Polytechnique Montréal – December 2018

What's new at EfficiOS ?

EfficiOS

mathieu.desnoyers@efficios.com

Outline

- Linux kernel and glibc contributions,
- LTTng 2.11,
- Babeltrace 2,
- LTTng Scope.

Linux Kernel and glibc Contributions

- Memory Barriers (membarrier(2)):
  - glibc: work in progress to integrate membarrier(3) libc library function,
- Restartable sequence (rseq):
  - System call ***merged*** into Linux 4.18,
  - glibc: work in progress to automatically register rseq TLS for each thread,
  - glibc: use rseq to speed up sched_getcpu(3).
  - librseq,
- CPU operation vectors (cpu_opv):
  - Linus requests examples of rseq real-life users before merging additional code.

LTTng 2.11

- Currently at 2.11-rc1 (release candidate 1),
- Working on stress-testing/fixing session rotation corner-cases and documentation before final version,
- Expect 2.11 final end of December,
- New features:
  - Session rotation,
  - Dynamic instrumentation with uprobes,
  - Filtering on array and sequence integers in LTTng-UST and LTTng-modules.
  - Filtering: bitwise operators,
  - Kernel tracer: kernel and user-space callstack contexts.

LTTng 2.11 – Session Rotation

- Split trace in self-contained traces on the fly,
- Allow processing portion of the trace without stopping tracing,
- Allows for pipelining and/or sharding of analyses (scale-out distributed analysis),
- Encryption, compression, cleanup of old chunks, integration with external message bus tools,
- *Fine-grained Distributed Application Monitoring Using LTTng*, Jérémie Galarneau, Open Source Summit 2018.

LTTng 2.11 – Dynamic instrumentation with uprobes

- Adding tracepoints without having to recompile or restart a process,
- Using the uprobe interface,
- Tracing userspace using the **kernel tracer**,
- Supported instrumentation point types:
  - ELF symbols,
  - SystemTap/SDT probe points (without semaphore).

```
lttng enable-event --kernel
  --userspace-probe=elf:/path/to/binary:symbol
    event_name
```

EfficiOS

LTTng 2.11 – Dynamic instrumentation with uprobes

- Limitations:
  - Slower than LTTng-UST, because of context-switches to the kernel,
  - No tracepoint payload recorded at the moment.

Filtering on array and sequence of integers

- Filter out event based on the content of arrays and sequence

```
[14:32:57.03] host lttng_ust_prov:event : { _field_length = 4,
field = [ [0] = 121, [1] = 55, [2] = 23, [3] = 42 ] }
```

- Define filter using indexes in sequence:

```
lttng enable-event --userspace lttng_ust_prov:event
          --filter='field[0]<100 && field[3]==42'
```

EfficiOS

Filtering: Bitwise Operators

- Support bitwise operators in both kernel and user-space tracers:
  - Bitwise NOT (~),
  - Bitwise left/right shift (<</>>),
  - Bitwise AND (&),
  - Bitwise OR (|),
  - Bitwise XOR (^).

Kernel and User-Space Callstack Contexts

- In lttng-modules kernel tracer,
- Sample kernel and user-space callstacks as a context,
- Main use-case: sample user-space callstack on system call entry,
- Requires applications and libraries to be built with frame pointers to unroll user-space stacks.

Upcoming LTTng 2.12 Features

- LTTng 2.12-rc1 planned for mid-January 2019, 2.12 final planned for February 2019,
- User ID tracker,
- Relay daemon enhancements:
  - Categorize trace hierarchy by session / hostname,
  - Allow overriding current working directory,
  - LRU tracking of open file descriptors.
- Fast LTTng clear.

