Tracing Summit 2016

Real-Time Response Time Measurement by Integration of Trace Buffering and Aggregation Tools

EfficiOS

mathieu.desnoyers@efficios.com

Presenter

- Mathieu Desnoyers
  - CEO at EfficiOS
  - LTTng, Linux, Userspace RCU, Babeltrace maintainer.

Content

- Trace buffering vs in-place aggregation
- Automate problem analysis by combining aggregation and post-processing tools
- Periodic use-case demo
  - Jack audio server
- Aperiodic use-cases demos
  - Memcached
- Benchmarks
- Future Work

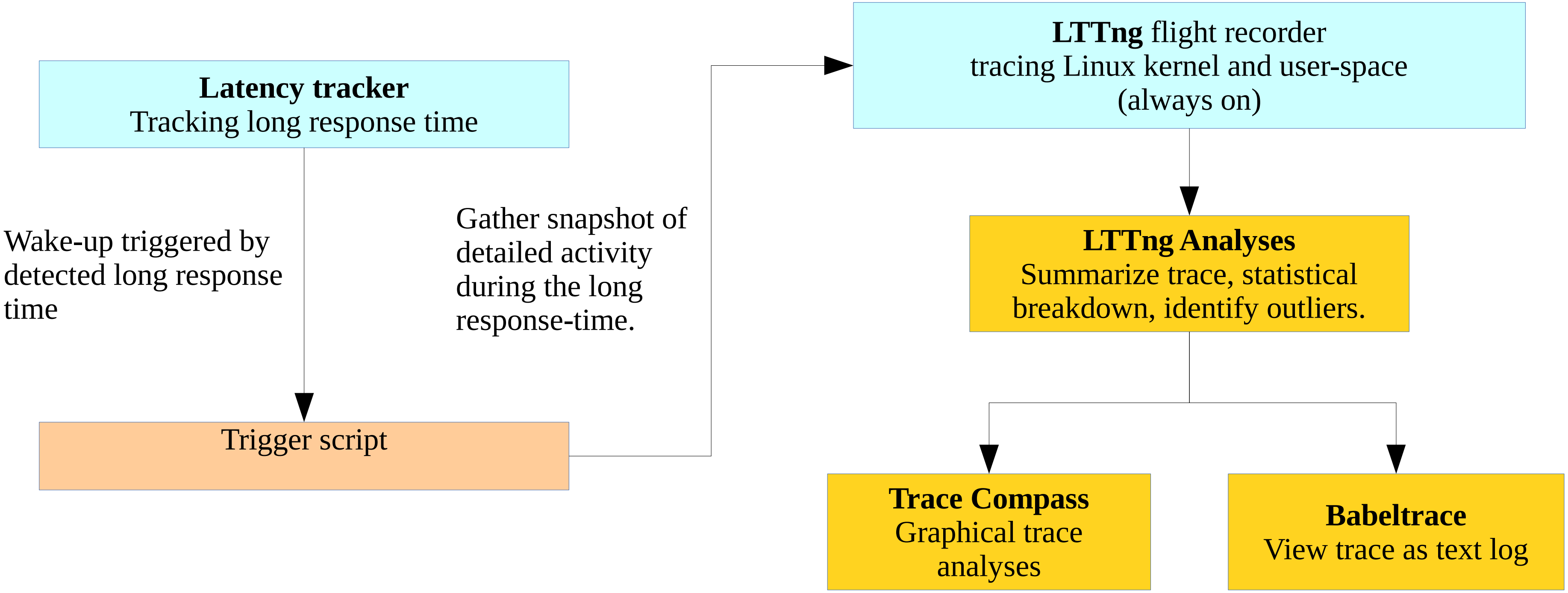
Trace Buffering vs In-Place Aggregation

- Trace buffering:
  - Store events into a buffer,
  - Analysis performed at post-processing,
  - Multiple analyses can be performed on the same recorded trace,
  - E.g. Ftrace, Perf, LTTng.
- In-place aggregation:
  - Run-time analysis directly using event input,
  - Aggregation performed in the traced execution context,
  - E.g. eBPF, DTrace, SystemTAP.

