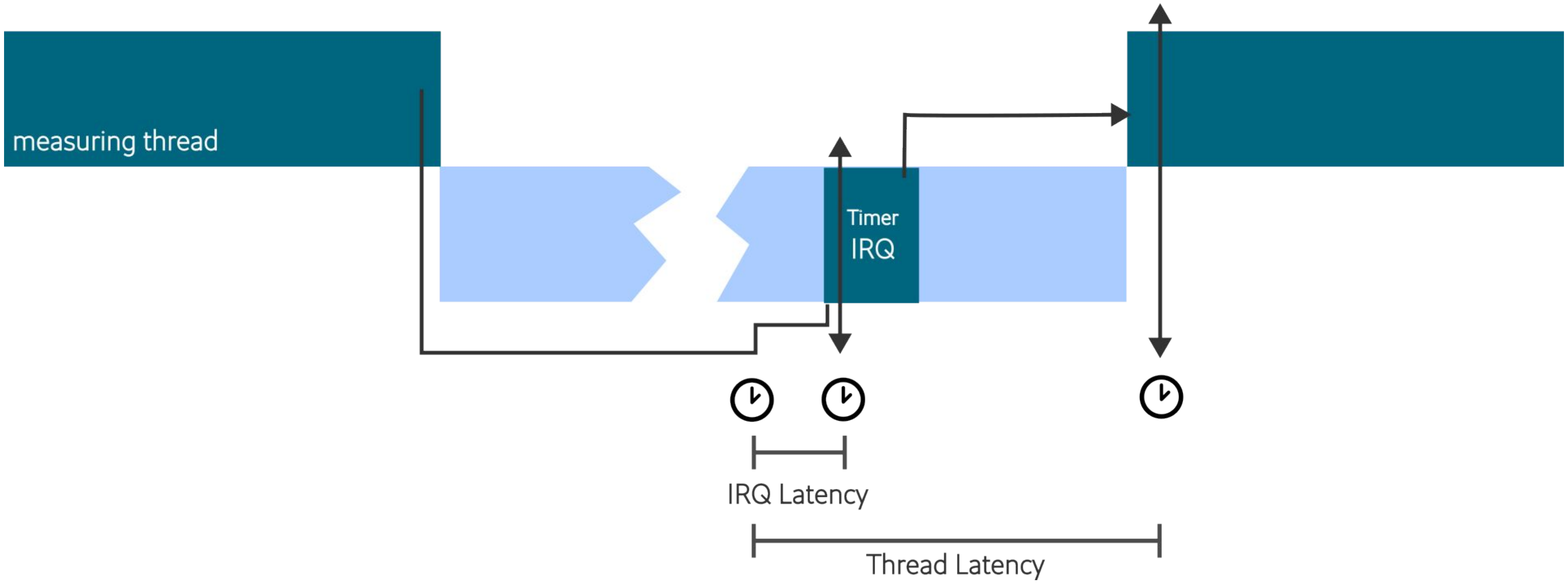


- **rtla timerlat** is part of **rtla** (the suite)
- Benchmark like interface
  - It sets up, collects, and parse trace data
    - top like
    - histogram
- Auto-analysis for long latencies
- User-space workload

