Linux Tools Documentation

The kernel development community

Jun 10, 2024

CONTENTS

This book covers user-space tools that are shipped with the kernel source; more additions are needed here:

CHAPTER

ONE

THE REALTIME LINUX ANALYSIS TOOL

RTLA provides a set of tools for the analysis of the kernel’s realtime behavior on specific hardware.

1.1 rtla

1.1.1 Real-time Linux Analysis tool

Manual section
1

SYNOPSIS

rtla *COMMAND* [*OPTIONS*]

DESCRIPTION

The **rtla** is a meta-tool that includes a set of commands that aims to analyze the real-time properties of Linux. But instead of testing Linux as a black box, **rtla** leverages kernel tracing capabilities to provide precise information about the properties and root causes of unexpected results.

COMMANDS

osnoise

Gives information about the operating system noise (osnoise).

timerlat

Measures the IRQ and thread timer latency.

OPTIONS

-h, **--help**

Display the help text.

For other options, see the man page for the corresponding command.

SEE ALSO

rtla-osnoise(1), **rtla-timerlat**(1)

AUTHOR

Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rtla is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2021 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

1.2 rtla-osnoise

1.2.1 Measure the operating system noise

Manual section
1

SYNOPSIS

rtla osnoise [*MODE*] ...

DESCRIPTION

The **rtla osnoise** tool is an interface for the *osnoise* tracer. The *osnoise* tracer dispatches a kernel thread per-cpu. These threads read the time in a loop while with preemption, softirq and IRQs enabled, thus allowing all the sources of operating system noise during its execution. The *osnoise*'s tracer threads take note of the delta between each time read, along with an interference counter of all sources of interference. At the end of each period, the *osnoise* tracer displays a summary of the results.

The *osnoise* tracer outputs information in two ways. It periodically prints a summary of the noise of the operating system, including the counters of the occurrence of the source of interference. It also provides information for each noise via the **osnoise:** tracepoints. The **rtla osnoise top** mode displays information about the periodic summary from the *osnoise* tracer. The **rtla osnoise hist** mode displays information about the noise using the **osnoise:** tracepoints. For further details, please refer to the respective man page.

MODES

top

Prints the summary from osnoise tracer.

hist

Prints a histogram of osnoise samples.

If no MODE is given, the top mode is called, passing the arguments.

OPTIONS

-h, **--help**

Display the help text.

For other options, see the man page for the corresponding mode.

SEE ALSO

rtla-osnoise-top(1), **rtla-osnoise-hist**(1)

Osnoise tracer documentation: <https://www.kernel.org/doc/html/latest/trace/osnoise-tracer.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rtla is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2021 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

1.3 rtla-osnoise-hist

1.3.1 Display a histogram of the osnoise tracer samples

Manual section
1

SYNOPSIS

rtla osnoise hist [*OPTIONS*]

DESCRIPTION

The **rtla osnoise** tool is an interface for the *osnoise* tracer. The *osnoise* tracer dispatches a kernel thread per-cpu. These threads read the time in a loop while with preemption, softirq and IRQs enabled, thus allowing all the sources of operating system noise during its execution. The *osnoise*'s tracer threads take note of the delta between each time read, along with an interference counter of all sources of interference. At the end of each period, the *osnoise* tracer displays a summary of the results.

The **rtla osnoise hist** tool collects all **osnoise:sample_threshold** occurrence in a histogram, displaying the results in a user-friendly way. The tool also allows many configurations of the *osnoise* tracer and the collection of the tracer output.

OPTIONS

-a, **--auto** *us*

Set the automatic trace mode. This mode sets some commonly used options while debugging the system. It is equivalent to use **-s** *us* **-T 1 -t**.

-p, **--period** *us*

Set the *osnoise* tracer period in microseconds.

-r, **--runtime** *us*

Set the *osnoise* tracer runtime in microseconds.

-s, **--stop** *us*

Stop the trace if a single sample is higher than the argument in microseconds. If **-T** is set, it will also save the trace to the output.

-S, **--stop-total** *us*

Stop the trace if the total sample is higher than the argument in microseconds. If **-T** is set, it will also save the trace to the output.

-T, **--threshold** *us*

Specify the minimum delta between two time reads to be considered noise. The default threshold is *5 us*.

-b, **--bucket-size** *N*

Set the histogram bucket size (default *1*).

-E, **--entries** *N*

Set the number of entries of the histogram (default 256).

--no-header

Do not print header.

--no-summary

Do not print summary.

--no-index

Do not print index.

--with-zeros

Print zero only entries.

-c, **--cpus** *cpu-list*

Set the osnoise tracer to run the sample threads in the cpu-list.

-H, **--house-keeping** *cpu-list*

Run rtla control threads only on the given cpu-list.

-d, **--duration** *time[s|m|h|d]*

Set the duration of the session.

-D, **--debug**

Print debug info.

-t, **--trace**[*=file*]

Save the stopped trace to [*file|osnoise_trace.txt*].

-e, **--event** *sys:event*

Enable an event in the trace (**-t**) session. The argument can be a specific event, e.g., **-e** *sched:sched_switch*, or all events of a system group, e.g., **-e** *sched*. Multiple **-e** are allowed. It is only active when **-t** or **-a** are set.

--filter *<filter>*

Filter the previous **-e** *sys:event* event with *<filter>*. For further information about event filtering see https://www.kernel.org/doc/html/latest/trace/events.html#event-filtering.

--trigger ***<trigger>***

Enable a trace event trigger to the previous **-e** *sys:event*. If the *hist:* trigger is activated, the output histogram will be automatically saved to a file named *system_event_hist.txt*. For example, the command:

rtla <command> <mode> -t -e osnoise:irq_noise --trigger=”hist:key=desc,duration/1000:sort=de

Will automatically save the content of the histogram associated to *osnoise:irq_noise* event in *osnoise_irq_noise_hist.txt*.

