Red Hat
Enterprise Linux

osnoise: what is missing?

Daniel Bristot de Oliveira
Senior Principal Software Engineer

Tracing + Workload

- osnoise is a **kernel tracer** that also dispatches the workload
- Per-cpu busy-loop tool that:
  - Measures how much time passed between two reads of the time
  - How many interrupts took place in between (sync)
- Traces sources of noise, aware of nesting:
  - NMI -> IRQs -> Softirqs -> Threads
  - Hardware & Virtualization sources of noise
    - SMIs/VM preemption by the host
- Interface via rtla
  - osnoise top: shows an interactive view of the osnoise summary output
  - osnoise hist: shows a histogram of the osnoise sample tracepoint

For a detailed description of the tracer, with the theory behind it, see this paper:

Operating system noise in the Linux kernel

In IEEE Transactions on Computers.

https://ieeexplore.ieee.org/document/9812514/

Red Hat

OS Noise tracer: summary output

```
[root@f32 ~]# cd /sys/kernel/tracing/
[root@f32 tracing]# echo osnoise > current_tracer
[root@f32 tracing]# cat trace
# tracer: osnoise
#
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth                                   MAX
#                              || /                                                     SINGLE      Interference counters:
#                              ||||              RUNTIME      NOISE   % OF CPU  NOISE    +-----------------------------+
#           TASK-PID      CPU# ||||   TIMESTAMP    IN US      IN US  AVAILABLE  IN US     HW    NMI    IRQ   SIRQ THREAD
#              | |         |   ||||      |           |          |    |           |       |      |      |      |      |
            <...>-859     [000] ....    81.637220: 1000000       190  99.98100       9     18      0   1007     18      1
            <...>-860     [001] ....    81.638154: 1000000       656  99.93440      74     23      0   1006     16      3
            <...>-861     [002] ....    81.638193: 1000000      5675  99.43250     202      6      0   1013     25     21
            <...>-862     [003] ....    81.638242: 1000000       125  99.98750      45      1      0   1011     23      0
            <...>-863     [004] ....    81.638260: 1000000      1721  99.82790     168      7      0   1002     49     41
            <...>-864     [005] ....    81.638286: 1000000       263  99.97370      57      6      0   1006     26      2
            <...>-865     [006] ....    81.638302: 1000000       109  99.98910      21      3      0   1006     18      1
            <...>-866     [007] ....    81.638326: 1000000      7816  99.21840     107      8      0   1016     39     19
```

The tracer is inspired on hwlat tracer.

Red Hat

OS Noise tracer: fine-grained tracing

OS Noise:

```
# cd /sys/kernel/tracing/
# echo osnoise > current_tracer
# echo osnoise > set_event
# echo 8 > osnoise/stop_tracing_us
# cat trace
[...]
  osnoise/8-960     [007] d.h.  5789.857530: irq_noise: local_timer:236 start 5789.857527123 duration 1867 ns
  osnoise/8-961     [008] d.h.  5789.857532: irq_noise: local_timer:236 start 5789.857529929 duration 1845 ns
  osnoise/8-961     [008] dNh.  5789.858408: irq_noise: local_timer:236 start 5789.858404871 duration 2848 ns
migration/8-54      [008] d...  5789.858413: thread_noise: migration/8:54 start 5789.858409300 duration 3068 ns
  osnoise/8-961     [008] ....  5789.858413: sample_threshold: start 5789.858404555 duration 8812 ns interferences 2
```

Hardware Noise:

```
  osnoise/1-32160   [001] d.h1. 31240.380886: irq_noise: thermal_apic:250 start 31240.380884026 duration 1715 ns
  osnoise/1-32160   [001] ..... 31240.380886: sample_threshold: start 31240.380883588 duration 2763 ns interference 1
  osnoise/1-32160   [001] ..... 31240.381105: sample_threshold: start 31240.381090803 duration 14384 ns interference 0
```

The tracer is inspired on hwlat tracer.