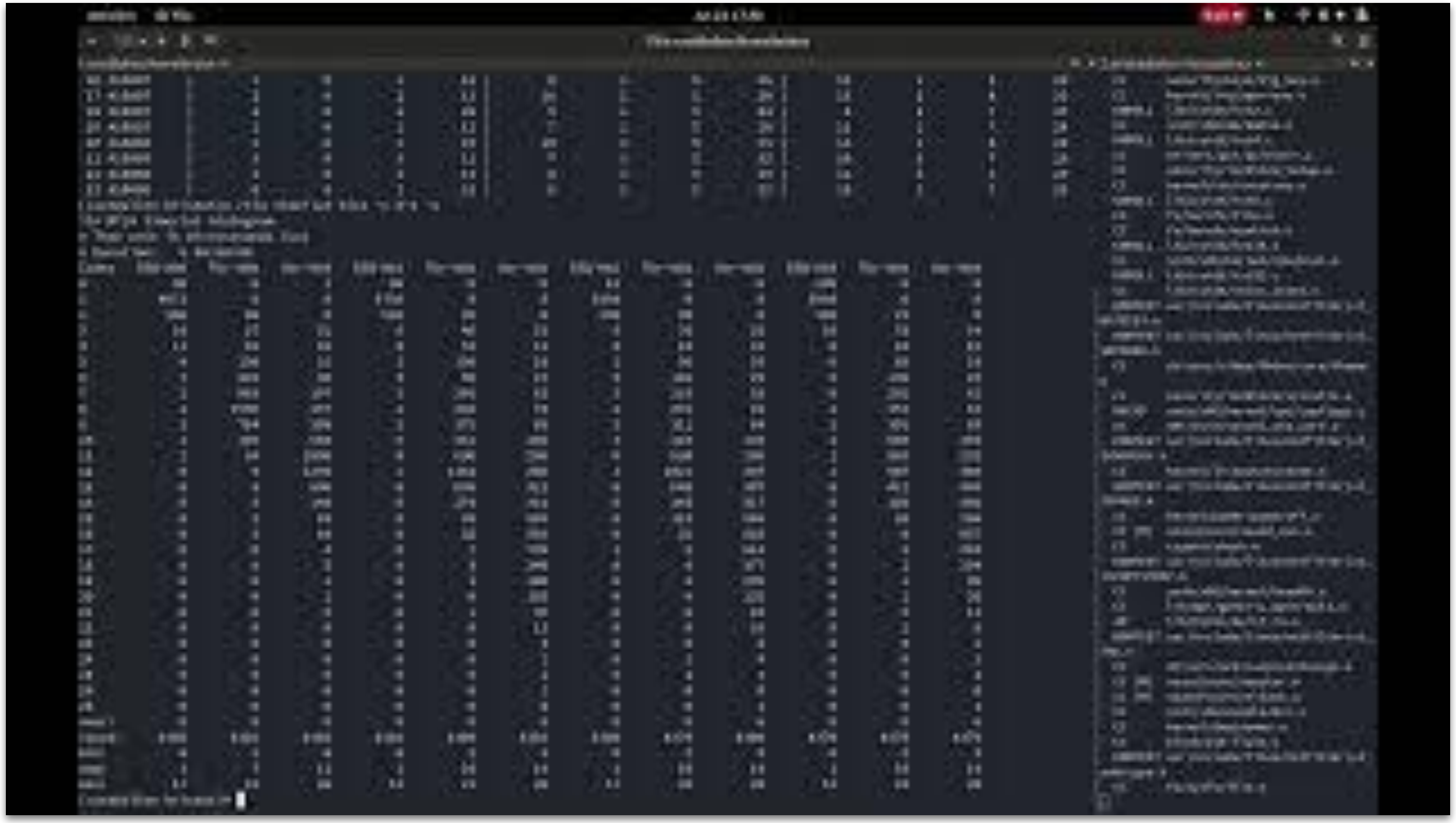


Practical intro

- Timerlat workload has two steps:
  - IRQ handler latency
  - Thread latency

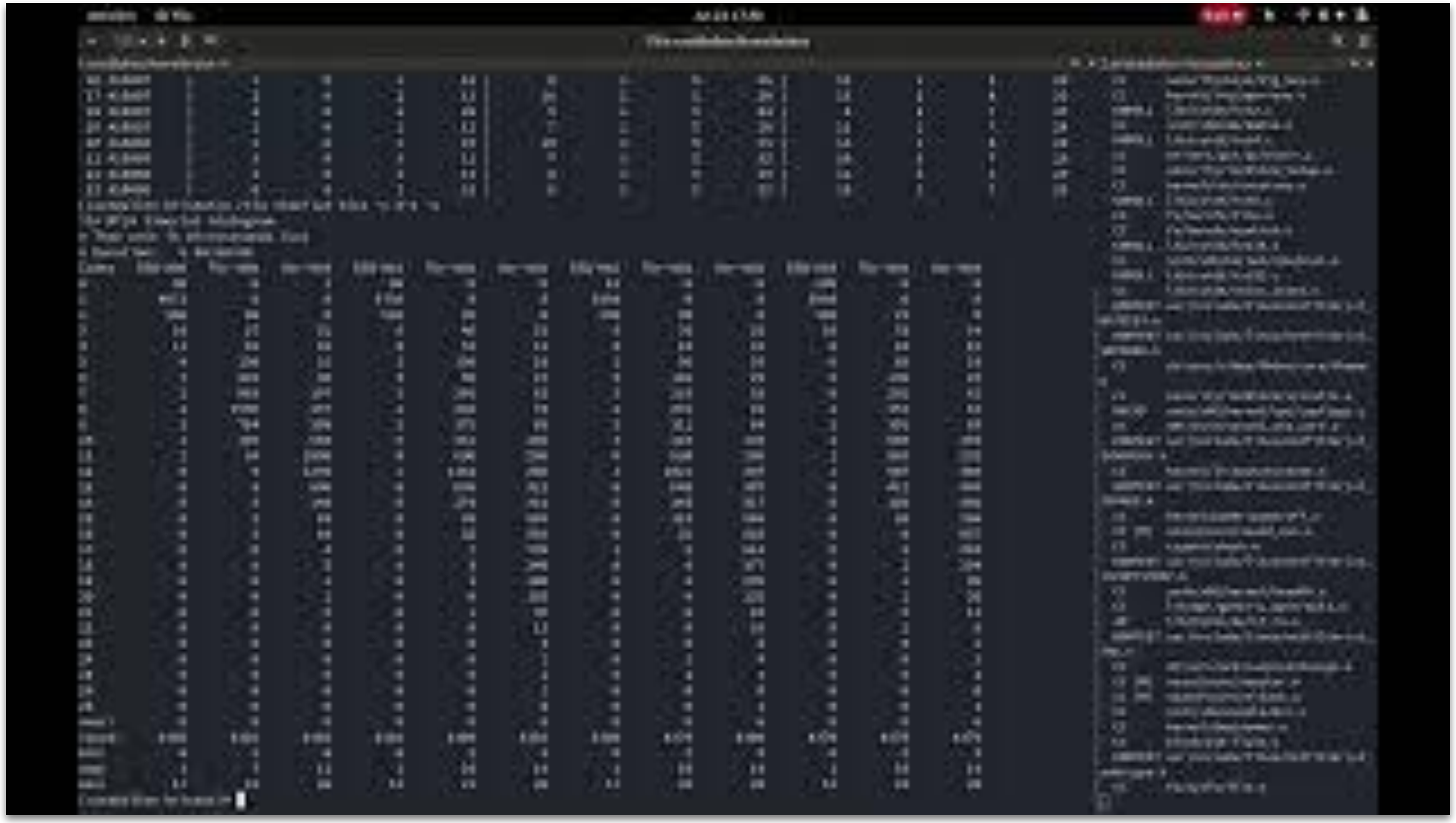


- Timerlat as a benchmark



- When testing a system, we generally have a **max acceptable** latency
  - Commonly, in the low microseconds scale, e.g., 100 us
- Timerlat can be set **to stop and produce a report** if a latency higher than a threshold is hit
    - if the thread is >
    - if the IRQ is >
- The -a <threshold> is a magic option
  - it enables a common set of options

- Timerlat auto-analysis



auto-analysis
analysis

- The auto-analysis **decomposes the latency** into a set of variables
  - Each of these variables can be analyzed independently
- IRQ and Thread latencies have different analysis
  - So the importance of having two metrics for the benchmark
- The auto analysis works for all preemption models
  - This is not only for PREEMPT_RT, but for any kernel

- **timerlat** uses **abstractions** from RT theory
  - **Execution time** is the time to accomplish the task
  - **Blocking** is caused by lower-priority tasks
  - **Interference** is caused by higher-priority tasks
- Linux has a set of task abstractions
  - **NMI**: Non-maskable interrupts preempt any other type of tasks
  - **IRQ:** Preempts all but NMIs.
  - **Softirq**: Preempts threads only (PREEMPT_RT: softirqs are threads)
  - **Threads:** Threads can only preempt other threads.

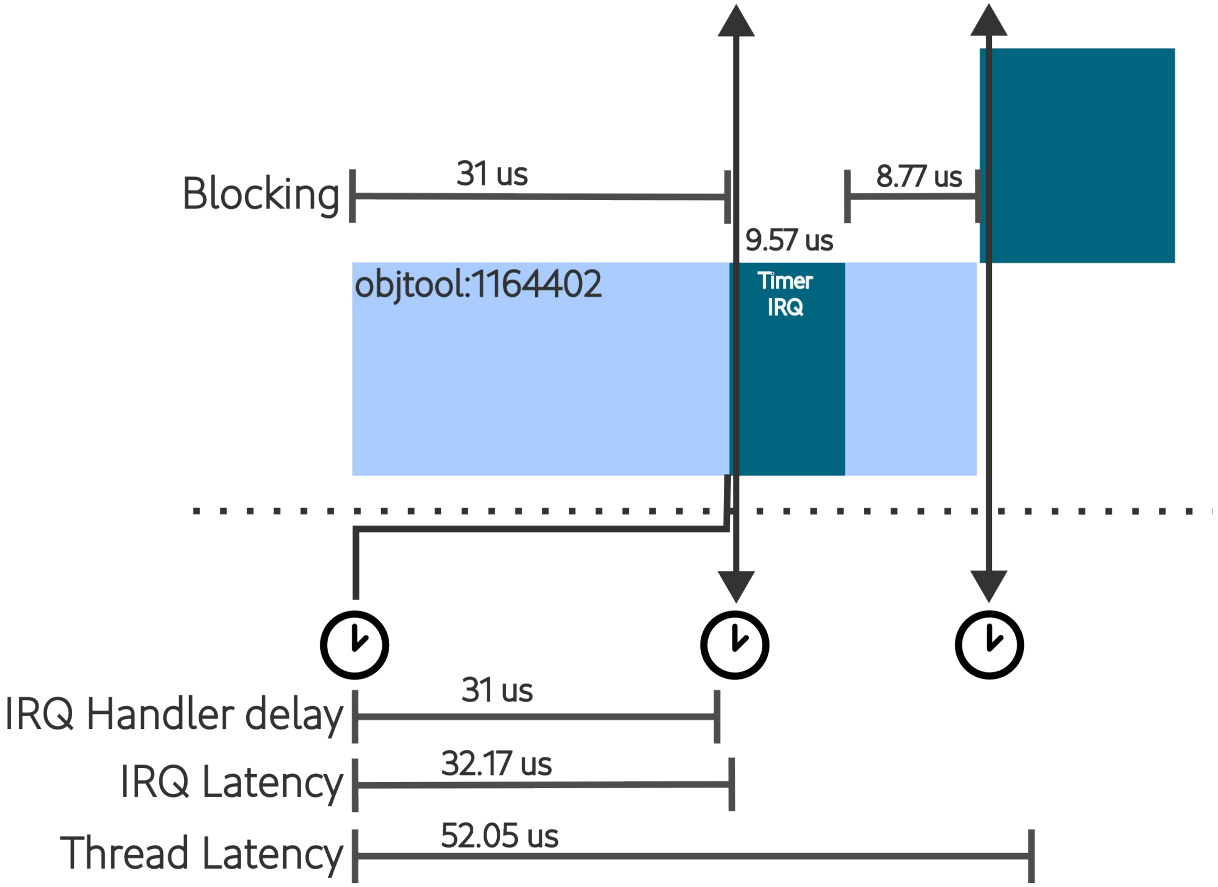
IRQ latency examples

```
## CPU 6 hit stop tracing, analyzing it ##
  IRQ handler delay:                                  31.00 us (59.56 %)
  IRQ latency:                                        32.17 us
  Timerlat IRQ duration:                              9.57 us (18.38 %)
  Blocking thread:                                    8.77 us (16.84 %)
                      objtool:1164402                 8.77 us
    Blocking thread stack trace
           -> timerlat_irq
           -> __hrtimer_run_queues
           -> hrtimer_interrupt
           -> __sysvec_apic_timer_interrupt
           -> sysvec_apic_timer_interrupt
           -> asm_sysvec_apic_timer_interrupt
           -> _raw_spin_unlock_irqrestore
           -> cgroup_rstat_flush_locked
           -> cgroup_rstat_flush_irqsafe
           -> mem_cgroup_flush_stats
           -> mem_cgroup_wb_stats
           -> balance_dirty_pages
           -> balance_dirty_pages_ratelimited_flags
           -> btrfs_buffered_write
           -> btrfs_do_write_iter
           -> vfs_write
           -> __x64_sys_pwrite64
           -> do_syscall_64
           -> entry_SYSCALL_64_after_hwframe
------------------------------------------------------------------------
  Thread latency:                                52.05 us (100%)

Max timerlat IRQ latency from idle: 19.93 us in cpu 12
  Saving trace to timerlat_trace.txt
```