For further information about event trigger see https://www.kernel.org/doc/html/latest/trace/events.html#event-triggers.

-P, **--priority** *o:prio|r:prio|f:prio|d:runtime:period*

Set scheduling parameters to the osnoise tracer threads, the format to set the priority are:

- *o:prio* - use SCHED_OTHER with *prio*;
- *r:prio* - use SCHED_RR with *prio*;
- *f:prio* - use SCHED_FIFO with *prio*;
- *d:runtime[us|ms|s]:period[us|ms|s]* - use SCHED_DEADLINE with *runtime* and *period* in nanoseconds.

-C, **--cgroup**[=*cgroup*]

Set a *cgroup* to the tracer’s threads. If the **-C** option is passed without arguments, the tracer’s thread will inherit **rtla**’s *cgroup*. Otherwise, the threads will be placed on the *cgroup* passed to the option.

-h, **--help**

Print help menu.

EXAMPLE

In the example below, *osnoise* tracer threads are set to run with real-time priority *FIFO:1*, on CPUs *0-11*, for *900ms* at each period (*1s* by default). The reason for reducing the runtime is to avoid starving the **rtla** tool. The tool is also set to run for *one minute*. The output histogram is set to group outputs in buckets of *10us* and *25* entries:

```
[root@f34 ~/]# rtla osnoise hist -P F:1 -c 0-11 -r 900000 -d 1M -b 10 -E 25
# RTLA osnoise histogram
# Time unit is microseconds (us)
# Duration:   0 00:01:00
Index    CPU-000    CPU-001    CPU-002    CPU-003    CPU-004    CPU-005    CPU-006  ␣
↪CPU-007    CPU-008    CPU-009    CPU-010    CPU-011
0          42982      46287      51779      53740      52024      44817      49898   ␣
↪ 36500      50408      50128      49523      52377
```

```
10        12224      8356      2912       878      2667     10155      4573   ␣
 ↪ 18894      4214      4836      5708      2413
20            8         5        12         2        13        24        20   ␣
 ↪    41        29        53        39        39
30            1         1         0         0        10         3         6   ␣
 ↪    19        15        31        30        38
40            0         0         0         0         0         4         2   ␣
 ↪     7         2         3         8        11
50            0         0         0         0         0         0         0   ␣
 ↪     0         0         1         1         2
over:         0         0         0         0         0         0         0   ␣
 ↪     0         0         0         0         0
count:    55215     54649     54703     54620     54714     55003     54499   ␣
 ↪ 55461     54668     55052     55309     54880
min:          0         0         0         0         0         0         0   ␣
 ↪     0         0         0         0         0
avg:          0         0         0         0         0         0         0   ␣
 ↪     0         0         0         0         0
max:         30        30        20        20        30        40        40   ␣
 ↪    40        40        50        50        50
```

SEE ALSO

rtla-osnoise(1), **rtla-osnoise-top**(1)

osnoise tracer documentation: <https://www.kernel.org/doc/html/latest/trace/osnoise-tracer.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rtla is Free Software licensed under the GNU GPLv2

COPYING

1.4 rtla-osnoise-top

1.4.1 Display a summary of the operating system noise

Manual section
1

SYNOPSIS

rtla osnoise top [*OPTIONS*]

DESCRIPTION

The **rtla osnoise** tool is an interface for the *osnoise* tracer. The *osnoise* tracer dispatches a kernel thread per-cpu. These threads read the time in a loop while with preemption, softirq and IRQs enabled, thus allowing all the sources of operating system noise during its execution. The *osnoise*'s tracer threads take note of the delta between each time read, along with an interference counter of all sources of interference. At the end of each period, the *osnoise* tracer displays a summary of the results.

rtla osnoise top collects the periodic summary from the *osnoise* tracer, including the counters of the occurrence of the interference source, displaying the results in a user-friendly format.

The tool also allows many configurations of the *osnoise* tracer and the collection of the tracer output.

OPTIONS

-a, **--auto** *us*

Set the automatic trace mode. This mode sets some commonly used options while debugging the system. It is equivalent to use **-s** *us* **-T 1 -t**.

-p, **--period** *us*

Set the *osnoise* tracer period in microseconds.

-r, **--runtime** *us*

Set the *osnoise* tracer runtime in microseconds.

-s, **--stop** *us*

Stop the trace if a single sample is higher than the argument in microseconds. If **-T** is set, it will also save the trace to the output.

-S, **--stop-total** *us*

Stop the trace if the total sample is higher than the argument in microseconds. If **-T** is set, it will also save the trace to the output.

-T, **--threshold** *us*

Specify the minimum delta between two time reads to be considered noise. The default threshold is *5 us*.

-q, **--quiet**

Print only a summary at the end of the session.

-c, **--cpus** *cpu-list*

Set the osnoise tracer to run the sample threads in the cpu-list.

-H, **--house-keeping** *cpu-list*

Run rtla control threads only on the given cpu-list.

-d, **--duration** *time[s|m|h|d]*

Set the duration of the session.

-D, **--debug**

Print debug info.

-t, **--trace**[=*file*]

Save the stopped trace to [*file|osnoise_trace.txt*].

-e, **--event** *sys:event*

Enable an event in the trace (**-t**) session. The argument can be a specific event, e.g., **-e** *sched:sched_switch*, or all events of a system group, e.g., **-e** *sched*. Multiple **-e** are allowed. It is only active when **-t** or **-a** are set.

