OSNOISE Tracer

In the context of high-performance computing (HPC), the Operating System Noise (*osnoise*) refers to the interference experienced by an application due to activities inside the operating system. In the context of Linux, NMIs, IRQs, SoftIRQs, and any other system thread can cause noise to the system. Moreover, hardware-related jobs can also cause noise, for example, via SMIs.

hwlat_detector is one of the tools used to identify the most complex source of noise: *hardware noise*.

In a nutshell, the hwlat_detector creates a thread that runs periodically for a given period. At the beginning of a period, the thread disables interrupt and starts sampling. While running, the hwlatd thread reads the time in a loop. As interrupts are disabled, threads, IRQs, and SoftIRQs cannot interfere with the hwlatd thread. Hence, the cause of any gap between two different reads of the time roots either on NMI or in the hardware itself. At the end of the period, hwlatd enables interrupts and reports the max observed gap between the reads. It also prints a NMI occurrence counter. If the output does not report NMI executions, the user can conclude that the hardware is the culprit for the latency. The hwlat detects the NMI execution by observing the entry and exit of a NMI.

The osnoise tracer leverages the hwlat_detector by running a similar loop with preemption, SoftIRQs and IRQs enabled, thus allowing all the sources of *osnoise* during its execution. Using the same approach of hwlat, osnoise takes note of the entry and exit point of any source of interferences, increasing a per-cpu interference counter. The osnoise tracer also saves an interference counter for each source of interference. The interference counter for NMI, IRQs, SoftIRQs, and threads is increased anytime the tool observes these interferences' entry events. When a noise happens without any interference from the operating system level, the hardware noise counter increases, pointing to a hardware-related noise. In this way, osnoise can account for any source of interference. At the end of the period, the osnoise tracer prints the sum of all noise, the max single noise, the percentage of CPU available for the thread, and the counters for the noise sources.

Usage

Write the ASCII text "osnoise" into the current_tracer file of the tracing system (generally mounted at /sys/kernel/tracing).

For example:

```
[root@f32 ~]# cd /sys/kernel/tracing/
[root@f32 tracing]# echo osnoise > current_tracer
```

It is possible to follow the trace by reading the trace file:

```
[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   SI
#              | |         |   ||||      |           |             |    |            |      |      |      |
           <...>-859     [000] ....    81.637220: 1000000        190  99.98100       9     18      0   1007
           <...>-860     [001] ....    81.638154: 1000000        656  99.93440      74     23      0   1006
           <...>-861     [002] ....    81.638193: 1000000       5675  99.43250     202      6      0   1013
           <...>-862     [003] ....    81.638242: 1000000        125  99.98750      45      1      0   1011
           <...>-863     [004] ....    81.638260: 1000000       1721  99.82790     168      7      0   1002
           <...>-864     [005] ....    81.638286: 1000000        263  99.97370      57      6      0   1006
           <...>-865     [006] ....    81.638302: 1000000        109  99.98910      21      3      0   1006
           <...>-866     [007] ....    81.638326: 1000000       7816  99.21840     107      8      0   1016
```

In addition to the regular trace fields (from TASK-PID to TIMESTAMP), the tracer prints a message at the end of each period for each CPU that is running an osnoise/ thread. The osnoise specific fields report:

- The RUNTIME IN US reports the amount of time in microseconds that the osnoise thread kept looping reading the time.
- The NOISE IN US reports the sum of noise in microseconds observed by the osnoise tracer during the associated runtime.
- The % OF CPU AVAILABLE reports the percentage of CPU available for the osnoise thread during the runtime window.
- The MAX SINGLE NOISE IN US reports the maximum single noise observed during the runtime window.
- The Interference counters display how many each of the respective interference happened during the runtime window.

Note that the example above shows a high number of HW noise samples. The reason being is that this sample was taken on a virtual machine, and the host interference is detected as a hardware interference.

Tracer options

The tracer has a set of options inside the osnoise directory, they are:

- osnoise/cpus: CPUs at which a osnoise thread will execute.
- osnoise/period_us: the period of the osnoise thread.
- osnoise/runtime_us: how long an osnoise thread will look for noise.
- osnoise/stop_tracing_us: stop the system tracing if a single noise higher than the configured value happens. Writing 0 disables this option.
- osnoise/stop_tracing_total_us: stop the system tracing if total noise higher than the configured value happens. Writing 0 disables this option.

- tracing_threshold: the minimum delta between two time() reads to be considered as noise, in us. When set to 0, the default value will be used, which is currently 5 us.

Additional Tracing

In addition to the tracer, a set of tracepoints were added to facilitate the identification of the osnoise source.

- osnoise:sample_threshold: printed anytime a noise is higher than the configurable tolerance_ns.
- osnoise:nmi_noise: noise from NMI, including the duration.
- osnoise:irq_noise: noise from an IRQ, including the duration.
- osnoise:softirq_noise: noise from a SoftIRQ, including the duration.
- osnoise:thread_noise: noise from a thread, including the duration.

Note that all the values are *net values*. For example, if while osnoise is running, another thread preempts the osnoise thread, it will start a thread_noise duration at the start. Then, an IRQ takes place, preempting the thread_noise, starting a irq_noise. When the IRQ ends its execution, it will compute its duration, and this duration will be subtracted from the thread_noise, in such a way as to avoid the double accounting of the IRQ execution. This logic is valid for all sources of noise.

Here is one example of the usage of these tracepoints:

```
     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 interferenc
```

In this example, a noise sample of 8 microseconds was reported in the last line, pointing to two interferences. Looking backward in the trace, the two previous entries were about the migration thread running after a timer IRQ execution. The first event is not part of the noise because it took place one millisecond before.

It is worth noticing that the sum of the duration reported in the tracepoints is smaller than eight us reported in the sample_threshold. The reason roots in the overhead of the entry and exit code that happens before and after any interference execution. This justifies the dual approach: measuring thread and tracing.