Can we blame the Kernel instead of Open vSwitch?

OVS/OVN Conference 2023

Eelco Chaudron
Principal Software Engineer
Agenda

- Explaining the problem
- Introduction to the *kernel_delay.py* tool
- Demo debugging the revalidator
- Questions?
Explaining the problem
### Explaining the problem

- **ovs-vswitchd** can generate weird log messages:
  - ... | WARN | Unreasonably long 1259ms poll interval (0ms user, 692ms system)
  - ... | WARN | blocked 1001 ms waiting for handler340 to quiesce

- Is this **ovs-vswitchd** or an overloaded/misbehaving kernel?

- Specific purpose tools exist
  - Coordinating and utilizing these tools efficiently can be a hurdle.

- Can we have a single tool to rule out the kernel?
Introduction to the `kernel_delay.py` tool
Introduction to the kernel_delay.py tool

- Python based script
- Uses eBPF hooks/programs to gather information
  - Uses the BCC framework
- Can be triggered on demand
- Will report on the following
  - SYSCALL statistics
  - THREAD RUN statistics
  - THREAD READY statistics
  - HARD IRQ statistics
  - SOFT IRQ statistics
- See blog and documentation for more information

[1] https://github.com/iovisor/bcc
Run the kernel_delay.py tool

```bash
$ sudo ./kernel_delay.py
# Start sampling @2023-06-08T12:17:22.725127 (10:17:22 UTC)
# Stop sampling @2023-06-08T12:17:23.224781 (10:17:23 UTC)
# Sample dump @2023-06-08T12:17:23.224855 (10:17:23 UTC)
```

### TID THREAD <RESOURCE SPECIFIC>

<table>
<thead>
<tr>
<th>TID</th>
<th>THREAD</th>
<th>&lt;RESOURCE SPECIFIC&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>31741</td>
<td>revalidator122</td>
<td>[SYSCALL STATISTICS]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NAME                 NUMBER</td>
</tr>
</tbody>
</table>
poll  7 | 5          | 184,193,176          | 184,191,520 |
recvmsg 47 | 494      | 125,208,756          | 310,331     |

... TOTAL( - poll): 519 144,405,334

### [THREAD RUN STATISTICS]

<table>
<thead>
<tr>
<th>SCHED_CNT</th>
<th>TOTAL ns</th>
<th>MIN ns</th>
<th>MAX ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>136,764,071</td>
<td>1,480</td>
<td>115,146,424</td>
</tr>
</tbody>
</table>

### [THREAD READY STATISTICS]

<table>
<thead>
<tr>
<th>SCHED_CNT</th>
<th>TOTAL ns</th>
<th>MAX ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>11,334</td>
<td>6,636</td>
</tr>
</tbody>
</table>

### [HARD IRQ STATISTICS]

<table>
<thead>
<tr>
<th>NAME</th>
<th>COUNT</th>
<th>TOTAL ns</th>
<th>MAX ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>eno8303-rx-1</td>
<td>1</td>
<td>3,586</td>
<td>3,586</td>
</tr>
</tbody>
</table>

TOTAL: 1 3,586

### [SOFT IRQ STATISTICS]

<table>
<thead>
<tr>
<th>NAME</th>
<th>VECT_NR</th>
<th>COUNT</th>
<th>TOTAL ns</th>
<th>MAX ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>net_rx</td>
<td>3</td>
<td>1</td>
<td>17,699</td>
<td>17,699</td>
</tr>
<tr>
<td>sched</td>
<td>7</td>
<td>6</td>
<td>13,820</td>
<td>3,226</td>
</tr>
<tr>
<td>rcu</td>
<td>9</td>
<td>16</td>
<td>13,586</td>
<td>1,554</td>
</tr>
<tr>
<td>timer</td>
<td>1</td>
<td>3</td>
<td>10,259</td>
<td>3,815</td>
</tr>
</tbody>
</table>

TOTAL: 26 55,364
kernel_delay.py has two modes of operation

- In time mode, the tool runs for a specific time and collects the information.
- In trigger mode, event collection can be started and/or stopped based on a specific eBPF probe.

