LTTng 2.1 : Advanced Linux tracing for everyone

Yannick Brosseau

LCA 2013
What is Tracing?
Definition:

Tracing is similar to logging: it consists in recording events that happen in a system. However, it usually records much lower-level events that occur much more frequently.
A way to see Inside...
A way to see Inside...

At specific points
Tracepoints can be Static or Dynamic
Tracepoints characteristics:
- High speed
- Low latency
- Timestamp
- Payload
What can it be used for?
Debugging
Events history
What is LTTng?
Tracers Timeline

- 1999 : LTT
- 2005 : LTTng 0.x
- 2005 : Dtrace
- 2005 : SystemTap
- 2008 : Ftrace
- 2009 : Perf
- 2012 : LTTng 2.0
- 2013 : LTTng 2.1
Userspace and kernel tracers
What's so special about LTTng 2.x?
Tracing for Everyone?
Easy installation
No kernel patch required

• Kernel tracing
  • Only kernel modules
  • Support kernel from 2.6.38
  • Runs on 2.6.32 with 3 small kernel patches

• Userspace tracing
  • Don't rely on the kernel
Linux distribution

- Ubuntu
- Debian
- Fedora (without kernel modules)
- Arch
- Suse
- Red Hat (pending)
Multi-arch and other OS

- x86 and x86-64
- ARM
- PowerPC, Sparc, Mips, Tile, Xeon Phi
- Android (Ongoing)
- FreeBSD (UST)
- Cygwin (UST)
Simple and unified command line (kernel + userspace)

Also available via a control API in C and python
(create, enable, start, stop, view, destroy)

# lttng create
# lttng enable-event -k -a
# lttng enable-event -u -a
# lttng start
# lttng stop
# lttng view
# lttng destroy
Tracing by non root users (tracing group)

Multiple active sessions
LTTng 2.0 Low-Overhead Tracing Architecture

Host-Side User Interfaces
- **Babeltrace** (MIT/BSD)
  - Trace converter
  - Trace pretty printer
  - Allow open source and proprietary plugins
  - libbabeltrace (MIT/BSD)

- **LTTV** (GPLv2)
  - Trace display and analysis
  - Trace control
  - Allow open-source plugins
  - libbabeltrace (MIT/BSD)

- **Eclipse Tracing and Monitoring Framework** (EPL)
  - Trace display and analysis
  - Trace control
  - Allow open source and proprietary plugins

Host

**LTTng Command Line Interface** (GPLv2)
- **liburcu** (LGPLv2.1)
- **liblttng-ctl** (GPLv2)

**LTTng Session Daemon** (GPLv2)
- Control multiple tracing sessions
- Centralized tracing status management
- **liburcu** (LGPLv2.1)
- **liblttng-ctl** (GPLv2)

**Custom Control Software**
- Interface with proprietary cluster management infrastructures
- **liburcu** (LGPLv2.1)
- **liblttng-ctl** (GPLv2)

**LTTng Consumer Daemon** (GPLv2)
- Zero-copy data transport or aggregator
- Export raw trace data, statistics and summary data
- Snapshots from in-memory flight recorder mode
- Store all trace data, discard on overrun
- **liburcu** (LGPLv2.1)
- **liblttng-ctl** (GPLv2)
- **liblttng-consumer** (GPLv2)

Target

**C/C++ Application**
- Tracepoint
- Tracepoint Probes
- **liburcu** (LGPLv2.1)
- **liblttng-ust** (GPLv2.1)

**Java/Erlang Application**
- Tracepoint
- **liburcu** (LGPLv2.1)

**Linux kernel**
- Tracepoint
- Dynamic probes
- **liburcu** (LGPLv2.1)

**LTTng VM adaptor**
- Tracepoint
- **liburcu** (LGPLv2.1)

**LTTng modules**
- **liburcu** (LGPLv2.1)
- **liblttng-ust** (GPLv2.1)

**Tracepoint and Probes Characteristics**
- Low overhead, no trap, no system call,
- Re-entrant: Signal, thread and NMI-safe,
- Wait-free read-copy update,
- Can be used in real-time systems,
- Use GCC asm goto and Linux kernel static jumps,
- Cycle-level time-stamp,
- Runtime activation of statically and dynamically inserted instrumentation,
- Non-blocking atomic operations,
- Allow tracing of proprietary applications and proprietary control software (LGPLv2.1 license).

Local storage

- **CTF†**

CTF† over TCP/UDP/SSH

† **Common Trace Format (CTF)**
- Compact binary format,
- Self-described,
- Handles HW&SW tracing,
- TCP and UDP network streaming,
- Flexible data layouts for expressiveness and highest throughput,
- Layout allows fast seek and processing of very large traces (> 10GB).
Multiples data sources...

