Tracing in the Real World

Julien Desfossez <julien.desfossez@polymtl.ca>

August 19, 2011
LinuxCon North America

Content

1. Debugging
    1. For developers
    2. For sysadmins
2. LTTng and LTTng 2.0
    1. Installation
    2. New features
    3. lttngtop
    4. Demos
3. Future of tracing for everyone

Tracing

- Monitor and record operations of a system
- Pretty much like logging for the entire OS
- Sysadmins love logs
- Why only (some) developers use tracing ?

Debugging for developers

- Developers work in their own environment
- Developers can take time to understand why a problem is happening
- Developers don't necessarily care about the efficiency of the tracer (printf, strace)... they should, but that's another story...

Debugging for sysadmins

- Sysadmins like text log files
- Sysadmins like console tools
- Sysadmins work remotely
- Sysadmins work on production systems
- Sysadmins work under pressure

Solving problems as a sysadmin

- Easy :
  - Most of load related problems can be found with top, vmstat, iotop, tcpdump, etc.
- Hard :
  - What about problems happening “sometimes” ?
  - How do you fix a “the server feels slow” symptom when nothing is obvious ?

Can non-kernel developers use kernel tracing ?

- Kernel level tracers give a lot of useful information
- Most of the time it is too detailled when you don't want to read the kernel source code
- Most of the time it is too much for a production environment
- Need for an external tool to analyse the recorded data
- **Too many tracers doing different but complementary things in their own format, how do you choose ?**

What is LTTng ?

- Highly efficient full system tracing solution
- Kernel and Userspace tracing
- Tools to analyse offline and live traces
- Trace streaming infrastructure
- Unified trace format
- GPLv2, LGPLv2.1 and MIT licensed
- Tested on x86, x86_64, PPC, ARM, Sparc
- But : used to be hard to use, needed to patch the kernel, complex GUI to read traces with nanoseconds accuracy

Introducing LTTng 2.0

- Same purpose as LTTng but :
  - Generic file format (Common Trace Format)
  - Module based (for 2.6.35+)
  - Secure and unified control tools (lttng-tools)
  - Easy to install on Ubuntu, Debian, Fedora and others (without reboot !)
  - Tracing group
  - Currently 2.0 pre-8, final 2.0 really soon

LTTng 2.0 kernel data sources

- Tracepoints
- Function tracer (aka **ftrace**)
- CPU Performance Monitoring Unit counters (aka **perf**)
- Kprobes
- Kretprobes

Installation on Ubuntu

apt-add-repository ppa:lttng

apt-get install lttng-modules-dkms lttng-tools babeltrace

Record a trace

```
# lttng create mysession
# lttng enable-event -k -a
# lttng add-context -k -t pid -t comm -t tid -t ppid
# lttng start
...do stuff...
# lttng stop
# lttng destroy
# babeltrace -n /path/to/trace
```

Introducing lttngtop

- A top-like application to read traces
- Ncurses
- GPLv2
- Browse through recorded traces
- Display various statistics
- Demo

Solving “weird” sysadmin problems with common tools

- “Sometimes at random time the server slows down”
  - while true; do d=$(date +%s); uptime > log-${d}; vmstat 2 5 >> log-${d}; ps aux >> log-${d}; sleep 10; done
  - Log files
  - Cacti, nagios, munin, ganglia, collectd...

Solving “weird” sysadmin problems with tracing

- Sometimes at random time the server “slows down”
  - Record as much information possible
  - Make sure the tracing itself does not impact the server too much
  - Wait the amount of time necessary until the problem reappears
  - Play back the interesting part of the trace

Using performance counters to solve Real World problems

- I have a server that does a lot of disk I/O and runs my Apache server, it is just a simple HTML site, it should be in cache and the disk activity should not have an impact on it right ?

Live tracing ?

- Early beta demo ?

Tracing in the Real World

- Kernel tracing is not limited to kernel developpers
- System administrators can benefit from this amount of information if it is presented in a useful way
- For solving complex problems, we can benefit from this information to track the problem back to kernel bug :
  - To get precise statistics, use the performance counter on the event you suspect
  - Once you know where the problem is happening, use the function tracer to identify the culprit
  - Once you know which function is posing problem, use kprobe to identify which instruction is problematic

Tracing in the Real World : tracing for end users ?

- Replace polling by tracing :
  - Wakeup when trace data is available
  - Read and process buffers directly in memory, absolutely no I/O
- Example applications :
  - Various load monitoring applet score really high in CPU wakeup (look at powertop) and generate a lot of I/O
  - Monitoring on embedded devices (Linux based smartphones...)

Informations

- LTTng : http://lttng.org/lttng2.0
- lttngtop : http://git.dorsal.polymtl.ca/?p=lttngtop.git;a=summary