LTTng and the love of development without `printf()`

dgoulet@efficios.com
David Goulet, Software Developer, EfficiOS,

Maintainer of LTTng-tools project

- https://git.lttng.org//lttng-tools.git
Content

- Quick overview of LTTng 2.x
- Everything else you need to know!
- Recent features & future work.
What is tracing?

• Recording runtime information without stopping the process
  – Enable/Disable event(s) at runtime

• Usually used during development to solve problems like performance, races, etc...

• Lots of possibilities on Linux: LTTng, Perf, ftrace, SystemTap, strace, ...
Overview of LTTng 2.x
Overview of LTTng 2.x

- Unified user interface, kernel and user space tracers combined. *(No need to recompile kernel)*

- Trace output in a unified format (CTF)
  - [https://git.efficios.com/ctf.git](https://git.efficios.com/ctf.git)

- Low overhead,

- Shipped in distros: Ubuntu, Debian, Suse, Fedora, Linaro, Wind River, etc.
Tracers

- lttnng-modules: kernel tracer module, compatible with kernels from 2.6.38* to 3.13.x,
- lttnng-ust: user-space tracer, in-process library.

* Kernel tracing is now possible on 2.6.32 to 2.6.37 by backport of 3 Linux Kernel patches.
• lttng-tools: cli utilities and daemons for trace control,
  - lttng: cli utility for tracing control,
  - lttng-ctl: tracing control API,
  - lttng-sessiond: tracing registry daemon,
  - lttng-consumerd: extract trace data,
  - lttng-relayd: network streaming daemon.
Viewers

- babeltrace: cli text viewer, trace converter, plugin system,
- ltntgtop: ncurses top-like viewer,
- Eclipse ltntg plugin: front-end for ltntg, collect, visualize and analyze traces, highly extensible.
Users instrument their applications with static tracepoints,

liblttng-ust, in-process library, dynamically linked with application,

Session setup, etc.,

Run app, collect traces,

Post analysis with viewers.
Tracing session - Setup

- **Session setup**: $ lttng create
- **User-space event enabling**: $ lttng enable-event -u -a
- **Start tracing**: $ lttng start
Tracing session - A wild app appears

Instrumented application

UST listener thread

- Listener thread spawned via constructor (GCC extension),
- App registration,
- Send SHM and wait fd.

sessiond

consumerd

UNIX Socket

SHM

Pipe
Time for the cool/useful stuff
Instrumentation of your app

TRACEPOINT_EVENT(
   /* Provider name */
   ust_tests_hello,

   /* Tracepoint name */
   tptest,

   /* Type, variable name */
   TP_ARGS(int, anint,
           long *, values,
           float, floatarg),

   /* Type, field name, expression */
   TP_FIELDS(ctf_integer(int, intfield, anint),
             ctf_array(long, arrfield1, values, 3),
             ctf_float(float, floatfield, floatarg))
)
void function(void)
{
    int i = 0;
    long vals[3] = { 0x42, 0xCC, 0xC001CAFE };
    float flt = M_PI;

    [...]?
    tracepoint(ust_tests_hello,
                tptest,
                i,
                &vals,
                flt);

    [...]?
}

Tracing session example

$ lttng create
$ lttng enable-event -u subsys1_*
$ lttng enable-event -u subsys42_*
$ lttng start
get(coffee);
$ lttng stop
$ lttng view
...

Human readable event (UST)

[13:52:13.523592640] (+0.100065120) thessa ust_tests_hello:tptest: 
{ cpu_id = 0 }, 

[13:52:13.623731676] (+0.100139036) thessa ust_tests_hello:tptest: 
{ cpu_id = 0 }, 

[13:52:13.723805959] (+0.100074283) thessa ust_tests_hello:tptest: 
{ cpu_id = 0 }, 
Human readable event (kernel)

[11:30:42.204505464] (+0.000026604) dalia
sys_read: { cpu_id = 3 }, { fd = 3, buf = 0x7FD06528E000, count = 4096 }
...

[11:30:42.204601549] (+0.000021061) dalia
sys_open: { cpu_id = 3 }, { filename = "/lib/x86_64-linux-gnu/libnss_compat.so.2", flags = 524288, mode = 54496 }
...

[11:30:42.205484608] (+0.000006973) dalia
sched_switch: { cpu_id = 1 }, { prev_comm = "swapper/1", prev_tid = 0, prev_prio = 20, prev_state = 0, next_comm = "rcuos/0", next_tid = 18, next_prio = 20 }
At any point in time, a snapshot can be taken of the current trace buffers. Overwrite mode meaning flight recorder

```
lttng_snapshot_record(..)
```

```
$ lttng snapshot record
```
$ lttng create --snapshot
$ lttng enable-event -a -u
$ lttng start
sell(dogecoin);
$ lttng snapshot record
Snapshot recorded successfully for session **auto-20140201-113803**

$ babeltrace /your/home/user/lttng-traces/\texttt{auto-20140201-113803}/snapshot-1-20140201-113813-0/ust/
Snapshot – Real world use case

Core dump
- Custom handler with lttng -> /proc/sys/kernel/core_pattern
- Snapshot record on coredump

IDS – Log Manager (ex: Splunk, Nagios)
- Trigger system snapshot on alert
- Gather system data regularly
- Correlate system events with logs

Performance profiling
- Server applications
- Kernel
- Hardware latency
As the trace is being created, you extract and can analyze the data.

🔄 **Continuous Analysis**
- Extract data with live streaming for analysis on an other machine

〓 **Cluster-level analysis**
- Gather traces from multiple machines
  - Load balancing analysis
  - Latency detection

🔧 **System Administration**
- Get data of faulty machine “on-demand”
Infrastructure integration

Server A (lttng-sessiond)

Server B (lttng-sessiond)

Server C (lttng-sessiond)

TCP

lttng-relayd

TCP

Viewer
Pretty awesome tool
Performance results

- The test runs for 50 minutes
- Each snapshot is around 7MB, 100 snapshots recorded (one every 30 sec.)
- The whole strace trace (text) is 5.4GB with 61 million events recorded
- The whole LTTng trace (binary CTF) is 6.8GB with 257 million events recorded with 1% of event lost.
Dedicated disk for trace

Number of database requests vs Number of threads

Dedicated disk for the DB

- No tracing
- Flight recorder
- Streaming
- Tracing to disk
- strace mysql
Shared disk with DB and trace

Number of database requests vs Number of threads

Writing the trace on the same disk as the DB

- No tracing
- LTTng syscall and sched_switch tracing
- strace mysql
Recent features & future work
Recent features

2.4 (Époque Opaque) – Upcoming

- Snapshot (local and remote), (2.3)
- Live tracing,
  - Analyze data while being created
- Java JUL support
  - Java Util Logging
Future work

- Hardware tracing support

- Trace trigger
  - Trigger custom actions

- Android port for kernel and UST tracers

- Automatic analysis for LTTng traces
Questions?

LTTng Project

🌐 https://{git | www}.lttng.org
✉️ lttnng-dev@lists.lttng.org
🐦 @lttng_project
👥 #lttng on irc.oftc.net