LTTng and Related Projects Update

DORSAL Progress Meeting

December 2023

EfficiOS

Outline

- Ongoing collaborations
- LTTng
- Babeltrace
- Common Trace Format 2 (CTF 2)
- Restartable Sequences: Concurrency IDs and extensible RSEQ GNU libc integration
- Libside
- Userspace RCU library
- Exa-Tracer: ROCm and MPI LTTng-UST instrumentation
- Tracing Summit

Ongoing Collaborations

- Ericsson,
- Ciena,
- AMD and Lawrence Livermore National Laboratory,
- Argonne National Laboratory,
- Internet Systems Consortium (ISC).

LTTng 2.14

- Release planned for Q1 2024
- Main feature:
  - Aggregation Maps (Trace Hit Counters)

Recording vs. aggregation: level of details

- Aggregation: simply count occurrences of event rule matches

```
+-----------------------------------------+-------------+----+----+
| key                                     |         val | uf | of |
+-----------------------------------------+-------------+----+----+
| syscall_entry_recvmsg                   |   3,404,391 |  0 |  0 |
+-----------------------------------------+-------------+----+----+
| kmem_kfree                              |     611,014 |  0 |  0 |
+-----------------------------------------+-------------+----+----+
```

Maps are presented like a regular back-end

- Create a user space map named **`my_map`** with session **`my_session`**

```
$ lttng add-map --userspace --session=my_session
                --bitness=64 --max-key-count=1024
                my_map
```

Performance of aggregation maps

- As expected, they are a lot cheaper to use than ring-buffer tracing

Method	**Time per event (ns)**	**σ (stdev)**
LTTng-UST ring-buffer (4 × 8 MiB)	158	0.222
LTTng-UST map	43.3	0.656
LTTng-modules ring-buffer (4 × 8 MiB)	151	0.824
LTTng-modules maps	44.8	0.219
eBPF per-CPU array	57.0	0.683

Benchmark code available, see reference slide

Future work for aggregation maps

- Native histogram support
- Decrement value
- Use event payload in the `incr-value` action
- Use event size in the `incr-value` action (dry run mode)

Babeltrace

- Babeltrace 2.0: Optimize filter.utils.muxer to use a priority heap instead of an array.
  - Measured acceleration of up to 12x with 200+ streams; our customer measures up to 6x (different hardware and use case).
  - Slightly faster with 4 streams.
  - Negligibly less efficient with one or two streams (could be fixed with special handling).
- Upcoming in Babeltrace 2.1 (Q1 2024): Common Trace Format (CTF) 2 support in all our component classes.

Common Trace Format 2.0

- CTF2-SPECRC-8.1rA was released on August 28, 2023.
  - Add the accuracy concept to a clock class.
  - While the precision property of a clock class already describes the random errors of an instance, the new accuracy property describes its systematic errors.
  - This feature makes it possible, for example, to sort more accurately packets and event records having timestamps from different clocks sharing the same origin.
  - To improve consistency: make the trace, data stream, event record, and clock class fragments have optional namespace, name, and UID properties to help identify them.

Common Trace Format 2.0

- CTF2-SPECRC-9.0 to be released before end of 2023.
  - New bit order property for all fixed-length bit array field classes: encode/decode first to last or last to first, whatever the endianness.
  - New fixed-length bit map field class: fixed-length bit array field class with flags (named groups of bits, overlaps allowed).
  - No more enumeration field classes: any integer field class may have a mapping property (equivalent).
  - Minor terminology changes.
  - UTF-16 BE/LE and UTF-32 BE/LE encodings for null-terminated, static-length, and dynamic-length string fields.
- Planned release in Babeltrace (2.1) and LTTng (2.15)
  - Allows us to validate the specification (produce and consume)

Restartable Sequences (RSEQ) ABI extensions

- Per memory-map concurrency id (`mm_cid`) (merged in Linux 6.3)
  - Ideal scaling of user space per-cpu data structures
  - Concurrency id is bounded by the number of concurrently running threads for a given memory map at any given time.
- Integration of extensible RSEQ with GNU libc (submitted)
- Per memory-map NUMA cid (`mm_numa_cid`) (work in progress)
  - Maintain NUMA-locality of per-cpu data structures
- Per-namespace (shared memory) concurrency id (future work)

libside: **S**oftware **I**nstrumentation **D**ynamically **E**nabled

- New instrumentation ABI
  - Tracer-agnostic application instrumentation framework
  - Usable from the purely user space tracers and from the kernel
  - The ABI can be used to instrument various runtimes
  - The C instrumentation API can be used to instrument C/C++
- Declare events statically or dynamically without code generation
  - Reduced code footprint (less impact on the instruction cache)
  - More flexible type system (variants, nested types, dynamic compound types)
- Spurred by the upstreaming of *User events* (Microsoft) into the Linux kernel
- Remaining work:
  - Finalize ABI
  - Integration with LTTng-UST

Userspace RCU library

- Now used by the BIND name server,
- Requirement that Userspace RCU QSBR and the liburcu-cds data structures support ThreadSanitizer (TSAN),
  - Moving liburcu memory model to C11 atomics,
  - Deprecating liburcu-signal
  - Add annotation infrastructure to validate multiple stores/loads associated with a single release/acquire barrier:
    - Acquire group,
    - Release group.
  - To be released in upcoming liburcu 0.15,
- There is interest in Userspace RCU for the C++26 standard and GNU libc
  - Multi-domain URCU prototypes could be a good fit

ROCm Tools: Exa-Tracer

- Work ongoing to add LTTng-UST integration to ROCm Tools
- Instrumentation of ROC APIs with LTTng-UST:
  - HIP
  - HSA
  - ROCtx
  - ROC Profiler
- Instrumentation of OpenMPI and CrayMPI with LTTng-UST

Roadmap

- LTTng 2.14: Q1 2024
- Babeltrace 2.1: Q1 2024
- LTTng 2.15: Q2 2024
- libside: Unknown, still evolving rapidly
- Userspace RCU 0.15: Q1 2024

Video recording of presentations at **tracingsummit.org**

References

- Aggregation maps benchmark repository

  https://github.com/jgalar/LinuxCon2022-Benchmarks

- CTF 2 Release Candidate 8.1rA

  https://diamon.org/ctf/files/CTF2-SPECRC-8.1rA.html

- libside repository

  https://github.com/efficios/libside