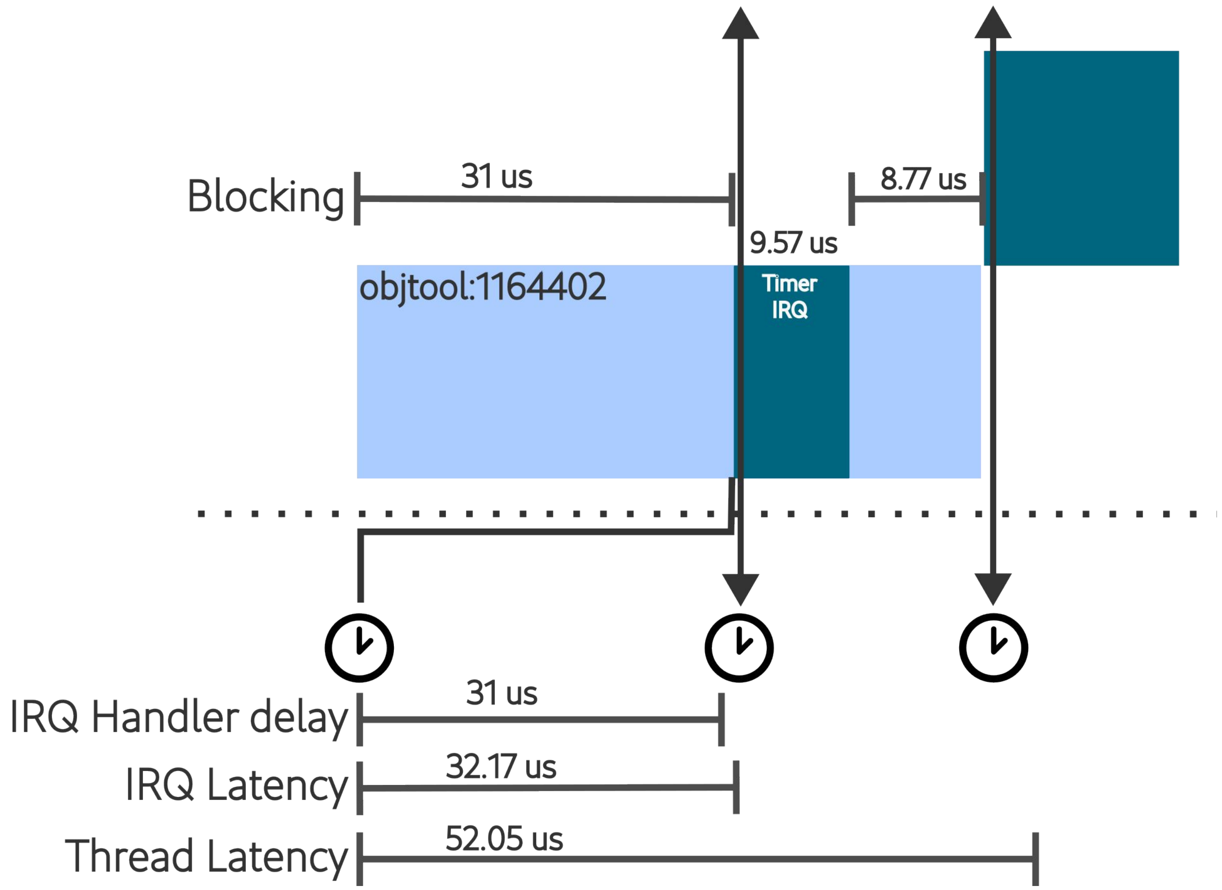
rtla timerlat

Debugging Real-time Linux Scheduling Latency

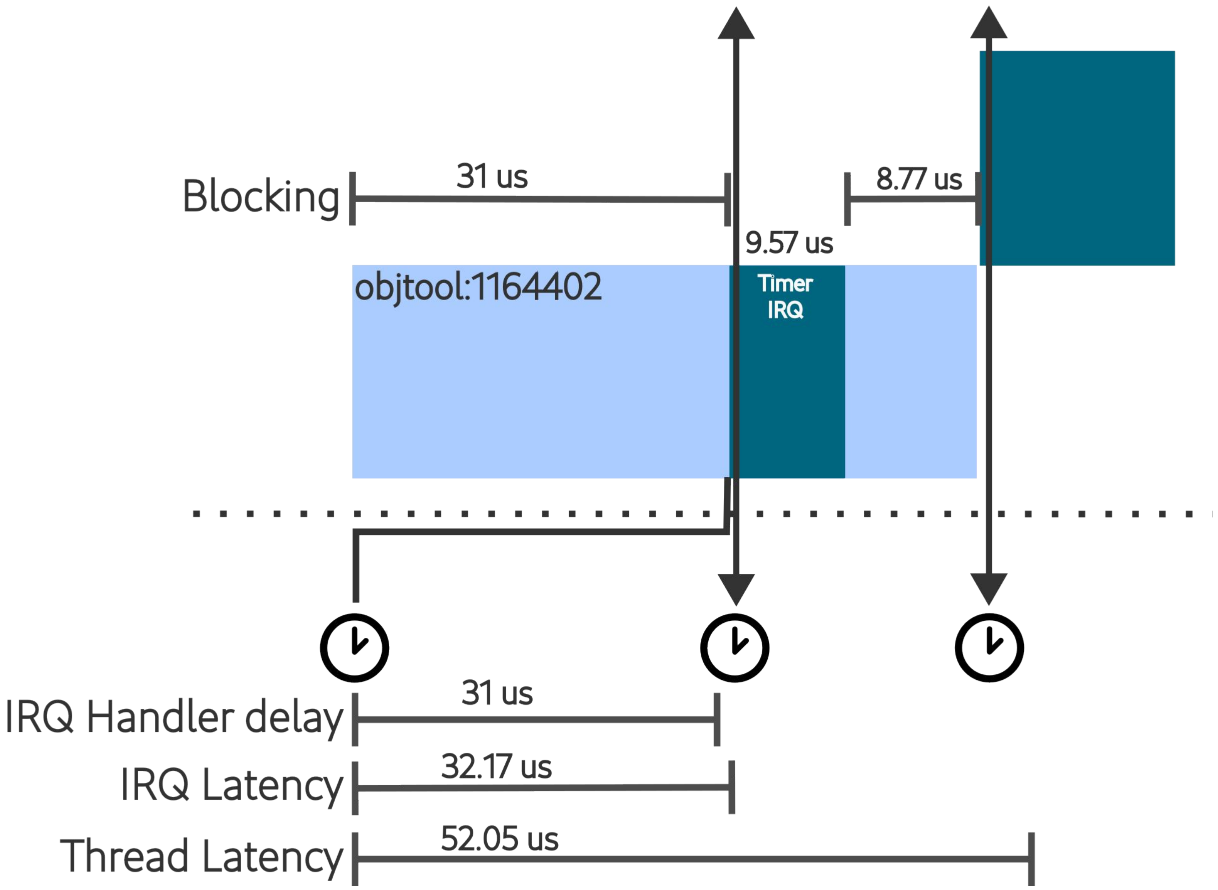
```
## CPU 6 hit stop tracing, analyzing it ##
  IRQ handler delay:                      31.00 us (59.56 %)
  IRQ latency:                            32.17 us
  Timerlat IRQ duration:                   9.57 us (18.38 %)
  Blocking thread:                         8.77 us (16.84 %)
                    objtool:1164402        8.77 us
    Blocking thread stack trace
          -> timerlat_irq
          -> __hrtimer_run_queues
          -> hrtimer_interrupt
          -> __sysvec_apic_timer_interrupt
          -> sysvec_apic_timer_interrupt
          -> asm_sysvec_apic_timer_interrupt
          -> _raw_spin_unlock_irqrestore
          -> cgroup_rstat_flush_locked
          -> cgroup_rstat_flush_irqsafe
          -> mem_cgroup_flush_stats
          -> mem_cgroup_wb_stats
          -> balance_dirty_pages
          -> balance_dirty_pages_ratelimited_flags
          -> btrfs_buffered_write
          -> btrfs_do_write_iter
          -> vfs_write
          -> __x64_sys_pwrite64
          -> do_syscall_64
          -> entry_SYSCALL_64_after_hwframe
------------------------------------------------------------------------
  Thread latency:                         52.05 us (100%)

Max timerlat IRQ latency from idle: 19.93 us in cpu 12
  Saving trace to timerlat_trace.txt
```



```
## CPU 6 hit stop tracing, analyzing it ##
  IRQ handler delay:                            31.00 us (59.56 %)
  IRQ latency:                                  32.17 us
  Timerlat IRQ duration:                         9.57 us (18.38 %)
  Blocking thread:                               8.77 us (16.84 %)
                    objtool:1164402              8.77 us
    Blocking thread stack trace
           -> timerlat_irq
           -> __hrtimer_run_queues
           -> hrtimer_interrupt
           -> __sysvec_apic_timer_interrupt
           -> sysvec_apic_timer_interrupt
           -> asm_sysvec_apic_timer_interrupt
           -> _raw_spin_unlock_irqrestore
           -> cgroup_rstat_flush_locked
           -> cgroup_rstat_flush_irqsafe
           -> mem_cgroup_flush_stats
           -> mem_cgroup_wb_stats
           -> balance_dirty_pages
           -> balance_dirty_pages_ratelimited_flags
           -> btrfs_buffered_write
           -> btrfs_do_write_iter
           -> vfs_write
           -> __x64_sys_pwrite64
           -> do_syscall_64
           -> entry_SYSCALL_64_after_hwframe
------------------------------------------------------------------------
  Thread latency:                               52.05 us (100%)

Max timerlat IRQ latency from idle: 19.93 us in cpu 12
  Saving trace to timerlat_trace.txt
```



