The LTTng Approaches to Solving Complex Problems in Production

jdesfossez@efficios.com
Content

- Trace buffering, aggregation and sampling.
- What is LTTng?
- Why LTTng compared to other tracing solutions?
- LTTng trace extraction modes with use-cases and examples:
  - Disk and streaming,
  - Live,
  - Snapshot,
  - Rotation.
- Conclusion.
Biography

- Julien Desfossez
  - Software Developer at EfficiOS,
  - Works on LTTng kernel and user-space tracers, Babeltrace,
  - Author and maintainer of the latency-tracker and LTTng-Analyses projects.
Trace Buffering

• Fast and efficient logging:
  – Generate events at specific locations in the code,
  – Extract parameters for later analysis,
  – Application-specific or system-wide.

• Common trace buffering solutions on Linux:
  – ftrace (kernel tracing),
  – perf in some modes,
  – LTTng (kernel and user-space tracing).
Trace Buffering Use-Cases

- Understanding complex problems that require low-level and a high volume of information (e.g: concurrency issues),
- Requires deep knowledge of the operating system or internal behavior of the application,
- Usually the “last line of defense” to fix a problem,
- With LTTng analyses tools, monitoring and cloud use-cases become possible.
Trace Aggregation

- Aggregation tools are used to perform run-time measurements or statistics based on tracing information.

- Common aggregation tools on Linux:
  - SystemTap,
  - eBPF/BCC,
  - latency-tracker.
Sampling or Profiling

- Periodically take a snapshot of the current activity of a system,
- Extract statistics and hot spots,
- Common profiling tools on Linux:
  - perf,
  - oprofile,
  - gprof.
LT-Tng Advantages

- **Fast kernel tracing** (same speed as ftrace but extracts the syscalls payload),
- **Fast user-space tracing** (does not rely on system calls at every event), native support for C/C++ applications, agents for Java and Python,
- Designed to run continuously in production environments,
- Multi-platform: x86, ARM, PPC, MIPS, s390, Tilera,
- Ability to merge kernel and user-space traces,
- Multi-host/clock support,
- Standard trace format (Common Trace Format),
- Packaged by the major distributions,
- Standalone kernel modules,
- Vast ecosystem of analysis and post-processing tools.
Tracing to disk with all kernel events enabled can quickly generate huge traces:

- 54,000 events/sec on an idle 4-cores laptop, 2.2 MB/sec
- 2,700,000 events/sec on a busy 8-cores server, 95 MB/sec

In addition to filtering and enabling specific events, LTTng offers various recording modes:

- Local disk and streaming mode,
- Live mode,
- Snapshot mode,
- Rotation mode (new in 2.11).
Disk and Streaming Modes

- Default mode,
- Write buffers to disk or the network when they are full,
- Only limited by disk space,
- Tracing session needs to be stopped to process the trace,
- Use-cases:
  - Understanding the complete life-cycle of a system or an application,
  - Trace exploration (need to identify what is relevant),
  - Post-mortem analyses,
  - Reverse engineering,
  - Continuous Integration.
Disk and Streaming Modes

$ lttng create # For streaming: -U net://<server>
$ lttng enable-event -k -a # All kernel events
$ lttng enable-event -u -a # All user-space events
$ lttng start
...
$ lttng stop
$ lttng view
$ lttng destroy
Sometimes users complain that the “website is slow”,

We do not see anything in the monitoring tools (averages, percentiles, etc),

Problem seems to happen periodically but we can only rely on users to report it,

Methodology:
  - Record all the I/O, scheduling and system calls activity on the webserver,
  - When a problem is reported, run statistics tools on the trace.

Full writeup on this case: https://lttng.org/blog/2015/02/04/web-request-latency-root-cause/
### Top system call latencies

```
ltng-ilatencytop /path/to/trace --limit=3 --minsize=2
```

#### Checking the trace for lost events...


##### Top open syscall latencies (usec)

<table>
<thead>
<tr>
<th>Begin</th>
<th>End</th>
<th>Name</th>
<th>Duration (usec)</th>
<th>Size</th>
<th>Proc</th>
<th>PID</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:18:59.432956815, 12:18:59.8078948568</td>
<td></td>
<td>open</td>
<td><strong>437697.753</strong></td>
<td>N/A</td>
<td>apache2</td>
<td>31517</td>
<td>/var/lib/php5/sess_0if1r2hangm8gg469g681</td>
</tr>
<tr>
<td>12:18:52.946606165, 12:18:52.946132276</td>
<td></td>
<td>open</td>
<td>52.113</td>
<td>N/A</td>
<td>apache2</td>
<td>31588</td>
<td>/var/lib/php5/sess_mr9045p1k55v1n1f</td>
</tr>
<tr>
<td>12:18:46.800846035, 12:18:46.800874916</td>
<td></td>
<td>open</td>
<td>28.881</td>
<td>N/A</td>
<td>apache2</td>
<td>31591</td>
<td>/var/lib/php5/sess_r7c12pccefvjitas1</td>
</tr>
</tbody>
</table>