--filter *<filter>*

Filter the previous **-e** *sys:event* event with *<filter>*. For further information about event filtering see https://www.kernel.org/doc/html/latest/trace/events.html#event-filtering.

--trigger *<trigger>*

Enable a trace event trigger to the previous **-e** *sys:event*. If the *hist:* trigger is activated, the output histogram will be automatically saved to a file named *system_event_hist.txt*. For example, the command:

rtla <command> <mode> -t -e osnoise:irq_noise --trigger=”hist:key=desc,duration/1000:sort=de

Will automatically save the content of the histogram associated to *osnoise:irq_noise* event in *osnoise_irq_noise_hist.txt*.

For further information about event trigger see https://www.kernel.org/doc/html/latest/trace/events.html#event-triggers.

-P, **--priority** *o:prio|r:prio|f:prio|d:runtime:period*

Set scheduling parameters to the osnoise tracer threads, the format to set the priority are:

- *o:prio* - use SCHED_OTHER with *prio*;

- *r:prio* - use SCHED_RR with *prio*;
- *f:prio* - use SCHED_FIFO with *prio*;
- *d:runtime[us|ms|s]:period[us|ms|s]* - use SCHED_DEADLINE with *runtime* and *period* in nanoseconds.

-C, **--cgroup**[=*cgroup*]

Set a *cgroup* to the tracer's threads. If the **-C** option is passed without arguments, the tracer's thread will inherit **rtla**'s *cgroup*. Otherwise, the threads will be placed on the *cgroup* passed to the option.

-h, **--help**

Print help menu.

EXAMPLE

In the example below, the **rtla osnoise top** tool is set to run with a real-time priority *FIFO:1*, on CPUs *0-3*, for *900ms* at each period (*1s* by default). The reason for reducing the runtime is to avoid starving the rtla tool. The tool is also set to run for *one minute* and to display a summary of the report at the end of the session:

```
[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
```

SEE ALSO

rtla-osnoise(1), **rtla-osnoise-hist**(1)

Osnoise tracer documentation: <https://www.kernel.org/doc/html/latest/trace/osnoise-tracer.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rtla is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2021 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

1.5 rtla-timerlat

1.5.1 Measures the operating system timer latency

Manual section
1

SYNOPSIS

rtla timerlat [*MODE*] ...

DESCRIPTION

The **rtla timerlat** tool is an interface for the *timerlat* tracer. The *timerlat* tracer dispatches a kernel thread per-cpu. These threads set a periodic timer to wake themselves up and go back to sleep. After the wakeup, they collect and generate useful information for the debugging of operating system timer latency.

The *timerlat* tracer outputs information in two ways. It periodically prints the timer latency at the timer *IRQ* handler and the *Thread* handler. It also enable the trace of the most relevant information via **osnoise:** tracepoints.

The *timerlat* tracer outputs information in two ways. It periodically prints the timer latency at the timer *IRQ* handler and the *Thread* handler. It also provides information for each noise via the **osnoise:** tracepoints. The **rtla timerlat top** mode displays a summary of the periodic output from the *timerlat* tracer. The **rtla hist hist** mode displays a histogram of each tracer event occurrence. For further details, please refer to the respective man page.

MODES

top

Prints the summary from *timerlat* tracer.

hist

Prints a histogram of timerlat samples.

If no *MODE* is given, the top mode is called, passing the arguments.

OPTIONS

-h, **--help**

Display the help text.

For other options, see the man page for the corresponding mode.

SEE ALSO

rtla-timerlat-top(1), **rtla-timerlat-hist**(1)

timerlat tracer documentation: <https://www.kernel.org/doc/html/latest/trace/timerlat-tracer.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rtla is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2021 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

1.6 rtla-timerlat-hist

1.6.1 Histograms of the operating system timer latency

Manual section
1

SYNOPSIS

rtla timerlat hist [*OPTIONS*] …

DESCRIPTION

The **rtla timerlat** tool is an interface for the *timerlat* tracer. The *timerlat* tracer dispatches a kernel thread per-cpu. These threads set a periodic timer to wake themselves up and go back to sleep. After the wakeup, they collect and generate useful information for the debugging of operating system timer latency.

The *timerlat* tracer outputs information in two ways. It periodically prints the timer latency at the timer *IRQ* handler and the *Thread* handler. It also enable the trace of the most relevant information via **osnoise:** tracepoints.

The **rtla timerlat hist** displays a histogram of each tracer event occurrence. This tool uses the periodic information, and the **osnoise:** tracepoints are enabled when using the **-T** option.

OPTIONS

-a, **--auto** *us*

Set the automatic trace mode. This mode sets some commonly used options while debugging the system. It is equivalent to use **-T** *us* **-s** *us* **-t**. By default, *timerlat* tracer uses FIFO:95 for *timerlat* threads, thus equilavent to **-P** *f:95*.

-p, **--period** *us*

Set the *timerlat* tracer period in microseconds.

-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* if a *Thread* latency is higher than the argument in us.

--dma-latency *us*

Set the /dev/cpu_dma_latency to *us*, aiming to bound exit from idle latencies. *cyclictest* sets this value to *0* by default, use **--dma-latency** *0* to have similar results.

-u, **--user-threads**

Set timerlat to run without a workload, and then dispatches user-space workloads to wait on the timerlat_fd. Once the workload is awakes, it goes to sleep again adding so the measurement for the kernel-to-user and user-to-kernel to the tracer output.

-b, **--bucket-size** *N*

Set the histogram bucket size (default *1*).

-E, **--entries** *N*

Set the number of entries of the histogram (default 256).

--no-header

Do not print header.

--no-summary

Do not print summary.

