Red Hat
Enterprise Linux

RTLA TODOs and requests

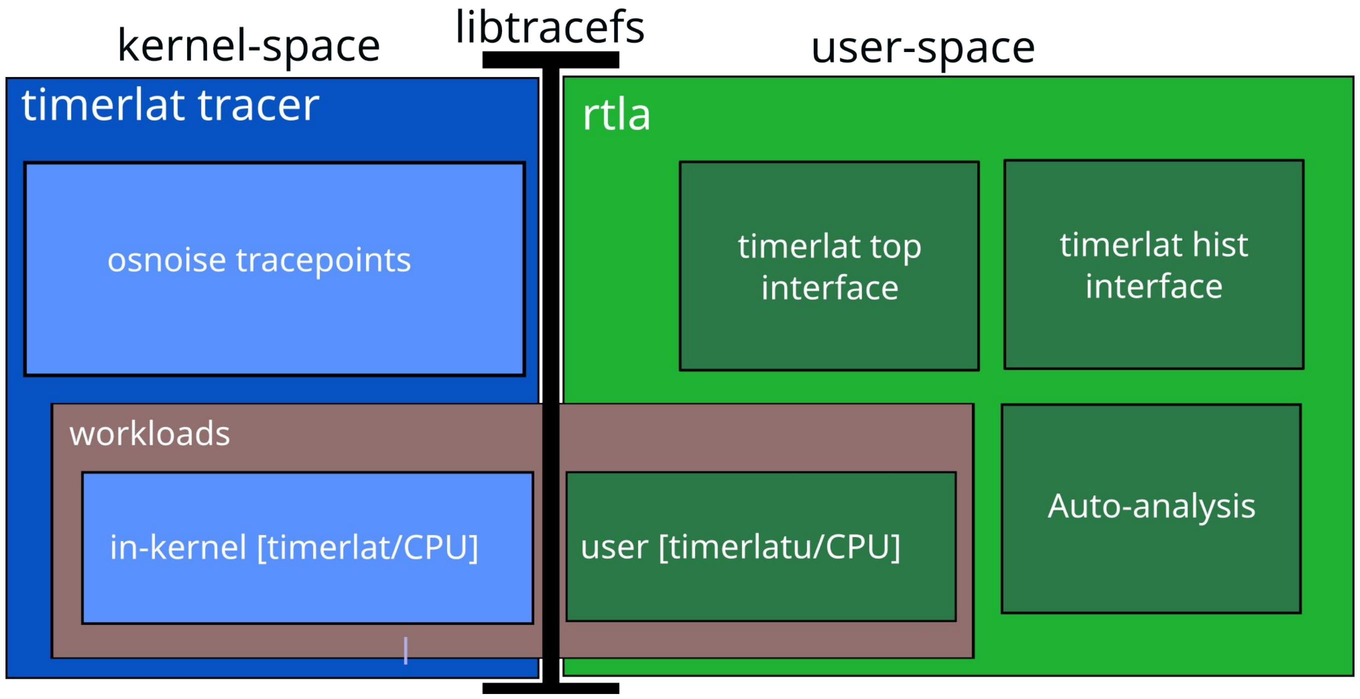
Tracing MC - LPC 2023

Daniel Bristot de Oliveira, Ph.D.
Senior Principal Software Engineer

RTLA & kernel tracers

- rtla is a suite aiming to give real-time users a set of tools to facilitate and automate the analysis
- rtla is an user-space binary that controls and parses (in-kernel) tracers
- It has three tools inside:
  - rtla timerlat
    - backed-by: timerlat tracer
  - osnoise
    - backed-by: osnoise tracer
  - hwnoise
    - backed-by: osnoise tracer with IRQs disabled (hwlat 2.0).

