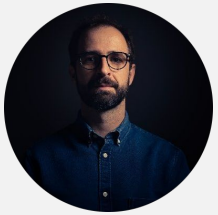
P99 CONF

OSNoise Tracer: Who Is Stealing My CPU Time?

Daniel Bristot de Oliveira, Ph.D.
Principal Software Engineer at Red Hat

Daniel Bristot de Oliveira

Principal Software Engineer at Red Hat



- Kernel developer with interest in RT/RV theoretical aspects
- I am here to share some research that landed into Linux
- Post-doc researcher at Scuola Superiore Sant'Anna
- AFK: Photography for mental health, cycling for body health

Red Hat

I am a regular user, how can I see who is stealing my cpu time?

Use top

```
top - 18:22:44 up 6 days, 10:05,  1 user,  load average: 1.06, 0.95, 0.77
Tasks: 341 total,   3 running, 338 sleeping,   0 stopped,   0 zombie
%Cpu(s): 10.1 us,  2.9 sy,  0.0 ni, 85.8 id,  0.1 wa,  0.6 hi,  0.4 si,  0.0 st
MiB Mem :  15765.3 total,   1518.5 free,   5059.2 used,   9187.5 buff/cache
MiB Swap:   8192.0 total,   8179.7 free,     12.2 used.   8544.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2156 bristot   20   0 5582816 304464 109084 S  25.7   1.9 125:07.16 gnome-shell
 281950 bristot   20   0   20.7g 383492 187800 S  19.1   2.4   2:15.53 chrome
 281405 bristot   20   0   16.9g 372696 215632 S  18.8   2.3   3:23.05 chrome
 281455 bristot   20   0   17.3g 196280 128200 S  18.5   1.2   3:11.31 chrome
   1961 bristot   20   0 1365748 200180 138840 R  10.2   1.2 104:28.12 Xorg
 286778 bristot   20   0   20.6g 152708 106008 S   2.3   0.9   0:19.14 chrome
 280507 bristot   20   0  767428  53824  37708 R   2.0   0.3   0:07.24 gnome-terminal-
 281456 bristot   20   0   16.5g 121220  91820 S   1.7   0.8   0:41.24 chrome
```

P99 CONF

Daniel Bristot de Oliveira

bristot@kernel.org

@bristot

That is it!?

Wait, there are still some slides for the non-regular users.

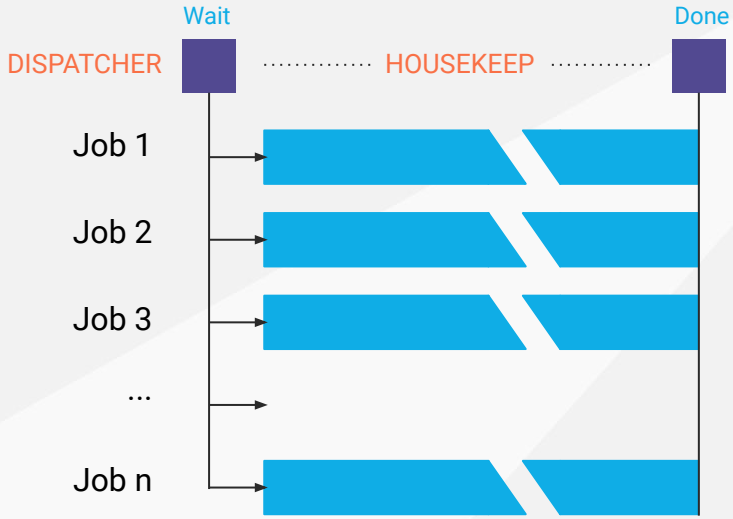
Introduction

What is OS Noise?

- The Operating Systems Noise (**OS Noise**) is a well defined High Performance Computing (**HPC**) metric
- It is the amount of **interference** experienced by an application due to **operating system activities**
- It is generally a fine grained metric

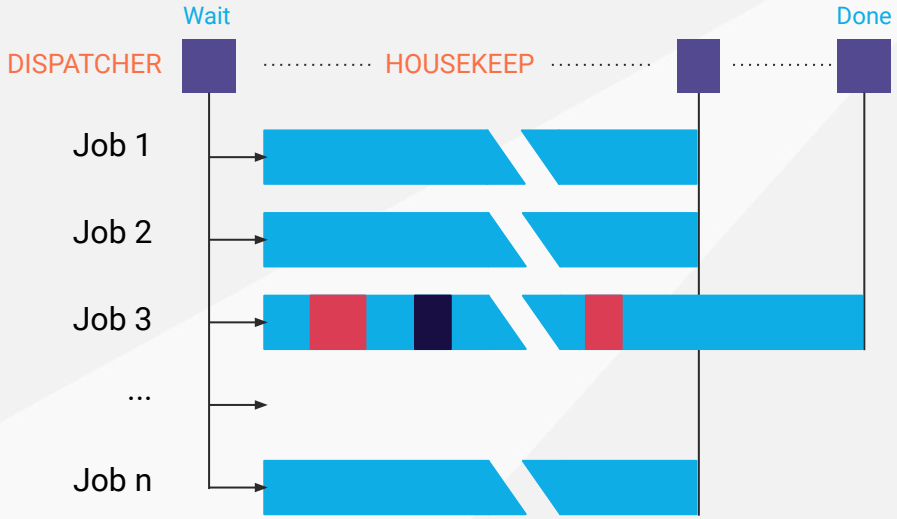
HPC Workload

- Generally, HPC workloads are composed of parallel jobs
- The system is configured with CPUs dedicated to the jobs
- A dispatcher lunches jobs to these CPUS and wait for completion.