Supported trigger probes:
- USDT probes
- Kernel tracepoints
- kprobe
- kretprobe
- uprobe
- uretprobe
Additional *sample* options exist:

- `--sample-count;` How many measurements you would like to perform.
- `--trigger-delta;` Ignore measurements if the delta is less than configured.
- `--sample-interval;` Delay the start of a new measurement.
Using triggers

- Supports start and stop triggers in any combination
  - Start only example:
    ```bash
    # ./kernel_delay.py --start-trigger up:bridge_run --sample-time 4 \   --sample-count 2 --sample-interval 1
    ```
  - Stop only example:
    ```bash
    # ./kernel_delay.py --stop-trigger upr:bridge_run \   --sample-count 4 --sample-interval 1
    ```
  - Start and stop example:
    ```bash
    # ./kernel_delay.py --start-trigger up:bridge_run \   --stop-trigger upr:bridge_run \   --sample-count 4 --sample-interval 1 \   --trigger-delta 50000
    ```
Supported trigger probes and syntax

- **USDT probes**[2]: \([u]:\{provider\}:\{probe\}\)
- **Kernel tracepoint**; \([t]:\{trace\}:\{system\}:\{event\}\)
- **Kprobe**; \([k]:\{kprobe\}:\{kernel\_function\}\)
- **Kretprobe**; \([kr]:\{kretprobe\}:\{kernel\_function\}\)
- **Uprobe**; \([up]:\{uprobe\}:\{function\}\)
- **Uretprobe**; \([upr]:\{uretprobe\}:\{function\}\)

What will it report?

- Currently has five statistics it’s gathering
  - SYSCALL statistics
  - THREAD RUN statistics
  - THREAD READY statistics
  - HARD IRQ statistics
  - SOFT IRQ statistics
- More can be easily added if needed in the future
SYSCALL STATISTICS

- Report all syscalls per thread for the measurement duration
- Per type; count of calls, total duration, and worst case duration
- It only counts syscall that are started AND stopped during the interval

<table>
<thead>
<tr>
<th>NAME</th>
<th>NUMBER</th>
<th>COUNT</th>
<th>TOTAL ns</th>
<th>MAX ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>poll</td>
<td>7</td>
<td>4</td>
<td>1,494,094,595</td>
<td>500,838,810</td>
</tr>
<tr>
<td>ioctl</td>
<td>16</td>
<td>14</td>
<td>7,105,005</td>
<td>3,284,088</td>
</tr>
<tr>
<td>read</td>
<td>0</td>
<td>10</td>
<td>1,088,265</td>
<td>225,845</td>
</tr>
<tr>
<td>accept</td>
<td>43</td>
<td>15</td>
<td>38,603</td>
<td>9,978</td>
</tr>
<tr>
<td>socket</td>
<td>41</td>
<td>14</td>
<td>36,769</td>
<td>6,520</td>
</tr>
<tr>
<td>openat</td>
<td>257</td>
<td>5</td>
<td>33,489</td>
<td>16,825</td>
</tr>
<tr>
<td>sendmsg</td>
<td>46</td>
<td>1</td>
<td>31,454</td>
<td>31,454</td>
</tr>
<tr>
<td>recvmsg</td>
<td>47</td>
<td>26</td>
<td>19,587</td>
<td>7,536</td>
</tr>
<tr>
<td>futex</td>
<td>202</td>
<td>12</td>
<td>10,003</td>
<td>3,649</td>
</tr>
<tr>
<td>close</td>
<td>3</td>
<td>19</td>
<td>4,885</td>
<td>531</td>
</tr>
<tr>
<td>ready</td>
<td>19</td>
<td>10</td>
<td>4,404</td>
<td>854</td>
</tr>
<tr>
<td>recvfrom</td>
<td>45</td>
<td>5</td>
<td>3,056</td>
<td>1,215</td>
</tr>
<tr>
<td>getusage</td>
<td>98</td>
<td>5</td>
<td>2,728</td>
<td>1,004</td>
</tr>
<tr>
<td>TOTAL ( - poll):</td>
<td>136</td>
<td>8,378,246</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

217258 ovs-vswitchd [SYSCALL STATISTICS]
Report how long the thread was running on a CPU
Counts time the thread was scheduled on and off
Records total, minimum and maximum CPU time
It only counts events that started AND stopped during the interval
Note PMD threads might show nothing for this statistic due to the above

217258 ovs-vswitchd  [SYSCALL STATISTICS]
...

