LTTng and Related Projects Updates

EfficiOS

Outline

- Babeltrace 2.1’s ongoing development
- CTF 2.0
- LTTng 2.13 and 2.14’s ongoing development
- Restartable Sequences
- LTTng-modules upstreaming

Babeltrace 2.1 — Ongoing Development

Development work on Babeltrace 2.1 is ongoing:

- Optional pretty-printing of bitmask enum values,

  ```
  ex: 1053696 vs ‘O_APPEND | O_SYNC’
  ```

- Changes required to support CTF 2.0.

Started the development of an experimental ftrace source component:

- No precise timeline, may live out of tree until it is production-ready.

CTF 2.0

The LTTng 2.x ecosystem has been using the CTF 1.8 format for ten years (2010).

Limitations of the CTF 1.8 format has stilted the development of some functionality in viewers:

- No extension mechanism,
- No way to define the semantics of events / fields,
- The format is hard to consume (mostly the TSDL metadata),
- Some areas of the specification were lacking in precision (mostly time handling).

EfficiOS

CTF 2.0

EfficiOS is leading the effort to address those shortcomings with a new version of the specification:

- Gathered feedback following the publication of the first draft,
- A new draft of the CTF 2.0 specification is available: https://diamon.org/ctf/files/CTF2-PROP-2.0.html

The ease of adoption of the new tracing format is a key concern. So far, our transition plan is in three phases:

1. Babeltrace 2 will support reading both CTF 1.8 and CTF 2.0,
2. LTTng will produce both CTF 1.8 and CTF 2.0,
3. CTF 1.8 is eventually phased-out and becomes unsupported by LTTng.

LTTng 2.13 – Ongoing Development

Expanding the *trigger* mechanism is the major focus of this release

- Allows users to specify actions to be taken when a specific condition is met.

Triggers were introduced in LTTng 2.10 to:

- Notify external applications when tracing buffer usage reached a given threshold,
- Allows a controller to disable certain events of lesser importance.

LTTng 2.13 – Ongoing Development

New actions and conditions indicated in bold.

Supported conditions:

- Buffer usage threshold
- Session consumed size
- Rotation ongoing or completed
- **On event**

Supported actions:

- Notify
- **Record snapshot**
- **Rotate session**
- **Start session**
- **Stop session**
- **Group of actions**

LTTng 2.13 – Ongoing Development

Allow the capture and transmission of specific event payload and context fields along with a notification.

- Allows external applications to use the context of an event to take a decision (such as taking a snapshot on another machine).

```
$ lttng add-trigger --condition on-event --userspace sample_component:*
                    --capture field1
                    --capture field2[4]
                    --capture '$ctx.vtid'
                    --action notify
```

EfficiOS

LTTng 2.13 – Ongoing Development

Removed the dependency on `liburcu` from the LTTng user space tracer:

- Avoid compatibility issues between applications and LTTng-UST which may be linked against different `liburcu` versions.

User space RCU is not stable yet (0.x release), major bumps will most likely still occur

- Linking two versions of `liburcu` in against an application results in a number of conflicts/symbol clashes,
- We can’t expect applications which use `liburcu` to be updated in lock-step with the tracing infrastructure to use the same library version.

The subset of liburcu needed by LTTng UST is integrated in the project directly.

LTTng 2.14 – Ongoing Development

Trace Hit Counters:

- Per-CPU array of counters
- New back-end (counting instead of serializing events)
- Non-blocking

Use cases:

- Count the number of event rule hits, without tracing to a ring buffer.
- Estimate the impact of a given tracing configuration on a live production workload.
- Collect statistics.

LTTng 2.14 – Ongoing Development

Implemented as part of `libcounter`

- Per-CPU split-counters implementation:
  - In kernel memory (LTTng kernel tracer),
  - In shared memory (LTTng user space tracer).

In summary:

- Flexible mapping between events and keys (named "slots" in a trace hit counter array), dynamically configurable.
- Mapping between keys and counter indexes done on a slow-path (when instrumentation is registered).
- Fast-path (in application/kernel) only needs to increment a per-cpu counter indexed within an array.
- Sum per-CPU counters for a given key when viewed.

libcounter

Multi-dimension (up to 4) array of counters

- Only one dimension exposed initially, but additional dimensions will be useful for future use-cases
  - e.g. Aggregation based on field value

Counters are configurable:

- Array length for each dimension,
- Per-CPU/Global,
- Modulo or saturation arithmetic,
- Size of each counter: 8, 16, 32, or 64-bit (limited by architecture atomic operation size).

LTTng will initially only expose the needed subset of the libcounter features to end users.

Restartable Sequences

Overview

- Restartable Sequence (`rseq`) is a Linux system call implemented by EfficiOS,
- Allow fast per-CPU operations in user space,
- End goal is to eliminate atomic operations from the user space tracer's fast-path,
- Useful for other use-cases (e.g. memory allocators),
- Merged in Linux 4.18.

Recent news:

- Google added support in `membarrier` system call for a "rseq fence" to abort pre-existing rseq critication sections (Linux 5.10, see `MEMBARRIER_CMD_PRIVATE_EXPEDITED_RSEQ`).
- Google tcmalloc started using `rseq` system call, allowing them to use per-cpu memory pools rather than per-thread.

Restartable Sequences

Ongoing:

- Integration of rseq into the GNU C Library. Almost merged for glibc 2.32, but will require exposing a new *extensible* Linux `rseq` ABI before we can agree on an implementation.

Still working on the missing pieces for LTTng-UST integration:

- Missing Linux kernel feature allowing modification of per-cpu user space data from remote CPUs safely against concurrent CPU-hotplug and cgroup `cpuset` configuration changes, without hurting partitioned latency-sensitive workloads.

LTTng-modules Upstreaming

Multiple attempts have been made to get the project merged in upstream Linux.

According to Linux tracing maintainer Steven Rostedt:

- Proposed changes must be presented in a way which shows their possible usefulness to other tracers,
- Identified LTTng system call tracing (with arguments) as an initial feature which would be a differentiator justifying the LTTng ABI,
- EfficiOS started the work to allow page faults in system call entry/exit tracepoints.

Initial lttng-modules for upstream would likely be a “redux” version of the kernel tracer featuring initially only system call entry/exit tracing.

Questions?

- lttng.org
- lttng-dev@lists.lttng.org
- @lttng_project
- `#lttng OFTC`

EfficiOS