Trace Buffering vs In-Place Aggregation

- Often presented as competing tracing solutions,
- In reality, can be ***combined*** to create powerful analysis tools.

Combining Trace Buffering with Aggregation



Latency Tracker

- Kernel module to track down latency problems at run-time,
- Simple API that can be called from anywhere in the kernel (tracepoints, kprobes, netfilter hooks, hardcoded in other module or the kernel tree source code),
- Keep track of entry/exit events and calls a callback if the delay between the two events is higher than a threshold.

Latency Tracker Usage

```
tracker = latency_tracker_create(threshold, timeout, callback);

latency_tracker_event_in(tracker, key);
....
latency_tracker_event_out(tracker, key);
```

If the delay between the event_in and event_out for the same key is higher than "threshold", the callback function is called.

The timeout parameter allows to launch the callback if the event_out takes too long to arrive (off-CPU profiling).

Latency Tracker: Low-Impact, Low-Overhead

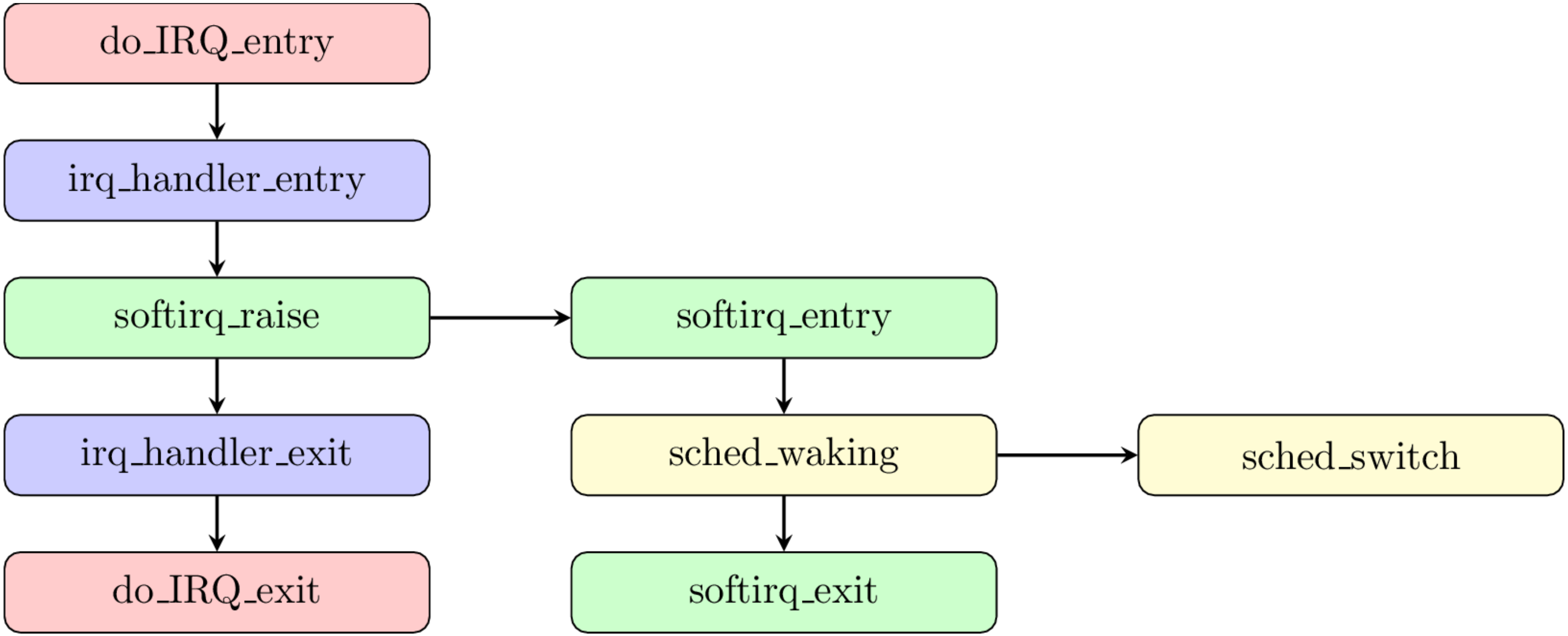
- Memory allocation:
  - Custom memory allocator implemented with lock-free per-CPU RCU free-lists and pre-allocated NUMA pools,
  - Out-of-context worker thread can expand the memory pools as needed up to a user-configurable limit,
  - Prior to 3.17, custom call_rcu thread to avoid wake-up deadlock. Starting from 3.17, use call_rcu_sched().
- State tracking:
  - Userspace-rcu hashtable ported to the Linux kernel:
    - Lock-free insertion and removal, wait-free lookups

