Enterprise End User Summit 2012

LTTng 2.0 : Kernel and Application tracing for the Enterprise.

E-mail:
mathieu.desnoyers@efficios.com

> Presenter

- Mathieu Desnoyers
- EfficiOS Inc.
  - http://www.efficios.com
- Author/Maintainer of
  - LTTng, LTTng-UST, Babeltrace, LTTV, Userspace RCU

> Content

- Tracing overview
- LTTng 2.0 features
- LTTng 2.0 UI examples
- Benchmarks
- Trace viewer & analysis tools

> Benefits of low-impact tracing in a multi-core world

- Understanding interaction between
  - Kernel
  - Libraries
  - Applications
  - Virtual Machines
- Debugging
- Performance tuning
- Monitoring

> Tracing use-cases

- Telecom
  - Operator, engineer tracing systems concurrently with different instrumentation sets.
  - In development and maintenance phases.
- High-availability, high-throughput servers
  - Development and maintenance: ensure high performance, low-latency in production.
- Embedded
  - System development, maintenance of deployed systems.

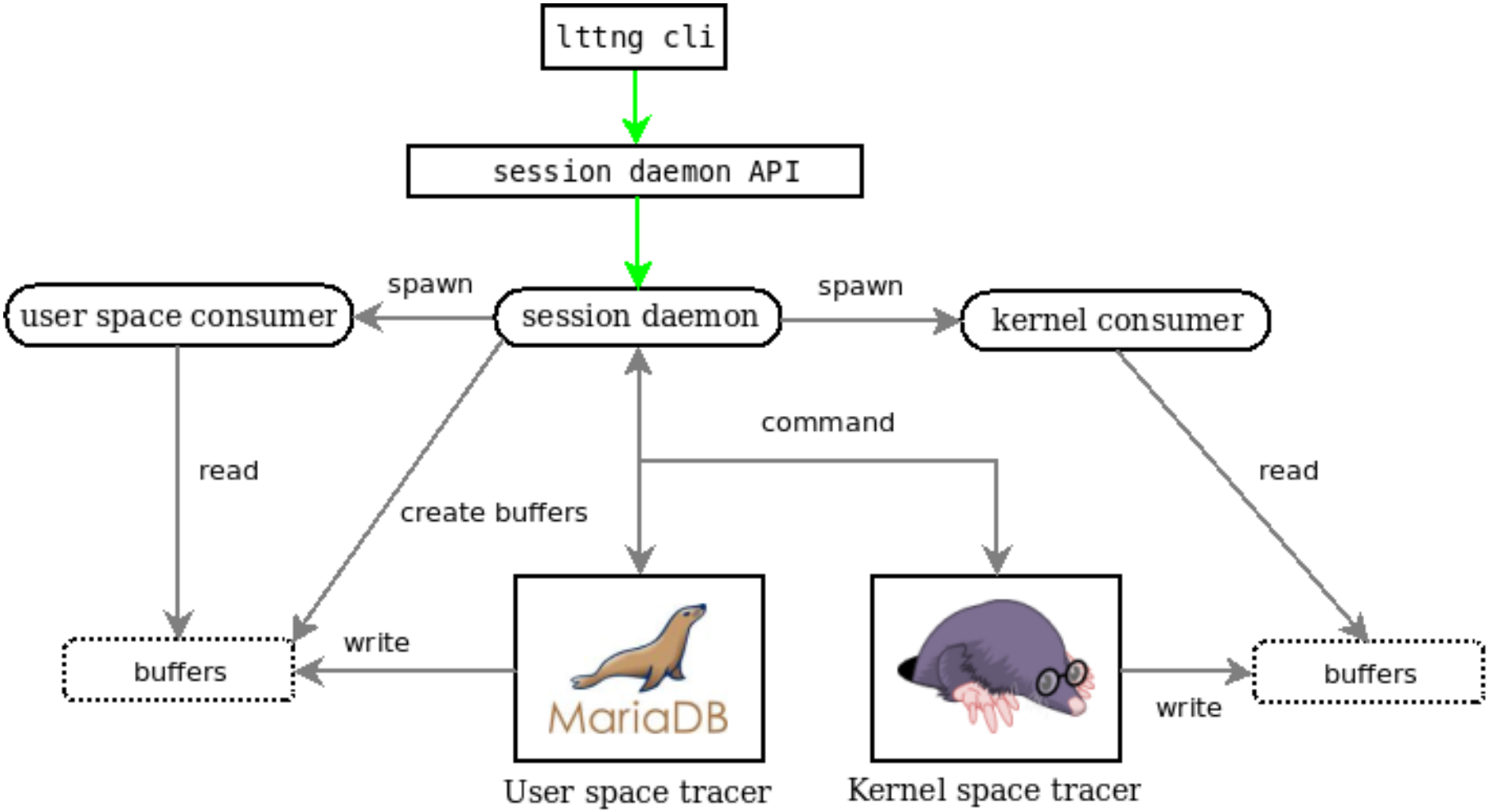
> Tracers timeline

- Tracing commonly used for embedded real-time systems (small traces) and for early high performance SMP servers (SGI, IBM). Then comes Linux...
- 1999: LTT
- 2005: LTTng
- 2005: Dtrace (Solaris/xBSD)
- 2005: SystemTap (RedHat)
- 2008: Ftrace
- 2009: Perf
- 2012: LTTng 2.0

