Progress Report Meeting, December 2016

EfficiOS Projects Status Update and Roadmap

jdesfossez@efficios.com
alexandre.montplaisir@efficios.com
jonathan.rajotte-julien@efficios.com

EfficiOS

Content

- New LTTng Features (2016),
- Project updates for 2016:
  - LTTng,
  - LTTng-Analyses,
  - Trace Compass,
  - Latency Tracker,
  - Babeltrace,
  - Common Trace Format (CTF) 2.0,
  - Barectf

New LTTng Features (2016)

LTTng is a low-overhead Linux kernel and user-space tracer

- Speeding up LTTng-UST (user-space tracer) on ARM32,
- Performance Monitoring Unit counters improvements,
- Linux kernel scheduler thread priority instrumentation for LTTng kernel tracer.

Speeding up LTTng-UST on ARM32

- Speed ups resulting from LTTng-UST profiling
- Propose new kernel system call: restartable sequences (rseq)
  - Expose CPU number through thread-local storage variable rather than system call on ARM32,
  - Expose Restartable Sequences ABI to speed up per-cpu atomic operations. Allows implementing atomic operations on per-cpu data as standard non-atomic operations,
  - Presented at Linux Plumbers Conference Referee Track: http://www.linuxplumbersconf.net/2016/ocw/proposals/3873

EfficiOS

Speeding up LTTng-UST

Benchmarks on Cubietruck ARM Cortex A7 @ 1GHz

		speedup
LTTng stable-2.8 (baseline), minus clock_gettime system call:	2288 ns/event	
Adding speed up commit resulting from profiling:	1624 ns/event	1.40:1
Adding use of restartable sequences:	1261 ns/event	1.81:1

- Also speed up LTTng-UST on i7-5600U @2.60GHz x86-64 from ~150ns to 90ns/event,
- Relevant improvements also implemented into LTTng modules kernel tracer.

EfficiOS

LTTng Performance Monitoring Unit Counters

- Added support for Performance Monitoring Unit counters for reader from user-space on ARM32,
  - Architectural limitation: requires a system call to read the counter value on ARM32.
- Added custom counter support on all architectures for LTTng-UST and LTTng modules:
  - Specify counter by raw value, associate name from user interface,
  - Useful for architectures with custom-made PMU counters.

Scheduler Thread Priority Instrumentation

- Linux kernel Tracepoints currently expose the “prio” value,
  - Internal scheduler value, should not have been exposed to user-space,
  - Does not convey deadline scheduler information,
  - Missing information at priority changes, only known on the next sched_switch event.

Scheduler Thread Priority Instrumentation

- New instrumentation proposed:
  - Expose Real-Time, Fair, and Deadline schedulers task state:
    - Scheduling policy,
    - Nice value, real-time priority,
    - Deadline scheduler: runtime, deadline, period,
    - Top waiter (priority inheritance).
  - Add missing instrumentation,
- Received feedback from scheduler maintainers, working on updated version.

LTTng Project Update (H2-2016)

- LTTng 2.9 (29-11-2016):
  - Discard mode buffers now available with snapshot tracing (single-shot),
  - New `lttng regenerate statedump` command,
    - Use-case: trigger state dump before taking flight recorder snapshot,
  - Allow override of trace name, path, destination URL when loading a session configuration.
- Titan LTTng-UST CTF logger plugin.