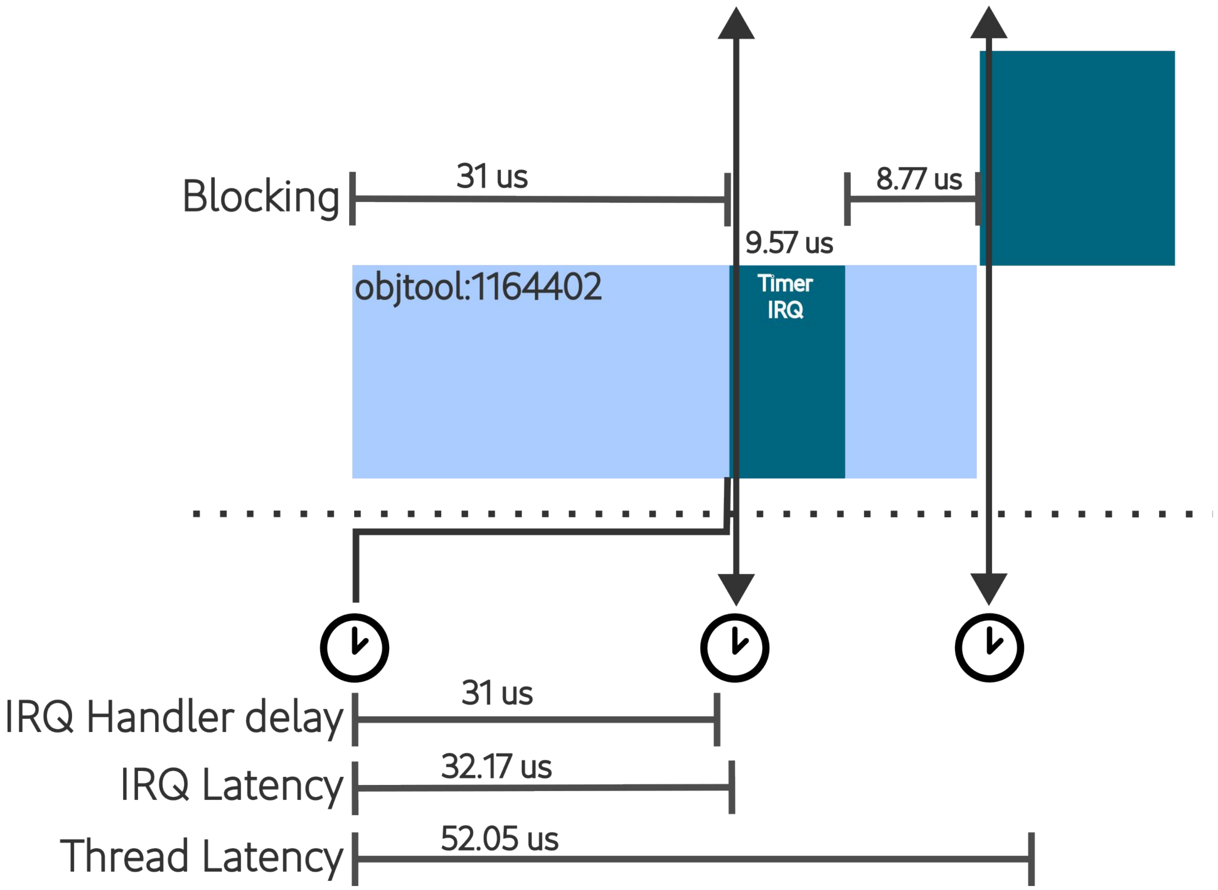
```
## CPU 6 hit stop tracing, analyzing it ##
  IRQ handler delay:                      31.00 us (59.56 %)
  IRQ latency:                            32.17 us
  Timerlat IRQ duration:                   9.57 us (18.38 %)
  Blocking thread:                         8.77 us (16.84 %)
                    objtool:1164402        8.77 us
    Blocking thread stack trace
          -> timerlat_irq
          -> __hrtimer_run_queues
          -> hrtimer_interrupt
          -> __sysvec_apic_timer_interrupt
          -> sysvec_apic_timer_interrupt
          -> asm_sysvec_apic_timer_interrupt
          -> _raw_spin_unlock_irqrestore
          -> cgroup_rstat_flush_locked
          -> cgroup_rstat_flush_irqsafe
          -> mem_cgroup_flush_stats
          -> mem_cgroup_wb_stats
          -> balance_dirty_pages
          -> balance_dirty_pages_ratelimited_flags
          -> btrfs_buffered_write
          -> btrfs_do_write_iter
          -> vfs_write
          -> __x64_sys_pwrite64
          -> do_syscall_64
          -> entry_SYSCALL_64_after_hwframe
------------------------------------------------------------------------
  Thread latency:                         52.05 us (100%)

Max timerlat IRQ latency from idle: 19.93 us in cpu 12
  Saving trace to timerlat_trace.txt
```



