LTTng and Related Projects Update

DORSAL Progress Report Meeting

July 2026

EfficiOS

Who is EfficiOS?

Our work
these days

What is LTTng?
What is Babeltrace?

What is LTTng?

Extremely low overhead troubleshooting tool

- First released in 2005
- Open Source

A collection of projects

- Kernel tracer (LTTng-modules)
- User space tracer (LTTng-UST)
- Tracing control tools (LTTng-tools)

What is Babeltrace 2?

Trace manipulation toolkit

- Allows decoding and transformation of trace files
- Extensible via a plug-in architecture
  - Support other trace formats
  - Implement analyses

Update Outline

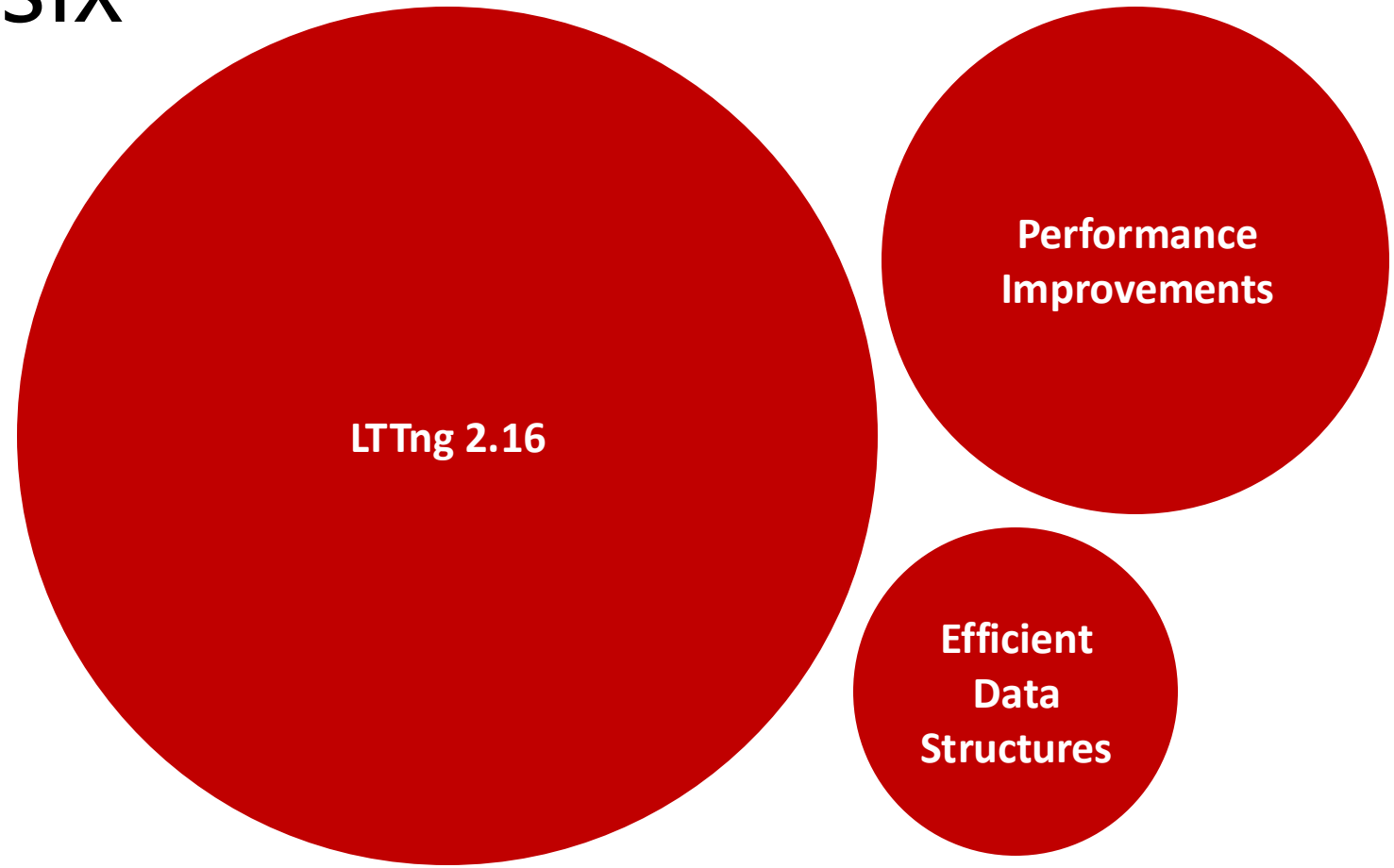
Research & Development

- Last 6 months
- Upcoming

In the

Last six months

R&D Activities in the last six months



LTTng 2.16
Landing this summer!

LTTng 2.16

Adds:

- Map channels
- BLOB support
- Mitigation of a performance regression

Lightweight insight

Feature: Map channels

- Count number of times an event (or event set) occurs
- Super lightweight – Doesn't require tracing buffers
- Powerful building block for responsive trace control

Lightweight insight

```
$ lttng add-map-channel OPTIONS
$ lttng add-trigger OPTIONS --action=incr-map-value

$ lttng start
$ lttng show-maps
```

LTTng 2.16

Try the release candidate

- Release candidate information
- Count events with a map channel
- Download page

Final 2.16 release planned for early August.

Experimental research

Efficient data structures

Fractal Trie

- Cache-dense, byte indexed ordered Trie.
- Arbitrary-byte keys, fixed or variable length.
- Automatic path compression; adaptive node layout.
- Wait-free RCU lookups (O(key length) complexity):
  - at most 2 cache lines per key byte
  - Exact, prefix, longest-match, ordered iteration, range queries (<=, >=, <, >), duplicate chains.
- Bulk mutation operations (graft, graft-swap, detach, merge-at)
  - Populate staging trie offline.
  - Graft it into live trie in O(1), minimal writer contention.
  - Readers observe pre or post graft, never partial state.
- Online, resumable DFS-order compaction (RCU GC copy).
- Main use-case: DNS resolver and authoritative servers.