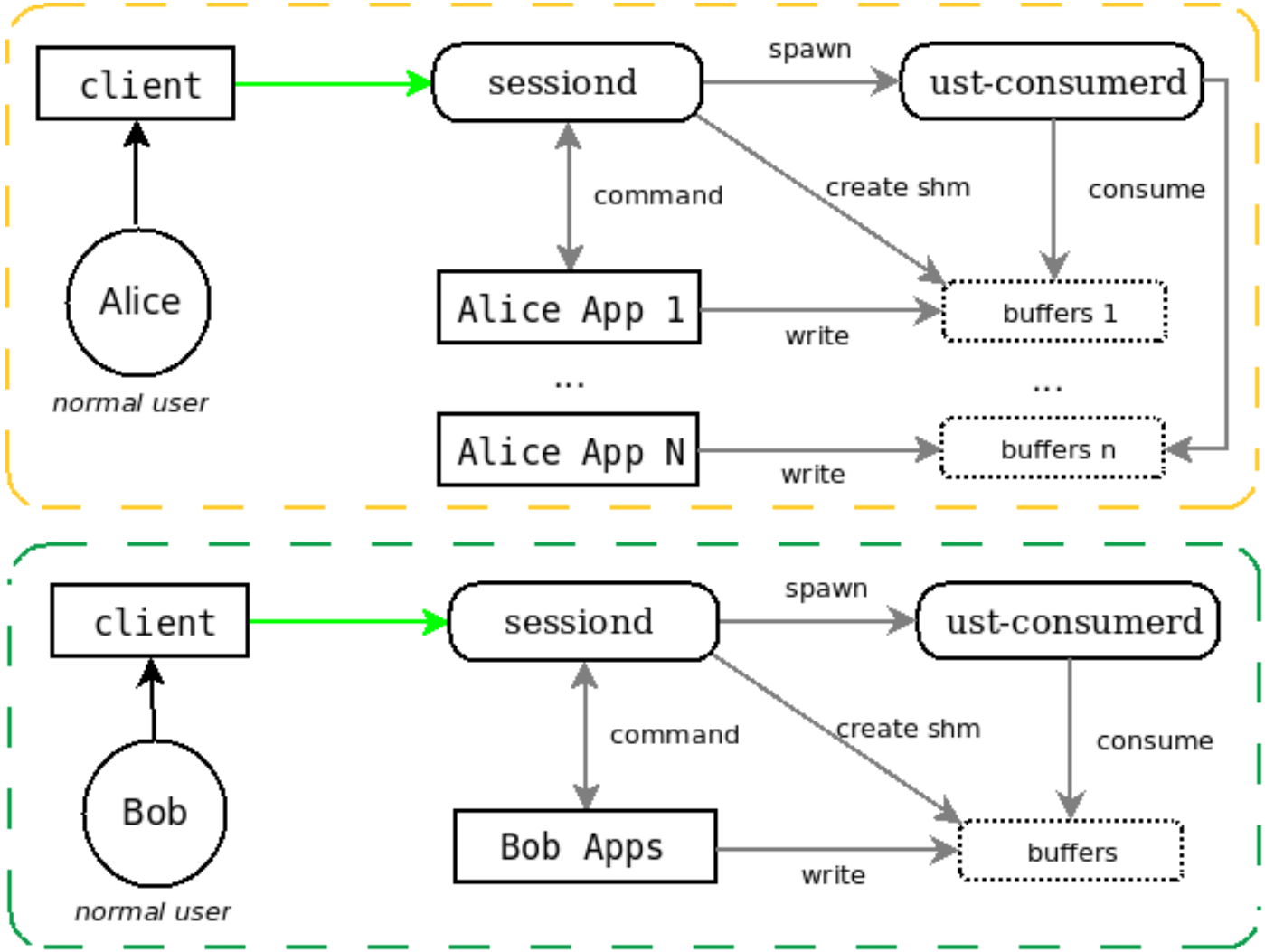
> Why do we need a LTTng 2.0 ?

- Need more flexible trace data layout format
  - Introduce Common Trace Format (CTF)
- Introduction of user-space tracing (UST)
  - Leverage common control infrastructure for kernel and user-space tracing
  - Simplification of the kernel-level infrastructure
- Need more flexible ring buffer
  - Snapshot, mmap and splice, global and per-cpu, kernel and user-space, configurable crash dump support.