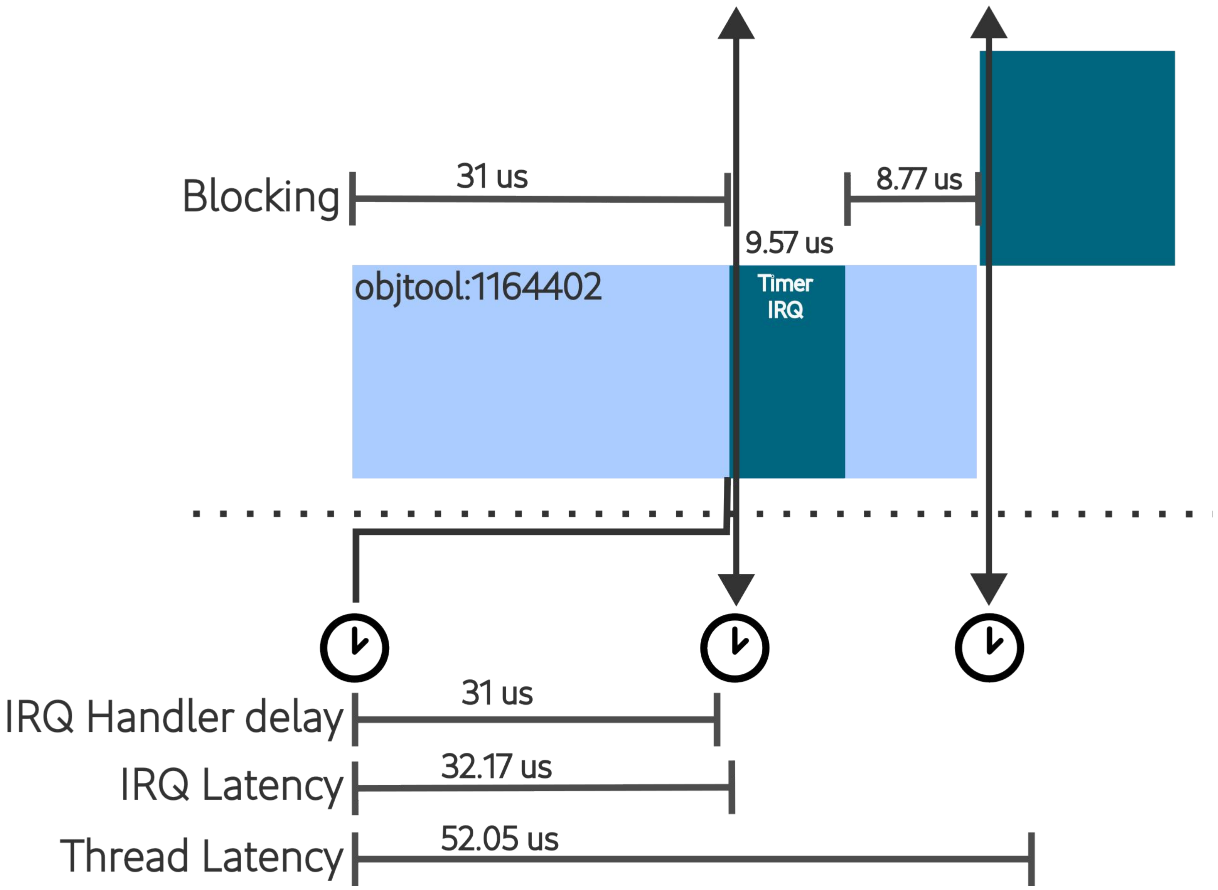
rtla timerlat

Debugging Real-time Linux Scheduling Latency

```
## CPU 6 hit stop tracing, analyzing it ##
  IRQ handler delay:                    31.00 us (59.56 %)
  IRQ latency:                          32.17 us
  Timerlat IRQ duration:                 9.57 us (18.38 %)
  Blocking thread:                       8.77 us (16.84 %)
                     objtool:1164402     8.77 us
    Blocking thread stack trace
           -> timerlat_irq
           -> __hrtimer_run_queues
           -> hrtimer_interrupt
           -> __sysvec_apic_timer_interrupt
           -> sysvec_apic_timer_interrupt
           -> asm_sysvec_apic_timer_interrupt
           -> _raw_spin_unlock_irqrestore
           -> cgroup_rstat_flush_locked
           -> cgroup_rstat_flush_irqsafe
           -> mem_cgroup_flush_stats
           -> mem_cgroup_wb_stats
           -> balance_dirty_pages
           -> balance_dirty_pages_ratelimited_flags
           -> btrfs_buffered_write
           -> btrfs_do_write_iter
           -> vfs_write
           -> __x64_sys_pwrite64
           -> do_syscall_64
           -> entry_SYSCALL_64_after_hwframe
------------------------------------------------------------------------
  Thread latency:                       52.05 us (100%)

Max timerlat IRQ latency from idle: 19.93 us in cpu 12
  Saving trace to timerlat_trace.txt
```



```
## CPU 6 hit stop tracing, analyzing it ##
  IRQ handler delay:                      31.00 us (59.56 %)
  IRQ latency:                            32.17 us
  Timerlat IRQ duration:                   9.57 us (18.38 %)
  Blocking thread:                         8.77 us (16.84 %)
                    objtool:1164402        8.77 us
    Blocking thread stack trace
          -> timerlat_irq
          -> __hrtimer_run_queues
          -> hrtimer_interrupt
          -> __sysvec_apic_timer_interrupt
          -> sysvec_apic_timer_interrupt
          -> asm_sysvec_apic_timer_interrupt
          -> _raw_spin_unlock_irqrestore
          -> cgroup_rstat_flush_locked      ????
          -> cgroup_rstat_flush_irqsafe
          -> mem_cgroup_flush_stats
          -> mem_cgroup_wb_stats
          -> balance_dirty_pages
          -> balance_dirty_pages_ratelimited_flags
          -> btrfs_buffered_write
          -> btrfs_do_write_iter
          -> vfs_write
          -> __x64_sys_pwrite64
          -> do_syscall_64
          -> entry_SYSCALL_64_after_hwframe
------------------------------------------------------------------------
  Thread latency:                         52.05 us (100%)

Max timerlat IRQ latency from idle: 19.93 us in cpu 12
  Saving trace to timerlat_trace.txt
```



```
## CPU 6 hit stop tracing, analyzing it ##
  IRQ handler delay:                          31.00 us (59.56 %)
  IRQ latency:                                32.17 us
  Timerlat IRQ duration:                       9.57 us (18.38 %)
  Blocking thread:                             8.77 us (16.84 %)
                    objtool:1164402            8.77 us
    Blocking thread stack trace
          -> timerlat_irq
          -> __hrtimer_run_queues
          -> hrtimer_interrupt
          -> __sysvec_apic_timer_interrupt
          -> sysvec_apic_timer_interrupt
          -> asm_sysvec_apic_timer_interrupt
          -> _raw_spin_unlock_irqrestore
          -> cgroup_rstat_flush_locked     ????
          -> cgroup_rstat_flush_irqsafe
          -> mem_cgroup_flush_stats
          -> mem_cgroup_wb_stats
          -> balance_dirty_pages
          -> balance_dirty_pages_ratelimited_flags
          -> btrfs_buffered_write
          -> btrfs_do_write_iter
          -> vfs_write
          -> __x64_sys_pwrite64
          -> do_syscall_64
          -> entry_SYSCALL_64_after_hwframe
------------------------------------------------------------------------
  Thread latency:                             52.05 us (100%)

Max timerlat IRQ latency from idle: 19.93 us in cpu 12
  Saving trace to timerlat_trace.txt
```

```
From: Sebastian Andrzej Siewior @ 2022-03-01 12:21 UTC (permalink / raw)
  To: cgroups, linux-mm
  Cc: Andrew Morton, Johannes Weiner, Tejun Heo, Zefan Li,
        Thomas Gleixner, Sebastian Andrzej Siewior

All callers of cgroup_rstat_flush_locked() acquire cgroup_rstat_lock
either with spin_lock_irq() or spin_lock_irqsave().
cgroup_rstat_flush_locked() itself acquires cgroup_rstat_cpu_lock which
is a raw_spin_lock. This lock is also acquired in cgroup_rstat_updated()
in IRQ context and therefore requires _irqsave() locking suffix in
cgroup_rstat_flush_locked().
Since there is no difference between spin_lock_t and raw_spin_lock_t
on !RT lockdep does not complain here. On RT lockdep complains because
the interrupts were not disabled here and a deadlock is possible.

Acquire the raw_spin_lock_t with disabled interrupts.

Signed-off-by: Sebastian Andrzej Siewior <bigeasy@linutronix.de>
---
 kernel/cgroup/rstat.c | 5 +++--
 1 file changed, 3 insertions(+), 2 deletions(-)

diff --git a/kernel/cgroup/rstat.c b/kernel/cgroup/rstat.c
index 9d331ba44870a..53b771c20ee50 100644
--- a/kernel/cgroup/rstat.c
+++ b/kernel/cgroup/rstat.c
@@ -153,8 +153,9 @@ static void cgroup_rstat_flush_locked(struct cgroup *cgrp, bool may_sleep)
 		raw_spinlock_t *cpu_lock = per_cpu_ptr(&cgroup_rstat_cpu_lock,
 						       cpu);
 		struct cgroup *pos = NULL;
+		unsigned long flags;
 
-		raw_spin_lock(cpu_lock);
+		raw_spin_lock_irqsave(cpu_lock, flags);
 		while ((pos = cgroup_rstat_cpu_pop_updated(pos, cgrp, cpu))) {
 			struct cgroup_subsys_state *css;
 
@@ -166,7 +167,7 @@ static void cgroup_rstat_flush_locked(struct cgroup *cgrp, bool may_sleep)
 				css->ss->css_rstat_flush(css, cpu);
 			rcu_read_unlock();
 		}
-		raw_spin_unlock(cpu_lock);
+		raw_spin_unlock_irqrestore(cpu_lock, flags);
 
 		/* if @may_sleep, play nice and yield if necessary */
 		if (may_sleep && (need_resched() ||
-- 
2.35.1
```

- IRQ Release jitter
  - IRQ delayed because of hw
- idle setup is required
  - e.g., limiting idle states
- rtla workaround
  - **--dma-latency 0** option

```
## CPU 9 hit stop tracing, analyzing it ##
  IRQ handler delay: (exit from idle) 39.01 us (76.59 %)
  IRQ latency:                         40.49 us
  Timerlat IRQ duration:                5.85 us (11.49 %)
  Blocking thread:                      3.99 us (7.83 %)
                   swapper/9:0          3.99 us
    Blocking thread stack trace
         -> timerlat_irq
         -> __hrtimer_run_queues
         -> hrtimer_interrupt
         -> __sysvec_apic_timer_interrupt
         -> sysvec_apic_timer_interrupt
         -> asm_sysvec_apic_timer_interrupt
         -> pv_native_safe_halt
         -> default_idle
         -> default_idle_call
         -> do_idle
         -> cpu_startup_entry
         -> start_secondary
         -> __pfx_verify_cpu
------------------------------------------------------------
  Thread latency:                      50.93 us (100%)

Max timerlat IRQ latency from idle: 40.49 us in cpu 9
```

Thread example

```
  IRQ handler delay:                                  0.00 us (0.00 %)
  IRQ latency:                                        1.64 us
  Timerlat IRQ duration:                              9.52 us (1.80 %)
  Blocking thread:                                  501.68 us (94.96 %)
                 kworker/u40:0:306130               501.68 us
    Blocking thread stack trace
           -> timerlat_irq
[...]
           -> asm_sysvec_apic_timer_interrupt
           -> ZSTD_compressBlock_fast
           -> ZSTD_buildSeqStore
           -> ZSTD_compressBlock_internal
[...]
           -> zstd_compress_pages
           -> btrfs_compress_pages
           -> compress_file_range
           -> async_cow_start
           -> btrfs_work_helper
           -> process_one_work
           -> worker_thread
           -> kthread
           -> ret_from_fork
  IRQ interference                                    3.68 us (0.70 %)
                   local_timer:236                    3.68 us
  Softirq interference                                4.21 us (0.80 %)
                         TIMER:1                      3.71 us
                           RCU:9                      0.49 us
  Thread interference                                 6.21 us (1.17 %)
                  migration/18:125                    6.21 us
------------------------------------------------------------------------
  Thread latency:                                   528.31 us (100%)
```

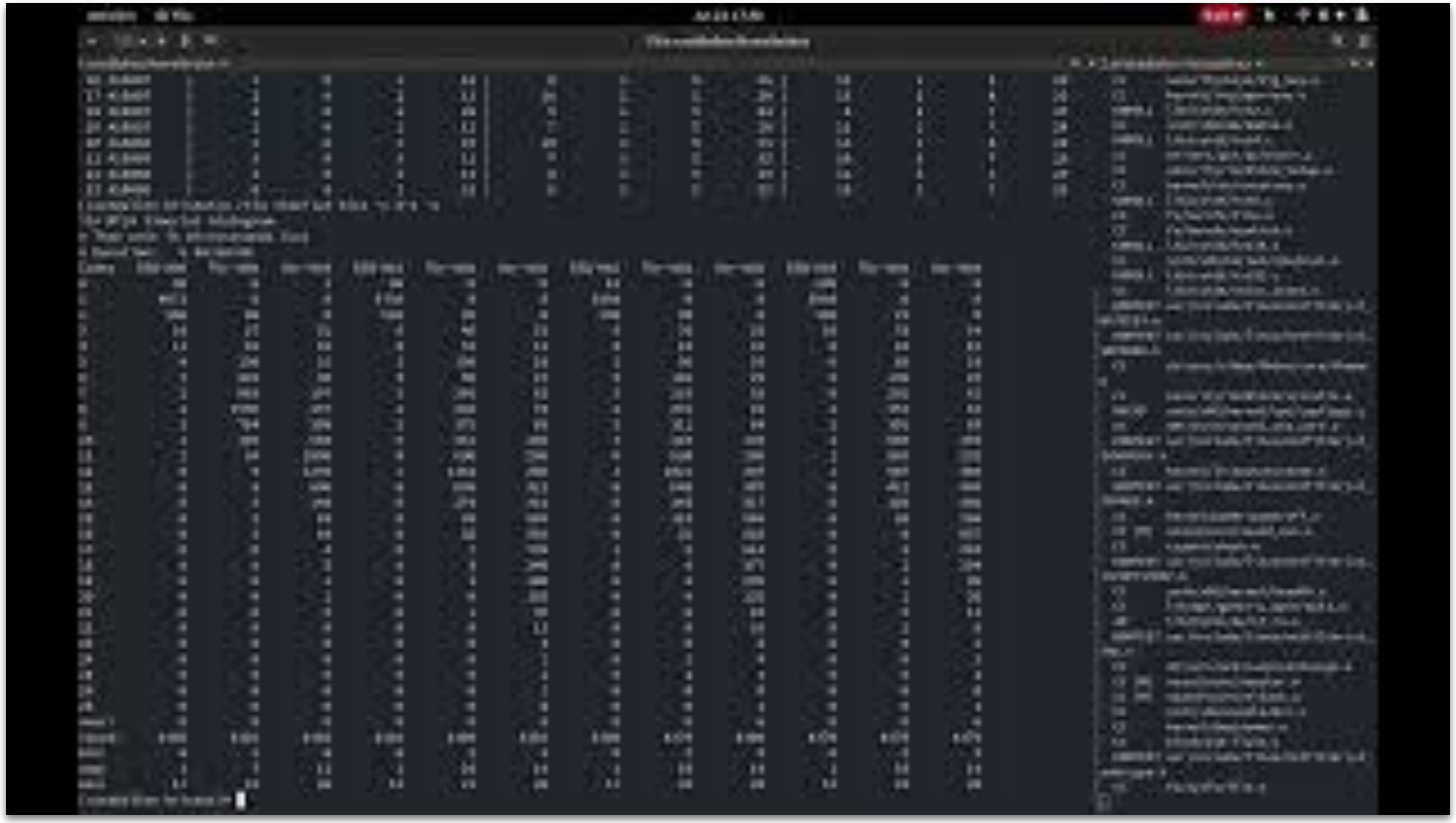
```
  IRQ handler delay:                                0.00 us (0.00 %)
  IRQ latency:                                      1.64 us
  Timerlat IRQ duration:                            9.52 us (1.80 %)
  Blocking thread:                                501.68 us (94.96 %)
                 kworker/u40:0:306130             501.68 us
    Blocking thread stack trace
           -> timerlat_irq
[...]
           -> asm_sysvec_apic_timer_interrupt
           -> ZSTD_compressBlock_fast
           -> ZSTD_buildSeqStore
           -> ZSTD_compressBlock_internal
[...]
           -> zstd_compress_pages
           -> btrfs_compress_pages
           -> compress_file_range
           -> async_cow_start
           -> btrfs_work_helper
           -> process_one_work
           -> worker_thread
           -> kthread
           -> ret_from_fork
  IRQ interference                                  3.68 us (0.70 %)
                   local_timer:236                  3.68 us
  Softirq interference                              4.21 us (0.80 %)
                         TIMER:1                    3.71 us
                           RCU:9                    0.49 us
  Thread interference                               6.21 us (1.17 %)
                  migration/18:125                  6.21 us
------------------------------------------------------------------------
  Thread latency:                                 528.31 us (100%)
```

```
  IRQ handler delay:                               0.00 us (0.00 %)
  IRQ latency:                                     1.64 us
  Timerlat IRQ duration:                           9.52 us (1.80 %)
  Blocking thread:                               501.68 us (94.96 %)
                 kworker/u40:0:306130            501.68 us
    Blocking thread stack trace
            -> timerlat_irq
[...]
            -> asm_sysvec_apic_timer_interrupt
            -> ZSTD_compressBlock_fast
            -> ZSTD_buildSeqStore
            -> ZSTD_compressBlock_internal
[...]
            -> zstd_compress_pages
            -> btrfs_compress_pages
            -> compress_file_range
            -> async_cow_start
            -> btrfs_work_helper
            -> process_one_work
            -> worker_thread
            -> kthread
            -> ret_from_fork
  IRQ interference                                 3.68 us (0.70 %)
                   local_timer:236                 3.68 us
  Softirq interference                             4.21 us (0.80 %)
                         TIMER:1                   3.71 us
                           RCU:9                   0.49 us
  Thread interference                              6.21 us (1.17 %)
                  migration/18:125                 6.21 us
------------------------------------------------------------------------
  Thread latency:                                528.31 us (100%)
```