--no-index

Do not print index.

--with-zeros

Print zero only entries.

-c, **--cpus** *cpu-list*

Set the osnoise tracer to run the sample threads in the cpu-list.

-H, **--house-keeping** *cpu-list*

Run rtla control threads only on the given cpu-list.

-d, **--duration** *time[s|m|h|d]*

Set the duration of the session.

-D, **--debug**

Print debug info.

-t, **--trace**[=*file*]

Save the stopped trace to [*file|osnoise_trace.txt*].

-e, **--event** *sys:event*

Enable an event in the trace (**-t**) session. The argument can be a specific event, e.g., **-e** *sched:sched_switch*, or all events of a system group, e.g., **-e** *sched*. Multiple **-e** are allowed. It is only active when **-t** or **-a** are set.

--filter *<filter>*

Filter the previous **-e** *sys:event* event with *<filter>*. For further information about event filtering see https://www.kernel.org/doc/html/latest/trace/events.html#event-filtering.

--trigger ***<trigger>***

Enable a trace event trigger to the previous **-e** *sys:event*. If the *hist:* trigger is activated, the output histogram will be automatically saved to a file named *system_event_hist.txt*. For example, the command:

rtla <command> <mode> -t -e osnoise:irq_noise --trigger="hist:key=desc,duration/1000:sort=de

Will automatically save the content of the histogram associated to *osnoise:irq_noise* event in *osnoise_irq_noise_hist.txt*.

For further information about event trigger see https://www.kernel.org/doc/html/latest/trace/events.html#event-triggers.

-P, **--priority** *o:prio|r:prio|f:prio|d:runtime:period*

Set scheduling parameters to the osnoise tracer threads, the format to set the priority are:

- *o:prio* - use SCHED_OTHER with *prio*;
- *r:prio* - use SCHED_RR with *prio*;
- *f:prio* - use SCHED_FIFO with *prio*;
- *d:runtime[us|ms|s]:period[us|ms|s]* - use SCHED_DEADLINE with *runtime* and *period* in nanoseconds.

-C, **--cgroup**[=*cgroup*]

Set a *cgroup* to the tracer's threads. If the **-C** option is passed without arguments, the tracer's thread will inherit **rtla**'s *cgroup*. Otherwise, the threads will be placed on the *cgroup* passed to the option.

-h, **--help**

Print help menu.

--dump-tasks

prints the task running on all CPUs if stop conditions are met (depends on !--no-aa)

--no-aa

disable auto-analysis, reducing rtla timerlat cpu usage

EXAMPLE

In the example below, **rtla timerlat hist** is set to run for *10* minutes, in the cpus *0-4*, *skipping zero* only lines. Moreover, **rtla timerlat hist** will change the priority of the *timerlat* threads to run under *SCHED_DEADLINE* priority, with a *100us* runtime every *1ms* period. The *1ms* period is also passed to the *timerlat* tracer. Auto-analysis is disabled to reduce overhead

```
[root@alien ~]# timerlat hist -d 10m -c 0-4 -P d:100us:1ms -p 1000 --no-aa
# RTLA timerlat histogram
# Time unit is microseconds (us)
# Duration:   0 00:10:00
Index   IRQ-000   Thr-000   IRQ-001   Thr-001   IRQ-002   Thr-002   IRQ-003   Thr-003   IRQ-004   Thr-004
0        276489         0    206089         0    466018         0    481102         0    205546         0
1        318327     35487    388149     30024     94531     48382     83082     71078    388026     55730
2          3282    122584      4019    126527     28231    109012     23311     89309      4568     98739
```

```
3           940     11815       837      9863      6209     16227      6895   ␣
 ↪ 17196       910      9780
4           444     17287       424     11574      2097     38443      2169   ␣
 ↪ 36736       462     13476
5           206     43291       255     25581      1223    101908      1304   ␣
 ↪101137       236     28913
6           132    101501        96     64584       635    213774       757   ␣
 ↪215471        99     73453
7            74    169347        65    124758       350     57466       441   ␣
 ↪ 53639        69    148573
8            53     85183        31    156751       229      9052       306   ␣
 ↪  9026        39    139907
9            22     10387        12     42762       161      2554       225   ␣
 ↪  2689        19     26192
10           13      1898         8      5770       114      1247       128   ␣
 ↪  1405        13      3772
11            9       560         9       924        71       686        76   ␣
 ↪   765         8       713
12            4       256         2       360        50       411        64   ␣
 ↪   474         3       278
13            2       167         2       172        43       256        53   ␣
 ↪   350         4       180
14            1        88         1       116        15       198        42   ␣
 ↪   223         0       115
15            2        63         3        94        11       139        20   ␣
 ↪   150         0        58
16            2        37         0        56         5        78        10   ␣
 ↪   102         0        39
17            0        18         0        28         4        57         8   ␣
 ↪    80         0        15
18            0         8         0        17         2        50         6   ␣
 ↪    56         0        12
19            0         9         0         5         0        19         0   ␣
 ↪    48         0        18
20            0         4         0         8         0        11         2   ␣
 ↪    27         0         4
21            0         2         0         3         1         9         1   ␣
 ↪    18         0         6
22            0         1         0         3         1         7         0   ␣
 ↪     3         0         5
23            0         2         0         4         0         2         0   ␣
 ↪     7         0         2
24            0         2         0         2         1         3         0   ␣
 ↪     3         0         5
25            0         0         0         1         0         1         0   ␣
 ↪     1         0         3
26            0         1         0         0         0         2         0   ␣
 ↪     2         0         0
27            0         0         0         3         0         1         0   ␣
 ↪     0         0         1
```

```
28            0         0         0         3         0         0         0   ␣
 ↪     1         0         0
29            0         0         0         2         0         2         0   ␣
 ↪     1         0         3
30            0         1         0         0         0         0         0   ␣
 ↪     0         0         0
31            0         1         0         0         0         0         0   ␣
 ↪     2         0         2
32            0         0         0         1         0         2         0   ␣
 ↪     0         0         0
33            0         0         0         2         0         0         0   ␣
 ↪     0         0         1
34            0         0         0         0         0         0         0   ␣
 ↪     0         0         2
35            0         1         0         1         0         0         0   ␣
 ↪     0         0         1
36            0         1         0         0         0         1         0   ␣
 ↪     1         0         0
37            0         0         0         1         0         0         0   ␣
 ↪     0         0         0
40            0         0         0         0         0         1         0   ␣
 ↪     1         0         0
41            0         0         0         0         0         0         0   ␣
 ↪     0         0         1
42            0         0         0         0         0         0         0   ␣
 ↪     0         0         1
44            0         0         0         0         0         1         0   ␣
 ↪     0         0         0
46            0         0         0         0         0         0         0   ␣
 ↪     1         0         0
47            0         0         0         0         0         0         0   ␣
 ↪     0         0         1
50            0         0         0         0         0         0         0   ␣
 ↪     0         0         1
54            0         0         0         1         0         0         0   ␣
 ↪     0         0         0
58            0         0         0         1         0         0         0   ␣
 ↪     0         0         0
over:         0         0         0         0         0         0         0   ␣
 ↪     0         0         0
count:   600002    600002    600002    600002    600002    600002    600002   ␣
 ↪600002    600002    600002
min:          0         1         0         1         0         1         0   ␣
 ↪     1         0         1
avg:          0         5         0         5         0         4         0   ␣
 ↪     4         0         5
max:         16        36        15        58        24        44        21   ␣
 ↪    46        13        50
```

