

Embedded Linux Conference 2010

Using the LTTng Tracer for System-Wide Performance Analysis and Debugging (Hands-on Tutorial)

Presentation and files at:

<http://www.efficios.com/pub/elc2010>

> Presenter

- Mathieu Desnoyers
- EfficiOS Inc.
 - <http://www.efficios.com>
- Author/Maintainer of
 - LTTng, LTTV, Userspace RCU
- Ph.D. in computer engineering
 - Low-Impact Operating System Tracing

> Plan

- LTTng Installation
- Tracing Strategy
- Trace Analysis (Hands-on Example)
- Questions

> LTTng installation

- <http://lttng.org>
 - LTTng
 - LTT control
 - LTTV
- Documentation
 - LTTng Kernel Tracer Manual
 - LTTng Compatibility List

> Lockless Trace Clock

- dmesg, check for LTT warnings
- Cycle counter is used
- For architectures with non-synchronized cycle counters (e.g. some x86):
 - cpufreq-set -g performance
 - idle=poll (kernel parameter)
- LTTng fully supports power management and frequency scaling on ARM OMAP3
 - funded by Nokia

> Tracing Strategy

- Problem Identification
- Trace Session Setup
- Anchor / Trigger

> Problem identification

- Bug report summary
 - What is going wrong with the system ?
 - What is the system configuration affected ?
 - Hardware
 - Software
 - Optionally: known good / known bad configurations

> Tracing strategy (decision factors)

- Tracing strategy decision factors
 - Reproducible on development setup or only in production ?
 - Tracing overhead the system can tolerate
 - Frequency of problem occurrence
 - Availability of the system
 - Remote/local
 - Controlled by third-party
 - Number of tracing iterations available

> Trace Session Setup

- Identify the tracer setup best suited to solve the problem
 - Producer-consumer tracing
 - Flight recorder tracing
 - Per-channel buffer size
 - Activated instrumentation

> Anchor / Trigger

- Traces are hard to analyze
 - Large volume of information collected
 - Hard to identify relevant information
- Add anchor instrumentation to the system
- Use triggers to stop flight recorder tracing

> Anchor

- Starting point for trace analysis
- Identify surrounding of problem occurrence
- Different types
 - Instrumentation anchors
 - Analysis-generated anchors

> Anchor

- Instrumentation anchors
 - Userland or kernel instrumentation
 - UST (Userspace Tracer)
 - Write to /debugfs/ltt/write_event
 - Add kernel TRACE_EVENT/markers
 - Events generated from user interaction
 - Input subsystem
 - Instrumentation of program error-handling
- Analysis-generated anchors
 - e.g. longest timer interrupt jitter

> Producer-consumer tracing

- Writes trace data to the file system
- Whole trace session duration
- Initial state dump : complete state collected
 - (+) Very accurate state representation
 - (-) Consumes disk or network I/O bandwidth

```
(as root)
ltt-armall
lttctl -C -w /tmp/trace-prod1 trace-prod1
...
lttctl -D trace-prod1
```

> Flight recorder tracing

- Gather trace data in circular ring buffers
- Kept in memory, oldest data overwritten
- Last events available when tracing is stopped
- Per-channel size is configurable
 - (+) Very low system throughput overhead
 - (-) Shorter available backlog
 - (-) System state is less accurate
 - Partially unknown

> Triggers

- Instrumentation with side-effect
 - Start/stop tracing when executed
- Particularly useful for flight recorder mode
 - Produces event backlog that lead to execution of “trigger”
- Kernel API
 - `ltt_trace_start(“name”), ltt_trace_stop(“name”)`
- From user-space
 - `lttctl -s name ; lttctl -p name`

> Userland trigger example

Userland trigger for flight recorder trace

(as root)

ltt-armall

lttctl -c -o channel.all.overwrite=1 -w /tmp/trace-uttrigger1 trace-uttrigger1

lttctl -s trace-uttrigger1

...

(trigger) → lttctl -p trace-uttrigger1

lttctl -d -w /tmp/trace-uttrigger1 trace-uttrigger1

> Kernel trigger example

Kernel trigger

(as root)

ltt-armall

lttctl -c -o channel.all.overwrite=1 -w /tmp/trace-ktrigger1 trace-ktrigger1

lttctl -s trace-ltrigger1

...