rtla timerlat tracing

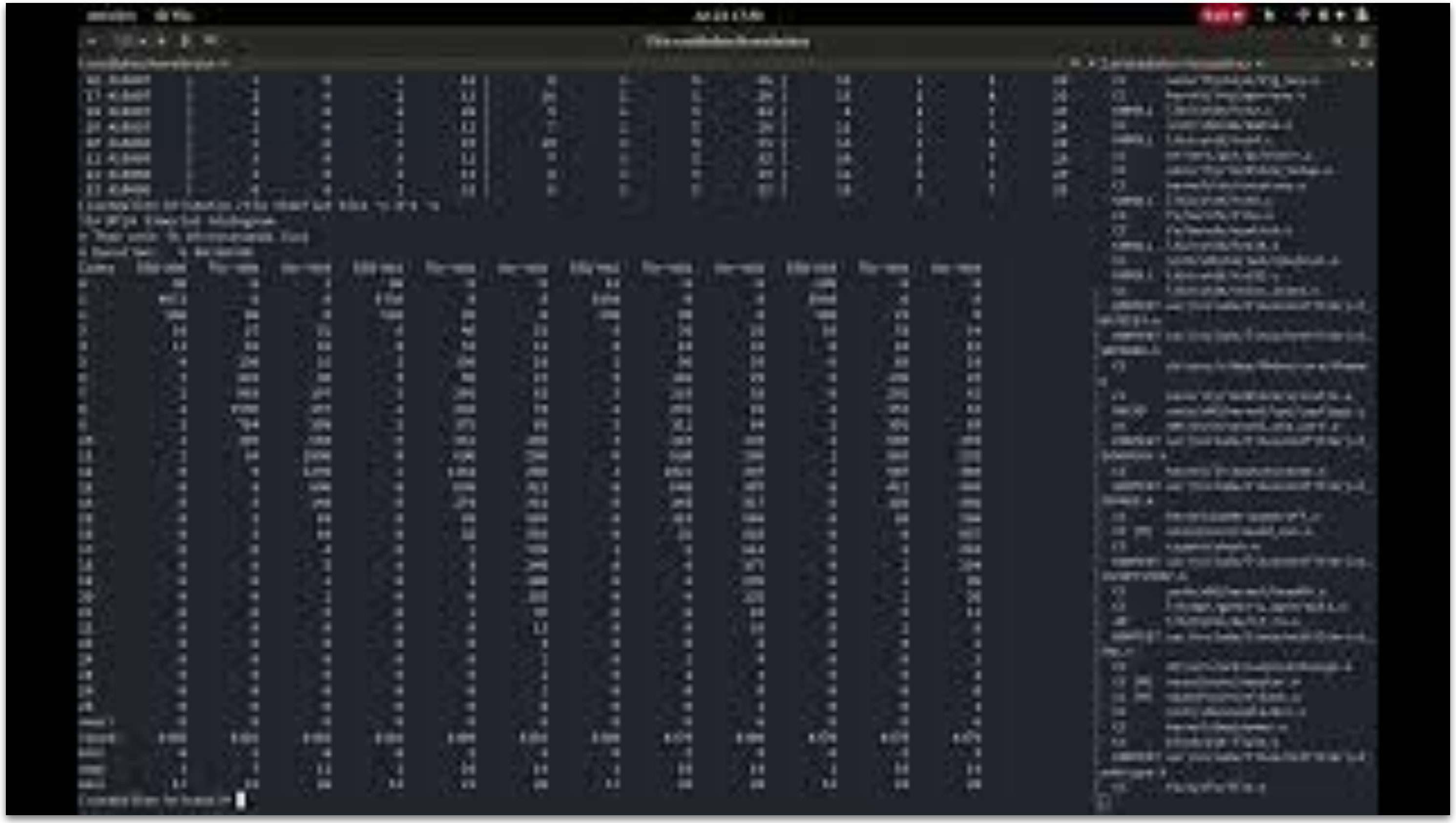
- rtla timerlat is a front-end for the timerlat tracer
- the tracer activates the osnoise: tracepoints
  - They report the amount of blocking and interference
- One tracepoint for each task
  - osnoise:**nmi_noise**
  - osnoise:**irq_noise**
  - osnoise:**softirq_noise**
  - osnoise:**thread_noise**
  - The values are free from nested interference
    - e.g., a thread_noise is free from any IRQ/Softirq/NMI interference that it could face

- Timerlat auto-analysis & trace



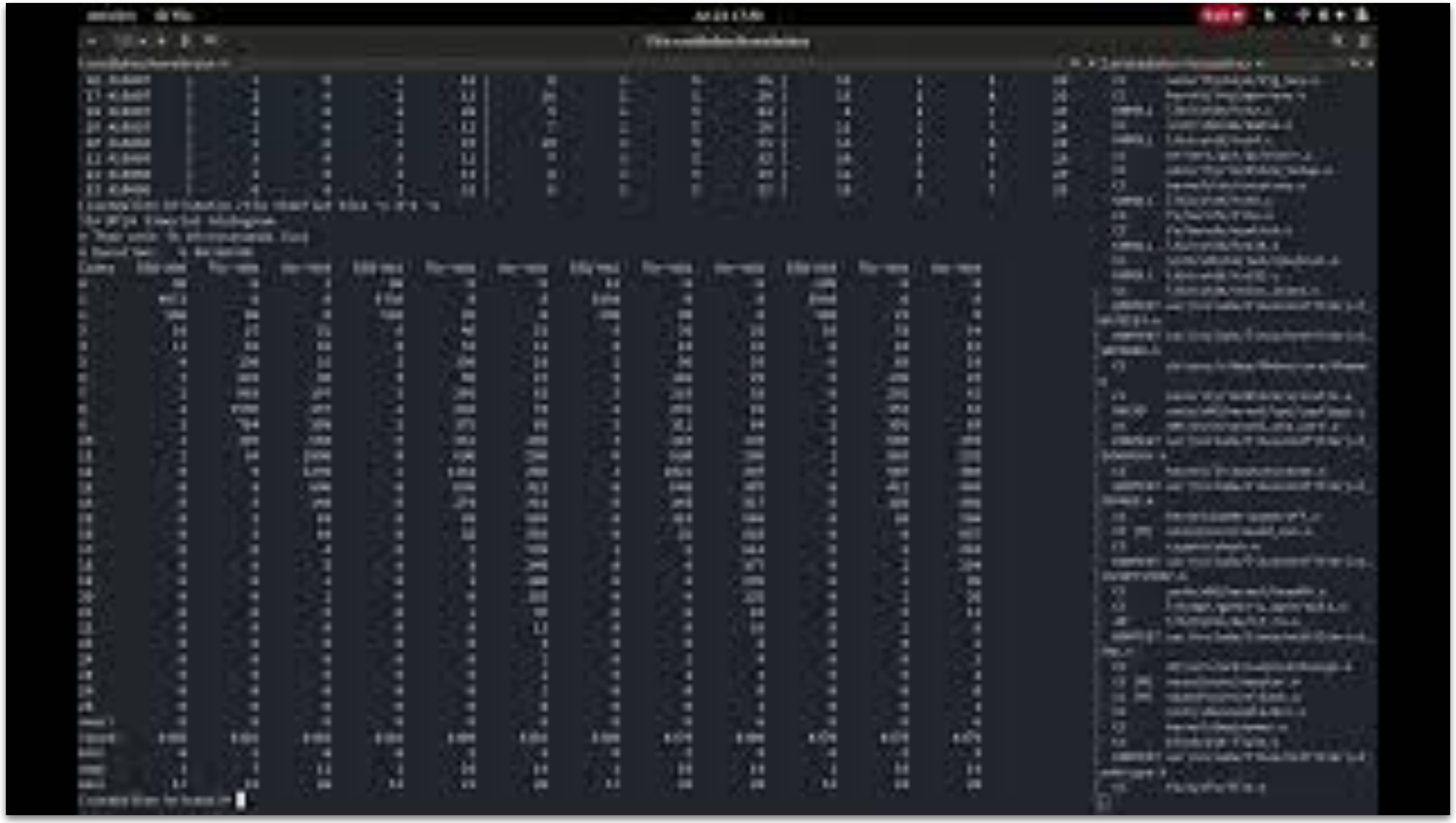
- rtla timerlat can also be used to enable other tracing features
  - **-e** tracepoint: enables a tracepoint
  - **--filter**: filters the previous -e tracepoint
  - **--trigger**: activates a trigger for the previous -e tracepoint

- Timerlat auto-analysis



- It is possible to leverage the osnoise: tracepoints to collect histograms for the sources of interference & blocking
- Example of histogram --trigger
  - https://bristot.me/rtla-histograms/