##### Top read syscall latencies (usec)

<table>
<thead>
<tr>
<th>Begin</th>
<th>End</th>
<th>Name</th>
<th>Duration (usec)</th>
<th>Size</th>
<th>Proc</th>
<th>PID</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:18:37.256073107, 12:18:37.256555967</td>
<td></td>
<td>read</td>
<td>482.860</td>
<td>7.00 B</td>
<td>bash</td>
<td>10237</td>
<td>unknown (origin not found) (fd=3)</td>
</tr>
<tr>
<td>12:18:52.000269798, 12:18:52.0009252304</td>
<td></td>
<td>read</td>
<td>42.506</td>
<td>1.00 KB</td>
<td>irqbalance</td>
<td>1337</td>
<td>/proc/interrupts (fd=3)</td>
</tr>
<tr>
<td>12:18:37.25659439, 12:18:37.25681615</td>
<td></td>
<td>read</td>
<td>42.176</td>
<td>5.00 B</td>
<td>bash</td>
<td>10237</td>
<td>unknown (origin not found) (fd=3)</td>
</tr>
<tr>
<td>12:18:42.000281918, 12:18:42.000320016</td>
<td></td>
<td>read</td>
<td>38.998</td>
<td>1.00 KB</td>
<td>irqbalance</td>
<td>1337</td>
<td>/proc/interrupts (fd=3)</td>
</tr>
</tbody>
</table>

##### Top write syscall latencies (usec)

<table>
<thead>
<tr>
<th>Begin</th>
<th>End</th>
<th>Name</th>
<th>Duration (usec)</th>
<th>Size</th>
<th>Proc</th>
<th>PID</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:18:49.913241516, 12:18:49.915988862</td>
<td></td>
<td>write</td>
<td>2667.346</td>
<td>95.00 B</td>
<td>apache2</td>
<td>31584</td>
<td>/var/log/apache2/access.log (fd=8)</td>
</tr>
<tr>
<td>12:18:37.991578372, 12:18:37.991612724</td>
<td></td>
<td>writev</td>
<td>34.352</td>
<td>21.97 KB</td>
<td>apache2</td>
<td>31589</td>
<td>unknown (origin not found) (fd=12)</td>
</tr>
</tbody>
</table>

##### Top sync syscall latencies (usec)

<table>
<thead>
<tr>
<th>Begin</th>
<th>End</th>
<th>Name</th>
<th>Duration (usec)</th>
<th>Size</th>
<th>Proc</th>
<th>PID</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:18:50.162777639, 12:18:50.157522361</td>
<td></td>
<td>sync</td>
<td><strong>994745.622</strong></td>
<td>N/A</td>
<td>sync</td>
<td>22791</td>
<td>None (fd=None)</td>
</tr>
<tr>
<td>12:18:37.227667532, 12:18:37.232289687</td>
<td></td>
<td>sync_file_range</td>
<td>4422.155</td>
<td>N/A</td>
<td>lttnng-consumerd</td>
<td>19964</td>
<td>/home/julien/lttnng-traces/analysis</td>
</tr>
<tr>
<td>12:18:37.236076585, 12:18:37.239912027</td>
<td></td>
<td>sync_file_range</td>
<td>935.442</td>
<td>N/A</td>
<td>lttnng-consumerd</td>
<td>19964</td>
<td>/home/julien/lttnng-traces/analysis</td>
</tr>
<tr>
<td>12:18:37.226947111, 12:18:37.221847124</td>
<td></td>
<td>sync_file_range</td>
<td>672.413</td>
<td>N/A</td>
<td>lttnng-consumerd</td>
<td>19964</td>
<td>/home/julien/lttnng-traces/analysis</td>
</tr>
</tbody>
</table>
Live Mode

- Tracing sessions of arbitrary duration and size (same as streaming mode),
- Can attach to a running session and start processing the events while the session is still running,
- The trace is still written to disk but we can limit its size with the `tracefile-size` and `tracefile-count` options (on-disk ring buffer),
- Use-cases:
  - Low throughput logging with quick feedback,
  - Distributed or embedded systems,
  - Continuous monitoring (extracting metrics from events out-of-bound).
Live Mode

$ lttng create --live # optional: -U net://<server>

$ lttng enable-event -k -a
$ lttng enable-event -u -a

$ lttng start
$ lttng view

$ lttng stop
$ lttng destroy
$ lttnng create --live # optional: -U
net://<server>