SEE ALSO

rtla-timerlat(1), **rtla-timerlat-top**(1)

timerlat tracer documentation: <https://www.kernel.org/doc/html/latest/trace/timerlat-tracer.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

1.7 rtla-timerlat-top

1.7.1 Measures the operating system timer latency

Manual section
1

SYNOPSIS

rtla timerlat top [*OPTIONS*] ...

DESCRIPTION

The **rtla timerlat** tool is an interface for the *timerlat* tracer. The *timerlat* tracer dispatches a kernel thread per-cpu. These threads set a periodic timer to wake themselves up and go back to sleep. After the wakeup, they collect and generate useful information for the debugging of operating system timer latency.

The *timerlat* tracer outputs information in two ways. It periodically prints the timer latency at the timer *IRQ* handler and the *Thread* handler. It also enable the trace of the most relevant information via **osnoise:** tracepoints.

The **rtla timerlat top** displays a summary of the periodic output from the *timerlat* tracer. It also provides information for each operating system noise via the **osnoise:** tracepoints that can be seem with the option **-T**.

OPTIONS

-a, **--auto** *us*

Set the automatic trace mode. This mode sets some commonly used options while debugging the system. It is equivalent to use **-T** *us* **-s** *us* **-t**. By default, *timerlat* tracer uses FIFO:95 for *timerlat* threads, thus equilavent to **-P** *f:95*.

-p, **--period** *us*

Set the *timerlat* tracer period in microseconds.

-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* if a *Thread* latency is higher than the argument in us.

--dma-latency us

Set the /dev/cpu_dma_latency to *us*, aiming to bound exit from idle latencies. *cyclictest* sets this value to *0* by default, use **--dma-latency** *0* to have similar results.

-u, **--user-threads**

Set timerlat to run without a workload, and then dispatches user-space workloads to wait on the timerlat_fd. Once the workload is awakes, it goes to sleep again adding so the measurement for the kernel-to-user and user-to-kernel to the tracer output.

-q, **--quiet**

Print only a summary at the end of the session.

-c, **--cpus** *cpu-list*

Set the osnoise tracer to run the sample threads in the cpu-list.

-H, **--house-keeping** *cpu-list*

Run rtla control threads only on the given cpu-list.

-d, **--duration** *time[s|m|h|d]*

Set the duration of the session.

-D, **--debug**

Print debug info.

-t, **--trace**[=*file*]

Save the stopped trace to [*file*|*osnoise_trace.txt*].

-e, **--event** *sys:event*

Enable an event in the trace (**-t**) session. The argument can be a specific event, e.g., **-e** *sched:sched_switch*, or all events of a system group, e.g., **-e** *sched*. Multiple **-e** are allowed. It is only active when **-t** or **-a** are set.

--filter *<filter>*

Filter the previous **-e** *sys:event* event with *<filter>*. For further information about event filtering see https://www.kernel.org/doc/html/latest/trace/events.html#event-filtering.

--trigger <trigger>

Enable a trace event trigger to the previous **-e** *sys:event*. If the *hist:* trigger is activated, the output histogram will be automatically saved to a file named *system_event_hist.txt*. For example, the command:

```
rtla <command> <mode> -t -e osnoise:irq_noise --trigger=”hist:key=desc,duration/1000:sort=de
```

Will automatically save the content of the histogram associated to *osnoise:irq_noise* event in *osnoise_irq_noise_hist.txt*.

For further information about event trigger see https://www.kernel.org/doc/html/latest/trace/events.html#event-triggers.

-P, **--priority** *o:prio|r:prio|f:prio|d:runtime:period*

Set scheduling parameters to the osnoise tracer threads, the format to set the priority are:

- *o:prio* - use SCHED_OTHER with *prio*;
- *r:prio* - use SCHED_RR with *prio*;
- *f:prio* - use SCHED_FIFO with *prio*;
- *d:runtime[us|ms|s]:period[us|ms|s]* - use SCHED_DEADLINE with *runtime* and *period* in nanoseconds.