> Combined kernel and user-space tracing



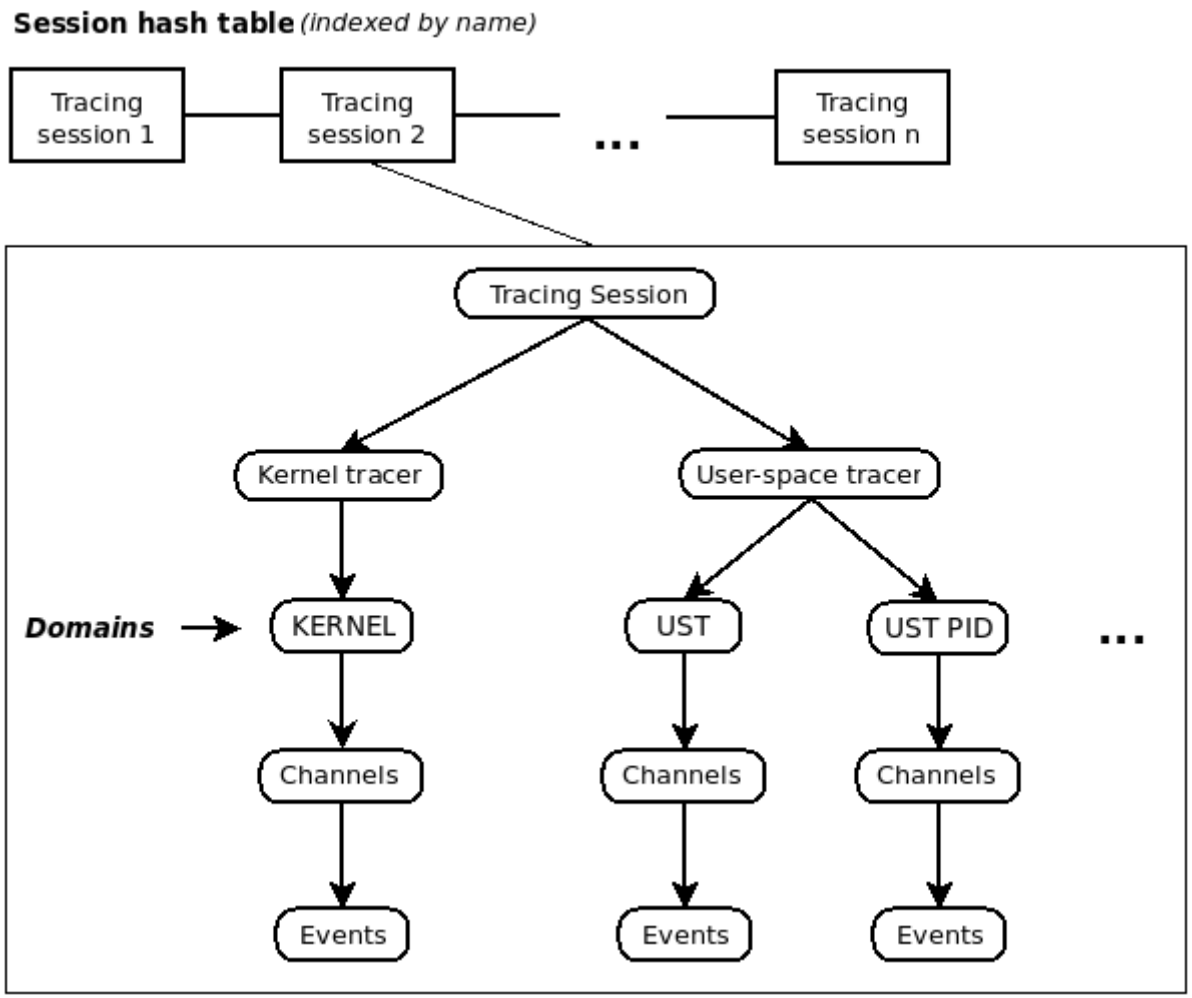
> Multi-user concurrent sessions



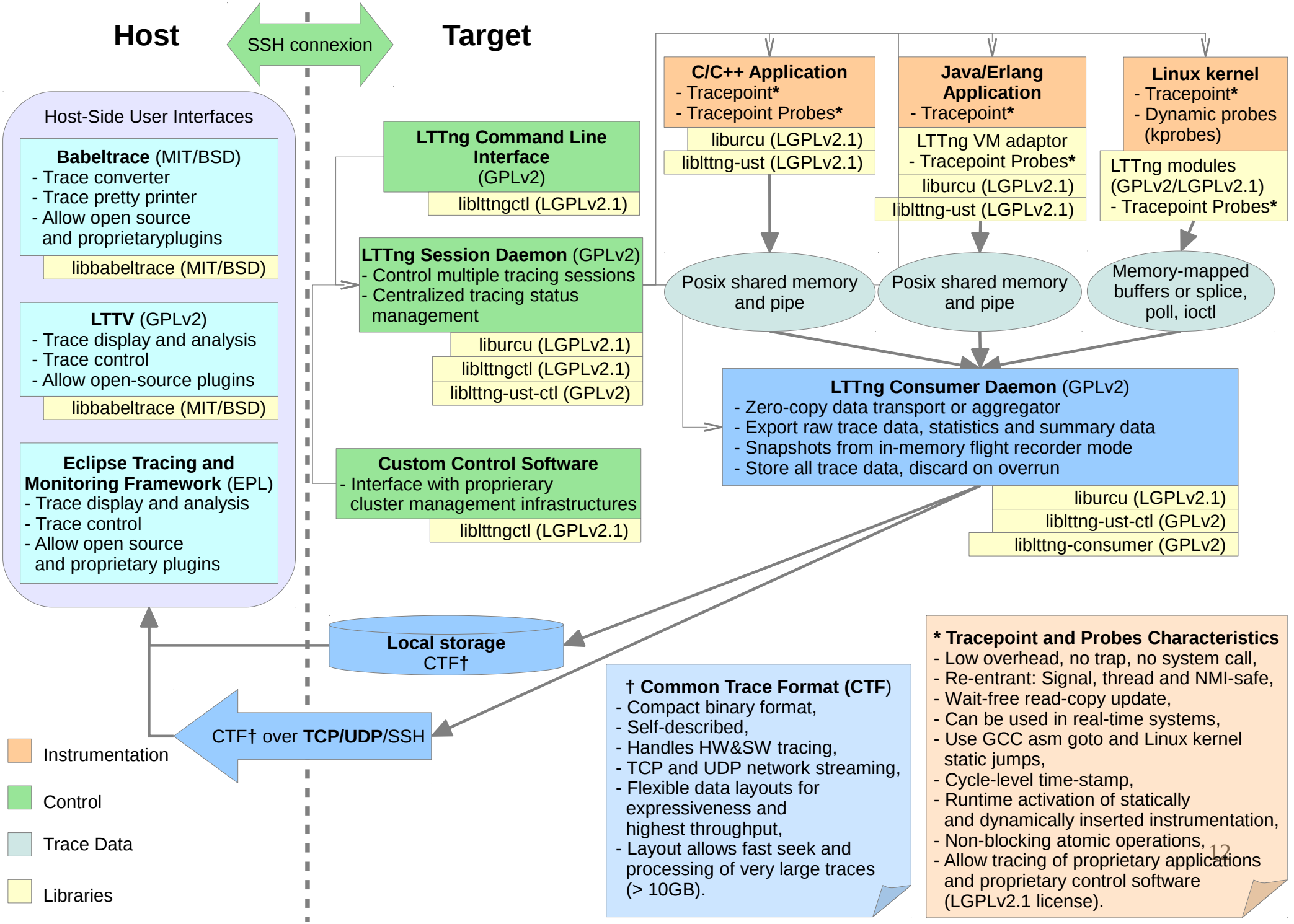
> LTTng 2.0 Tracing Session

- Multiple domains:
  - Kernel, User-space
  - Eventually: Hypervisor, multiple hosts
- Controlled through same UI/API:
  - lttng -k ...
  - lttng -u …
- Correlation across domains (common time-line)
- Viewed by pointing trace viewer to the top-level trace collection directory

> Session, domain, channel and event



LTTng 2.0 Low-Overhead Tracing Architecture



> LTTng 2.0 Kernel Tracer

- Build against a vanilla or distribution kernel, without need for additional patches,
- Tracepoints, System calls, Function tracer, Perf CPU Performance Monitoring Unit (PMU) counters, kprobes, and kretprobes support,
- Supports multiple tracing sessions,
- Flight recorder mode, snapshots, supported at the tracer level, not supported by lttng-tools 2.0 yet (coming in 2.1).

> LTTng 2.0 Kernel Tracer

- Supports dynamically selectable “context” information to augment event payload
  - Any Perf Performance Monitoring Unit counter
  - PID, PPID, TID, process name, VPID, VTID, …
  - Dynamic Priority, nice value

> LTTng-UST 2.0 User-space Tracer Features

- TRACEPOINT_EVENT() API for application/library static instrumentation with sdt.h gdb/systemtap integration.
- Per-user tracing.
- System-wide tracing.
  - “tracing” group: no need to be root to perform system-wide tracing.

> TRACEPOINT_EVENT

In header:

```
TRACEPOINT_EVENT(ust_tests_hello, tptest,
        TP_ARGS(int, anint, long *, values,
                char *, text, size_t, textlen,
                double, doublearg, float, floatarg),
        TP_FIELDS(
                ctf_integer(int, intfield, anint)
                ctf_integer_hex(int, intfield2, anint)
                ctf_array(long, arrfield1, values, 3)
                ctf_sequence(char, seqfield1, text,
                        size_t, textlen)
                ctf_string(stringfield, text)
                ctf_float(float, floatfield, floatarg)
                ctf_float(double, doublefield, doublearg)
        )
)
```

Tracepoint name convention

> User-level Tracepoint

Name convention

< [com_company_]project[_component] >, < event >

Where "company" is the name of the company,
"project" is the name of the project,
"component" is the name of the project component (which may include several levels of sub-components, e.g. ...component_subcomponent_...) where the tracepoint is located (optional),
"event" is the name of the tracepoint event.