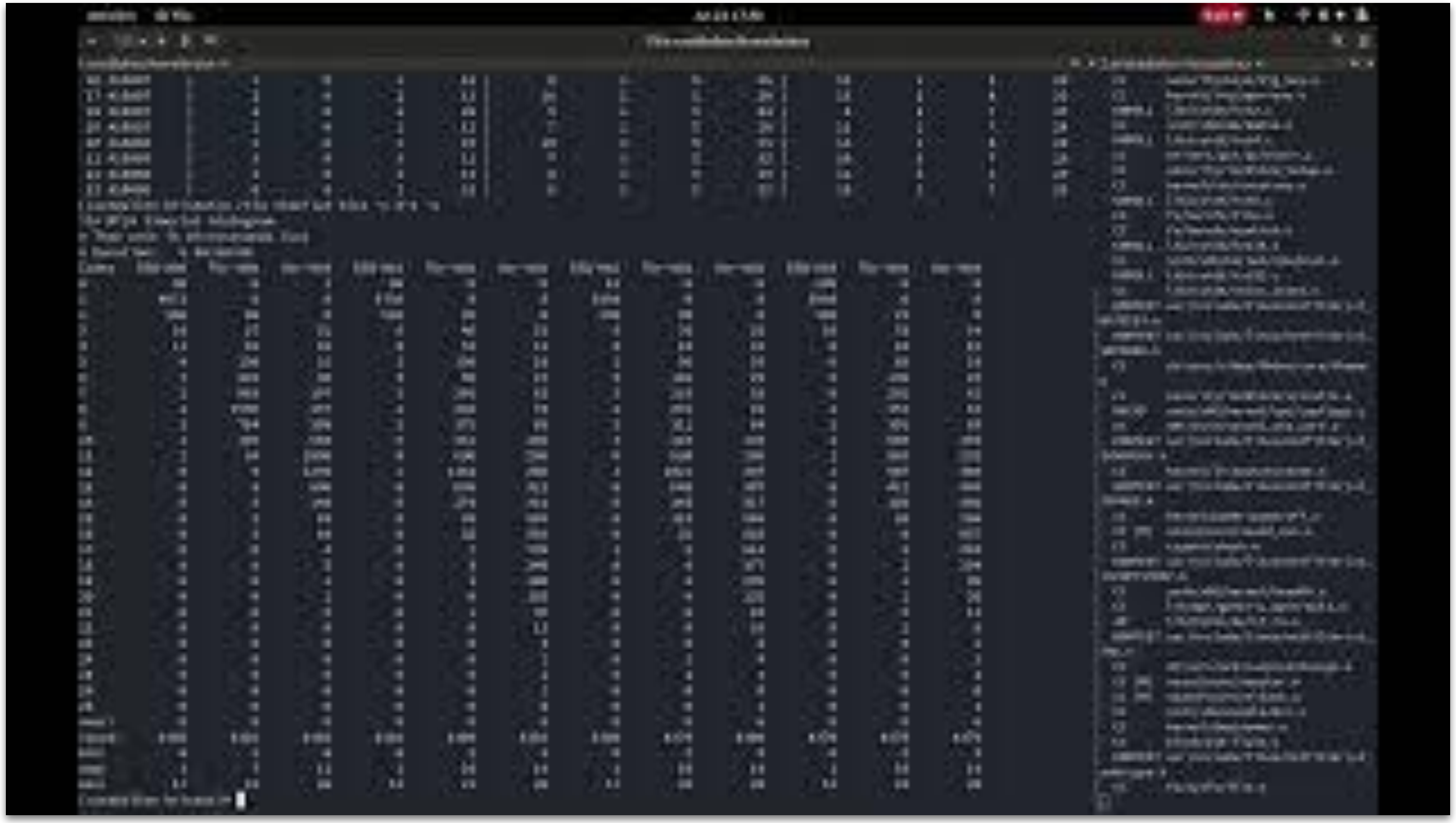
- Timerlat histograms



rtla timerlat -u

- any user-space workload is now supported
- **timerlat exposes a fd where a thread can sleep** waiting for a period in a loop.
  - **timerlat activates and traces the IRQ and Thread latency.**
- If the thread returns to kernel-space, timerlat prints the return to user-space
  - this can be used to measure the **kernel-user-kernel** latency
  - or to report the response time for a task!
  - the kernel tracer works for any workload, rtla dispatches its own.

- timerlat user-space



btw...

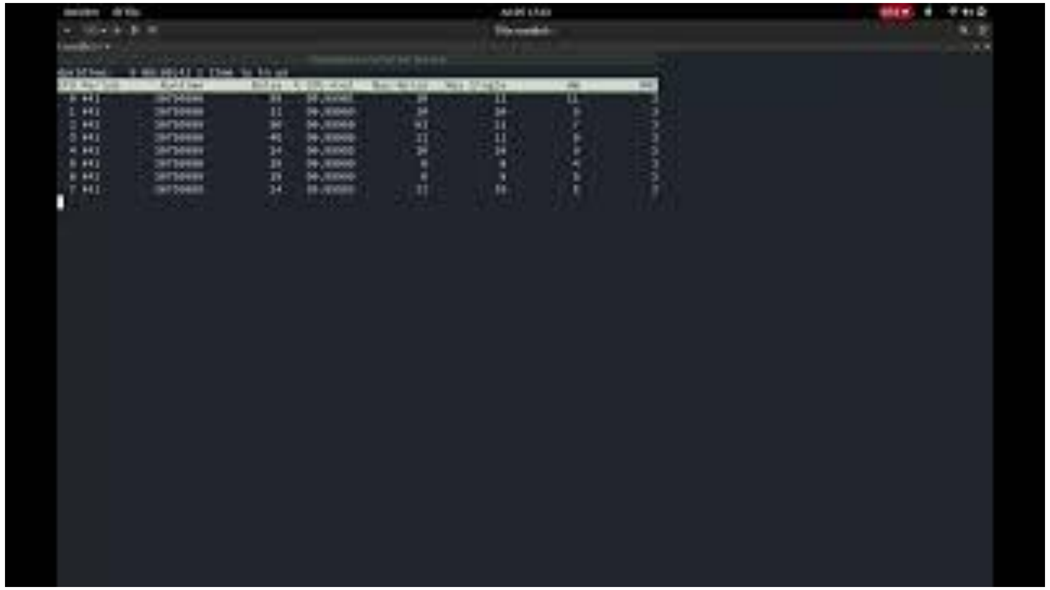
- rtla timerlat has a set of config options:
  - **-p/--period us**: timerlat period in us
  - **-c/--cpus cpus**: run the tracer only on the given cpus
  - **-d/--duration time[m|h|d]**: duration of the session in seconds
  - **-D/--debug**: print debug info
  - **-P/--priority**: set scheduling parameters
    - **o:0** use SCHED_OTHER with *nice*
    - **r:prio** use SCHED_RR with priority
    - **f:prio** use SCHED_FIFO with priority
    - **d:runtime[us|ms]:period[us|ms]** use SCHED_DEADLINE

- **-H/--house-keeping cpus**: run rtla control threads only on the given cpus
- **-C/--cgroup[=cgroup]**: set cgroup, if no cgroup is passed, the rtla's cgroup will be inherited
- **--dma-latency us**: set /dev/cpu_dma_latency latency <us>
- **--aa-only us**: stop if <us> latency is hit, only printing the auto-analysis
- **--no-aa**: disable auto-analysis, reducing rtla timerlat cpu usage
- **--dump-tasks**: on auto analysis, prints the task running on all CPUs if stop
- **-t/--trace[=file]:** save the stopped trace to [file|timerlat_trace.txt]
- **-i/--irq us:** stop trace if the irq latency is higher than the argument in us
- **-T/--thread us:** stop trace if the thread latency is higher than the argument in us
- **-s/--stack us:** save the stack trace at the IRQ printing if a thread latency is higher than the argument in us

rtla timerlat

Debugging Real-time Linux Scheduling Latency

- btw... run hwnoise before starting with timerlat
- rtla hwnoise measures the ... hw noise :)
- The latency is always, at least, the hw noise long



Final remarks

- rtla timerlat integrates workload, tracing and auto-analysis in a single tool!
- it produces an summary of the root cause for latency spikes
  - that is a good starting point for the analysis, even for a non-expert
- the tool also allows the usage of more advanced tracing

- rtla is the home of other tools for rt analysis
  - timerlat: scheduling latency via sampling
  - osnoise: operating system noise
  - hwnoise: hardware related noise
- it can only get better...
  - execution time tracer
  - IRQ noise/execution time
  - the worst case scheduling latency (formally proof)
  - Integration with KVM
  - ... and whatever the community needs

- A tutorial-like version of this talk can be found here:
  - https://bristot.me/linux-scheduling-latency-debug-and-analysis/

Thanks! questions?