-C, **--cgroup**[=*cgroup*]

Set a *cgroup* to the tracer's threads. If the **-C** option is passed without arguments, the tracer's thread will inherit **rtla**'s *cgroup*. Otherwise, the threads will be placed on the *cgroup* passed to the option.

-h, **--help**

Print help menu.

--dump-tasks

prints the task running on all CPUs if stop conditions are met (depends on !--no-aa)

--no-aa

disable auto-analysis, reducing rtla timerlat cpu usage

--aa-only *us*

Set stop tracing conditions and run without collecting and displaying statistics. Print the auto-analysis if the system hits the stop tracing condition. This option is useful to reduce rtla timerlat CPU, enabling the debug without the overhead of collecting the statistics.

EXAMPLE

In the example below, the timerlat tracer is dispatched in cpus *1-23* in the automatic trace mode, instructing the tracer to stop if a *40 us* latency or higher is found:

```
# timerlat -a 40 -c 1-23 -q
                                   Timer Latency
  0 00:00:12   |          IRQ Timer Latency (us)        |         Thread Timer␣
↪Latency (us)
CPU COUNT      |      cur       min       avg       max |      cur       min  ␣
↪    avg       max
  1 #12322     |        0         0         1        15 |       10         3  ␣
↪      9        31
  2 #12322     |        3         0         1        12 |       10         3  ␣
```

```
 ↪      9        23
  3 #12322     |        1         0         1        21 |        8         2  ␣
 ↪      8        34
  4 #12322     |        1         0         1        17 |       10         2  ␣
 ↪     11        33
  5 #12322     |        0         0         1        12 |        8         3  ␣
 ↪      8        25
  6 #12322     |        1         0         1        14 |       16         3  ␣
 ↪     11        35
  7 #12322     |        0         0         1        14 |        9         2  ␣
 ↪      8        29
  8 #12322     |        1         0         1        22 |        9         3  ␣
 ↪      9        34
  9 #12322     |        0         0         1        14 |        8         2  ␣
 ↪      8        24
 10 #12322     |        1         0         0        12 |        9         3  ␣
 ↪      8        24
 11 #12322     |        0         0         0        15 |        6         2  ␣
 ↪      7        29
 12 #12321     |        1         0         0        13 |        5         3  ␣
 ↪      8        23
 13 #12319     |        0         0         1        14 |        9         3  ␣
 ↪      9        26
 14 #12321     |        1         0         0        13 |        6         2  ␣
 ↪      8        24
 15 #12321     |        1         0         1        15 |       12         3  ␣
 ↪     11        27
 16 #12318     |        0         0         1        13 |        7         3  ␣
 ↪     10        24
 17 #12319     |        0         0         1        13 |       11         3  ␣
 ↪      9        25
 18 #12318     |        0         0         0        12 |        8         2  ␣
 ↪      8        20
 19 #12319     |        0         0         1        18 |       10         2  ␣
 ↪      9        28
 20 #12317     |        0         0         0        20 |        9         3  ␣
 ↪      8        34
 21 #12318     |        0         0         0        13 |        8         3  ␣
 ↪      8        28
 22 #12319     |        0         0         1        11 |        8         3  ␣
 ↪     10        22
 23 #12320     |       28         0         1        28 |       41         3  ␣
 ↪     11        41
rtla timerlat hit stop tracing
## CPU 23 hit stop tracing, analyzing it ##
IRQ handler delay:                                        27.49 us (65.52 %)
IRQ latency:                                              28.13 us
Timerlat IRQ duration:                                     9.59 us (22.85 %)
Blocking thread:                                           3.79 us (9.03 %)
                       objtool:49256                       3.79 us
  Blocking thread stacktrace
              -> 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:                                          41.96 us (100%)

The system has exit from idle latency!
  Max timerlat IRQ latency from idle: 17.48 us in cpu 4
Saving trace to timerlat_trace.txt
```

In this case, the major factor was the delay suffered by the *IRQ handler* that handles **timerlat** wakeup: *65.52%*. This can be caused by the current thread masking interrupts, which can be seen in the blocking thread stacktrace: the current thread (*objtool:49256*) disabled interrupts via *raw spin lock* operations inside mem cgroup, while doing write syscall in a btrfs file system.

The raw trace is saved in the **timerlat_trace.txt** file for further analysis.

Note that **rtla timerlat** was dispatched without changing *timerlat* tracer threads' priority. That is generally not needed because these threads have priority *FIFO:95* by default, which is a common priority used by real-time kernel developers to analyze scheduling delays.

SEE ALSO

rtla-timerlat(1), **rtla-timerlat-hist**(1)

timerlat tracer documentation: <https://www.kernel.org/doc/html/latest/trace/timerlat-tracer.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rtla is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2021 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

1.8 rtla-hwnoise

1.8.1 Detect and quantify hardware-related noise

Manual section
1

SYNOPSIS

rtla hwnoise [*OPTIONS*]

DESCRIPTION

rtla hwnoise collects the periodic summary from the *osnoise* tracer running with *interrupts disabled*. By disabling interrupts, and the scheduling of threads as a consequence, only non-maskable interrupts and hardware-related noise is allowed.

The tool also allows the configurations of the *osnoise* tracer and the collection of the tracer output.

OPTIONS

-a, **--auto** *us*

Set the automatic trace mode. This mode sets some commonly used options while debugging the system. It is equivalent to use **-s** *us* **-T 1 -t**.

-p, **--period** *us*

Set the *osnoise* tracer period in microseconds.