rtla osnoise

bristot@x1:~ — sudo osnoise top

Operating System Noise

duration: 0 00:00:03 | time is in us

CPU	Period	Runtime	Noise	% CPU Aval	Max Noise	Max Single	HW	NMI	IRQ	Softirq	Thread
0	#3	3000000	61022	97.96593	22403	2849	2408	0	3669	917	887
1	#3	3000000	54360	98.18800	22041	8337	2403	0	3049	286	166
2	#3	3000000	35663	98.81123	12155	200	2424	1	3010	270	49
3	#3	3000000	37135	98.76216	13282	288	2425	1	3017	249	57
4	#3	3000000	37056	98.76480	13008	546	2468	1	3014	272	65
5	#3	3000000	35689	98.81036	12462	352	2436	0	3017	310	52
6	#3	3000000	36138	98.79540	13494	1820	2423	1	3007	273	31
7	#3	3000000	49978	98.33406	21294	3024	2415	0	3044	314	209

The tracer is inspired on hwlat tracer.

Osnoise: what is missing

- IPI tracing
  - We have IPI tracing in ARM but not in x86.
  - We need to beautify the results
- User-space/Other workloads
  - Can I hook any arbitrary workload?
  - How do I register a user-space PID as the workload?
- Integration with KVM/XEN
  - To know when the VM was preempted
  - Using the VM was a workload
- Integration with CI
  - Daniel Wagner integrated with lava!
  - We know when VM preempted the CPU, we can discard the results.

The tracer and rtla osnoise can be considered a single tool: what is possible from user space: do it on rtla (e.g., setting priorities), otherwise do in kernel.

Red Hat

Osnoise: user-space workload

- Other workload
  - Now we have a workload in the kernel, and we use its PID in the osnoise: events
  - How can we add other workloads? Events to hook?
- We can have a user-space workload
  - We will lose the synchronization with tracepoints
- The workload needs to be pinned to a CPU
  - Return -einval if not? Force pin from the kernel?
  - Return kernel workload if the workload dies.

Possible interface:

```
# cat osnoise/workload_pid
CPU  -    PID
0         1211
1         1212
2         1213

# echo 0:123123 > osnoise/workload_pid
```

We can also register a VCPU as the workload!

Red Hat

Osnoise: what is missing - IPI tracing

```
[root@ampere-hr330a-04 tracing]# cat available_events | grep ipi:
ipi:ipi_exit
ipi:ipi_entry
ipi:ipi_raise
# echo 'hist:name=ipi:key=stacktrace,target_cpus,reason:val=hitcount' > events/ipi/ipi_raise/trigger
```


Red Hat

Osnoise: what is missing - IPI tracing

```
[root@ampere-hr330a-04 ipi_raise]# head -200 hist
[...]
{ stacktrace:
         smp_send_reschedule+0x3c/0x50
         resched_curr+0x50/0x9c
         check_preempt_curr+0x58/0x94
         ttwu_do_wakeup+0x2c/0x1dc
         ttwu_do_activate+0x7c/0xf0
         try_to_wake_up+0x214/0x6d0
         wake_up_state+0x20/0x30
         wake_page_function+0xcc/0x120
         __wake_up_common+0x90/0x1b4
         __wake_up_locked_key_bookmark+0x2c/0x40
         folio_wake_bit+0x94/0x140
         folio_unlock+0x40/0x50
         unlock_page+0x28/0x70
         end_page_read+0x68/0xd0
         end_bio_extent_readpage+0x294/0x4a0
         bio_endio+0x15c/0x230
, target_cpus:   33554456, reason: 18446603336374800216} hitcount:          1
```


Red Hat

Osnoise: integration with hypervisor & CI

- Now hypervisors are identified as "hw noise"
- Knowing that the hypervisor preempted the workload is helpful to debug
  - Having the reason for it is even better
- Using the VM was a workload
  - It is a use-case of the user-space workload
- It is also helpful for CI
  - Most of the CI work is done on VMs
  - Knowing that the noise was from the hardware, we can ignore it - to focus on the "guest" only noise.