[THREAD RUN STATISTICS]

<table>
<thead>
<tr>
<th>SCHED_CNT</th>
<th>TOTAL ns</th>
<th>MIN ns</th>
<th>MAX ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>230</td>
<td>1,459,544</td>
<td>1,381</td>
<td>155,609</td>
</tr>
</tbody>
</table>
- Report how long the thread was waiting for CPU time
- Records total and maximum schedule delay
- It only counts events that started AND stopped during the interval
- Note PMD threads might show nothing for this statistic due to the above

```
217258 ovs-vswitchd     [SYSCALL STATISTICS]
...
[THREAD RUN STATISTICS]
SCHED_CNT   TOTAL ns   MIN ns   MAX ns
 230        1,459,544   1,381    155,609

[THREAD READY STATISTICS]
SCHED_CNT   TOTAL ns   MAX ns
 230        66,745     2,984
```
HARD IRQ STATISTICS

- Report time spent servicing hard interrupts during the threads run time
- Records per irq vector count, total duration, and worst case duration

217331 revalidator48  [SYSCALL STATISTICS]
...
...

[HARD IRQ STATISTICS]

<table>
<thead>
<tr>
<th>NAME</th>
<th>COUNT</th>
<th>TOTAL ns</th>
<th>MAX ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>eno8303-rx-1</td>
<td>1</td>
<td>3,586</td>
<td>3,586</td>
</tr>
<tr>
<td>TOTAL:</td>
<td>1</td>
<td>3,586</td>
<td></td>
</tr>
</tbody>
</table>
- Report time spent servicing soft interrupts during the threads run time
- Records per irq vector count, total duration, and worst case duration

<table>
<thead>
<tr>
<th>NAME</th>
<th>VECT_NR</th>
<th>COUNT</th>
<th>TOTAL ns</th>
<th>MAX ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>sched</td>
<td>7</td>
<td>1</td>
<td>2,149</td>
<td>2,149</td>
</tr>
<tr>
<td>rcu</td>
<td>9</td>
<td>1</td>
<td>890</td>
<td>890</td>
</tr>
<tr>
<td>TOTAL:</td>
<td></td>
<td>2</td>
<td>3,039</td>
<td></td>
</tr>
</tbody>
</table>
The --syscall-events option

- The **--syscall-events** option will report individual syscalls
- Has an optional argument to only report call taking more than x ns
- Does support backtraces, but are not that useful\[1\], can be disabled with **--stack-trace-size 0**
- Skip poll() system calls with **--skip-syscall-poll-events**

```bash
# ./kernel_delay.py  --syscall-events 50000 --skip-syscall-poll-events
...
...
# SYSCALL EVENTS:
ENTRY (ns)           EXIT (ns)        TID COMM             DELTA (us)  SYSCALL
------------------- ------------------- ---------- ---------------- ----------  ----------------
2161821694935486    2161821695031201    3359699 revalidator14            95  futex
  syscall_exit_to_user_mode_prepare+0x161 [kernel]
  syscall_exit_to_user_mode_prepare+0x161 [kernel]
  syscall_exit_to_user_mode+0x9 [kernel]
  do_syscall_64+0x68 [kernel]
  entry_SYSCALL_64_after_hwframe+0x72 [kernel]
    __GI___lll_lock_wait+0x30 [libc.so.6]
    ovs_mutex_lock_at+0x18 [ovs-vswitchd]
    [unknown] 0x696c003134313a63
2161821695276882    2161821695333687    3359698 revalidator13            56  futex
  syscall_exit_to_user_mode_prepare+0x161 [kernel]
    ovs_mutex_lock_at+0x18 [ovs-vswitchd]
    [unknown] 0x696c003134313a63
```

\[1\] https://github.com/chaudron/perf_scripts/blob/master/analyze_perf_pmd_syscall.py
Demo debugging the revalidator
Questions?
Thank you

Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.