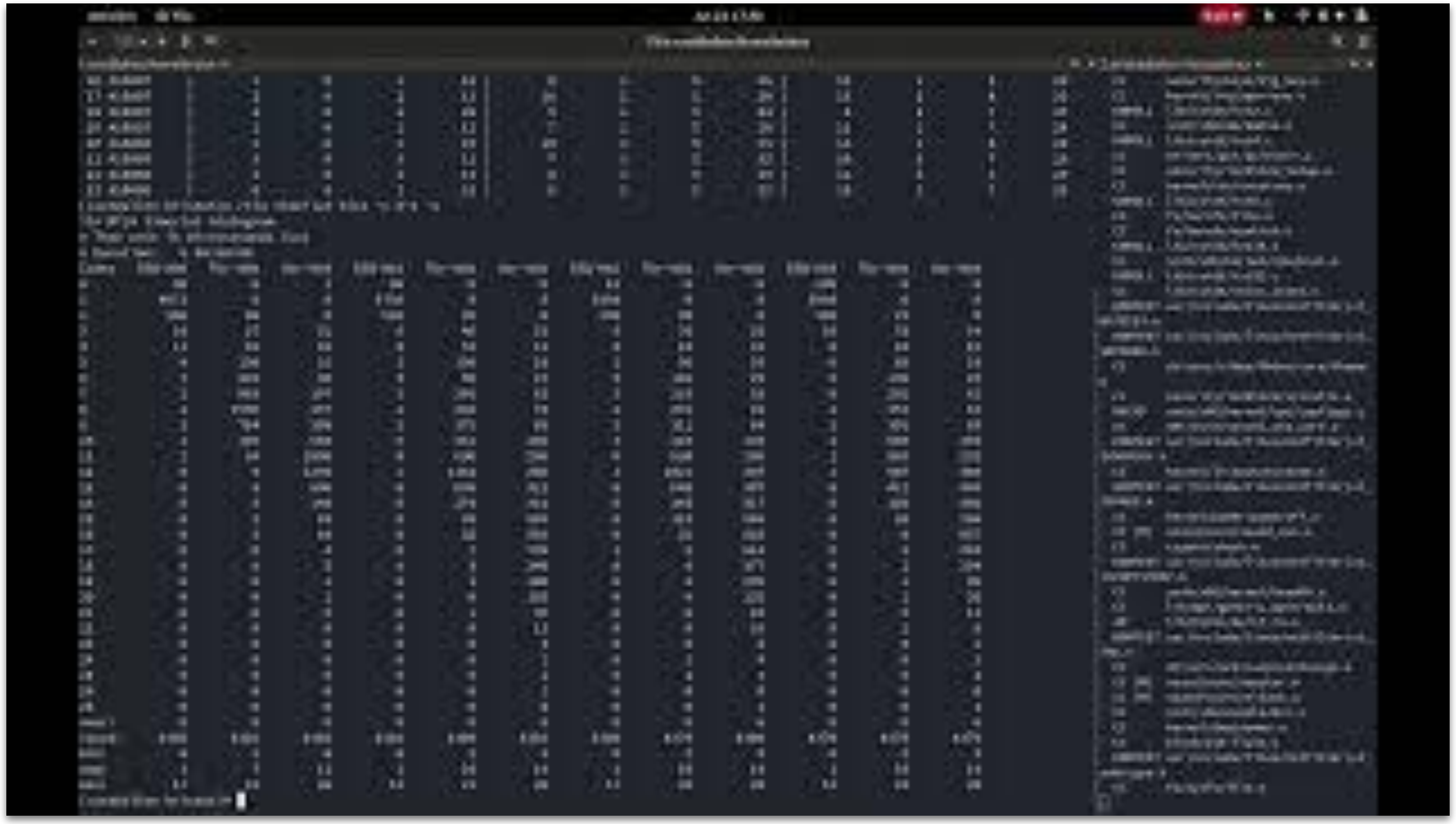
rtla timerlat components:



Red Hat

RTLA & kernel tracers

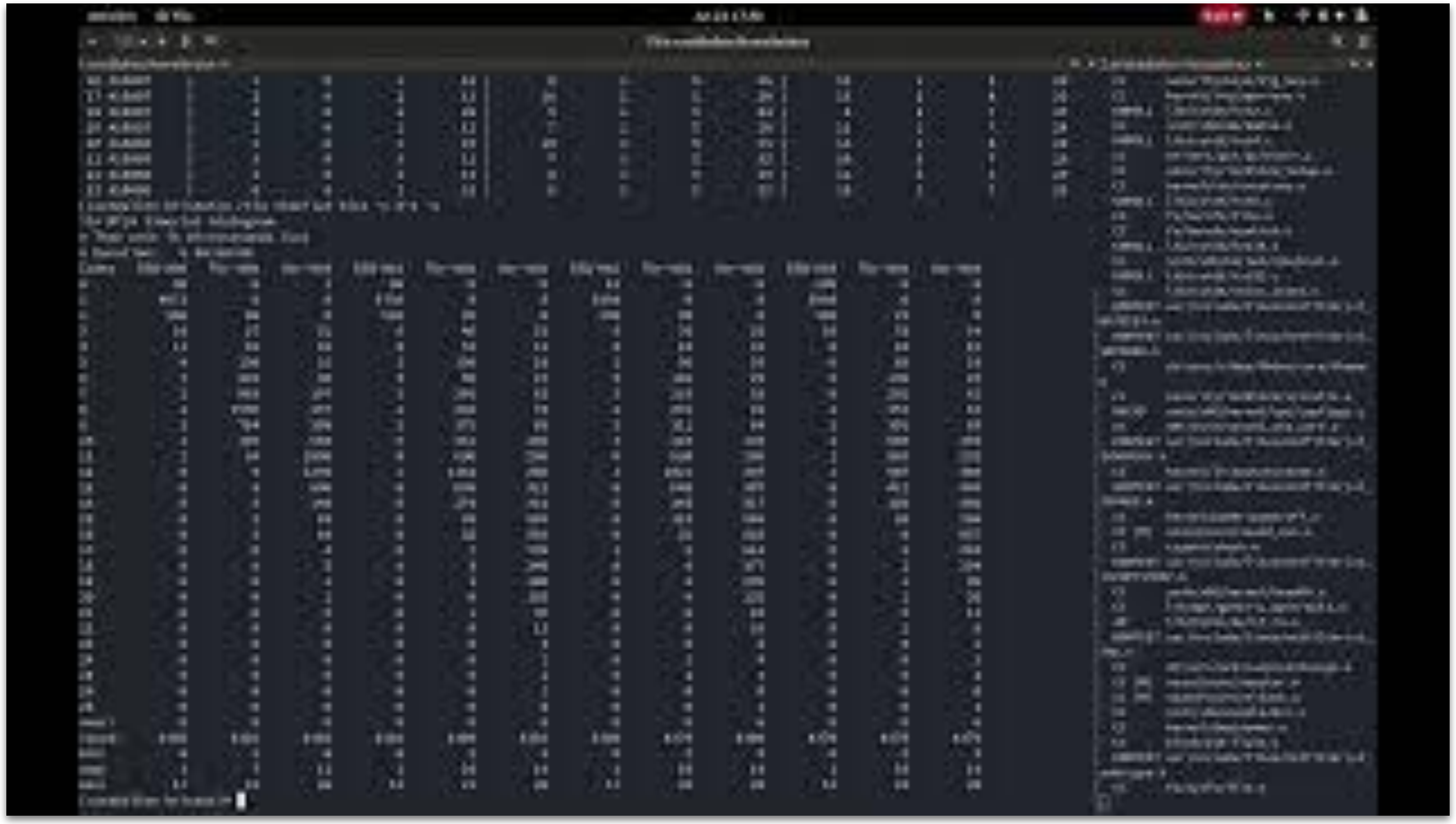
- rtla timerlat auto analysis example



```
[illegible]
```