EfficiOS

Implemented 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.

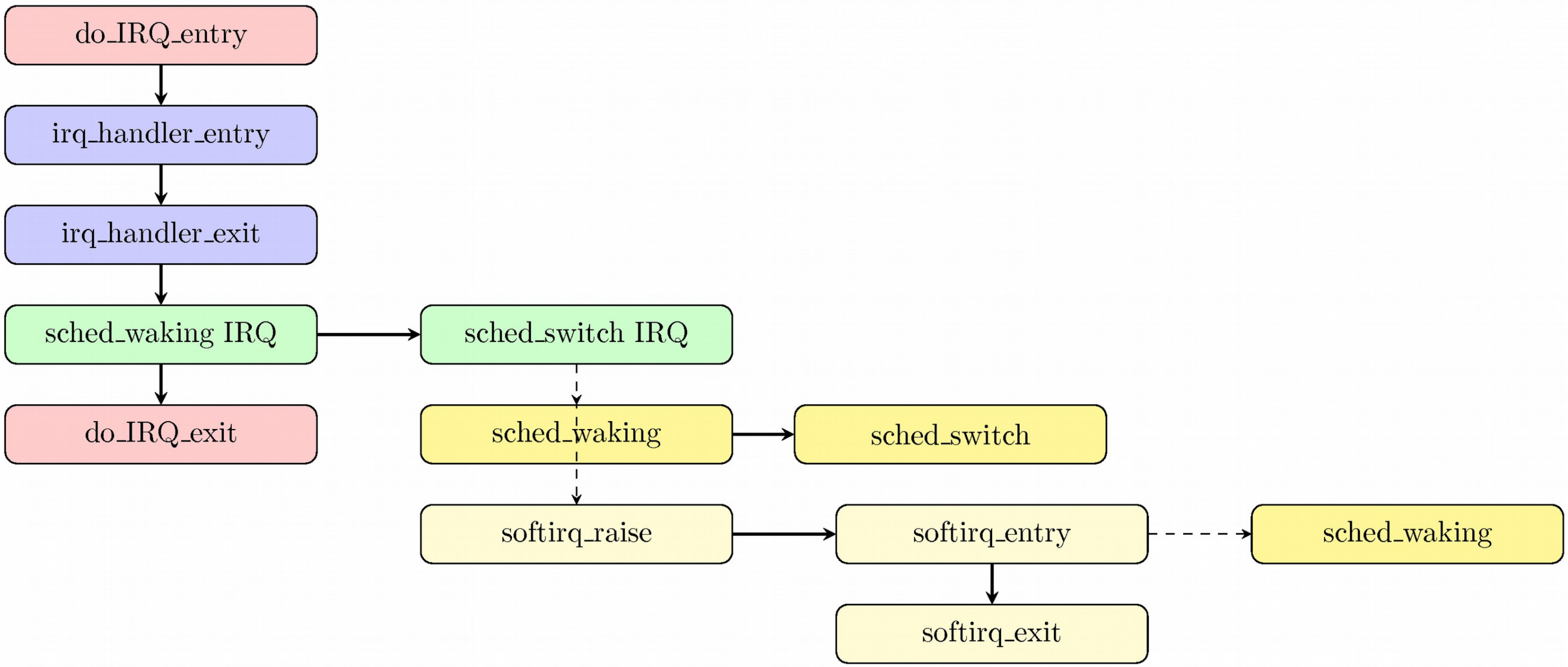
Response Time: Interrupt to Thread Execution



