MIRANTIS
Pure Play OpenStack®

2016.02.17
Kernel debugging notes

Fabrizio Soppelsa
Escalations engineer

www.mirantis.com

Agenda

MIRANTIS
Pure Play OpenStack

- Kernel problems
- Kernel logs
- Increasing the verbosity
- ftracing the kernel
- Dumps and debuggers
- lttng
- Q&A&D

Kernel errors - why

- Bugs as in userspace
- Wrong synchronization
- Security issues
- Incorrect code

Kernel errors - results

- Performance issues
- Incorrect behaviours
- Data corruptions
- Kernel crashes

Kernel logs

- **messages** (and klogd)
- **dmesg**
- /proc/kmsg
- /dev/kmsg

Verbosity

MIRANTIS
Pure Play OpenStack

- Use `printk()`
- dmesg is circular!
- Klogd
- Look for info in /proc
- `echo "8" > /proc/sys/kernel/printk`
- Remember to restore the `printk` value!

Klogd

- Dumps kernel logs to messages
- Requires to install the **klogd** daemon
- `apt-get install klogd`

Super hyper verbosity

MIRANTIS
Pure Play OpenStack

Hard tasks:

- Use of kernel ftrace https://www.kernel.org/doc/Documentation/trace/ftrace.txt
- Tracer: function
- Collect bomb-proof proves
- Make customer confess their code modifications
- Make customer to fix their code modifications
- Resolved

Using kernel ftrace

```
# sysctl kernel.ftrace_enabled=1
# cd /sys/kernel/debug/tracing
# echo function > current_tracer
# echo 1 > tracing_on
# … Reproduce error
# echo 0 > tracing_on
# less trace
```

Example of a kernel ftrace

```
# tracer: function
#
# entries-in-buffer/entries-written: 409852/3735717   #P:8
#
#                              _-----=> irqs-off
#                              / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth
#                              ||| / delay
#           TASK-PID   CPU#  ||||   TIMESTAMP  FUNCTION
#           | |    |   ||||    |      |
      <idle>-0    [004] d.h.  5819.204206: handle_irq_event <-handle_edge_irq
      <idle>-0    [004] d.h.  5819.204224: _raw_spin_unlock <-handle_irq_event
      <idle>-0    [004] d.h.  5819.204225: handle_irq_event_percpu <-handle_irq_event
      <idle>-0    [004] d.h.  5819.204225: vring_interrupt <-handle_irq_event_percpu
      <idle>-0    [004] d.h.  5819.204225: skb_recv_done <-vring_interrupt
      <idle>-0    [004] d.h.  5819.204226: virtqueue_disable_cb <-skb_recv_done
      <idle>-0    [004] d.h.  5819.204226: __napi_schedule <-skb_recv_done
      <idle>-0    [004] d.h.  5819.204226: __raise_softirq_irqoff <-__napi_schedule
      <idle>-0    [004] d.h.  5819.204226: add_interrupt_randomness <-handle_irq_event_percpu
```

Enable kernel core dumps

```
# apt-get install linux-crashdump
```

Change **USE_KDUMP=1** in **/etc/default/kdump-tool**

```
# reboot
```

Analyzing a dump

- Requires good knowledge of the kernel
- Dumps are stored in **/var/crash**
- Activity: Time consuming
- Using gdb, kdbg and other tools

Book
http://www.amazon.com/Systems-Performance-Enterprise-Brendan-Gregg/dp/0133390098

Advanced - lttng

LTTng uses the Tracepoint instrumentation of the Linux kernel, as well as various other information sources such as kprobes, and the Perf performance monitoring counters. It is useful for debugging a wide range of bugs that are otherwise extremely challenging. These include, for example, performance problems on parallel systems and on real-time systems. Custom instrumentation is easy to add. LTTng is designed for minimal performance impact and has a near zero impact when not tracing. LTTng has at least basic support for all Linux-supported architectures (see the LTTng-modules README file for more details).

```
root@node-6:~# apt-get install liblttng \
lttng-modules-dkms lttng-tools
```

Advanced - lttng & kernel

MIRANTIS
Pure Play OpenStack

http://www.lttng.org/docs/#doc-tracing-the-linux-kernel

```
root@node-6:~# lsmod | grep ltt
lttng_tracer            1193921 0
lttng_statedump          669725 1 lttng_tracer
lttng_ftrace             13274  1 lttng_tracer
lttng_kprobes            13091  1 lttng_tracer
lttng_lib_ring_buffer    55166  1 lttng_tracer
lttng_kretprobes    13762  1 lttng_tracer
```

Advanced - lttng create

```
root@node-6:~# lttng create
Session auto-20160216-132912 created.
Traces will be written in
/root/lttng-traces/auto-20160216-132912
root@node-6# lttng enable-event --kernel --all
All Kernel events are enabled in channel channel0
root@node-6# lttng start
Tracing started for session auto-20160216-132912
root@node-6# lttng stop
Waiting for data availability.
Tracing stopped for session auto-20160216-132912
```

Advanced - read results

MIRANTIS
Pure Play OpenStack

```
# babeltrace lttng-traces/
[15:00:41.017823948] (+?.??????????) node-6 kmem_kmalloc: { cpu_id = 5 }, { call_site = 0xFFFFFFFFA0A34267, ptr =
0xFFFF88082F280000, bytes_req = 708, bytes_alloc = 1024, gfp_flags = 208 }
[15:00:41.017828520] (+0.000004572) node-6 kmem_kfree: { cpu_id = 5 }, { call_site = 0xFFFFFFFFA0A3433A, ptr =
0xFFFF88082F280000 }
[15:00:41.017831385] (+0.000002865) node-6 sched_wakeup: { cpu_id = 5 }, { comm = "lttng-consumerd", tid = 1013,
prio = 20, success = 1, target_cpu = 6 }
[15:00:41.017832548] (+0.000001163) node-6 power_cpu_idle: { cpu_id = 6 }, { state = 4294967295, cpu_id = 6 }
[15:00:41.017833985] (+0.000001437) node-6 kmem_kmalloc: { cpu_id = 5 }, { call_site = 0xFFFFFFFFA0A34267, ptr =
0xFFFF88085063FC00, bytes_req = 288, bytes_alloc = 512, gfp_flags = 208 }
[15:00:41.017834556] (+0.000000571) node-6 timer_hrtimer_cancel: { cpu_id = 6 }, { hrtimer = 18446612168818088896 }
[15:00:41.017835293] (+0.000000737) node-6 timer_hrtimer_start: { cpu_id = 6 }, { hrtimer = 18446612168818088896,
function = 18446744071579764256, expires = 527768660000000, softexpires = 527768660000000 }
[15:00:41.017836030] (+0.000000737) node-6 kmem_kfree: { cpu_id = 5 }, { call_site = 0xFFFFFFFFA0A3433A, ptr =
0xFFFF88085063FC00 }
```

Q&A&D

You