-r, **--runtime** *us*

Set the *osnoise* tracer runtime in microseconds.

-s, **--stop** *us*

Stop the trace if a single sample is higher than the argument in microseconds. If **-T** is set, it will also save the trace to the output.

-S, **--stop-total** *us*

Stop the trace if the total sample is higher than the argument in microseconds. If **-T** is set, it will also save the trace to the output.

-T, **--threshold** *us*

Specify the minimum delta between two time reads to be considered noise. The default threshold is *5 us*.

-q, **--quiet**

Print only a summary at the end of the session.

-c, **--cpus** *cpu-list*

Set the osnoise tracer to run the sample threads in the cpu-list.

-H, **--house-keeping** *cpu-list*

Run rtla control threads only on the given cpu-list.

-d, **--duration** *time[s|m|h|d]*

Set the duration of the session.

-D, **--debug**

Print debug info.

-t, **--trace**[=*file*]

Save the stopped trace to [*file|osnoise_trace.txt*].

-e, **--event** *sys:event*

Enable an event in the trace (**-t**) session. The argument can be a specific event, e.g., **-e** *sched:sched_switch*, or all events of a system group, e.g., **-e** *sched*. Multiple **-e** are allowed. It is only active when **-t** or **-a** are set.

--filter *<filter>*

Filter the previous **-e** *sys:event* event with *<filter>*. For further information about event filtering see https://www.kernel.org/doc/html/latest/trace/events.html#event-filtering.

--trigger ***<trigger>***

Enable a trace event trigger to the previous **-e** *sys:event*. If the *hist:* trigger is activated, the output histogram will be automatically saved to a file named *system_event_hist.txt*. For example, the command:

rtla <command> <mode> -t -e osnoise:irq_noise --trigger="hist:key=desc,duration/1000:sort=de

Will automatically save the content of the histogram associated to *osnoise:irq_noise* event in *osnoise_irq_noise_hist.txt*.

For further information about event trigger see https://www.kernel.org/doc/html/latest/trace/events.html#event-triggers.

-P, **--priority** *o:prio|r:prio|f:prio|d:runtime:period*

Set scheduling parameters to the osnoise tracer threads, the format to set the priority are:

- *o:prio* - use SCHED_OTHER with *prio*;
- *r:prio* - use SCHED_RR with *prio*;
- *f:prio* - use SCHED_FIFO with *prio*;
- *d:runtime[us|ms|s]:period[us|ms|s]* - use SCHED_DEADLINE with *runtime* and *period* in nanoseconds.

-C, **--cgroup**[=*cgroup*]

Set a *cgroup* to the tracer's threads. If the **-C** option is passed without arguments, the tracer's thread will inherit **rtla**'s *cgroup*. Otherwise, the threads will be placed on the *cgroup* passed to the option.

-h, **--help**

Print help menu.

EXAMPLE

In the example below, the **rtla hwnoise** tool is set to run on CPUs *1-7* on a system with 8 cores/16 threads with hyper-threading enabled.

The tool is set to detect any noise higher than *one microsecond*, to run for *ten minutes*, displaying a summary of the report at the end of the session:

```
# rtla hwnoise -c 1-7 -T 1 -d 10m -q
                                        Hardware-related Noise
duration:   0 00:10:00 | time is in us
CPU Period       Runtime        Noise  % CPU Aval   Max Noise   Max Single    ␣
 ↪     HW          NMI
  1 #599       599000000          138    99.99997           3            3    ␣
 ↪      4           74
  2 #599       599000000           85    99.99998           3            3    ␣
 ↪      4           75
  3 #599       599000000           86    99.99998           4            3    ␣
 ↪      6           75
  4 #599       599000000           81    99.99998           4            4    ␣
 ↪      2           75
```

```
  5 #599       599000000           85     99.99998           2            2     ␣
 ↪      2          75
  6 #599       599000000           76     99.99998           2            2     ␣
 ↪      0          75
  7 #599       599000000           77     99.99998           3            3     ␣
 ↪      0          75
```

The first column shows the *CPU*, and the second column shows how many *Periods* the tool ran during the session. The *Runtime* is the time the tool effectively runs on the CPU. The *Noise* column is the sum of all noise that the tool observed, and the *% CPU Aval* is the relation between the *Runtime* and *Noise*.

The *Max Noise* column is the maximum hardware noise the tool detected in a single period, and the *Max Single* is the maximum single noise seen.

The *HW* and *NMI* columns show the total number of *hardware* and *NMI* noise occurrence observed by the tool.

For example, *CPU 3* ran *599* periods of *1 second Runtime*. The CPU received *86 us* of noise during the entire execution, leaving *99.99997 %* of CPU time for the application. In the worst single period, the CPU caused *4 us* of noise to the application, but it was certainly caused by more than one single noise, as the *Max Single* noise was of *3 us*. The CPU has *HW noise,* at a rate of *six occurrences/ten minutes*. The CPU also has *NMIs*, at a higher frequency: around *seven per second*.