Linux Mainline Hardware Interrupt
Processing Critical Path

EfficiOS

Interrupt to Thread Execution (Preempt-RT)



Linux Preempt-RT Hardware Interrupt Processing Critical Path

Latency Tracker: Online Critical Path Analysis

- Measure response time,
- Execution contexts and wakeup chains tracking in kernel module
  - For both mainline kernel and preempt-rt,
  - IRQ, SoftIRQ, wakeup/scheduling chains, NMI (eventually).
- Follow critical path from interrupt servicing to completion of task,
- Can perform user-defined action when latencies are higher than a specified threshold,

Online Critical Path Analysis Configuration

- Passing parameters to latency tracker kernel module
  - Latency threshold,
  - Chain filters:
    - User-space task, pid, process name, RT task, Interrupt source (timer or IRQ/SoftIRQ number),
  - Chain stops when target task starts to run,
  - Chain stops when target task blocks,
- Track work begin/end with identifiers from instrumented user-space
  - Complex asynchronous use-cases.

LTTng Kernel and User-Space Tracers

- **Low-overhead**, **correlated** kernel and user-space tracing,
  - Ring buffers in shared memory.
- User-defined filtering on event arguments,
- System-wide or tracking of specific processes,
- Optionally gather performance counters and extra fields as contexts.
- Support disk I/O output, in-memory flight recorder, network streaming, live reading.

LTTng Kernel Tracer (LTTng-modules)

- Load kernel tracer modules (**no kernel patching required!**), or build into the Linux kernel image,
- LTTng kernel tracer hooks on:
  - Tracepoints,
  - System call entry/exit with detailed argument content,
  - Kprobes,
  - Kretprobes.

LTTng User-Space Tracer (LTTng-UST)

- Dynamically loaded shared library,
- Fast user-space tracing, fast-path entirely in user-space,
- Instruments:
  - Application and libraries with lttng-ust tracepoints, tracef, tracelog,
  - Java JUL and Log4j loggers, Python logger,
  - Malloc, pthread mutex with symbol override,
  - Function entry/exit by compiling with -finstrument-functions.
- Dumps base address information required to map process addresses to executable and library functions/source code using ELF and DWARF.

EfficiOS

LTTng Analyses

- Offline analysis based on LTTng traces,
- Analyze CPU, memory, I/O, interrupts, scheduling, system calls,
- Distribution, top, log over threshold:
  - I/O latency,
  - IRQ handler duration, SoftIRQ raise latency, handler duration,
  - Thread wakeup latency (sched_waking to sched_switch in),
  - User-defined periods based on kernel and user-space events.
- Integrated with Trace Compass graphical user interface.

