Red Hat
Enterprise Linux

RTLA: what is next?

Daniel Bristot de Oliveira
Senior Principal Software Engineer

Red Hat

Real-time Linux Analysis!

- RTLA is a meta-tool that includes a **set of commands that aims to analyze the real-time properties of Linux**.
- **rtla is a user-space tool** that serves as the front-end for setup, tracing, and interpretation of data.
- **It is in C, hosted inside the kernel source.**
- **rtla osnoise** is an interface for osnoise tracer
  - osnoise top: shows an interactive view of the osnoise summary output
  - osnoise hist: shows a histogram of the osnoise sample tracepoint
- **rtla timerlat** is an interface for timerlat tracer
  - timerlat top: shows an interactive view of the timer latencies
  - timerlat hist: shows a histogram of the timer latencies
- People in the community are giving good feedback about them:
  - Mainly on the "hard to get back" side

It is a meta tool because it does not aim to do only one analysis, but to become a tool set, home for multiple types of analysis.

timerlat top

```
[root@alien ~]# rtla timerlat top -s 30 -t 30 -T
                Timer Latency
  0 00:00:59   |          IRQ Timer Latency (us)        |        Thread Timer Latency (us)
CPU COUNT      |      cur       min       avg       max |      cur       min       avg       max
  0 #58634     |        1         0         1        10 |       11         2        10        23
  1 #58634     |        1         0         1         9 |       12         2         9        23
  2 #58634     |        0         0         1        11 |       10         2         9        23
  3 #58634     |        1         0         1        11 |       11         2         9        24
  4 #58634     |        1         0         1        10 |       11         2         9        26
  5 #58634     |        1         0         1         8 |       10         2         9        25
  6 #58634     |       12         0         1        12 |       30         2        10        30 <--- CPU with spike
  7 #58634     |        1         0         1         9 |       11         2         9        23
  8 #58633     |        1         0         1         9 |       11         2         9        26
  9 #58633     |        1         0         1         9 |       10         2         9        26
 10 #58633     |        1         0         1        13 |       11         2         9        28
 11 #58633     |        1         0         1        13 |       12         2         9        24
 12 #58633     |        1         0         1         8 |       10         2         9        23
```

osnoise top

```
[root@f34 ~]# rtla osnoise top -P F:1 -c 0-3 -r 900000 -d 1M -q
                                     Operating System Noise
duration:   0 00:01:00 | time is in us
CPU Period       Runtime        Noise  % CPU Aval   Max Noise   Max Single          HW          NMI          IRQ      Softirq       Thread
  0 #59         53100000       304896    99.42580        6978           56         549            0        53111         1590           13
  1 #59         53100000       338339    99.36282        8092           24         399            0        53130         1448           31
  2 #59         53100000       290842    99.45227        6582           39         855            0        53110         1406           12
  3 #59         53100000       204935    99.61405        6251           33         290            0        53156         1460           12
```

Timerlat events limitation

- Timerlat collects timerlat events:

```
                                Timer Latency
  0 00:00:13   |          IRQ Timer Latency (us)        |         Thread Timer Latency (us)
CPU COUNT      |      cur       min       avg       max |      cur       min       avg       max
  0 #12920     |        3         0        48      2091 |       23         3        84      2126
  1 #12919     |       63         0        54      2047 |       95         4        89      2102
  2 #12920     |        6         0        53      1931 |       27         3        97      1978
```

- Timerlat top collects timerlat events in the trace output:

```
    <idle>-0       [001] d.h1. 313167.456096: #21003 context    irq timer_latency     59190 ns
timerlat/1-375527  [001] ..... 313167.456126: #21003 context thread timer_latency     90002 ns
```

- It generates tons of events on 100+ CPUs with a period of 100 ns.
  - This is not a problem on osnoise because the osnoise:noise_sample is a regular tracepoint, so I can do histograms, but:

```
[root@x1 ~]# cd /sys/kernel/tracing/
[root@x1 tracing]# ls events/ftrace/
bprint  bputs  branch  context_switch  funcgraph_entry  funcgraph_exit  func_repeats  function  hwlat ...  osnoise timerlat
[root@x1 tracing]# ls events/ftrace/timerlat/
format  hist
```

Timerlat Improvements

- Add SMI counters to timerlat && osnoise tools
- Add support for parallel hwlat measurements
  - osnoise is sufficient but not necessary for hwlat measurements

```
                                Timer Latency
  0 00:00:13   |          IRQ Timer Latency (us)        |         Thread Timer Latency (us)       |  SMI  |  HW noise
CPU COUNT      |      cur       min       avg       max |      cur       min       avg       max  |    0  |        10
  0 #12920     |        3         0        48      2091 |       23         3        84      2126  |    1  |        11
  1 #12919     |       63         0        54      2047 |       95         4        89      2102  |    0  |
  2 #12920     |        6         0        53      1931 |       27         3        97      1978  |    0  |
  3  #3 -----  | ---------------------------------------- | ---------------------------------------- |    3  |       100
```

trace.dat and trace sync support

- osnoise and timerlat -t/--trace together with a form of stop trace save the end of the log to a text file
- But it would be better to save it in the trace.dat format
  - It allows better search
  - I can use it with kernel shark
- But we could synchronize multiple trace sessions, e.g.,
  - timerlat
  - Function tracer
- It shouldn't be hard

Integration with trace-cmd format

- osnoise and timerlat -t/--trace together with a form of stop trace save the end of the log to a text file
- But it would be better to save it in the trace.dat format
  - It allows better search
  - I can use it with kernel shark
- But we could synchronize multiple trace sessions, e.g.,
  - timerlat
  - Function tracer
- It shouldn't be hard

Real-time scheduling latency

- Hey Daniel, where is all that thing you presented last year?
- The (formally proved) scheduling latency tool is composed of a:
  - Kernel tracer
  - User-space tool to parse & analyze the data
- It is La Ferrari, while timerlat is the Fiat 500
  - It sums all possible worst case - timerlat only the cases it could observe
- It will get in when the preempt model gets in the RV
  - It depends on the WIP model working well
  - Once there, we will have the proof being verified
- It will depend on kernel options that are not enabled by default
  - preemptirq: tracepoints
  - That is why I did timerlat