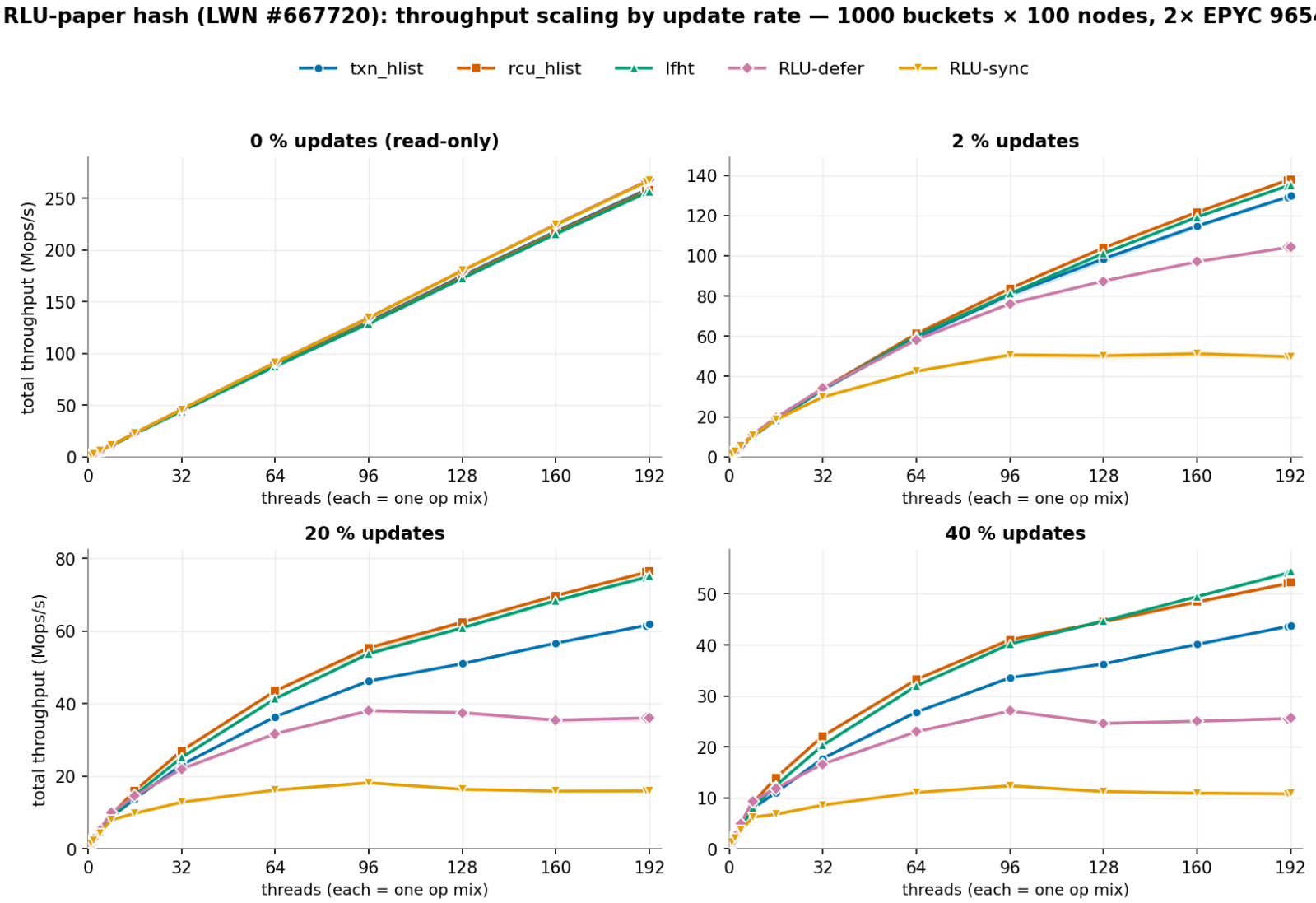
RCU Pseudo-Transactions

- Solution to Paul E. McKenney's Issaquah Challenge (2016)
- Answers to a key RCU data structure design problem:
  - How to publish a set of pointers atomically ?
- State of the Art:
  - Read-Log Update (RLU)
  - Paul E. Mckenney's "Existence structures"
  - Software Transactional Memory (STM)

RCU Pseudo-Transactions: Multi-Writer

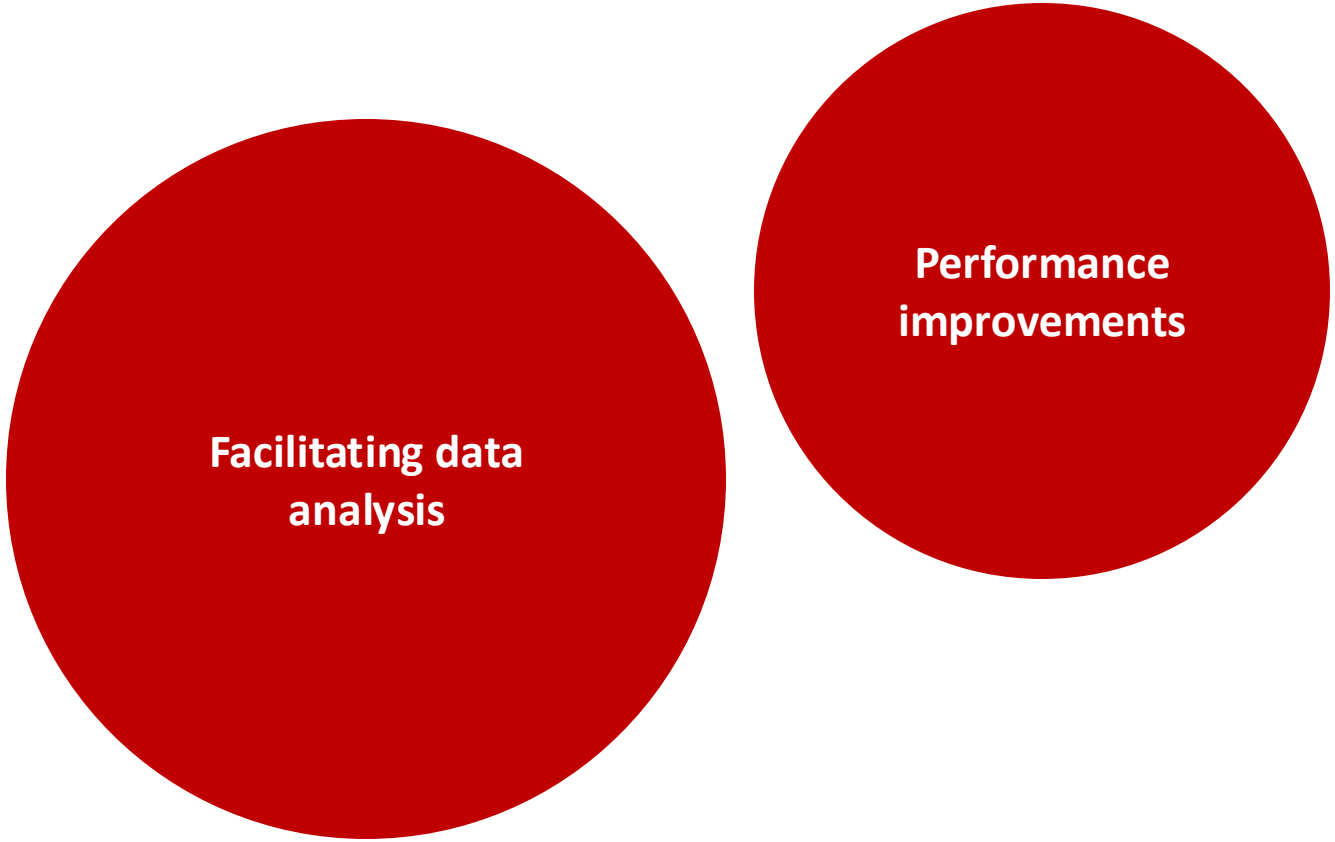
- Enable updates to proceed concurrently.
- Updates synchronize only through the pointers being modified through a protocol which state is in a transient "transaction" object.
- Remove the need for sequence locks, mutual exclusion.
- Bounded-blocking updates, with mutual help/abort conflict resolution.

RCU Transactions: Benchmarks



Upcoming* Work

Upcoming* Work



Towards facilitating data analysis

Moving towards CTF 2

Add reading and producing CTF 2 traces

- Babeltrace 2.1 (Q1 2025)

Add producing CTF 2 traces

- LTTng 2.15 (Q1 2026)

Benefits of CTF 2

- Better trace readability
- Broader type support
- Cleaner trace streaming
- Easier decoding

Benefits of CTF 2

- Better trace readability
- Broader type support
- **Cleaner trace streaming (2.15)**
- **Easier decoding (2.15)**

Benefits of CTF 2

- **Better trace readability (TBD)**
- **Broader type support (TBD)**
- Cleaner trace streaming
- Easier decoding

Moving towards CTF 2

Add reading and producing CTF 2 traces

- Babeltrace 2.1 (Q1 2025)

Add producing CTF 2 traces

- LTTng 2.15 (Q1 2026)

Supporting new CTF 2 annotations and types

- TBD

Performance improvements

Reducing memory usage

- Per-container max concurrency limits.
  - Allow expressing number of CPUs which can be used by a container with a numeric upper bound rather than topology-aware CPU mask.
- Per-container concurrency IDs allocator.
  - Extension of RSEQ per-process concurrency allocator (Linux scheduler) to containers.
  - Allow using concurrency ID to index shared memory content within containers.

Reducing latency

- RSEQ time slice extension (Linux v7.0)
- Allow userspace to ask the kernel not to preempt userspace for a short time (default 5µs, max. 50µs)
- Useful to reduce tail latency induced by scheduler preemption at key points in lock-free algorithms.
  - LTTng-UST ring buffer (from reserve to commit),
  - Userspace spinlocks.
  - Queues, stacks with wait-free enqueue/push.

Wrap up!

LTTng 2.16

Try the release candidate

- Release candidate information
- Count events with a map channel
- Download page

Final 2.16 release planned for early August.

Questions ?

- Links:
  - https://www.efficios.com
  - https://lttng.org
  - https://babeltrace.org
  - https://diamon.org
  - https://barectf.org

Contacts

Mathieu Desnoyers – mathieu.desnoyers@efficios.com

Jérémie Galarneau – jgalar@efficios.com

Erica Bugden – ebugden@efficios.com

Annex

Resources

- Common Trace Format 2 Specification

  https://diamon.org/ctf

- libside repository

  https://github.com/efficios/libside

Field classes common to CTF 1 and CTF 2

Field class	CTF 1.8	CTF 2
Fixed-length integer	✓	✓
UTF-8 string	✓	✓
Floating point number	✓	✓
Fixed-length array	✓	✓
Dynamic-length array	✓	✓
Structure	✓	✓
Variant	✓	✓

What does CTF 2 do better than CTF 1?

	CTF 1.8	CTF 2
Metadata format	TSDL (custom DSL) • Non-trivial to parse.	JSON text sequences • Widely used standard format with pre-existing parser libraries in various languages.
Augment events and fields with user-defined metadata	✗	✓ • Associate user-defined name to a value. • Used to tailor analysis or pretty printing of trace data.

What does CTF 2 do better than CTF 1?

Field class	CTF 1.8	CTF 2
BLOB	✗	✓ • Record opaque binary blobs • IANA media type attribute
Optional	✗	✓
LEB128 variable length integer	✗	✓ • Values > 64-bit range • Common need in scientific computing
UTF-16 and UTF-32 string character encoding	✗	✓ • Native string encoding on some platforms • E.g. Windows, Java VM
Fixed-length bit map	✗	✓ • Associate names to specific bits in a bitmap • Useful to represent flags
Boolean	✗	✓

Linux Kernel & Community Work

Laying the foundation to...

Reduce userspace tracer CPU and memory overhead

- Reduce CPU execution constraints by replacing hardware atomic instructions with kernel-managed software transactions
  - Restartable sequences (RSEQ) system call and GNU C library integration
- Bound memory allocation to max number of concurrently running threads (rather than allocate for each CPU)
  - RSEQ concurrency IDs (mm_cid)
- Reduce CPU data cache & branch prediction buffer impact of static instrumentation
  - Concurrent code patching (XMC), page deduplication
  - Also useful for code specialization at runtime

Linux Kernel & Community Work

Laying the foundation to...

Enable kernel tracer to have previously unavailable data

- Handle page faults while tracing system calls
  - Faultable system call tracepoints

Expand integration of LTTng-UST with the open source ecosystem

- Instrumentation coverage of runtimes, libraries, applications
- Tracer-agnostic "SIDE" instrumentation specification
- libside reference implementation for C/C++
- ABI targets instrumentation of various runtimes natively

SIDE ABI RFC (libside)

- The SIDE ABI is currently at RFC stage, aiming to create a specification.
  - https://github.com/efficios/libside/blob/master/doc/rfc-side-abi.txt
- Runtime/language agnostic,
- Supports multiple concurrent tracers,
- Instrumentation is not specific to a tracer,
  - No need to rebuild applications if using a different tracer,
- Instrumentation can be either static or dynamic,
- Supports complex/nested types,
- Supports both static and dynamic types,
- libside is a C/C++ reference implementation for the System V ELF ABI.