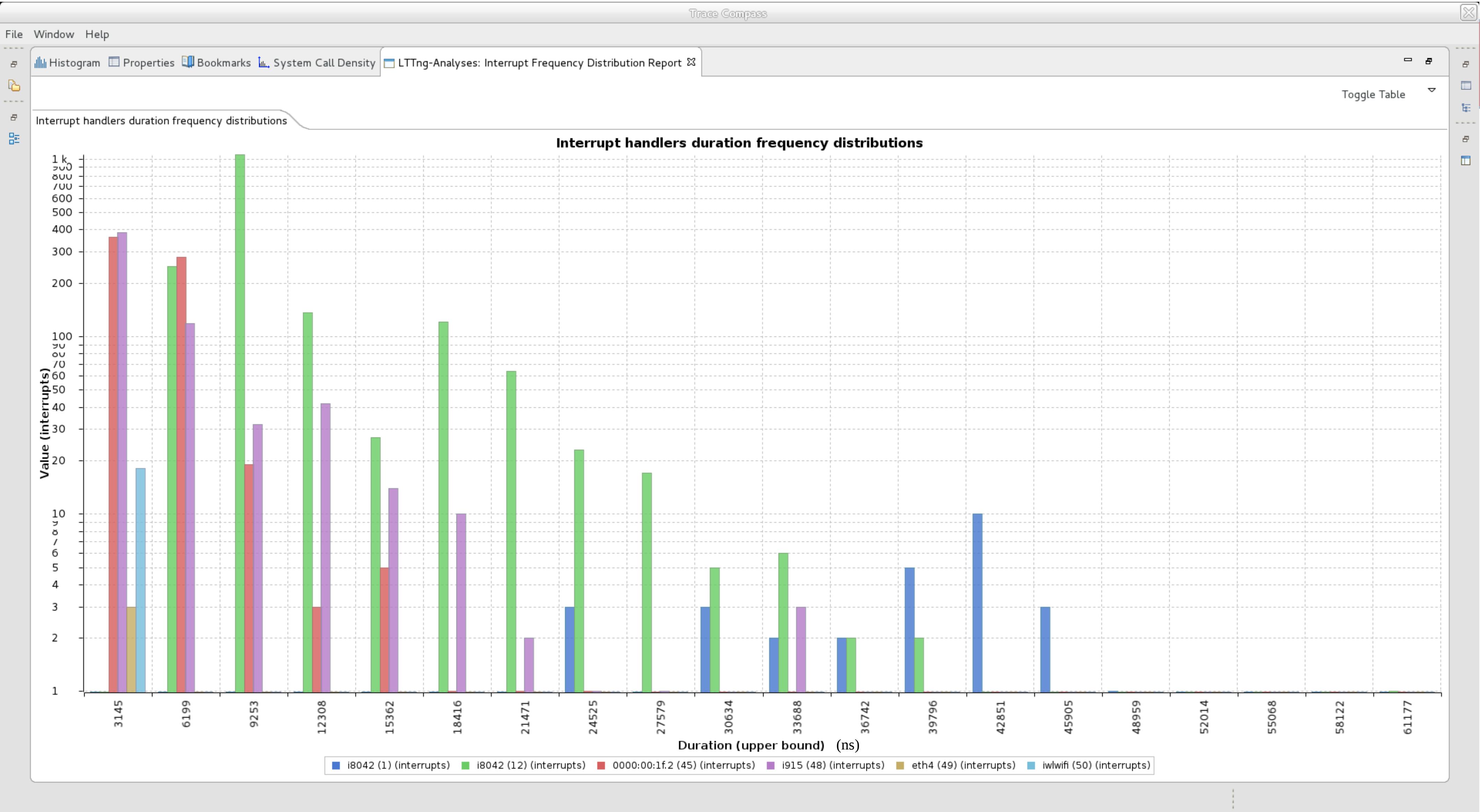
LTTng Monitoring

http://grafana.ini-tech.com:3002/dashboard/db/response-time
Login: demo Password: demo123