Tracepoint invocation within the code:

```
void fct(void)
{
    tracepoint(ust_tests_hello, tptest, i, values,
                text, strlen(text), dbl, flt);
}
```

> LTTng-UST 2.0 Buffering

- Port of the lib ring buffer to user-space.
- Supports buffering between processes through POSIX shared memory maps.
- Fast-paths stay in user-space (no system call).
- Wake-up though pipes.
- Buffers per process (for security), shared with consumer. Faster/lower memory consumption per-user global buffers feature planned for 2.1.

> LTTng Tracing Session Daemon

- Central (system-wide) and per-user instances.
- Controls
  - LTTng kernel tracer
  - LTTng-UST application/library tracer
  - Right management by UNIX socket file access rights.
  - System-wide tracing controlled by tracing group.
  - File descriptors passed through UNIX sockets
- Presents a unified notion of system-wide tracing session, with multiple “domains”.

> LTTng UI examples

```
lttng list -k                          # list available kernel tracepoints
lttng create mysession                 # create session “mysession”
lttng enable-event -k -a               # enable all syscalls/tracepoints
lttng enable-event -k --syscall -a # trace system calls
lttng enable-event sched_switch,sched_wakeup -k
lttng enable-event aname -k --probe symbol+0x3
lttng enable-event aname -k --function <symbol_name>
lttng add-context -k -e sched_switch -t pid        # add PID context
lttng add-context -k -e sched_switch -t perf:cpu-cycles
lttng start                            # start tracing
...
lttng stop                                 # stop tracing
lttng destroy                              # teardown session
# text output
babeltrace -n $HOME/lttng-traces/mysession-<date>-<time>
```

> LTTng 2.0 high-speed “strace”

lttng enable-event --syscall -a

compudj@squeeze-amd64: ~

```
lp
name = sys_brk, stream.packet.context = { cpu_id = 1 }, event.fields = { brk = 28622848 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 28622848 }
name = sys_read, stream.packet.context = { cpu_id = 1 }, event.fields = { fd = 3, buf = 0x1B48008, count = 9645 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 9645 }
name = sys_close, stream.packet.context = { cpu_id = 1 }, event.fields = { fd = 3 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 0 }
name = sys_open, stream.packet.context = { cpu_id = 1 }, event.fields = { filename = "/root/.bash_history", flags = 513, mode
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 3 }
name = sys_write, stream.packet.context = { cpu_id = 1 }, event.fields = { fd = 3, buf = 0x1B48081, count = 9524 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 9524 }
name = sys_close, stream.packet.context = { cpu_id = 1 }, event.fields = { fd = 3 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 0 }
name = sys_rt_sigprocmask, stream.packet.context = { cpu_id = 1 }, event.fields = { how = 0, nset = 0x7FFF28A2A040, oset = 0x
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 0 }
name = sys_ioctl, stream.packet.context = { cpu_id = 1 }, event.fields = { fd = 255, cmd = 21520, arg = 140733875134380 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 0 }
name = sys_rt_sigprocmask, stream.packet.context = { cpu_id = 1 }, event.fields = { how = 2, nset = 0x7FFF28A29FC0, oset = 0x
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 0 }
name = sys_setpgid, stream.packet.context = { cpu_id = 1 }, event.fields = { pid = 0, pgid = 4235 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 0 }
name = sys_exit_group, stream.packet.context = { cpu_id = 1 }, event.fields = { error_code = 0 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 1 }
name = sys_gettimeofday, stream.packet.context = { cpu_id = 1 }, event.fields = { tv = 0x7FFF0E61DC10, tz = 0x0 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 0 }
```

> Common Trace Format

- Trace format specification
  - Funded by
    - Linux Foundation CE Linux Forum and Ericsson
  - In collaboration with Multi-Core Association Tool Infrastructure Workgroup
    - Freescale, Mentor Graphics, IBM, IMEC, National Instruments, Nokia Siemens Networks, Samsung, Texas Instruments, Tilera, Wind River, University of Houston, Polytechnique Montréal, University of Utah.
  - Gathered feedback from Linux kernel developers and SystemTAP communities.