$ lttnng enable-channel -k chan --tracefile-size 10M --tracefile-count 4

$ lttnng enable-event -k -a -c chan

$ lttnng start

$ lttnng view

$ lttnng stop

$ lttnng destroy
Snapshot Mode

- Memory-only tracing (ring-buffer),
- Low overhead while tracing (no I/O),
- On demand, “lttng snapshot record” extracts tracing buffers content from memory to disk or the network,
- Triggers to extract the snapshots can be errors detected by an application, high latencies measured, segmentation faults, time-based sampling, etc,
- The time span covered by a snapshot depends on the buffer size configuration, number of events enabled and the event rate.
Snapshot Mode

• Use-cases:

  – Fault investigation: get the full activity a few seconds before an error or high latency occurred,
  – Profiling: get a sense of the machine activity periodically,
  – When a Continuous Integration worker detects an error.
Snapshot Mode

$ lttng create --snapshot # optional: -U net://<server>
$ lttng enable-event -k -a
$ lttng enable-event -u -a
$ lttng start
...
$ lttng snapshot record
...
$ lttng snapshot record
...
$ lttng snapshot record
Snapshot Mode - Example

- We sometimes measure high response times with an aggregation tool (latency-tracker),
- We want to know what is happening around the time the latencies are detected,
- Methodology:
  - Start a snapshot session with scheduling, I/O, and system calls events,
  - Every time a high latency is detected, record a snapshot,
  - Send the snapshot to an automated post-processing tool that generates activity reports,
  - Plot all the response times in Grafana and link the spikes to the snapshot analyses.
### I/O request latency distribution

```bash
lttng-iolatencyfreq /path/to/trace
```

**Timerange:** [2015-01-06 10:58:26.140545481, 2015-01-06 10:58:27.229358936]

Open latency distribution (usec)

<table>
<thead>
<tr>
<th>Latency (usec)</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.562</td>
<td>25</td>
</tr>
<tr>
<td>9.168</td>
<td>4</td>
</tr>
<tr>
<td>12.774</td>
<td>8</td>
</tr>
<tr>
<td>16.380</td>
<td>3</td>
</tr>
<tr>
<td>19.986</td>
<td>2</td>
</tr>
<tr>
<td>23.592</td>
<td>0</td>
</tr>
<tr>
<td>27.198</td>
<td>0</td>
</tr>
<tr>
<td>30.894</td>
<td>0</td>
</tr>
<tr>
<td>34.416</td>
<td>1</td>
</tr>
<tr>
<td>38.016</td>
<td>0</td>
</tr>
<tr>
<td>41.623</td>
<td>0</td>
</tr>
<tr>
<td>45.229</td>
<td>0</td>
</tr>
<tr>
<td>48.835</td>
<td>0</td>
</tr>
<tr>
<td>52.441</td>
<td>0</td>
</tr>
<tr>
<td>56.047</td>
<td>0</td>
</tr>
<tr>
<td>59.653</td>
<td>0</td>
</tr>
<tr>
<td>63.259</td>
<td>0</td>
</tr>
<tr>
<td>66.865</td>
<td>0</td>
</tr>
<tr>
<td>70.471</td>
<td>0</td>
</tr>
<tr>
<td>74.077</td>
<td>2</td>
</tr>
</tbody>
</table>
Rotation Mode

- New in LTTng 2.11 (expected to be released in March 2018),
- Archive a tracing session’s current chunk,
- Allows to process/archive/delete/compress a chunk of a trace while it is still writing in a separate directory,
- The trace can run indefinitely but the chunks can be processed like offline traces (disk or streaming mode),
- Timer-based or size-based auto-rotation available.
Rotation Mode

• Use-cases:
  – Continuous monitoring: periodically rotate and extract/plot low-level metrics from the trace,
  – Smaller traces to process than with the default mode,
  – Spreading the post-processing load (send chunks for analysis to available worker servers),
  – Archiving/Compression.
## Rotation Mode

$ lttng create # optional: -U net://<server>

$ lttng enable-event -k -a

$ lttng enable-event -u -a

$ lttng start

...

$ lttng rotate

Output files of session auto-20180125-155317 rotated to
/home/julien/lttng-traces/auto-20180125-155317/20180125T155319-0500-20180125T155320-0500-1

$ lttng rotate

...

$ lttng rotate
Conclusion

- LTTng allows to extract low-level, high volume tracing information in production environments,
- Efficient kernel and user-space combined tracing,
- Used for monitoring and fault investigation in at least cloud, telecommunication and automotive environments,
- There are five main ways to extract LTTng traces, flexibility based on the use-case,
- Not just a tracer to use when all else has failed.
Questions?

EfficiOS

www.efficios.com

lttng.org

lttng-dev@lists.lttng.org

@lttng_project

OFTC / #lttng