RTLA & kernel tracers

- rtla timerlat and others options of tracing



```
[illegible]
```

RTLA TODOs

- The osnoise tracer has a workload (busy-loop per CPU) and a set of tracepoints to measure execution time
  - We can run osnoise tracer without the work
  - We can extend it to work with (any) user-space workload adding auto-analysis
    - **Need to find a way to sync a per-cpu variable with user-space**
  - Add ipi root cause analysis (goooooo Valentin!)

```
[root@x1 bristot]# cd /sys/kernel/debug/tracing/ && echo osnoise > set_event && echo NO_OSNOISE_WORKLOAD > osnoise/options && echo osnoise > current_tracer
[root@x1 tracing]# cat trace
# tracer: osnoise
[...]
#                              ||| / _-=> migrate-disable                         MAX
#                              |||| /     delay                                   SINGLE      Interference counters:
#                              |||||               RUNTIME      NOISE  %% OF CPU  NOISE    +------------------------------+
#           TASK-PID      CPU# |||||   TIMESTAMP    IN US       IN US  AVAILABLE  IN US     HW    NMI    IRQ   SIRQ THREAD
#              | |         |   |||||      |           |             |    |            |      |      |      |      |      |
          <idle>-0       [011] d..3. 34832.839504: thread_noise: swapper/11:0 start 0.000000000 duration 34832839502655 ns
 ibus-engine-sim-4045    [011] d..3. 34832.839543: thread_noise: ibus-engine-sim:4045 start 34832.839505329 duration 37043 ns
          <idle>-0       [007] d..3. 34832.850596: thread_noise: swapper/7:0 start 0.000000000 duration 34832850595038 ns
          chrome-30840   [007] d.h1. 34832.851167: irq_noise: local_timer:236 start 34832.851151387 duration 15422 ns
          chrome-30840   [007] ..s1. 34832.851173: softirq_noise:    SCHED:7 start 34832.851168367 duration 4410 ns
          chrome-30840   [007] ..s1. 34832.851175: softirq_noise:      RCU:9 start 34832.851173714 duration 804 ns
          chrome-30840   [007] d.h1. 34832.851322: irq_noise: call_function_single:251 start 34832.851321171 duration 910 ns
          chrome-30840   [007] d.h1. 34832.851397: irq_noise: call_function_single:251 start 34832.851396030 duration 811 ns
          chrome-30840   [007] d.h1. 34832.852153: irq_noise: local_timer:236 start 34832.852150044 duration 2723 ns
          chrome-30840   [007] ..s1. 34832.852153: softirq_noise:      RCU:9 start 34832.852152992 duration 312 ns
          chrome-30840   [007] d.h1. 34832.853153: irq_noise: local_timer:236 start 34832.853149933 duration 3297 ns
          chrome-30840   [007] d.h1. 34832.854152: irq_noise: local_timer:236 start 34832.854149908 duration 2105 ns
```