• Kernel :
  • Static Tracepoints
  • Syscalls
  • Kprobes
  • Function tracer
  • Context (including perf counters)

• Userspace
  • Static Tracepoints
    – With filtering
  • Context
… That produce CTF data

- Common Trace Format, defined by the MultiCore Association
TRACE_EVENT(sched_process_fork,

    TP_PROTO(struct task_struct *parent, struct task_struct *child),

    TP_ARGS(parent, child),

    TP_STRUCT__entry(
        __array( char, parent_comm, TASK_COMM_LEN )
        __field( pid_t, parent_pid )
        __array( char, child_comm, TASK_COMM_LEN )
        __field( pid_t, child_pid )
    ),

    TP_fast_assign(
        memcpy(__entry->parent_comm, parent->comm, TASK_COMM_LEN);
        __entry->parent_pid = parent->pid;
        memcpy(__entry->child_comm, child->comm, TASK_COMM_LEN);
        __entry->child_pid = child->pid;
    ),

    TP_printk("comm=%s pid=%d child_comm=%s child_pid=%d",
        __entry->parent_comm, __entry->parent_pid,
        __entry->child_comm, __entry->child_pid)
);
Simple userspace tracepoint generation

TRACEPOINT_EVENT(  
sample_tracepoint,  
message,  
TP_ARGS(char *, text),  
TP_FIELDS(  
  ctf_string(message, text)  
)  
)
Generated by lttnng-gen-tp:

In C:

#include "sample_tracepoint.h"
tracepoint(sample_tracepoint, message, "Hello World\n");
Strace VS LTTng Tracing

Timing of a find of 100000 files (seconds)

- **find**: 0.54 seconds
- **find + lttng**: 1.4 seconds
- **find + strace**: 38.8 seconds
Top vs LTTngTop

- Top:
  - Total CPU time: 2.4
  - # syscalls: 24,1933

- LTTngTop:
  - Total CPU time: 0.9
  - # syscalls: 13,981
Userspace Tracing

Approx time by event – 1 thread (nanoseconds)

- LTTng UST: 280
- UST (W/O OPT): 500
- Dtrace: 2400
- SystemTap: 6000
Userspace Tracing

Approx time by event – 8 threads
(nanoseconds)

- UST: 280 nanoseconds
- Dtrace: 19400 nanoseconds
- SystemTap: 56000 nanoseconds
Looking at the trace data
Babeltrace

[13:58:29.128909723] (+0.000002475) sys_read: { 0 }, { "firefox-bin", 3363 }, { fd = 5, buf = count = 16 }
[13:58:29.128911513] (+0.000001790) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = -11 }
[13:58:29.128919672] (+0.0000008159) sys_write: { 0 }, { "firefox-bin", 3363 }, { fd = 5, buf count = 8 }
[13:58:29.128921404] (+0.0000001732) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = 8 }
[13:58:29.128922884] (+0.0000001480) sys_read: { 0 }, { "firefox-bin", 3363 }, { fd = 19, buf count = 1 }
[13:58:29.128925765] (+0.0000002881) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = 1 }
[13:58:29.128928120] (+0.0000002355) sys_write: { 0 }, { "firefox-bin", 3363 }, { fd = 5, buf count = 8 }
[13:58:29.128929552] (+0.0000001432) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = 8 }
[13:58:29.129020005] (+0.00000090453) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 1 }
[13:58:29.129025587] (+0.0000005582) sys_rt_sigprocmask: { 0 }, { "acpid", 1536 }, { how = 0, offset = 0x00, sigsetsize = 8 }
[13:58:29.129027993] (+0.0000002406) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 0 }
[13:58:29.129030188] (+0.0000002195) sys_poll: { 0 }, { "acpid", 1536 }, { ufds = 0x7FF2A055D mcount msecs = 0 }
[13:58:29.129032570] (+0.0000002382) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 0 }
[13:58:29.129033929] (+0.0000001359) sys_rt_sigprocmask: { 0 }, { "acpid", 1536 }, { how = 1, offset = 0x00, sigsetsize = 8 }
[13:58:29.129035144] (+0.0000001215) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 0 }
[13:58:29.129037520] (+0.0000002376) sys_read: { 0 }, { "acpid", 1536 }, { fd = 4, buf = 0x7FF = 24 }
Statistics for interval [1330053201794942051, 1330053202795131720]

CPUs 4 (max/cpu : 25.00%)
Processes N/A (0, 0)
Threads N/A (0, 0)
Files N/A (0, 0) N/A kbytes/sec
Network N/A (0, 0) N/A Mbytes/sec

CPU Top

<table>
<thead>
<tr>
<th>CPU(%)</th>
<th>TGID</th>
<th>PID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.00</td>
<td>23844</td>
<td>23844</td>
<td>gnome-shell</td>
</tr>
<tr>
<td>5.50</td>
<td>20627</td>
<td>20627</td>
<td>firefox-bin</td>
</tr>
<tr>
<td>0.93</td>
<td>23653</td>
<td>23653</td>
<td>Xorg</td>
</tr>
<tr>
<td>0.29</td>
<td>4788</td>
<td>4788</td>
<td>epiphany-browse</td>
</tr>
<tr>
<td>0.05</td>
<td>11223</td>
<td>11223</td>
<td>kworker/2:2</td>
</tr>
<tr>
<td>0.05</td>
<td>11173</td>
<td>11173</td>
<td>kworker/0:0</td>
</tr>
<tr>
<td>0.05</td>
<td>11222</td>
<td>11222</td>
<td>kworker/1:1</td>
</tr>
<tr>
<td>0.05</td>
<td>10843</td>
<td>10843</td>
<td>kworker/3:1</td>
</tr>
<tr>
<td>0.04</td>
<td>14809</td>
<td>14809</td>
<td>hald</td>
</tr>
<tr>
<td>0.04</td>
<td>24103</td>
<td>24103</td>
<td>xchat</td>
</tr>
<tr>
<td>0.02</td>
<td>31261</td>
<td>31261</td>
<td>synergyc</td>
</tr>
<tr>
<td>0.02</td>
<td>20247</td>
<td>20247</td>
<td>emacs</td>
</tr>
<tr>
<td>0.02</td>
<td>6251</td>
<td>6251</td>
<td>emacs</td>
</tr>
<tr>
<td>0.02</td>
<td>2403</td>
<td>2403</td>
<td>soffice.bin</td>
</tr>
<tr>
<td>0.01</td>
<td>25701</td>
<td>25701</td>
<td>emacs</td>
</tr>
<tr>
<td>0.01</td>
<td>2719</td>
<td>2719</td>
<td>nmbd</td>
</tr>
<tr>
<td>0.01</td>
<td>13085</td>
<td>13085</td>
<td>icedove-bin</td>
</tr>
<tr>
<td>0.01</td>
<td>1534</td>
<td>1534</td>
<td>dbus-daemon</td>
</tr>
<tr>
<td>0.00</td>
<td>11193</td>
<td>11193</td>
<td>kworker/u:1</td>
</tr>
<tr>
<td>0.00</td>
<td>10985</td>
<td>10985</td>
<td>kworker/u:2</td>
</tr>
<tr>
<td>0.00</td>
<td>577</td>
<td>577</td>
<td>ips-monitor</td>
</tr>
<tr>
<td>0.00</td>
<td>9750</td>
<td>9750</td>
<td>ksoftirqd/3</td>
</tr>
<tr>
<td>0.00</td>
<td>17301</td>
<td>17301</td>
<td>kworker/1:2</td>
</tr>
<tr>
<td>0.00</td>
<td>23813</td>
<td>23813</td>
<td>gnome-settings-</td>
</tr>
</tbody>
</table>

Status
Starting display
Pause

F2:CPUtop  F3:PerfTop  F6:IOTop  Enter:Details  F:Quit  P:Perf  P:Pause
```
me/pmf/tmp/trace-dd-hdc-hda/cpu_0), 0, 0, swapper, UNBRANDED, 0, 0x0, IRQ { major = 3, minor = 0, direction = 1 }
me/pmf/tmp/trace-dd-hdc-hda/cpu_0), 0, 0, swapper, UNBRANDED, 0, 0x0, IRQ { device = 3145728, sector = 80207, size = 4096, what = 16908288, error = 0 }
me/pmf/tmp/trace-dd-hdc-hda/cpu_0), 4187, 4187, /bin/dd, UNBRANDED, 4176, 0x0, SYSCALL { device = 23068672, sector = 0, size = 0, what = 16842752, error = 0 }
me/pmf/tmp/trace-dd-hdc-hda/cpu_0), 4187, 4187, /bin/dd, UNBRANDED, 4176, 0x0, SYSCALL { major = 22, minor = 0, direction = 0 }
```
Eclipse Linux Tools
lttng-graph
The Road Ahead

Live tracing
Per user global buffers
Remote tracing
Filtering (kernel)
uprobes
Other language binding
lttv
What about research?

- Dependency analysis
- Multi level event abstraction
- Time synchronisation
- Integration of hardware tracing
- Tracing the cloud!
Demo!
Contact

• Me :
  • http://yannickbrosseau.com
  • yannick.brosseau@gmail.com
  • @greenscientist

• LTTng project :
  • http://lttng.org
  • lttnng-dev@lists.lttng.org
  • irc.oftc.net #lttng
http://lttng.org/download
Image references

https://secure.flickr.com/photos/andreweadie/3746534415
https://secure.flickr.com/photos/-bast-/34949798
https://secure.flickr.com/photos/jwyg/3746351826
https://secure.flickr.com/photos/pedromourapinheiro/4270640863
https://secure.flickr.com/photos/puuikebeach/4781504955
https://secure.flickr.com/photos/b4b2/3954106061
https://secure.flickr.com/photos/edvvc/1972546648
https://secure.flickr.com/photos/anirudhkouli/2632880868
https://secure.flickr.com/photos/iita-media-library/6031540791
https://secure.flickr.com/photos/38471709@N02/5353281594
https://commons.wikimedia.org/wiki/File:Yvain-draggon.jpg
https://commons.wikimedia.org/wiki/File:Chevrolet_VOLT_under_the_hood_WAS_2011_1107.jpg
http://www.flickr.com/photos/15708236@N07/2754478731/
https://commons.wikimedia.org/wiki/File:Mount_Royal_Montreal_Lookout.jpg