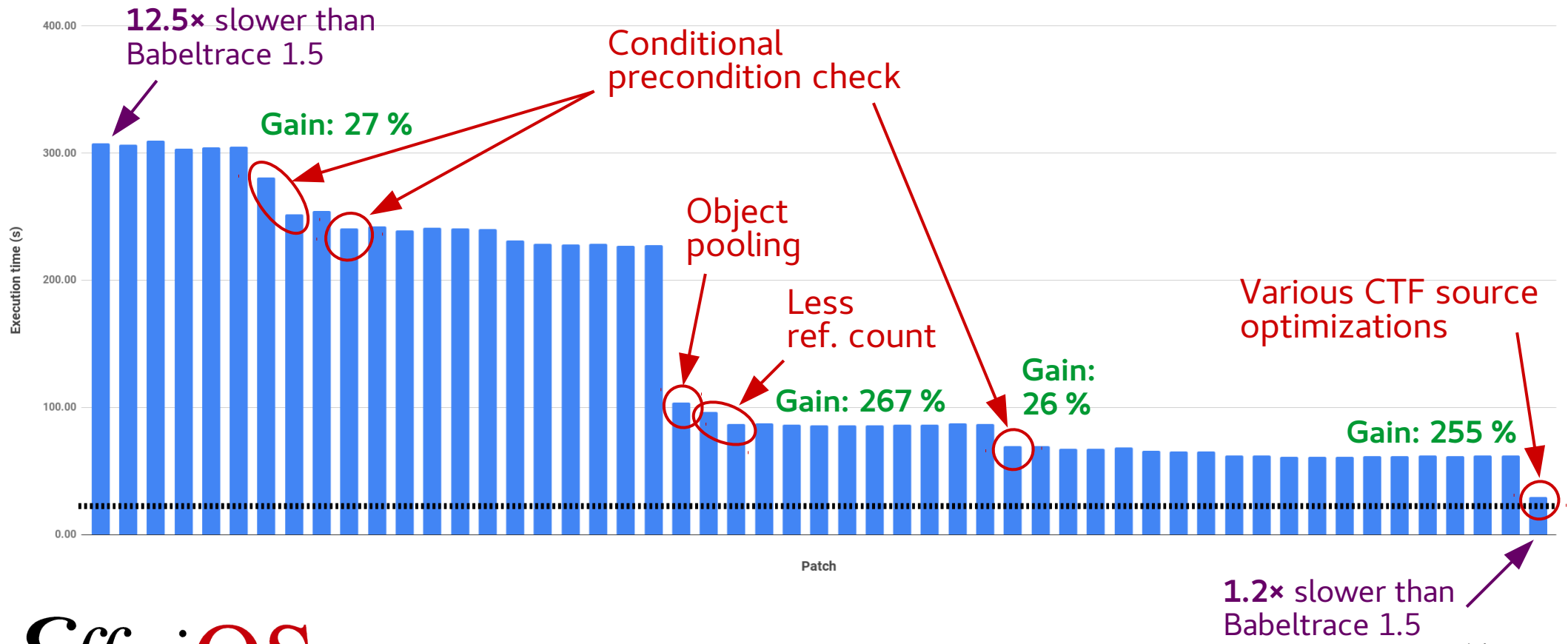
2019 (LTTng 2.13+)

- LTTng dynamic snapshot and event notification.
- LTTng strace-alike follow children:
  - Trace a hierarchy of processes with the PID tracker.
- Trace hit counters per tracepoint,
- Multiple liburcu flavors per applications,
- Data throughput counters per tracepoint.

Babeltrace 2.0 - Performance

- Babeltrace 2.0-pre measured to be 12.5x slower than Babeltrace 1.x,
- Focused on optimisations requiring changes to the API:
  - Reducing object allocation:
    - Object pooling.
  - Removing precondition checks:
    - Introducing “Developer Mode”.
  - Remove superfluous reference counting.
- Now 1.2x slower than Babeltrace 1.x,
- Aiming at least to be as fast as Babeltrace 1.x.

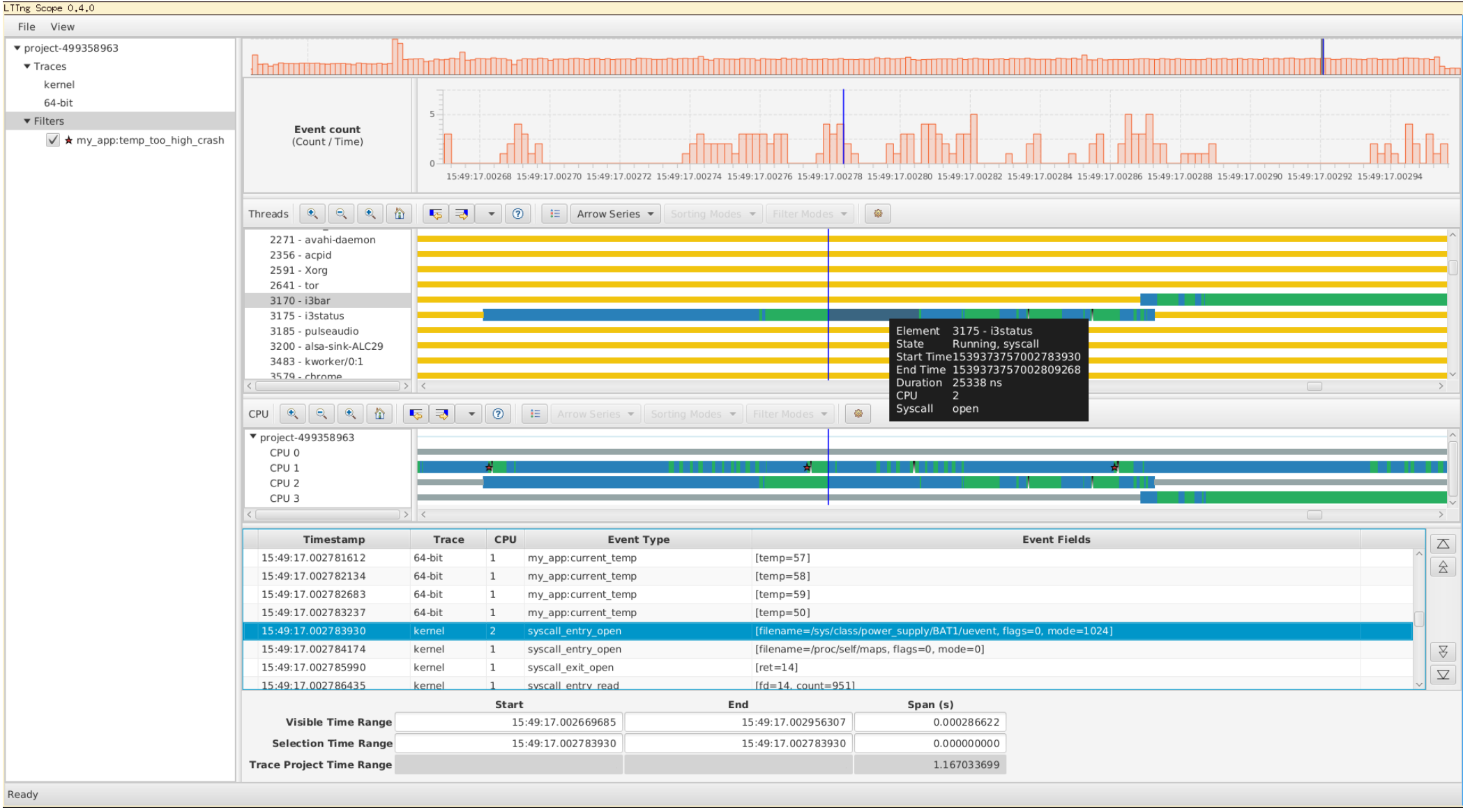
Babeltrace 2 Optimisation Results



EfficiOS

LTTng Scope

- LTTng Scope 0.4 (released October 15, 2018)
- Highlights:
  - Correlate multiple traces within a trace project:
    - E.g. kernel trace and UST traces,
  - Event count chart improvements,
  - Bug fixes.


LTTng Scope 0.4.0
File
View
project-499358963
Traces
kernel
64-bit
Filters
my_app:temp_too_high_crash
Event count
(Count / Time)
5
0
15:49:17.00268 15:49:17.00270 15:49:17.00272 15:49:17.00274 15:49:17.00276 15:49:17.00278 15:49:17.00280 15:49:17.00282 15:49:17.00284 15:49:17.00286 15:49:17.00288 15:49:17.00290 15:49:17.00292 15:49:17.00294
Threads
Arrow Series
Sorting Modes
Filter Modes
2271 - avahi-daemon
2356 - acpid
2591 - Xorg
2641 - tor
3170 - i3bar
3175 - i3status
3185 - pulseaudio
3200 - alsa-sink-ALC29
3483 - kworker/0:1
3579 - chrome
Element 3175 - i3status
State Running, syscall
Start Time 1539373757002783930
End Time 1539373757002809268
Duration 25338 ns
CPU 2
Syscall open
CPU
Arrow Series
Sorting Modes
Filter Modes
project-499358963
CPU 0
CPU 1
CPU 2
CPU 3
Timestamp Trace CPU Event Type Event Fields
15:49:17.002781612 64-bit 1 my_app:current_temp [temp=57]
15:49:17.002782134 64-bit 1 my_app:current_temp [temp=58]
15:49:17.002782683 64-bit 1 my_app:current_temp [temp=59]
15:49:17.002783237 64-bit 1 my_app:current_temp [temp=50]
15:49:17.002783930 kernel 2 syscall_entry_open [filename=/sys/class/power_supply/BAT1/uevent, flags=0, mode=1024]
15:49:17.002784174 kernel 1 syscall_entry_open [filename=/proc/self/maps, flags=0, mode=0]
15:49:17.002785990 kernel 1 syscall_exit_open [ret=14]
15:49:17.002786435 kernel 1 syscall_entry_read [fd=14, count=951]
Start End Span (s)
Visible Time Range 15:49:17.002669685 15:49:17.002956307 0.000286622
Selection Time Range 15:49:17.002783930 15:49:17.002783930 0.000000000
Trace Project Time Range 1.167033699
Ready