(trigger in kernel) → ltt_trace_stop("trace-ktrigger1");

lttctl -d -w /tmp/trace-ktrigger1 trace-ktrigger1

> Trace Analysis (Hands-on Example)

- Identify sources of audio latency
- Scheduler latency
 - `wakeup-latency.c`
 - `write_event` anchor
- With 2.6.33.2 kernel

maximum latency: 44614.1 μ s
average latency: 3851.4 μ s
missed timer events: 0

> Find anchors

Filter expression:

<field> <comparator> <value>

<field> <comparator> <value> <logical> <expression>

(comparators: =, !=, <, <=, >, >=)

(logical: &, |, !, ^)

(see lttv-gui "filter" plugin for expression examples)

Find all events in channel "userspace"

(written through write_event):

```
% lttv -m textDump -e "channel.name=userspace" \  
-t /tmp/trace-prod1
```

Filter by channel.event (list with "find /debugfs/ltt/markers") :

```
% lttv -m textDump -e "event.name=kernel.sched_schedule" \  
-t /tmp/trace-prod1
```

Filter by PID :

```
% lttv -m textDump -e "state.pid=1" -t /tmp/trace-prod1
```

```
event.name  
event.subname  
event.category  
event.time  
event.tsc  
event.target_pid  
channel.name  
trace.name  
state.process_name  
state.thread_brand  
state.pid  
state.ppid  
state.creation_time  
state.insertion_time  
state.execution_mode  
state.execution_submode  
state.process_status  
state.cpu
```

Filter fields

> Finding "wakeup-latency" Anchor

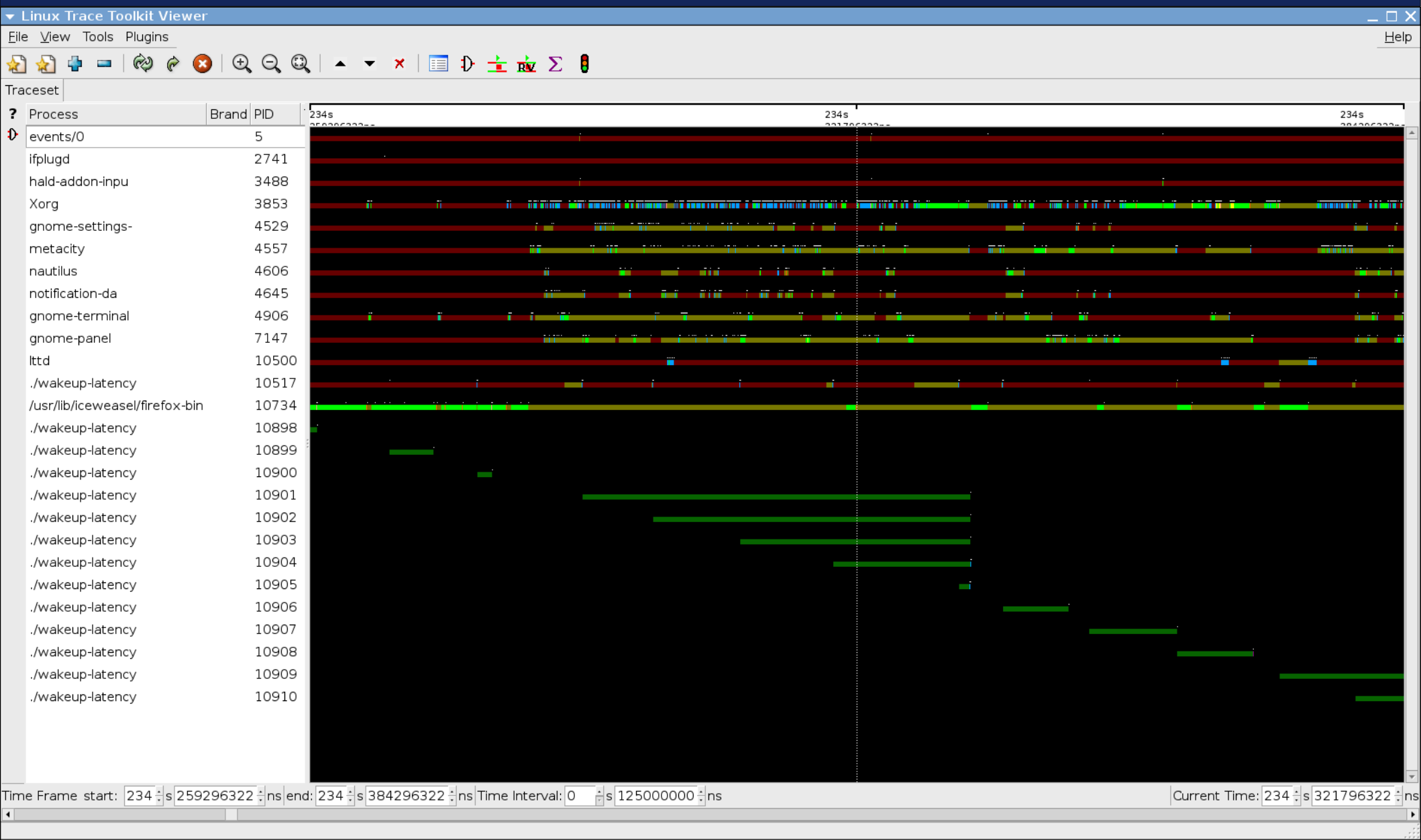
```
% lttv -m textDump -e "channel.name=userspace" -t /tmp/trace-cfs-2/
```

```
...
```

```
userspace.event: 234.334734395 (/tmp/trace-cfs-2/userspace_0),  
    10905, 10516, ./wakeup-latency, , 10517, 0x0,  
    SYSCALL { string = "late by: 44614.1  $\mu$ s" }
```

```
...
```