EfficiOS

LTTng-Analyses Project Update

The LTTng analyses are a set of various executable analyses to extract and visualize monitoring data and metrics from LTTng kernel traces on the command line. It models some kernel subsystems to track their state:

- *Latency statistics and distributions (IO, Scheduling, IRQ),*
- *System call statistics,*
- *IRQ handler duration,*
- *Top resource users (CPU, memory, ...).*

EfficiOS

LTTng-Analyses Project Update (H2-2016)

- Added support for nested period analyses: log, frequency distribution, statistics, top,
- Now uses stream intersection mode by default,

EfficiOS

LTTng analyses

```
[jgalar@XThink ~/LinuxCon2016/investigation/simple-trace]$ lttng-cputop /home/jgalar/lttng-traces/lttng-analysis-20987-20160820-235617
Checking the trace for lost events...
Processing the trace: 100% [##################################################################################################################]
Timerange: [2016-08-20 23:56:18.203115455, 2016-08-20 23:56:18.921040130]
Per-TID Usage                                                                   Process                    Migrations   Priorities
##########################################################################
                                                                        3.04 %  watch (16125)                     0   [20]
                                                                        2.87 %  Xorg (832)                         0   [20]
                                                                        1.56 %  chromium (15418)                   0   [20]
                                                                        0.99 %  lttng-sessiond (9343)             0   [20]
                                                                        0.93 %  alsa-sink-USB A (999)            0   [-6]
                                                                        0.58 %  watch (20771)                     0   [20]
                                                                        0.55 %  watch (20769)                     0   [20]
                                                                        0.55 %  watch (20775)                     0   [20]
                                                                        0.55 %  watch (20777)                     0   [20]
                                                                        0.55 %  pulseaudio (990)                  0   [9]

Per-CPU Usage
##########################################################################
                                                                        3.77 % CPU 0
                                                                        4.44 % CPU 1
                                                                        6.42 % CPU 2
                                                                        3.44 % CPU 3

Total CPU Usage: 4.52%
```

LTTng analyses

```
[jgalar@XThink ~/LinuxCon2016/investigation/simple-trace]$ lttng-iousagetop /home/jgalar/lttng-traces/lttng-analysis-20987-20160820-235617
Checking the trace for lost events...
Processing the trace: 100% [##################################################################################################################]
Timerange: [2016-08-20 23:56:18.203115455, 2016-08-20 23:56:18.921040130]
Per-process I/O Read                                                                  Process                    Disk        Net         Unknown
##########################################################################
                                                                          113.62 KiB sh (20765)                113.62 KiB      0   B       0   B
                                                                          113.62 KiB sh (20767)                113.62 KiB      0   B       0   B
                                                                          113.62 KiB sh (20769)                113.62 KiB      0   B       0   B
                                                                          113.62 KiB sh (20771)                113.62 KiB      0   B       0   B
                                                                          113.62 KiB sh (20773)                113.62 KiB      0   B       0   B
                                                                          113.62 KiB sh (20777)                113.62 KiB      0   B       0   B
                                                                          113.62 KiB sh (20775)                113.62 KiB      0   B       0   B
                                                                           25.70 KiB lttng-sessiond (9296)          0   B       0   B  25.70 KiB
                                                                           23.77 KiB lttng (20778)               15.21 KiB   8.57 KiB       0   B
                                                                            5.25 KiB watch (16125)                   0   B       0   B   5.25 KiB

Per-process I/O Write                                                                 Process                   Disk        Net         Unknown
##########################################################################
                                                                          25.70 KiB lttng (20778)                  0   B  25.70 KiB       0   B
                                                                           8.59 KiB lttng-sessiond (9296)          0   B       0   B   8.59 KiB
                                                                           2.41 KiB Chrome_ChildIOT (15407)        0   B       0   B   2.41 KiB
                                                                           1.47 KiB xcompmgr (939)                 0   B       0   B   1.47 KiB
                                                                            777   B terminator (15005)             0   B       0   B    777   B
                                                                            768   B sh (20765)                     0   B       0   B    768   B
                                                                            768   B sh (20767)                     0   B       0   B    768   B
                                                                            768   B sh (20769)                     0   B       0   B    768   B
                                                                            768   B sh (20771)                     0   B       0   B    768   B
                                                                            768   B sh (20773)                     0   B       0   B    768   B

Per-file I/O Read                                                                     Path
##########################################################################
```

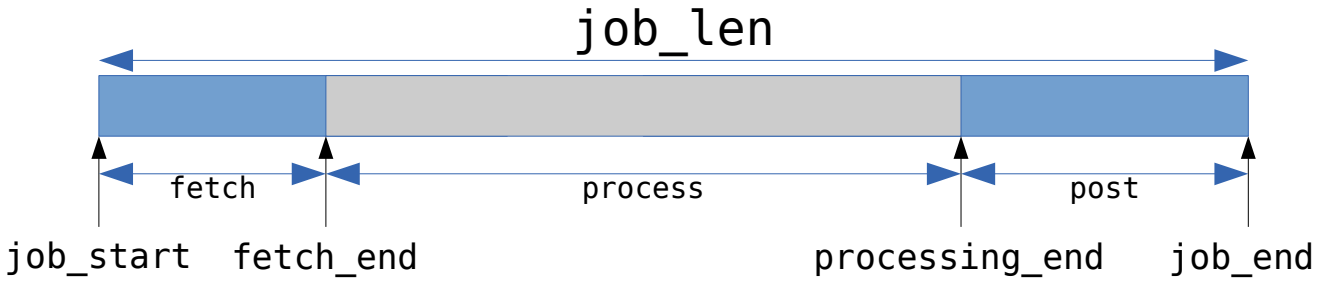
LTTng analyses

```
[jgalar@XThink ~/LinuxCon2016/investigation/simple-trace]$ lttng-syscallstats kernel/
Checking the trace for lost events...
Processing the trace: 100% [##################################################################################################################
Timerange: [2016-08-20 23:56:18.203115455, 2016-08-20 23:56:18.921040130]
Per-TID syscalls statistics (usec)
watch (16125, TID: 16125)                    Count           Min       Average           Max       Stdev  Return values
 - read                                      44107           0.138         0.797      4251.704      50.316  {'success': 44108}
 - rt_sigaction                                 14           0.247         0.489         0.924       0.247  {'success': 15}
 - close                                        14            0.39          0.61         1.058       0.193  {'success': 15}
 - poll                                         14           0.538         1.613         3.075       1.036  {'success': 15}
 - pipe                                          7           5.466         6.542         7.635       0.885  {'success': 8}
 - wait4                                         7           3.255          3.83         5.179       0.642  {'success': 8}
 - fcntl                                         7           0.519         1.045         1.255        0.25  {'success': 8}
 - clone                                         7           64.16        68.005        74.077        3.14  {'success': 8}
 - newfstat                                      7           0.772         0.878         1.195       0.151  {'success': 8}
 - newstat                                       7           8.248         9.101        10.215        0.67  {'success': 8}
 - newuname                                      7           1.044          1.23         1.517       0.174  {'success': 8}
 - nanosleep                                     6      100096.694     100102.787    100107.064       4.424  {'success': 7}
Total:                                       44204
-----------------------------------------------------------------------------------------------------------------
lttng-consumerd (?, TID: 9423)               Count           Min       Average           Max       Stdev  Return values
 - ioctl                                       325           0.177         1.234        55.117       3.355  {'EAGAIN': 2, 'success': 325}
 - splice                                      162           1.382         3.313        12.323       2.013  {'success': 163}
 - sync_file_range                             162           6.949          30.1       512.894      42.733  {'success': 163}
 - fadvise64                                    81           1.889         2.676        24.711       2.783  {'success': 82}
Total:                                         730
-----------------------------------------------------------------------------------------------------------------
threaded-ml (?, TID: 7926)                   Count           Min       Average           Max       Stdev  Return values
 - read                                        162            0.24         1.971        17.789       2.599  {'EAGAIN': 51, 'success': 113}
 - write                                        97           0.456         2.508         9.372       1.444  {'success': 98}
 - fcntl                                        93           0.178         0.589         1.799       0.433  {'success': 94}
 - poll                                         49         166.732      14489.45     20299.653    5428.461  {'success': 50}
 - select                                       31           3.987         4.974          9.53       0.972  {'success': 32}
 - ioctl                                        31           0.781           1.1         3.642        0.49  {'success': 32}
Total:                                         463
```

LTTng analyses

```
[jgalar@XThink ~/LinuxCon2016/investigation/simple-trace]$ lttng-iolatencyfreq kernel/
Checking the trace for lost events...
Processing the trace: 100% [##################################################################
Timerange: [2016-08-20 23:56:18.203115455, 2016-08-20 23:56:18.921040130]
I/O request latency distribution open (µs)
##########################################################################
  0.218                                                                  7
  0.561                                                                 24
  0.903                                                                 53
  1.246                                                                 21
  1.589                                                                 31
  1.932                                                                 37
  2.274                                                                 31
  2.617                                                                  5
  2.960                                                                  0
  3.303                                                                  2
  3.646                                                                  1
  3.988                                                                  0
  4.331                                                                  0
  4.674                                                                  1
  5.016                                                                  5
  5.359                                                                  0
  5.702                                                                  1
  6.045                                                                  0
  6.388                                                                  0
  6.730                                                                  2
```

Custom Period




```
$ lttng-periodlog --period ‘job_len:        $evt.$name == “job_start” :
                                            $evt.$name == “job_end”
                  --period ‘fetch(job_len): $evt.$name == “job_start” :
                                            $evt.$name == “fetch_end”’
                  [...]
```

lttng-period{log,top,stats,freq}

- Extract statistics, log, top, frequency distributions of period durations
- Allow to identify the longest periods (high latency)
- Keep relationship and data between child/parent period definitions
- Group statistics and frequency distributions based on payload value(s)

Custom Periods

```
$ lttng-periodlog
    --period ‘[ NAME [ (PARENT) ] ] : BEGIN_EXPR [ : END_EXPR ]’
    --period-capture ‘NAME : BEGINCAPTURES [ : ENDCAPTURES ]’
    --select PERIODS
    --aggregate-by PERIOD
    --group-by FIELD
```

Example period

```
sched_switch: { cpu_id = 1 }, { prev_comm = "swapper/1", prev_tid = 0,
|        prev_prio = 20, prev_state = 0,
|        next_comm = "bash", next_tid = 12421, next_prio = 20 }
|
|    syscall_entry_open: { cpu_id = 1 }, { filename = "/etc/ld.so.cache",
|    |      flags = 524288, mode = 1 }
|    |
|    |    kmem_cache_alloc: { cpu_id = 1 }, { call_site = 0xFFFFFFFF811CDB1F,
|    |    |   ptr = 0xFFFF88037BF67000, bytes_req = 4096, bytes_alloc = 4096,
|    |    |   gfp_flags = 208 }
|    |    |
|    |    kmem_cache_free: { cpu_id = 1 }, { call_site = 0xFFFFFFFF811CDAA2,
|    |        ptr = 0xFFFF88037BF67000 }
|    |
|    syscall_exit_open: { cpu_id = 1 }, { ret = 3 }
|
sched_switch: { cpu_id = 1 }, { prev_comm = "lttng", prev_tid = 12421,
         prev_prio = 20, prev_state = 1, next_comm = "swapper/1",
         next_tid = 0, next_prio = 20 }
```

Delay between the 2 sched_switch: 3.6ms
Files opened: 22
Events generated during that period: 1570

EfficiOS

Period definition

```
$ ./lttng-periodlog /path/to/trace
   --period 'switch :
             $evt.$name == "sched_switch" :
             $evt.$name =="sched_switch" &&
                   $begin.$evt.next_tid == $evt.prev_tid &&
                   $begin.$evt.cpu_id == $evt.cpu_id' \
  --period 'open(switch) :
             $evt.$name == "syscall_entry_open" &&
                   $parent.$begin.$evt.cpu_id == $evt.cpu_id :
             $evt.$name == "syscall_exit_open" &&
                   $begin.$evt.cpu_id == $evt.cpu_id' \
  --period 'alloc(open) : $evt.$name == "kmem_cache_alloc" &&
             $parent.$begin.$evt.cpu_id == $evt.cpu_id :
             $evt.$name == "kmem_cache_free" && $evt.ptr == $begin.$evt.ptr' \
  --period-captures 'switch : comm = $evt.next_comm, cpu = $evt.cpu_id,
             tid = $evt.next_tid' \
  --period-captures 'open : filename = $evt.filename : fd = $evt.ret' \
  --period-captures 'alloc : ptr = $evt.ptr' \
  --select "open, alloc" \
```

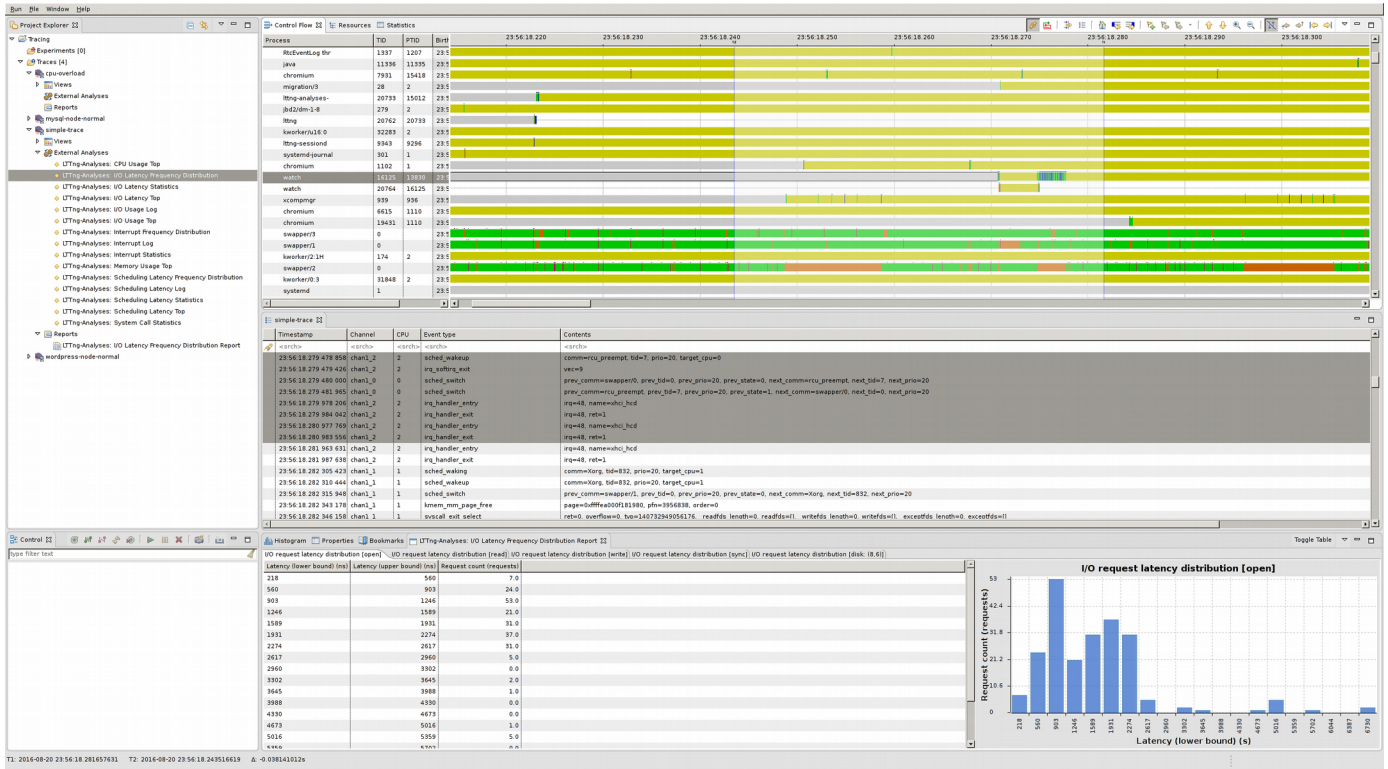
Trace Compass Project Update (H2-2016)

- *Eclipse Trace Compass provides views, graphs, metrics, and more to help extract useful information from traces.*

- Speed up single-stepping of kernel events when following one thread,
- Integration of LTTng-Analyses machine interface,
- Pin & New View features (proposed upstream),
- Control Flow View dynamic filter on active threads with CPU set selection (proposed upstream),
- Stream intersection mode,
- Trace cut feature.

EfficiOS

LTTng analyses - Trace Compass Integration



- Invoke custom analyses
- LAMI 1.0
  - Open Specification
  - JSON based

TRACE COMPASS

EfficiOS

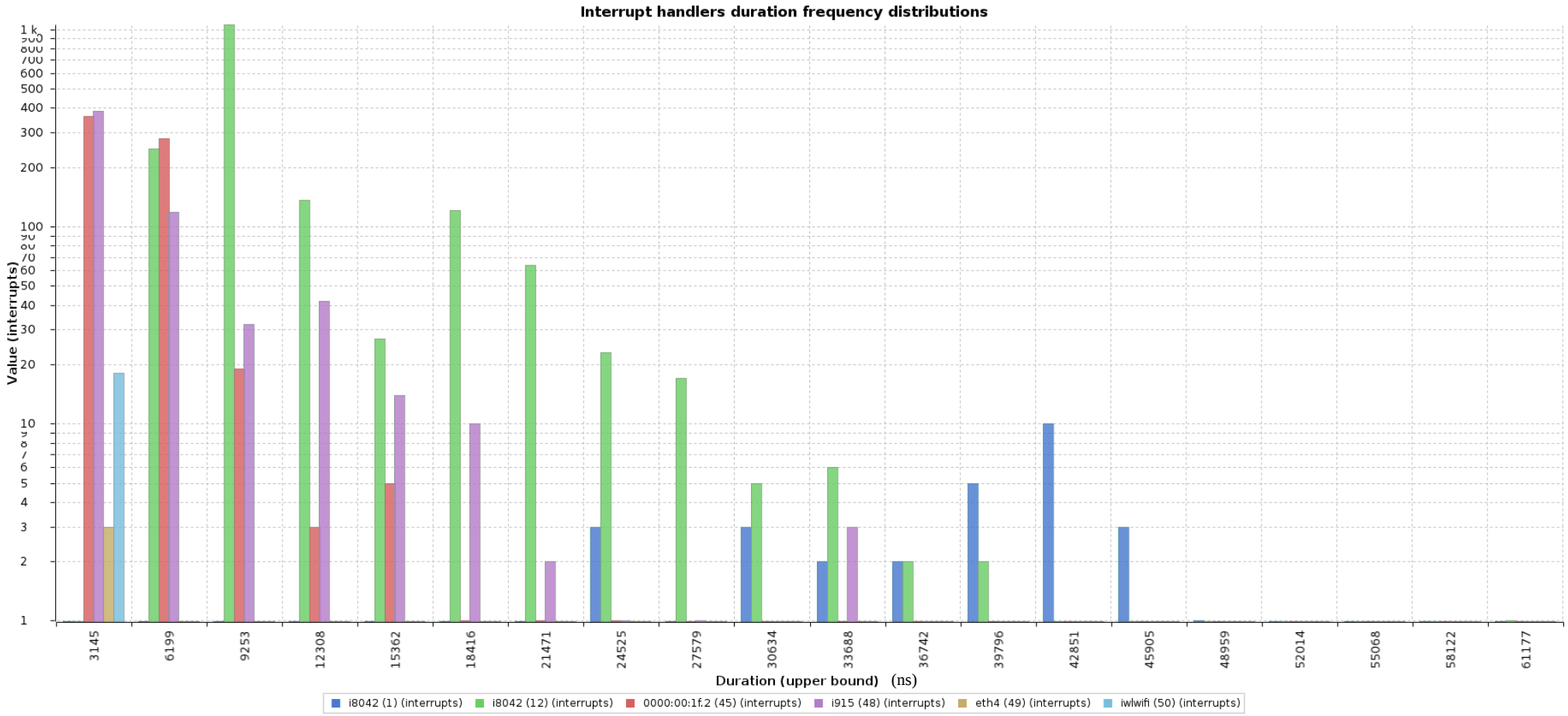
Trace Compass

File Window Help

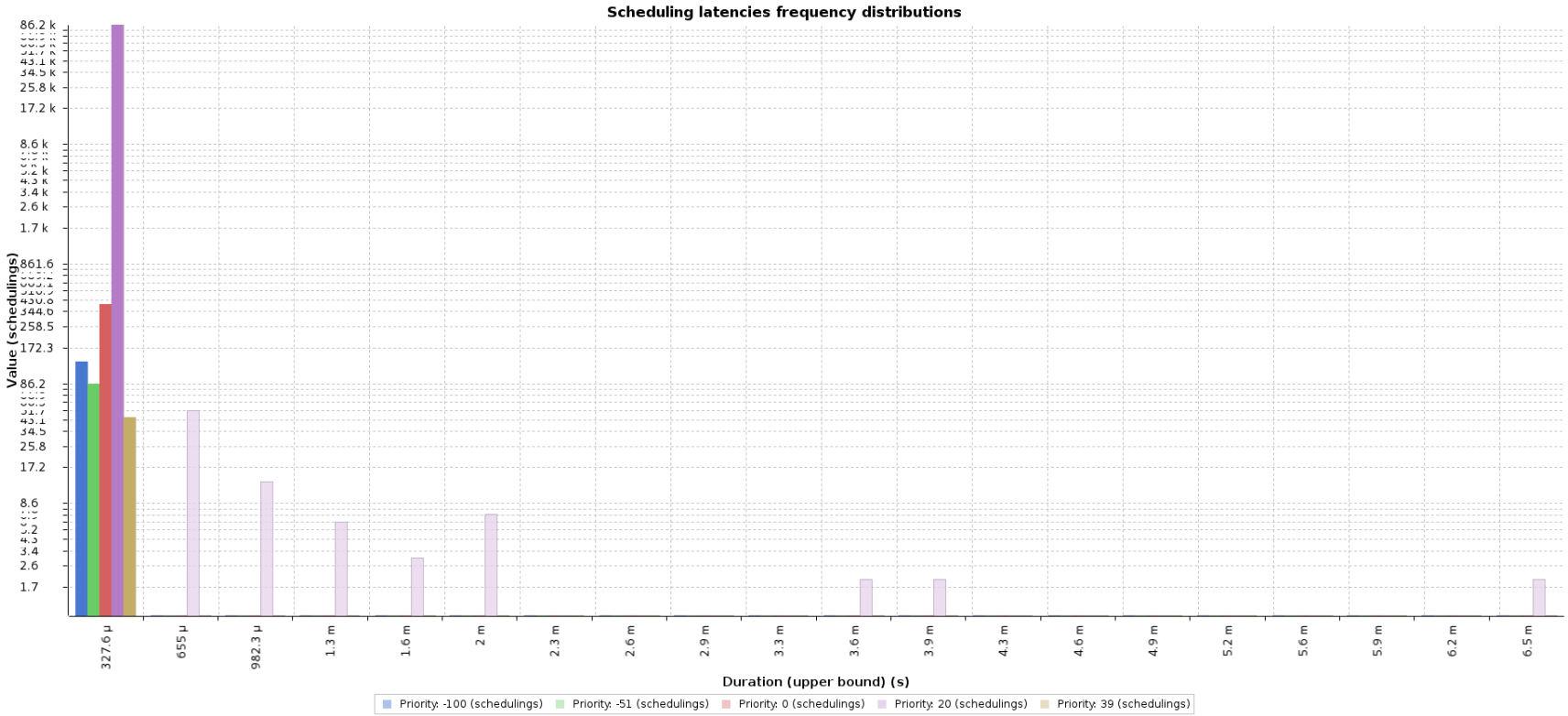
Histogram | Properties | Bookmarks | System Call Density | LTTng-Analyses: Interrupt Frequency Distribution Report

Toggle Table

Interrupt handlers duration frequency distributions



Scheduling Latencies



EfficiOS

Latency Tracker Project Update (H2-2016)

The Latency Tracker is a kernel module performing statistical latency trend aggregation, and identification of outliers. It can trigger user-configurable actions such as recording a flight recorder snapshot when outliers are detected,

- Track work begin/end with identifiers from instrumented user-space,
- Time-to-first-byte tracker.

Available Latency Trackers

- Block layer: from block request issue to completion,
- Network: from socket buffer receive to consumption by user-space,
- Wake-up: from each thread wake-up to next scheduling of that thread,
- Off-cpu: from each thread preemption/blocking to next execution of that thread,
- IRQ handler: from irq handler entry to exit,
- System call: from system call entry to exit,
- Time-to-first-byte: from accept system call return to write system call family entry on the same inode,
- Online critical path analysis: from interrupt servicing to completion of task.

Babeltrace Project Update (H2-2016)

The Babeltrace project provides a library, Python bindings, as well as a command-line tool to view and convert traces. It is a reference implementation of the Common Trace Format (CTF).

- Babeltrace 1.4 (06-2016)
  - Mapping events to C/C++ source code (DWARF debug info, ELF),
  - Stream intersection mode (for LTTng snapshots),
  - Lost packet reporting.
- Babeltrace 1.5 (12-2016)
  - Expose APIs required by Perf to CTF converter.

EfficiOS

Babeltrace Project Update (2017)

- Babeltrace 2.0 planned in January 2017
  - Trace Intermediate Representation,
  - Modular source/filter/sink architecture,
  - Plugin architecture,
  - C/C++/Python APIs,
  - Allows analyses to read live traces,
  - CTF 1.8 source/sink (reader/writer),
  - Trace cut feature,
  - Multi-clock support (e.g. Epoch time and BFN clock).

Babeltrace Project Update (2017)

- Babeltrace 2.1 (2017)
  - Event filtering,
  - CTF 2.0.

CTF 2.0

The Common Trace Format (CTF) is a binary trace format designed to be very fast to write without compromising great flexibility. It allows traces to be natively generated by any C/C++ application or system, as well as by bare-metal (hardware) components.

- Main change in CTF 2 is to move from custom metadata language to JSON, for flexibility and extensibility purposes,
- CTF 2 proposal document sent for comments on lttng-dev and diamon-discuss mailing lists.

Barectf Project Update (H2-2016)

barectf is a command-line generator of ANSI C tracers which output Common Trace Format packets natively.

- Demo of instrumented Parallella bare-metal application, with flight recorder snapshots, and custom Trace Compass view.

Links

LTTng:
http://lttng.org

LTTng analyses scripts:
https://github.com/lttng/lttng-analyses

Latency tracker:
https://github.com/efficios/latency-tracker

barectf:
https://github.com/efficios/barectf

TraceCompass:
http://tracecompass.org/

Babeltrace
http://diamon.org/babeltrace

Common Trace Format
http://diamon.org/ctf

Trace Compass Demos

- Dynamic filters
  - Active Threads
  - Per CPU filtering
- Pin & Clone of views
- Trace cutting

EfficiOS

Questions ?

EfficiOS

www.efficios.com

lttng.org

lttng-dev@lists.lttng.org

@lttng_project