HPC Workload

- The problem with OS Noise is that the OS interference on a single job can cause a delay in the entire task:



HPC meets Low Latency

- Low latency communication are touching the sub millisecond range
  - Allowing low latency services in the cloud
- Many new options enabled by 5G
- And Linux is becoming part of the core of the **network with of NFV**
- This work is not parallel like regular HPC, but serial among the hops.
  - **The effect of OS Noise are cumulative in the round-trip time**
- Many providers request latency in the order of microseconds.

the state-of-art

How OS Noise is measured today?

- A tool in **user-space** reads the time in a loop
  - Computes the delta between two time reads
  - Report each delta > threshold as a "jitter" or "latency"
- For the bugging, the user needs to setup a set of tracing events
  - The user-space tool stops the trace when hitting a "spike"
- Human interpretation of the trace

Problems with the current approach

- The **trace** and the **benchmark tool** are not synchronized
  - This leaves gaps for interpretation and "doubts"
  - Requires the trace of multiple events - to be interpreted by a human
  - Too much room for speculation
- There is no clear definition of the metric
  - And so no clear method to debug it
- Other tools are required for hw/virtualization induced noise:
  - most notably hwlat detector

How can these problems be solved?

- Improve the information given by the measuring tool
  - Informing reasons for a given "noise" occurrence
- Making the workload and the trace to be in sync
  - The workload and the trace needs to "atomically" in sync
- Tracing automation
  - Define/standardize the most essential information
  - Reduce the amount of events passed to the user (to reduce overhead)
  - Do the common interpretation before "printing" the trace

osnoise tracer

osnoise tracer

- Osnoise is a **kernel tracer** that also dispatches the workload
  - The workload runs in kernel
- The workload and the trace are synchronized
  - Likewise other tools, `osnoise` measures the time delta
  - But it also measures the amount of interference from OS Operations

Enabling osnoise

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

osnoise tracer output

```
                  _-----=> 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
```

osnoise tracer config

- Configuration files inside /sys/kernel/trace/osnoise
  - cpus: CPUs at which a osnoise thread will execute.
  - period_us: the period of the osnoise thread.
  - runtime_us: how long an osnoise thread will look for noise in the period
  - stop_tracing_us: stop the system tracing if a single noise is >= than set here
  - stop_tracing_total_us: stop the system tracing if total noise is >= than set here
- /sys/kernel/trace/tracing_threshold
  - The minimum delta between two time() reads to be considered as noise, in us.
  - When set to 0, the default value will will be used, which is currently 5 us.

Finding sources of noise

What can steal your cpu time?

- Characterization of osnoise:
  - Any sort of task tha interference (preempt) the `osnoise` workload
- Linux task abstractions:
  - NMI
  - IRQs
  - Softirqs
  - Threads
- But also the hardware can interfere on your task
  - SMIs
  - VMs

Osnoise tracepoints

- The osnoise: tracepoints process all the data in kernel, in sync with the tracer
  - To reduce overhead when enabled
  - No overhead when disabled
- The events are:
  - 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.
  - osnoise:sample_threshold: printed anytime a noise is found, including the $ of interferences

osnoise tracer output

```
[root@f32 ~]# cd /sys/kernel/tracing/
[root@f32 tracing]# echo osnoise > current_tracer
[root@f32 tracing]# echo osnoise > set_event
[root@f32 tracing]# echo 8 > osnoise/stop_tracing_us
[root@f32 tracing]# cat trace
[...]
  osnoise/8-960  [007]  5789.857530: irq_noise: local_timer:236 start 5789.857527123 duration 1867 ns
  osnoise/8-961  [008]  5789.857532: irq_noise: local_timer:236 start 5789.857529929 duration 1845 ns
  osnoise/8-961  [008]  5789.858408: irq_noise: local_timer:236 start 5789.858404871 duration 2848 ns
migration/8-54   [008]  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
```

How about hardware noise?

- `osnoise` tracks all sources of OS Noise
- osnoise computes the **delta time** and the **interference** counter on every loop
- When the interference counter == 0:
  - The cause of the noise is from outside the OS
  - It is computed as hardware, like **hwlat detector** does
  - The hardware can be either physical or virtual VMs

hardware counter output

```
                  _-----=> 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
```

hardware noise output

```
[root@f32 ~]# cd /sys/kernel/tracing/
[root@f32 tracing]# echo osnoise > current_tracer
[root@f32 tracing]# echo osnoise > set_event
[root@f32 tracing]# echo 8 > osnoise/stop_tracing_us
[root@f32 tracing]# cat trace
[...]
  osnoise/0-713  [000]  66.570788: irq_noise: local_timer:236 start 66.570786364 duration 1593 ns
  osnoise/2-715  [002]  66.570788: irq_noise: local_timer:236 start 66.570786364 duration 1628 ns
  osnoise/4-717  [004]  66.570788: irq_noise: local_timer:236 start 66.570786377 duration 1568 ns
  osnoise/0-713  [000]  66.570871: sample_threshold: start 66.570862574 duration 8038 ns interference 0
  osnoise/3-716  [003]  66.571788: irq_noise: local_timer:236 start 66.571786373 duration 1555 ns
  osnoise/7-720  [007]  66.571788: irq_noise: local_timer:236 start 66.571786396 duration 1536 ns
```

Final thoughts

- osnoise tracer puts the workload and the tracer in a single tool
- Provides information and tracing that points to the root cause of OS Noise
  - Processing the data to reduce the overhead to the minimum possible
- It can also be used to detect hardware latency
- The **timerlat** tracer is osnoise's sibling for interrupt based latency.
- It is part of the kernel and is enabled on recent Fedora/CentOS/Red Hat
- rtla tool adds an intuitive interface for osnoise tracer

P99 CONF

Daniel Bristot de Oliveira

bristot@kernel.org

@bristot