> Viewing execution patterns



> Instrumentation addition iterations

Instrumentation example (with markers) :

Index: linux-2.6-lttng.git/kernel/sched_fair.c

=====

=====

--- linux-2.6-lttng.git.orig/kernel/sched_fair.c 2010-04-10

11:47:04.000000000 -0400

+++ linux-2.6-lttng.git/kernel/sched_fair.c 2010-04-10

11:50:59.000000000 -0400

....

@@ -764,12 +768,18 @@
 thresh >>= 1;

vruntime -= thresh;

+ trace_mark(test, cfs_place_sleeper,
+ "pid %d thresh %lu vruntime %llu",
+ task_of(se)->pid, thresh, vruntime);
}

> Digging into Xorg scheduling

Linux Trace Toolkit Viewer

File View Tools Plugins Help

Traceset

Event Description

kernel.send_signal: 234.321790689 (/tmp/trace-cfs-2/kernel_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 3853, signal = 29 }

test.wakeup_state: 234.321793993 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { caller = 0xC1073781 }

test.cfs_enqueue: 234.321795160 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 3853, vruntime = 37793709218 }

test.cfs_place: 234.321796322 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 3853, orig_vruntime = 37793709218, vruntime = 37793709218 }

test.cfs_place_sleeper: 234.321797137 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 3853, thresh = 2500000, vruntime = 37791209218 }

test.cfs_place_end: 234.321797808 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 3853, vruntime = 37793709218 }

test.cfs_int_enqueue: 234.321798430 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 3853, vruntime = 37793709218 }

test.cfs_enqueue: 234.321799062 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 0, vruntime = 55069985566 }

test.cfs_place: 234.321799834 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 0, orig_vruntime = 55069985566, vruntime = 55073363633 }

test.cfs_place_sleeper: 234.321800598 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 0, thresh = 2500000, vruntime = 55070863633 }

test.cfs_place_end: 234.321801261 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 0, vruntime = 55070863633 }

test.cfs_int_enqueue: 234.321801815 (/tmp/trace-cfs-2/test_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 0, vruntime = 55070863633 }

kernel.sched_try_wakeup: 234.321802314 (/tmp/trace-cfs-2/kernel_0), 10734, 10734, /usr/lib/iceweasel/firefox-bin, , 10733, 0x0, IRQ { pid = 3853, cpu_id = 0, state = 1 }

Process Brand PID

Xorg 3853

metacity 4557

gnome-panel 7147

/usr/lib/iceweasel/firefox-bin 10734

Time Frame start: 234 s 320819760 ns end: 234 s 322772885 ns Time Interval: 0 s 1953125 ns Current Time: 234 s 321800598 ns

> Looking at alsa

- Adding instrumentation of buffer underrun into aplay as anchor
 - `write_event`

```
./aplay --period-size=128 --buffer-size=512 -D hw:0 test.wav
```

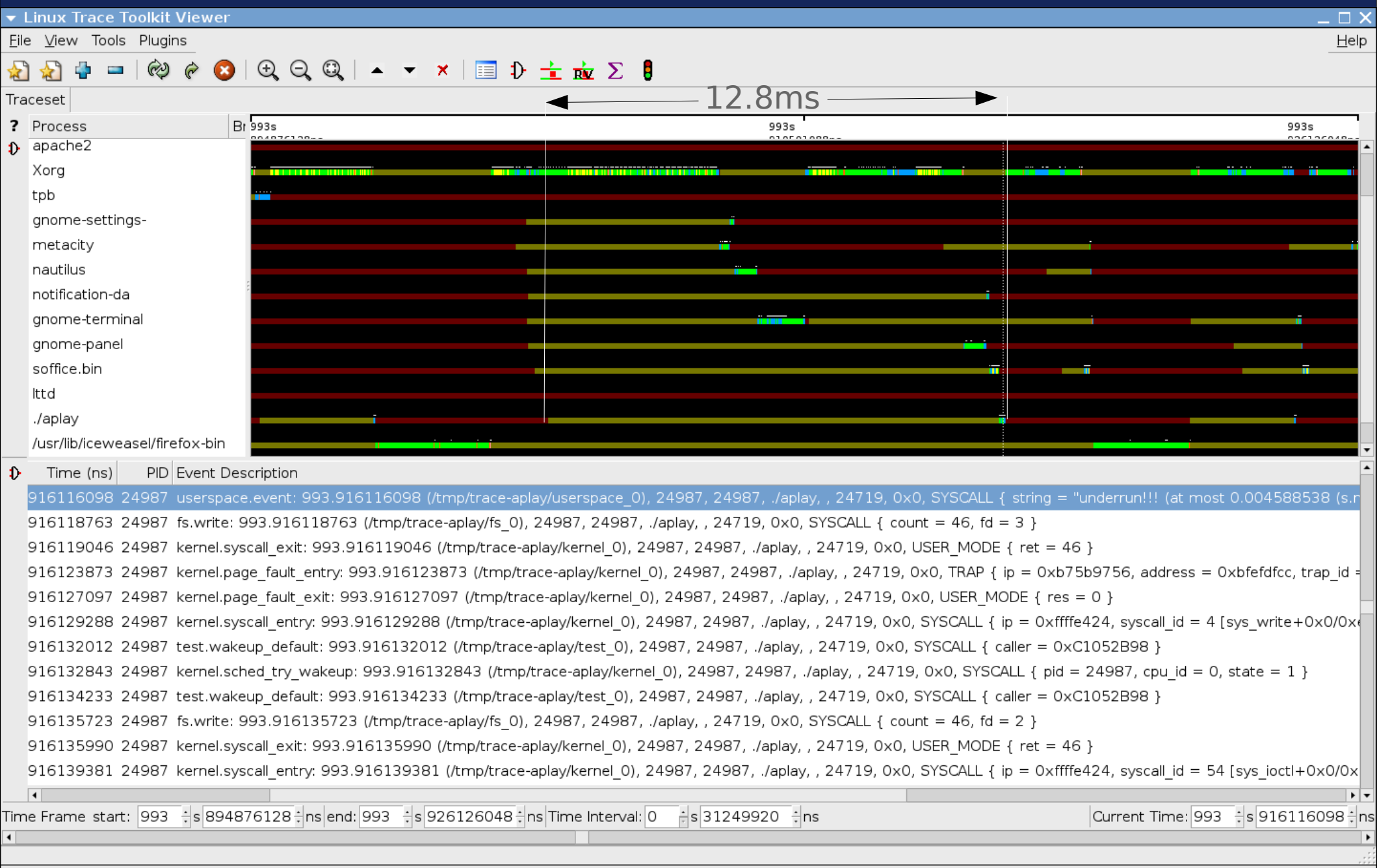
Buffer:

512 samples / 48000 samples/s = 10.6ms

Period:

128 samples / 48000 samples/s = 2.6ms

> View of a buffer underrun



> sched_fair vruntime analysis

Linux Trace Toolkit Viewer

File View Tools Plugins

18.62 minutes vruntime offset !