RTLA TODOs

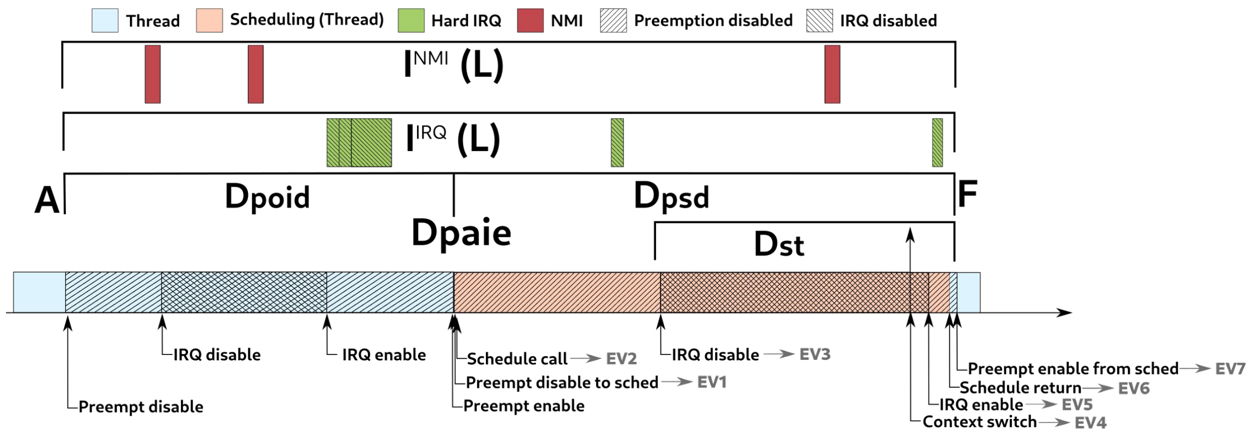
- The osnoise tracer tracepoints can be leveraged for two other purposes:
- rtla exec-time

```
    ibus-engine-sim-4045    [011] d..3. 34832.839543: thread_noise: ibus-engine-sim:4045 start 34832.839505329 duration 37043 ns
             chrome-30840   [007] d.h1. 34832.851167: irq_noise: local_timer:236 start 34832.851151387 duration 15422 ns
             chrome-30840   [007] ..s1. 34832.851173: softirq_noise:    SCHED:7 start 34832.851168367 duration 4410 ns
```

- Not only min/max/avg… But also probabilistic analysis (pWCET)
- rtla cache-noise
  - Get per-cpu counters to measure the net noise - free from other interferences
- rtla workload <params like cpu> <seed to recreate the same workload> <workload> <prioritization>
  - Parameterized synthetic workload generator
    - Pseudo-random
    - Schedulable task set generator
  - Workload other than just spinning
    - Like… using stress-ng workloads called from main()
  - osnoise/exectime/cache-noise collect

RTLA TODOs

- RTSL: the formally proved scheduling latency analysis
- It is the thing that inspired RTLA
- It gives the worst case scheduling latency!
- But it depends on preemptirq tracepoints
    - They are heavy and not enabled by default
- I need to find ways to mitigate the overheads of preemptirq tracepoints to have them enabled by default



```
Interference Free Latency:
    paie is lower than 1 us -> neglectable
    latency = max(poid,    dst) + paie +   psd
      42212 = max(22510, 19312) +    0 + 19702
Cyclictest:
    Latency =     27000 with Cyclictest
No Interrupts:
    Latency =     42212 with No Interrupts
Sporadic:
    INT:      oWCET             oMIAT
    NMI:          0                 0
     33:      16914            257130
     35:      12913              1843 <- oWCET > oMIAT
    236:      20728              1558 <- oWCET > oMIAT
    246:       3299           1910321
    Did not converge.
```

```
continuing....
Sliding window:
    Window: 42212
           NMI:        0
            33:    16914
            35:    14588
           236:    20728
           246:     3299
    Window: 97741
           236:    21029 <- new!
    Window: 98042
    Converged!
    Latency =     98042 with Sliding Window
```

Red Hat

RTLA Requests: kernel side

- Two tracers at once!
  - There are tracers that does not make sense to run together
  - But, we could run timerlat/osnoise/hwnoise with other tracers
    - Like timerlat & function
  - Is that... to hard?
  - Can we have an in-kernel "file" to merge multiple instances?
- Tracer histogram
  - We can create histograms for tracepoints, but not for tracers
  - It would be good to have histograms for timerlat
  - Add it for all tracers, or make a special file with stats for timerlat on osnoise/dir

RTLA Requests: library side

- rtla uses libtracefs
    - It enables the trace instances, set all data, set prio and parses the trace
    - It currently parses single-cpu
        - Can it parse on per-cpu file?
- Using libtrace-cmd would be better
    - rtla record to set things and save data to trace.dat
    - rtla report to report data
    - Is it possible to record with libtrace-cmd?
        - Just save a buffer...
- Find a better way to list dependencies on Makefile
    - Today we point the dependencies by hand (Linus asked us)
    - Is there another way to do this, with these new tools
        - Also for eBPF
        - How perf does it?

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat