# LTTng: from Low-Level Tracing to High-Level Analyses

EfficiOS

mathieu.desnoyers@efficios.com

# Content

- LTTng
- Babeltrace
- Common Trace Format
- LTTng analyses
- Latency tracker
- Histogram generator
- TraceCompass

EfficiOS

# LTTng

- Performs user-space and kernel tracing,
- Kernel tracing performed by out-of-tree module
  - **No kernel patching required**
  - Supports kernel from 2.6.38 to 4.2+
- Enables seemless analysis of correlated kernel and user-space data.
- Flexible and fast.

EfficiOS

# What's new in LTTng ?

- LTTng 2.6 (01/2015)
  - Java Log4j support,
  - Kernel tracer per system call tracing,
  - Kernel tracer system call content (input/output) fetching,
  - Tracing NMI handlers (with Linux 3.17 or better).
  - LTTng MI (Machine Interface),

EfficiOS

# What's new in LTTng ?

- LTTng 2.7 (currently in RC)
  - Persistent memory UST ring buffer
    - pramfs (out of tree), or
    - DAX (Linux 4.0) and pmem driver (upcoming Linux 4.1)
    - Either BIOS does not reset memory on soft reboot, or use kexec(8)
    - Allows recovering user-space traced when system crash with new lttng-crash tool.

EfficiOS

# What's new in LTTng ?

- LTTng 2.7 (currently in RC)
  - LTTng filtering for kernel domain,
  - Per-process user-space and kernel tracing,
    - Select a set of PIDs
  - Wildcards for kernel tracepoints,
  - LTTng modules clock plugin support,
  - LTTng UST clock and getcpu plugin support,
  - LTTng Python logger support.

EfficiOS

# Babeltrace

- Babeltrace 1.x
  - CTF reader
  - Merge CTF traces by timestamp,
  - Supports live LTTng tracing,
  - C, C++, Python APIs.
- Babeltrace 2.0 (approx. 10/2015)
  - Plugin system overhaul,
  - Intermediate Representation,
  - Event filtering.

EfficiOS

# Common Trace Format (CTF)

- Currently working on CTF 2.0
- Goal: transition from own metadata grammar (TSDL) to JSON.
- Will be easier to extend, and easier to parse by alternative CTF reader implementations.
- Specification of CTF 1.8 available at http://diamon.org/ctf

EfficiOS

# LTTng Analyses

- Set of Python scripts providing summarized trace information,
- Each analysis typically classified as:
  - Top N
  - Statistics table (avg., std. dev, min, max)
  - Frequency histogram
- Available at https://github.com/lttng/lttng-analyses

EfficiOS

# Available Analyses

- CPU usage for the whole system
- CPU usage per-process
- Process CPU migration count
- Memory usage per-process (as seen by the kernel)
- Memory usage system-wide (as seen by the kernel)
- I/O usage (syscalls, disk, network)
- I/O operations log (with latency and usage)
- I/O latency statistics (open, read, write, sync operations)
- I/O latency frequency distribution
- Interrupt handler duration statistics (count, min, max, average stdev)
- Interrupt handler duration top
- Interrupt handler duration log
- Interrupt handler duration frequency distribution
- SoftIRQ handler latency statistics
- Syscalls usage statistics

EfficiOS

# LTTng Analyses (live demo)

EfficiOS

# 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

# Usage

```
tracker = latency_tracker_create();


latency_tracker_event_in(tracker, key,
      threshold, timeout, callback);

....

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).

EfficiOS

# Implemented use-cases

- Block layer latency
  - Delay between block request issue and complete
- Wake-up latency
  - Delay between sched_wakeup and sched_switch
- Network latency (prototype)
- IRQ handler latency (prototype)
- System call latency
  - Delay between the entry and exit of a system call
- Offcpu latency
  - How long a process has been scheduled out and why did it get woken up

EfficiOS

# Example: system call latency

- Developed in collaboration with François Doray