The tool should report *0* hardware-related noise in the ideal situation. For example, by disabling hyper-threading to remove the hardware noise, and disabling the TSC watchdog to remove the NMI (it is possible to identify this using tracing options of **rtla hwnoise**), it was possible to reach the ideal situation in the same hardware:

```
# rtla hwnoise -c 1-7 -T 1 -d 10m -q
                                        Hardware-related Noise
duration:   0 00:10:00 | time is in us
CPU Period       Runtime        Noise  % CPU Aval   Max Noise   Max Single     ␣
 ↪     HW          NMI
  1 #599       599000000            0   100.00000           0            0     ␣
 ↪      0            0
  2 #599       599000000            0   100.00000           0            0     ␣
 ↪      0            0
  3 #599       599000000            0   100.00000           0            0     ␣
 ↪      0            0
  4 #599       599000000            0   100.00000           0            0     ␣
 ↪      0            0
  5 #599       599000000            0   100.00000           0            0     ␣
 ↪      0            0
  6 #599       599000000            0   100.00000           0            0     ␣
 ↪      0            0
  7 #599       599000000            0   100.00000           0            0     ␣
 ↪      0            0
```

SEE ALSO

rtla-osnoise(1)

Osnoise tracer documentation: <https://www.kernel.org/doc/html/latest/trace/osnoise-tracer.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rtla is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2021 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

CHAPTER

TWO

RUNTIME VERIFICATION (RV) TOOL

rv tool provides the interface for a collection of runtime verification (rv) monitors.

2.1 rv

2.1.1 Runtime Verification

Manual section
1

SYNOPSIS

rv *COMMAND* [*OPTIONS*]

DESCRIPTION

Runtime Verification (**RV**) is a lightweight (yet rigorous) method for formal verification with a practical approach for complex systems. Instead of relying on a fine-grained model of a system (e.g., a re-implementation a instruction level), RV works by analyzing the trace of the system's actual execution, comparing it against a formal specification of the system behavior.

The **rv** tool provides the interface for a collection of runtime verification (rv) monitors.

COMMANDS

list

List all available monitors.

mon

Run monitor.

OPTIONS

-h, **--help**

Display the help text.

For other options, see the man page for the corresponding command.

SEE ALSO

rv-list(1), **rv-mon**(1)

Linux kernel *RV* documentation: <https://www.kernel.org/doc/html/latest/trace/rv/index.html>

AUTHOR

Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rv is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2022 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

2.2 rv-list

2.2.1 List available monitors

Manual section
1

SYNOPSIS

rv list [*OPTIONS*]

DESCRIPTION

The **rv list** command prints all available monitors. These monitors can be enabled using the **rv mon** command.

OPTIONS

-h, **--help**

Print help menu.

SEE ALSO

rv(1), **rv-mon**(1)

Linux kernel *RV* documentation: <https://www.kernel.org/doc/html/latest/trace/rv/index.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rv is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2022 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

2.3 rv-list

2.3.1 List available monitors

Manual section
1

SYNOPSIS

rv mon [*-h*] **monitor_name** [*-h*] [*MONITOR OPTIONS*]

DESCRIPTION

The **rv mon** command runs the monitor named *monitor_name*. Each monitor has its own set of options. The **rv list** command shows all available monitors.

OPTIONS

-h, **--help**

Print help menu.

AVAILABLE MONITORS

The **rv** tool provides the interface for a set of monitors. Use the **rv list** command to list all available monitors.

Each monitor has its own set of options. See man **rv-mon**-*monitor_name* for details about each specific monitor. Also, running **rv mon monitor_name -h** display the help menu with the available options.

SEE ALSO

rv(1), **rv-mon**(1)

Linux kernel *RV* documentation: <https://www.kernel.org/doc/html/latest/trace/rv/index.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rv is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2022 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

2.4 rv-mon-wip

2.4.1 Wakeup In Preemptive monitor

Manual section
1

SYNOPSIS

rv mon wip [*OPTIONS*]

DESCRIPTION

The wakeup in preemptive (**wip**) monitor is a sample per-cpu monitor that checks if the wakeup events always take place with preemption disabled.

See kernel documentation for further information about this monitor: <https://docs.kernel.org/trace/rv/monitor_wip.html>

OPTIONS

-h, **--help**

Print the monitor's options and the available reactors list.

-r, **--reactor** *reactor*

Enables the *reactor*. See **-h** for a list of available reactors.

-s, **--self**

When tracing (**-t**), also print the events that happened during the **rv** command itself. If the **rv** command itself generates too many events, the tool might get busy processing its own events only.

-t, **--trace**

Trace monitor's events and error.

-v, **--verbose**

Print debug messages.

SEE ALSO

rv(1), **rv-mon**(1)

Linux kernel *RV* documentation: <https://www.kernel.org/doc/html/latest/trace/rv/index.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rv is Free Software licensed under the GNU GPLv2

COPYING

Copyright (C) 2022 Red Hat, Inc. Free use of this software is granted under the terms of the GNU Public License (GPL).

2.5 rv-mon-wwnr

2.5.1 Wakeup While Not Running monitor

Manual section
1

SYNOPSIS

rv mon wip [*OPTIONS*]

DESCRIPTION

The wakeup while not running (**wwnr**) is a per-task sample monitor.

See kernel documentation for further information about this monitor: <https://docs.kernel.org/trace/rv/monitor_wwnr.html>

OPTIONS

-h, **--help**

Print the monitor's options and the available reactors list.

-r, **--reactor** *reactor*

Enables the *reactor*. See **-h** for a list of available reactors.

-s, **--self**

When tracing (**-t**), also print the events that happened during the **rv** command itself. If the **rv** command itself generates too many events, the tool might get busy processing its own events only.

-t, **--trace**

Trace monitor's events and error.

-v, **--verbose**

Print debug messages.

SEE ALSO

rv(1), **rv-mon**(1)

Linux kernel *RV* documentation: <https://www.kernel.org/doc/html/latest/trace/rv/index.html>

AUTHOR

Written by Daniel Bristot de Oliveira <bristot@kernel.org>

REPORTING BUGS

Report bugs to <linux-kernel@vger.kernel.org> and <linux-trace-devel@vger.kernel.org>

LICENSE

rv is Free Software licensed under the GNU GPLv2

COPYING