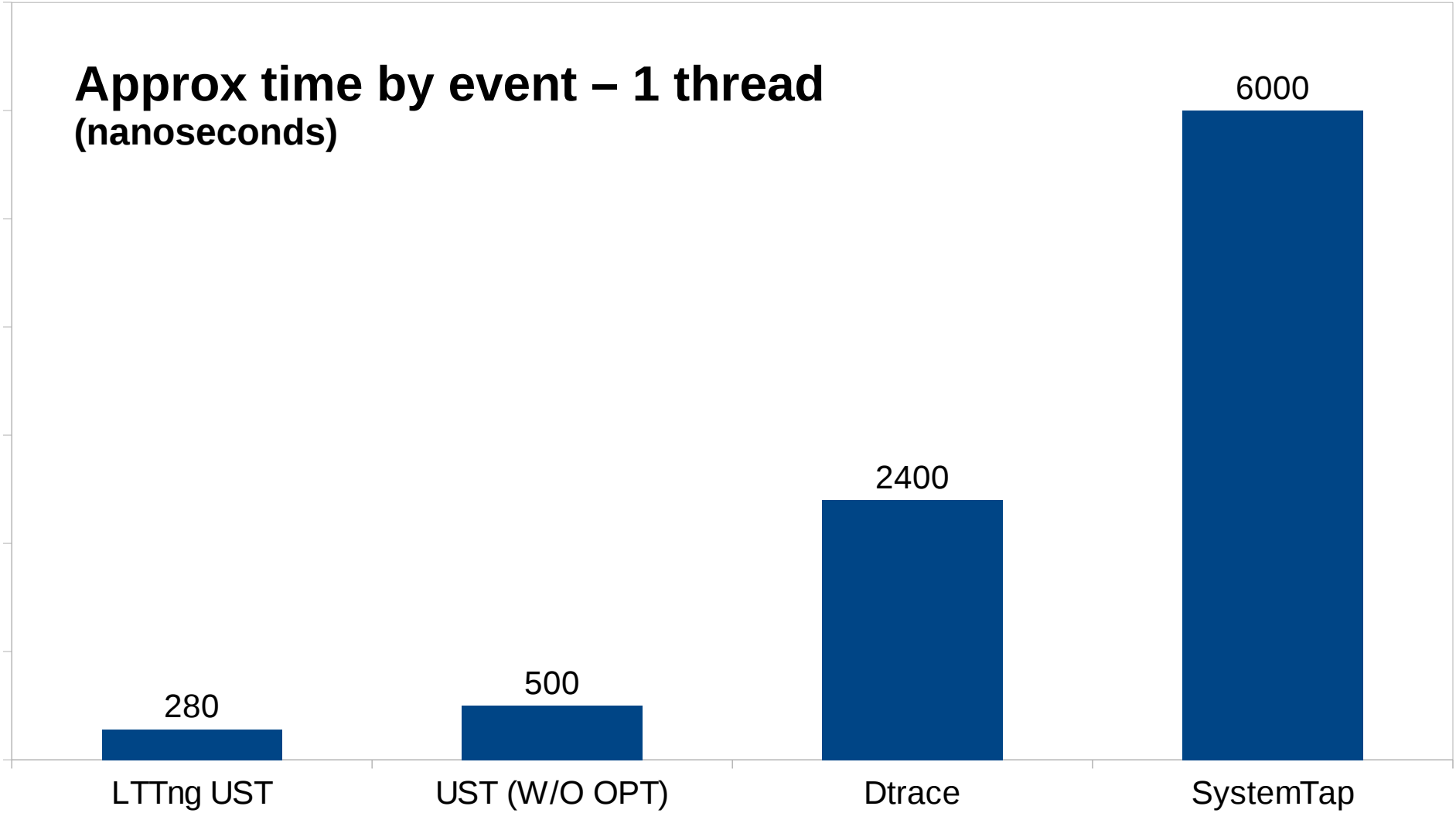
> Common Trace Format

- Targets system-wide and multi-system trace representation in a common format, for integrated analysis:
  - Software traces
    - Across multiple CPUs
    - Across the software stack (Hypervisor, kernel, library, applications)
  - Hardware traces
    - DSPs, device-specific tracing components.
    - GPUs.

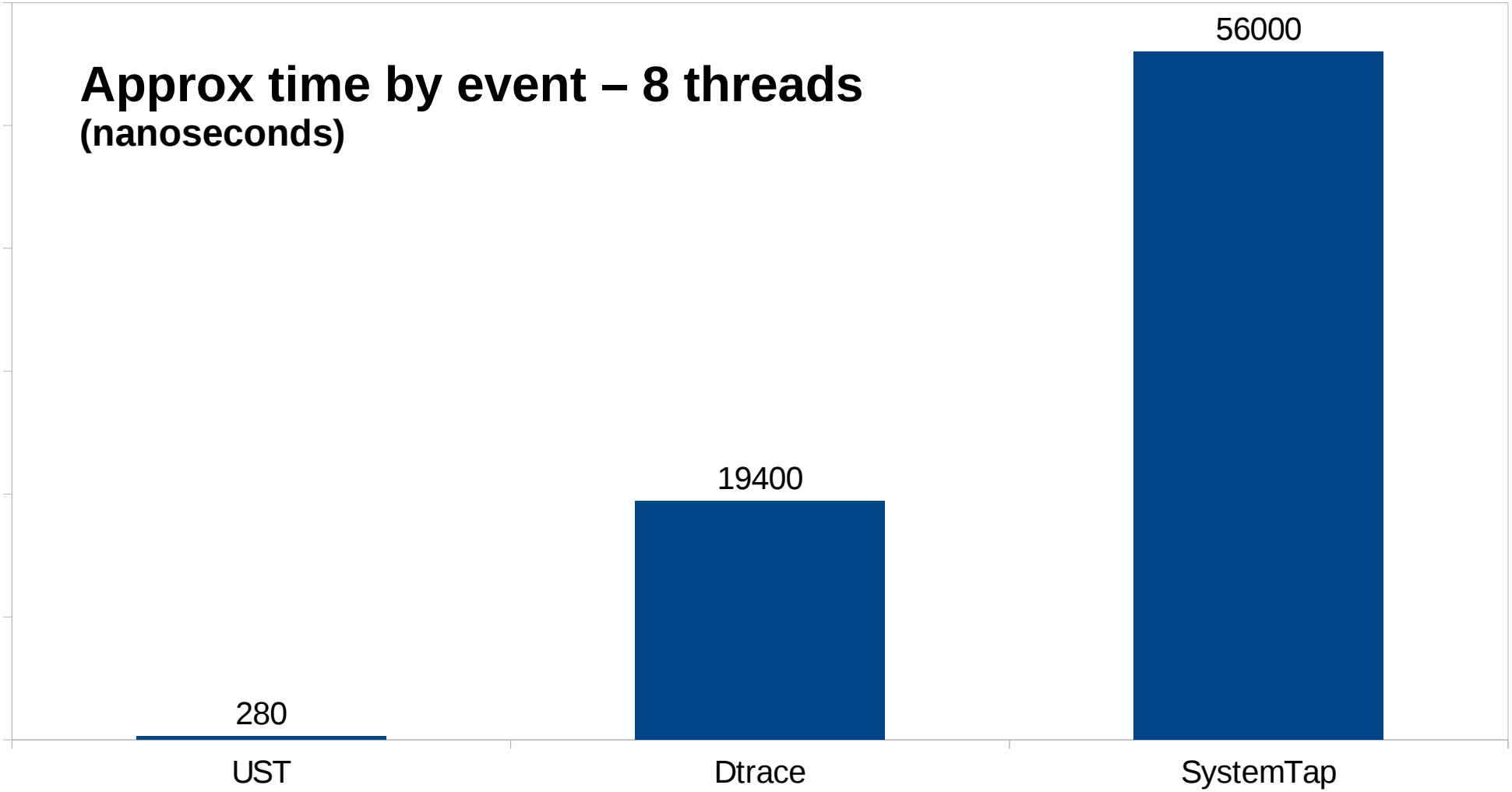
> Common Trace Format

- Babeltrace
  - Reference implementation trace conversion tool and read/seek API for trace collections.
  - Initially converts
    - From CTF to text
    - From dmesg text log to CTF
- LTTng kernel 2.0 and LTTng-UST 2.0
  - Native CTF producer reference implementation.
- Spec. available at: http://www.efficios.com/ctf