```
on syscall_entry:
        latency_tracker_event_in(current_pid);
on syscall_exit:
        latency_tracker_event_out(current_pid);
on sched_switch:
        event =
        latency_tracker_get_event(next_pid);
        if event && ((now – event->start) >
        threshold):
             dump_stack(next_pid);
```

EfficiOS

# System call latency example

**syscall_latency_stack: comm=sync, pid=32224**

| **81136.460929** | **81136.461482** | **81136.467357** | **81136.470176** |
|---|---|---|---|
| schedule | _cond_resched | _cond_resched | schedule |
| schedule_timeout | sync_inodes_sb | mempool_alloc | schedule_timeout |
| wait_for_completion | sync_inodes_one_sb | __split_and_process_bio | wait_for_completion |
| sync_inodes_sb | iterate_supers | dm_request | submit_bio_wait |
| sync_inodes_one_sb | sys_sync | generic_make_request | blkdev_issue_flush |
| iterate_supers | tracesys | submit_bio | ext4_sync_fs |
| sys_sync | | submit_bio_wait | sync_fs_one_sb |
| tracesys | | blkdev_issue_flush | iterate_supers |
| | | ext4_sync_fs | sys_sync |
| | | sync_fs_one_sb | tracesys |

Dynamically change the threshold:
# echo 1000000 > /sys/module/latency_tracker_syscalls/parameters/usec_threshold

EfficiOS

# Off-cpu profiling

```
on sched_switch(prev, next):
    latency_tracker_event_in(prev, cb)
    latency_tracker_event_out(next)

cb():
    dump_stack(pid)

on sched_wakeup(pid):
    event = latency_tracker_get_event(pid)
    if event && ((now – event->start) > threshold):
        dump_stack(current)
```

EfficiOS

# Off-cpu profiling example

```
offcpu_sched_wakeup:
  waker_comm=swapper/3 (0),
  wakee_comm=qemu-system-x86 (7726),
  wakee_offcpu_delay=10000018451,
  waker_stack=
    ttwu_do_wakeup
ttwu_do_activate.constprop.74
    try_to_wake_up
    wake_up_process
    hrtimer_wakeup
    __run_hrtimer
    hrtimer_interrupt
  local_apic_timer_interrupt
    smp_apic_timer_interrupt
    apic_timer_interrupt
```

```
offcpu_sched_switch:
  comm=qemu-system-x86,
  pid=7726,
  delay=10000140896,
  stack=
    schedule
    futex_wait_queue_me
    futex_wait
    do_futex
    SyS_futex
    system_call_fastpath
```

EfficiOS

# Runtime latency distributions

- For system calls, file system, I/O scheduler and block requests
- Show the distribution of requests latencies
- Clearly see in one screen the latencies of all disk I/O at various level
- Available at https://github.com/jdesfossez/latency_tracker
- Video demo (demo-latency_tracker.ogv)

EfficiOS

# Overhead on sysbench oltp (MySQL)

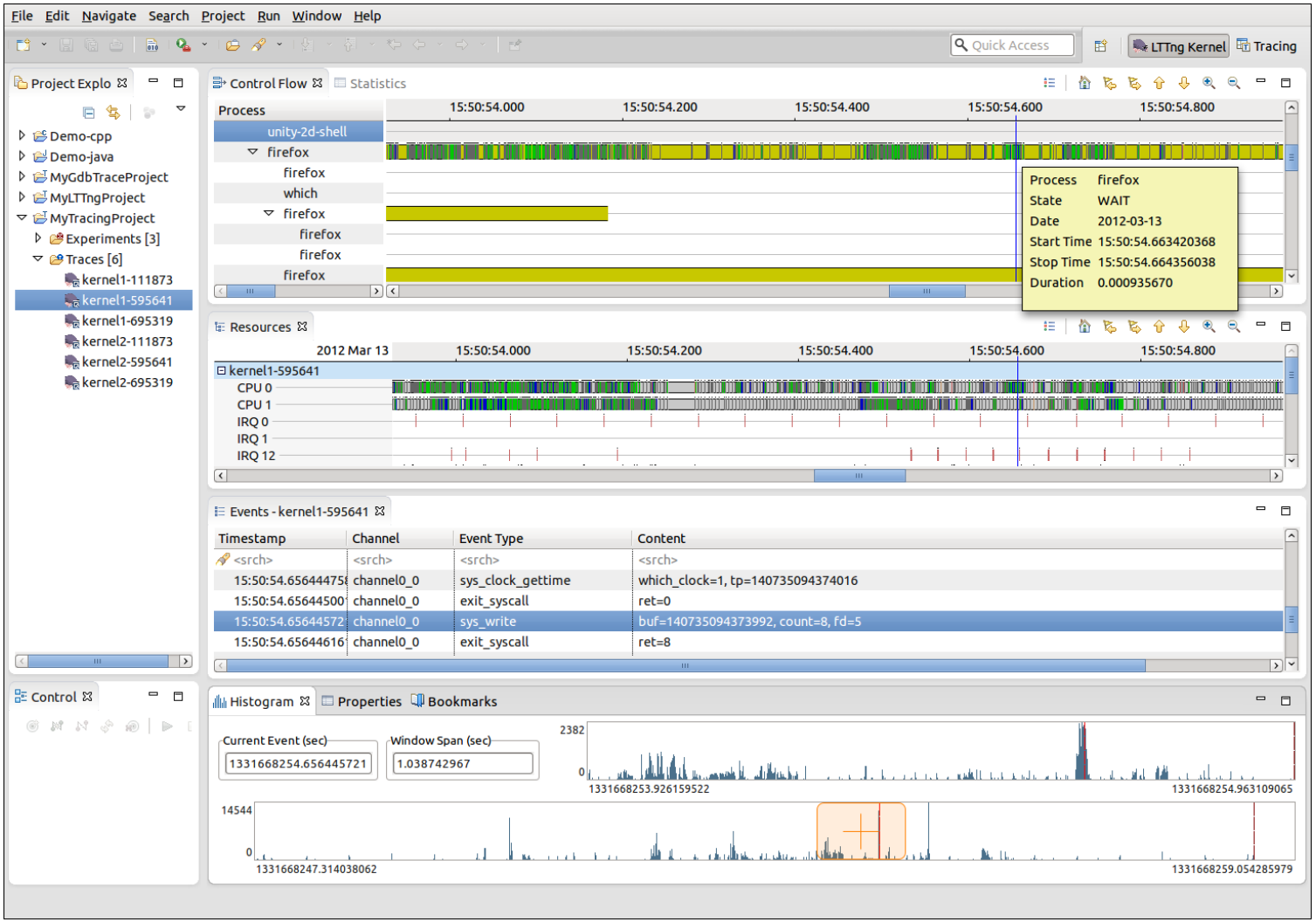
| Test | Average | Overhead |
|---|---|---|
| Baseline | 63.26s | |
| LTTng sched | 63.65s | 0.61% |
| LTTng syscalls | 64.95s | 2.66% |
| **Latency_tracker** | **65.36s** | **3.31%** |
| Latencytop | 66.24s | 4.70% |
| LTTng all | 70.24s | 11% |

EfficiOS

# TraceCompass

- Now available as a standalone application (requires only a Java Virtual Machine)
- Available at http://tracecompass.org
- We are currently working at facilitating workflows involving frequent back-and-forth between LTTng analyses and TraceCompass,
- Can now read Perf traces converted to CTF.

# TraceCompass Screenshot




EfficiOS

# Questions ?

?

EfficiOS

EfficiOS

www.efficios.com

lttng.org

lttng-dev@lists.lttng.org

@lttng_project