LTTng & Tools Roadmap

Content

- LTTng
  - new and upcoming features,
  - release schedule,
- Babeltrace 2,
- Restartable Sequences,
- Membarrier system call

New Features LTTng 2.10 (KeKriek) 2017-08-01

- Multi-wildcard support for event name and filtering strings:
  - lttng enable-event -u ‘myapp_abc*def*’ --filter ‘field == “a*b*c”’
- New trigger and notification API:
  - Buffer usage conditions,
- LTTng-UST blocking mode:
  - http://lttng.org/blog/2017/11/22/lttng-ust-blocking-mode/
- LTTng-modules for Linux 4.10, 4.11, 4.12, 4.13, 4.14,
- Extended socketpair(2) syscall tracing data,
- Embedded man pages configuration option.

Upcoming Features for LTTng 2.11

- Trace bandwidth monitoring:
  - Query tracing throughput per-channel and per-session,
- Filter on array/sequence index, with bitwise ops (&, |, ^, ~, <<, >>):
  - Filter on network protocol headers,
  - lttng enable-event -u myevent –filter ‘(field[2] & 0xF) == 0x3’

Upcoming Features for LTTng 2.11

- Session rotation:
  - Similar to log rotation, for trace output,
  - Split trace data output time-wise,
  - Provide notifications when rotations are completed,
  - Enables chunk-wise:
    - Trace data transport, integration with external message-passing infrastructures, compression, encryption, ...
  - Opens the door to pipelining of analyses, and sharding for map-reduce style of distributed analysis.

Upcoming Features for LTTng 2.11

- Uprobes instrumentation from kernel tracer
  - User-space function entry,
  - SDT (without semaphore),
- User-space and kernel stack dump from kernel tracer (if sufficient testing).

LTTng Release Schedule

- LTTng 2.11
  - RC1 in January 2018
  - Final release in February 2018

Container Filtering PoC

- Filter all syscalls from a docker container:

```
# Get the pid of the docker container init process
$ pid=$(docker inspect --format '{{.State.Pid}}' my-container)

# Get the pid namespace id from this pid
$ pid_ns=$(lsns -n -t pid -o NS -p ${pid})

# Create a session and add the required contexts
$ lttng create my-container
$ lttng add-context -k -t procname -t pid -t vpid -t tid -t vtid -t pid_ns

# Enable all syscalls, filter by pid namespace for my-container
$ lttng enable-event -k --syscall --all --filter="\$ctx.pid_ns == ${pid_ns}"
```

EfficiOS

Containers Analyses PoC

- Combine all this information to run kernel-level analysis per-container,

Per-TID Usage

	Process	Migrations	Priorities
38.80 %	ab (23205)	0	[20]
14.52 %	lttng-consumerd (11032)	0	[20]
7.20 %	apache2 (23033)	0	[20]
7.19 %	apache2 (23007)	0	[20]
3.90 %	gcc (23297)	0	[20]
2.58 %	gcc (23294)	0	[20]
1.62 %	lxd (23364)	0	[20]
1.59 %	gcc (23300)	0	[20]
1.54 %	lxd (23363)	0	[20]
1.51 %	lxd (23362)	0	[20]

Per-CPU Usage

81.80 % CPU 0
77.57 % CPU 1

Per-Container Usage

	Container	Type
40.16 %	[HOST] (4026531836)	host
0.14 %	goofy_haibt (4026532295)	docker
119.15 %	ample-adder (4026532356)	lxd
0.01 %	coherent-macaque (4026532424)	lxd
0.02 %	master-cheetah (4026532491)	lxd
0.12 %	thirsty_meninsky (4026532572)	docker
0.11 %	some-redis (4026532637)	docker

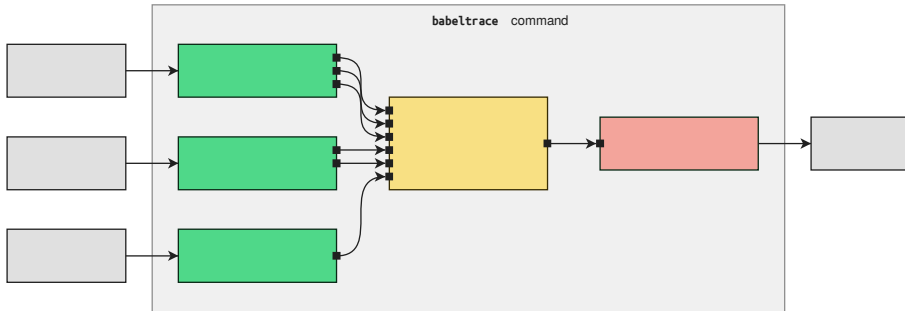
EfficiOS

Babeltrace 2

- The babeltrace client becomes a “host” application for trace processing graphs
- Cross-platform
  - Linux
  - Windows (native and Cygwin)
  - Solaris
  - BSDs
  - macOS

Babeltrace 2

- Provides components which allow everything Babeltrace 1.x could do
  - CTF file system source, sink,
  - LTTng-live source
  - dmesg source
  - Muxer, trimmer
  - Debugging information injecter



- Components can be written in C, C++, and Python
- Stable ABI allows out-of-tree components

Babeltrace Roadmap

- Babeltrace 2
  - Currently at v2.0.0-pre4, released in September 2017,
  - Feature-complete,
  - Works on all supported platforms,
  - Current focus on optimizations which may affect APIs,
  - Ongoing work on Documentation,
  - Targetting the first Release Candidate for December 2017 (API freeze).

Babeltrace 2 Roadmap

- Babeltrace 2.1
  - Support for CTF 2,
  - Filtering component
    - Filter by event name, context and event payload field content,
  - Multi-clock support (need use-cases):
    - Reference clock choice, how to define priority between clocks: automatically or through user interaction,
  - State tracker, Period/span tracking, ideas?

Restartable Sequences

- Implemented a system call handling debugger single-stepping failure due to restartable sequences:
  - Pinning of accessed user-space pages upon entry into the system call,
  - Execution of operation vector with preemption disabled within the kernel.
- Presented at Kernel Summit 2017,
- Ongoing work on the approach, receiving lots of community feedback,
- Linus Torvalds interested in merging this approach.

Restartable Sequences Use-Cases

- Per-CPU statistics counters in user-space,
- Memory allocator per-CPU memory pools (glibc, jemalloc),
- Per-CPU RCU grace-period tracking in user-space:
  - Single and multi-process,
- Per-CPU ring buffers in user-space:
  - Speed up LTTng-UST reserve and commit operations,
- Reliable use of PMU counters vs migration:
  - E.g. allow reading PMU counters from user-space on ARM 64 big.LITTLE without triggering a trap.

membarrier(2) system call

- Speed up liburcu and lttng-ust fast-path by removing memory barriers, issuing heavier synchronization on tracing configuration update,
- Removed use of SHARED membarrier command by liburcu, which caused significant process startup delays (10-20ms per call, quicky sums up to seconds when enabling many lttng-ust events),
- Contributed PRIVATE_EXPEDITED membarrier to upstream Linux (4.14), completes in few µs (single-process, e.g. liburcu),
- Proposed SHARED_EXPEDITED membarrier for upstream Linux (multi-process shared memory, e.g. lttng-ust ring buffer).