Trace Compass

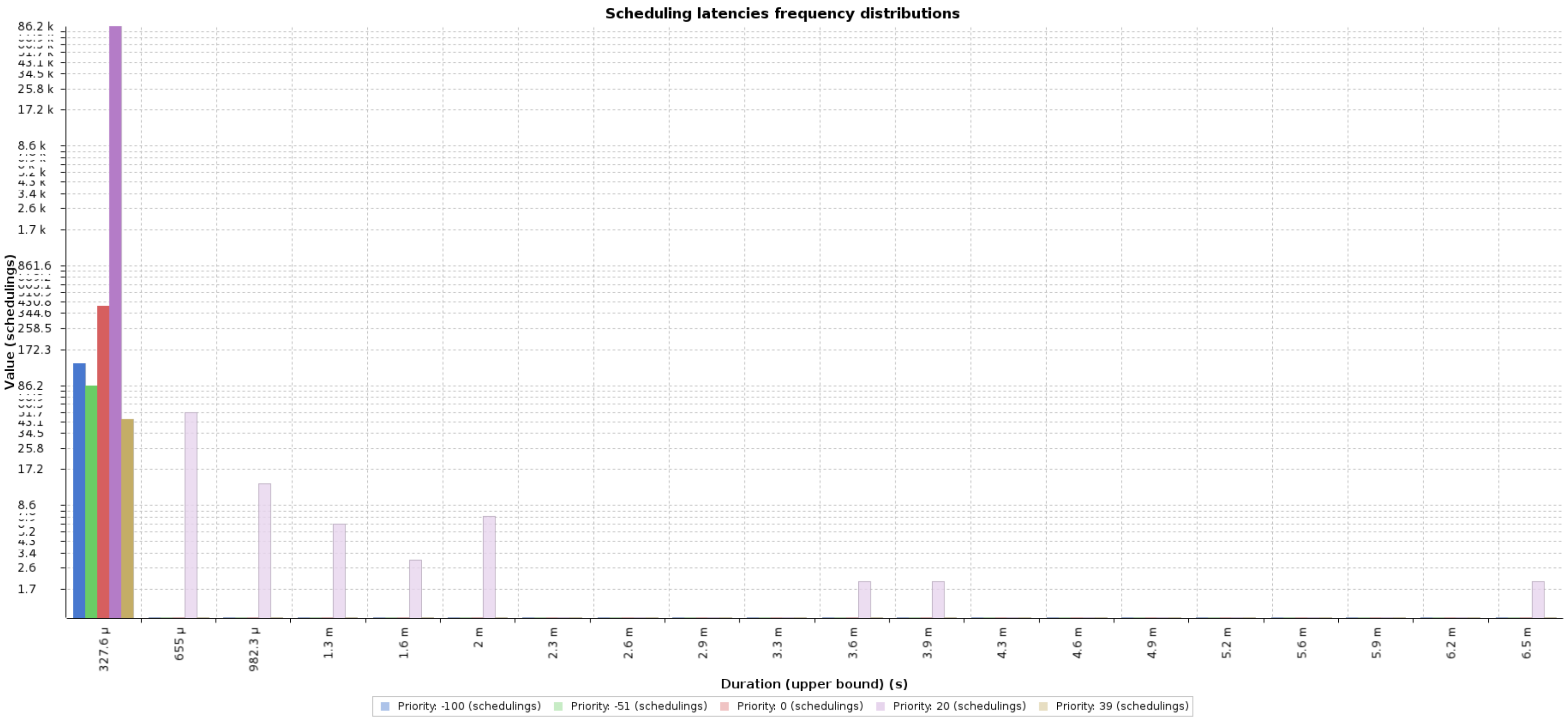
- Graphical user interface,
- Useful for correlating trace analysis results with detailed graphical representation,
- Implements its own analyses,
- Implements LAMI JSON interface to interact with external analysis scripts.