Traceset

Event Description

```
test.sched_switch: 993.901654560 (/tmp/trace-aplay/test_0), 3768, 3768, Xorg, , 3755, 0x0, USER_MODE { next = 3768, vruntime = 95351578124 }
test.sched_switch: 993.908105835 (/tmp/trace-aplay/test_0), 19999, 19999, metacity, , 19812, 0x0, SYSCALL { next = 19999, vruntime = 1213021553424 }
test.sched_switch: 993.908407361 (/tmp/trace-aplay/test_0), 19982, 19982, gnome-settings-, , 1, 0x0, SYSCALL { next = 19982, vruntime = 1213021553424 }
test.sched_switch: 993.908534017 (/tmp/trace-aplay/test_0), 20057, 20057, nautilus, , 19812, 0x0, SYSCALL { next = 20057, vruntime = 1213021553424 }
test.sched_switch: 993.909177730 (/tmp/trace-aplay/test_0), 20314, 20314, gnome-terminal, , 1, 0x0, SYSCALL { next = 20314, vruntime = 1213021553424 }
test.sched_switch: 993.910533596 (/tmp/trace-aplay/test_0), 3768, 3768, Xorg, , 3755, 0x0, SYSCALL { next = 3768, vruntime = 95358020608 }
test.sched_switch: 993.914999738 (/tmp/trace-aplay/test_0), 20769, 20769, gnome-panel, , 19812, 0x0, SYSCALL { next = 20769, vruntime = 1213021553424 }
test.sched_switch: 993.915641107 (/tmp/trace-aplay/test_0), 20097, 20097, notification-da, , 1, 0x0, SYSCALL { next = 20097, vruntime = 1213021553424 }
```

Process

nautilus

notification-da

gnome-terminal

gnome-panel

soffice.bin

./aplay

/usr/lib/iceweasel/firefox-bin

/usr/lib/iceweasel/firefox-bin

Event Description

```
kernel_irq_entry: 993.903268234 (/tmp/trace-aplay/kernel_0), 3768, 3768, Xorg, , 3755, 0x0, IRQ { ip = 3075750182, handler = 0xfabbb490, irq_id = 22, kernel_mode = 0 }
test.wakeup_default: 993.903279480 (/tmp/trace-aplay/test_0), 3768, 3768, Xorg, , 3755, 0x0, IRQ { caller = 0xc1052b98 }
test.cfs_enqueue: 993.903280844 (/tmp/trace-aplay/test_0), 3768, 3768, Xorg, , 3755, 0x0, IRQ { pid = 24987, vruntime = 1213018367508 }
test.cfs_place: 993.903281949 (/tmp/trace-aplay/test_0), 3768, 3768, Xorg, , 3755, 0x0, IRQ { pid = 24987, orig_vruntime = 1213018367508, vruntime = 1213024053424 }
test.cfs_place_sleeper: 993.903282859 (/tmp/trace-aplay/test_0), 3768, 3768, Xorg, , 3755, 0x0, IRQ { pid = 24987, thresh = 2500000, vruntime = 1213021553424 }
test.cfs_place_end: 993.903283609 (/tmp/trace-aplay/test_0), 3768, 3768, Xorg, , 3755, 0x0, IRQ { pid = 24987, vruntime = 1213021553424 }
test.cfs_int_enqueue: 993.903284202 (/tmp/trace-aplay/test_0), 3768, 3768, Xorg, , 3755, 0x0, IRQ { pid = 24987, vruntime = 1213021553424 }
```

Time Frame start: 993 s 902754538 ns end: 993 s 903731098 ns Time Interval: 0 s 976560 ns Current Time: 993 s 903281155 ns

> Questions ?



*Effici*OS

- <http://www.efficios.com>
- LTTng Information
 - <http://lttng.org>
 - ltt-dev@lists.casi.polymtl.ca