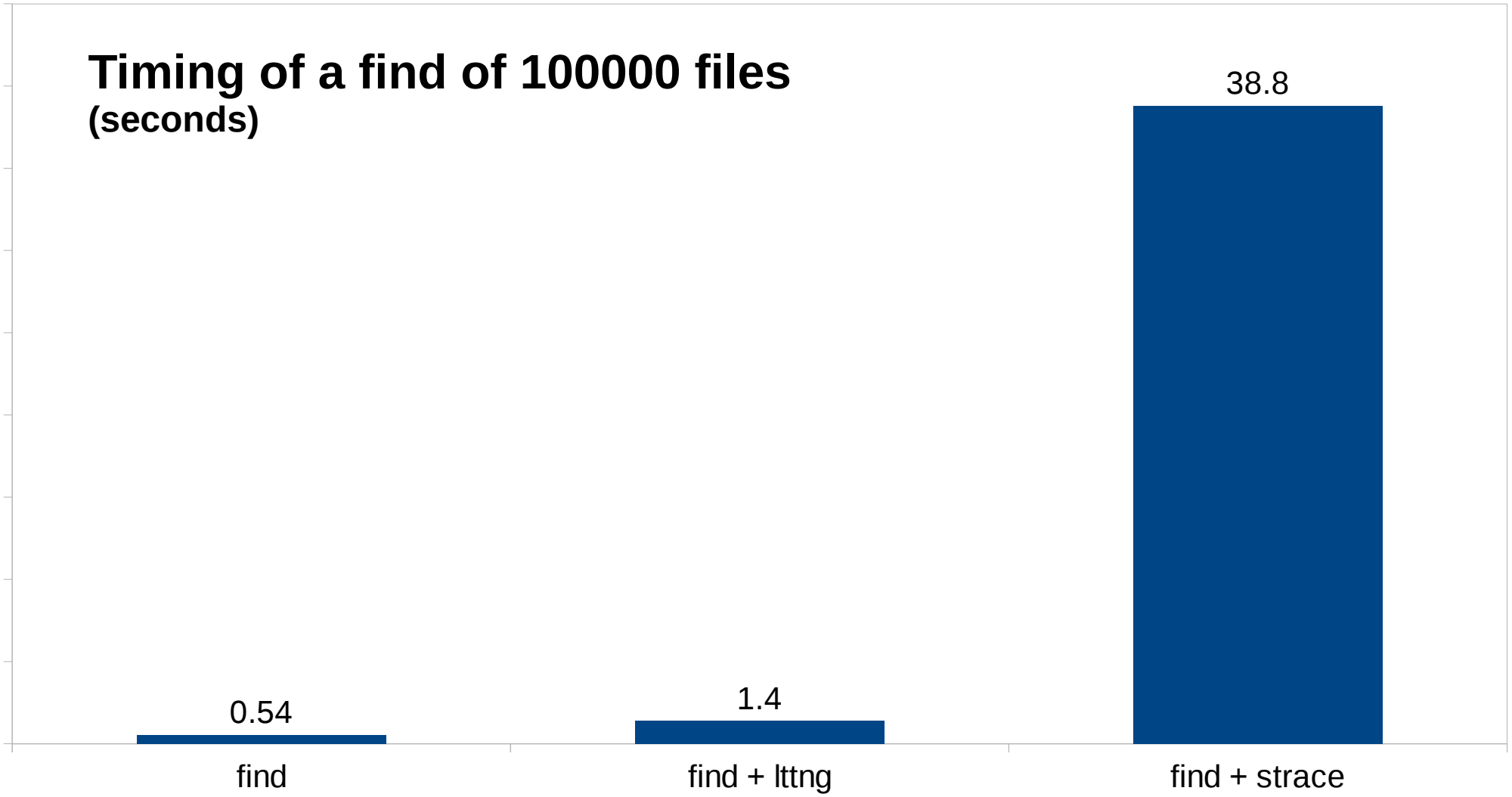
> Userspace Tracing Benchmark



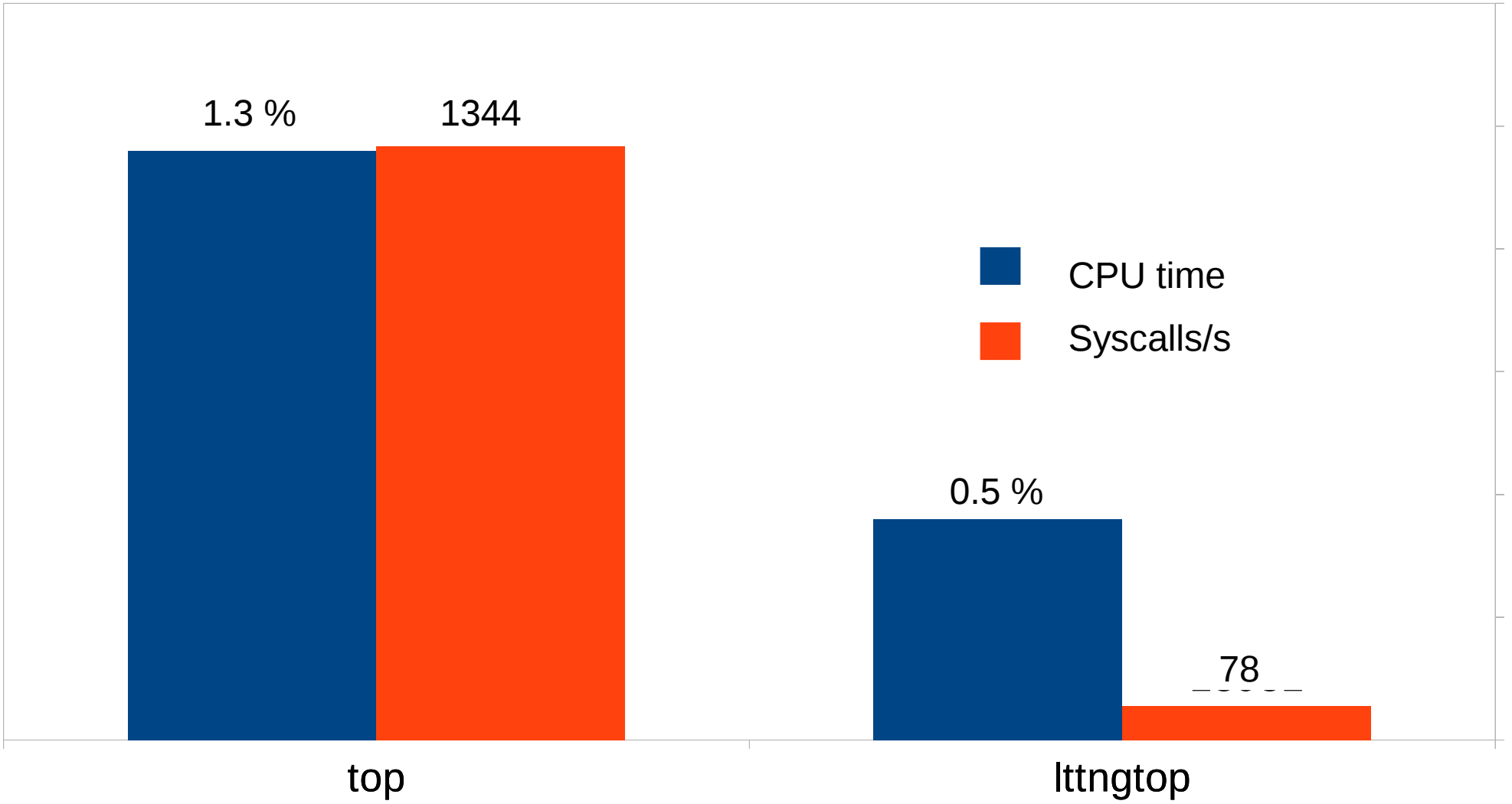
> Userspace Tracing Benchmark



> Strace vs LTTng Tracing



> Top vs LTTngTop



> Distributions / Integration

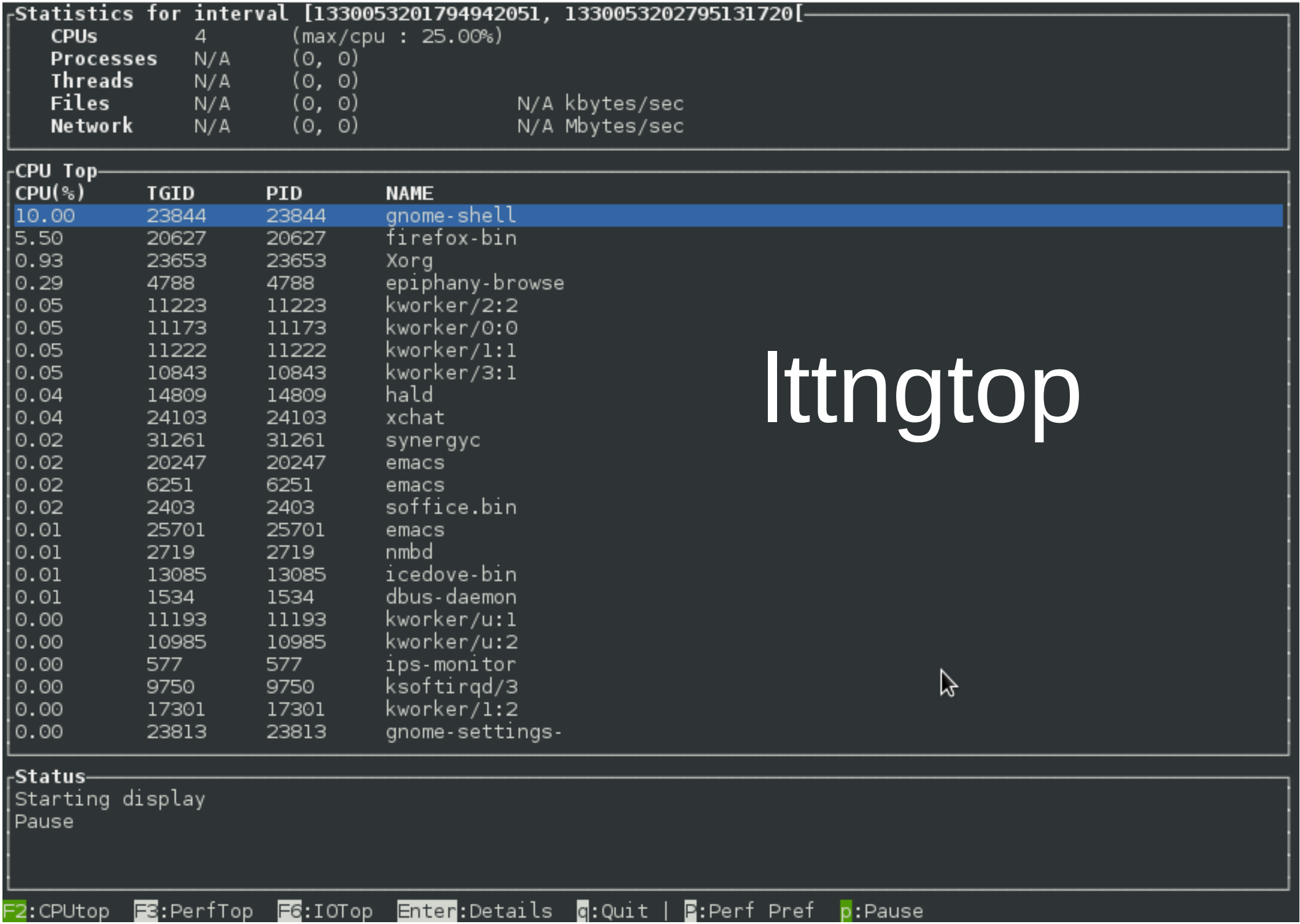
- LTTng 0.x
  - Wind River Linux, Montavista, STlinux, Linaro, Yocto, Mentor Embedded Linux, ELinOS.
- LTTng 2.0
  - Ubuntu 12.04 LTS
  - Debian
  - Novell SuSE Enterprise RT Linux
  - Linux Foundation LTSI
  - Fedora / RedHat : process ongoing

> Collaborations

- Financial support: CAE, DRDC, Ericsson, Google, Opal-RT, Revolution Linux, with matching contributions from CRIAQ, NSERC, Prompt.
- Maintainer and main developer: Mathieu Desnoyers, EfficiOS
- Integrating and redistributing LTTng: Wind River, MontaVista, Linaro, LTSI, Debian, Ubuntu, Suse, Fedora/Red Hat (pending).
- Interfacing: GDB tracepoints can interoperate with LTTng UST tracepoints, The Multi Core Association is defining a Common Trace Format (CTF), for which LTTng 2.0 is a reference implementation.
- Code contributions: over 70 individuals from more than 20 companies including Ericsson, Google, IBM, Red Hat...