Trace Compass
File
Window
Help
kernel
Control Flow
Process
TID
kernel
systemd
1
gnome-terminal-
2958
bash
32649
gdm3
1127
gdm-session-wor
2647
gnome-session
2662
gnome-shell
2822
firefox-esr
29690
Timer
29711
Compositor
29717
Xorg
1158
swapper/2
0
kthreadd
2
kworker/u16:2
31802
16:23:08.240100
16:23:08.240200
16:23:08.240300
16:23:08.240400
16:23:08.240500
re
wri
select
ioctl
sele
Statistics
Resources
2016 Aug 21
CPU 0
CPU 1
CPU 2
CPU 3
IRQ 1
IRQ 12
IRQ 45
IRQ 48
bash
gnome-shell
Compositor
firefox-esr
writev
Xorg
gnome-terminal-
Histogram
Properties
Bookmarks
System Call Density
Selection Start
16:23:08.240 434 007
Selection End
16:23:08.240 434 007
16:23:08.240 013 051
16:23:08.240 673 556
18034
T1: 2016-08-21 16:23:08.240434007



Trace Compass
File Window Help
Histogram Properties Bookmarks System Call Density LTTng-Analyses: Interrupt Frequency Distribution Report
Toggle Table
Interrupt handlers duration frequency distributions
Interrupt handlers duration frequency distributions
Value (interrupts)
Duration (upper bound) (ns)
3145 6199 9253 12308 15362 18416 21471 24525 27579 30634 33688 36742 39796 42851 45905 48959 52014 55068 58122 61177
i8042 (1) (interrupts) i8042 (12) (interrupts) 0000:00:1f.2 (45) (interrupts) i915 (48) (interrupts) eth4 (49) (interrupts) iwlwifi (50) (interrupts)

Scheduling Latencies



EfficiOS

Babeltrace

- Common Trace Format (CTF) trace reader/converter,
- Performs time-based trace correlation/merge,
- Expose APIs (C, C++, Python) for reading CTF traces,
- Pretty-print traces into text log.

Periodic Use-Case Demo

- Jack
  - Infrastructure for communication between audio applications and with audio hardware
  - http://www.jackaudio.org
  - Scheduling latency caused by unsuitable priorities.

Aperiodic Use-Cases Demos

- Memcached
  - Distributed in-memory object caching system
  - http://memcached.org
  - Response-time to start handling client query
    - Interrupt servicing latency caused by long driver interrupt handler
  - Response-time to complete client query handling
    - I/O latency caused by logging

Benchmarks

- Latency tracker online critical path
  - Memcached, through gigabit interface,
  - 10k requests,
  - Baseline: 491 ms
  - With tracker: 520 ms
  - Overhead: 5.9 %

Latency Tracker Critical Path Bechmarks

Test	Baseline	Tracker	Overhead
CPU	19.20s	19.20s	0.00%
Memory	32.33s	32.37s	0.30%
File Read/Write	9.04 s	9.50 s	5.10%
Network 1Gbps	942Mbps/s	942Mbps/s	0.00%
Network 10Gbps	8.02Gbps/s	7.70Gbps/s	3.89%
OLTP (MySQL)	2.27s	2.38s	4.84%

Latency Tracker Critical Path Benchmarks

Metric	Transition	No transition
Ratio of requests	0.6%	99.4 %
Average latency	1136.93 ns	259.13 ns
Standard deviation	278.71 ns	28.42 ns
Minimum latency	565 ns	237 ns
Maximum latency	3028 ns	1938 ns
Average instruction count	2024	756
Average L1 misses	38.78	3.04
Average LLC misses	3.66	0.003
Average TLB misses	0.12	0.002
Average branch misses	3.08	0.15

Future Work

- Expose API to lock-free memory allocator, hash table, and latency tracker for use in eBPF scripts. Would provide:
  - NMI-safe lock-free memory allocator vs per-freelist spin lock with interrupts off,
  - NMI-safe lock-free hash table vs per-bucket locking with interrupts off,
  - Would allow hooking eBPF scripts to perf NMIs triggered on performance counter overflows.
- Re-implement latency tracker online critical path module state-machine as eBPF high-level code (bcc).

Links

LTTng:
http://lttng.org

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

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

TraceCompass:
http://tracecompass.org/

Babeltrace
http://diamon.org/babeltrace

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

Questions ?

?

www.efficios.com

lttng.org

lttng-dev@lists.lttng.org

@lttng_project