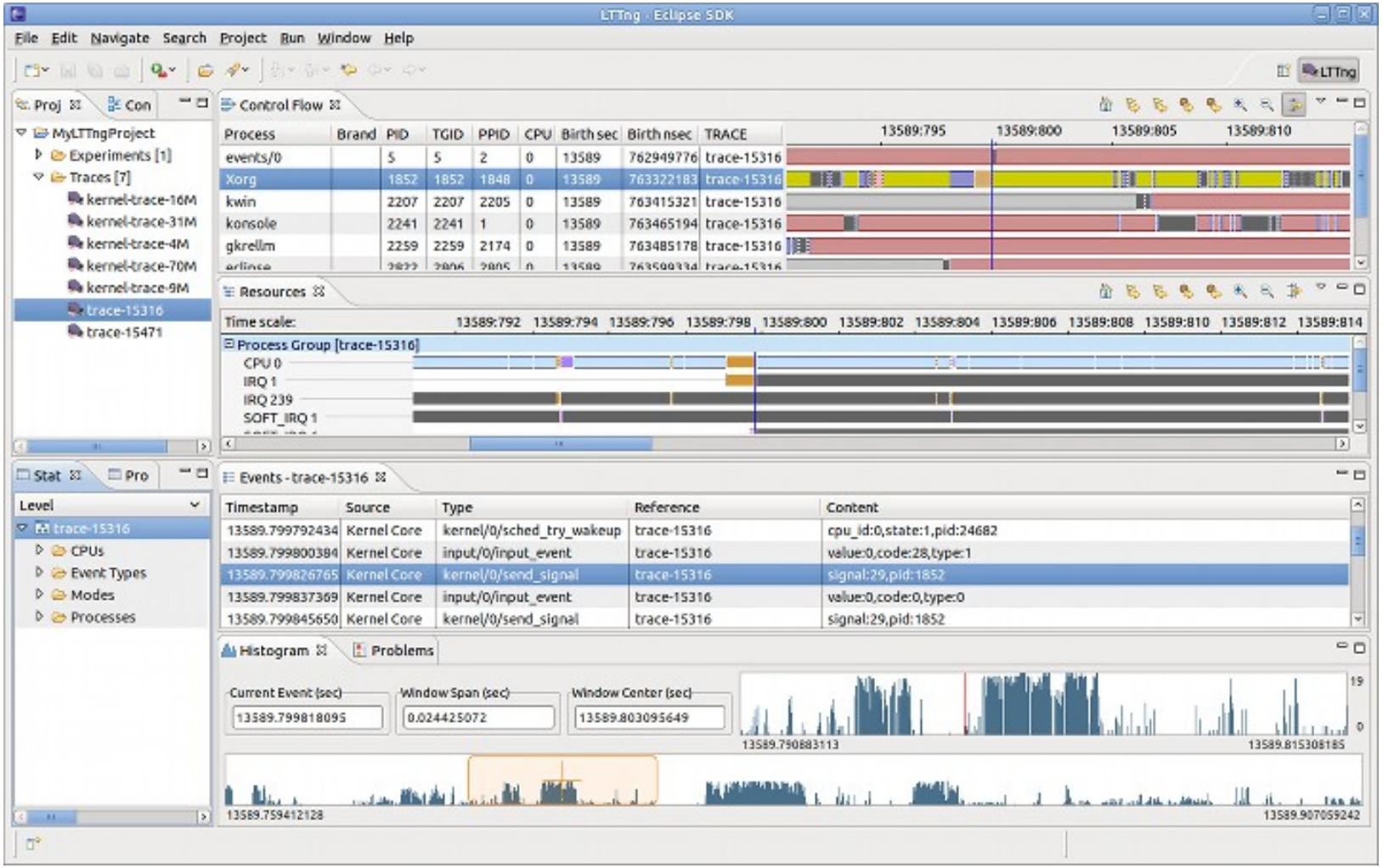
> Babeltrace

```
[13:58:29.128909723] (+0.000002475) sys_read: { 0 }, { "firefox-bin", 3363 }, { fd = 5, buf =
 count = 16 }
[13:58:29.128911513] (+0.000001790) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = -11
[13:58:29.128919672] (+0.000008159) sys_write: { 0 }, { "firefox-bin", 3363 }, { fd = 5, buf
, count = 8 }
[13:58:29.128921404] (+0.000001732) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = 8 }
[13:58:29.128922884] (+0.000001480) sys_read: { 0 }, { "firefox-bin", 3363 }, { fd = 19, buf
, count = 1 }
[13:58:29.128925765] (+0.000002881) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = 1 }
[13:58:29.128928120] (+0.000002355) sys_write: { 0 }, { "firefox-bin", 3363 }, { fd = 5, buf
, count = 8 }
[13:58:29.128929552] (+0.000001432) exit_syscall: { 0 }, { "firefox-bin", 3363 }, { ret = 8 }
[13:58:29.129020005] (+0.000090453) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 1 }
[13:58:29.129025587] (+0.000005582) sys_rt_sigprocmask: { 0 }, { "acpid", 1536 }, { how = 0,
 oset = 0x0, sigsetsize = 8 }
[13:58:29.129027993] (+0.000002406) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 0 }
[13:58:29.129030188] (+0.000002195) sys_poll: { 0 }, { "acpid", 1536 }, { ufds = 0x7FFF2A055D
meout_msecs = 0 }
[13:58:29.129032570] (+0.000002382) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 0 }
[13:58:29.129033929] (+0.000001359) sys_rt_sigprocmask: { 0 }, { "acpid", 1536 }, { how = 1,
 oset = 0x0, sigsetsize = 8 }
[13:58:29.129035144] (+0.000001215) exit_syscall: { 0 }, { "acpid", 1536 }, { ret = 0 }
[13:58:29.129037520] (+0.000002376) sys_read: { 0 }, { "acpid", 1536 }, { fd = 4, buf = 0x7FF
 = 24 }
```


Statistics for interval [13300532017949420 51, 13300532027951317 20[
CPUs 4 (max/cpu : 25.00%)
Processes N/A (0, 0)
Threads N/A (0, 0)
Files N/A (0, 0) N/A kbytes/sec
Network N/A (0, 0) N/A Mbytes/sec
CPU Top
CPU(%) TGID PID NAME
10.00 23844 23844 gnome-shell
5.50 20627 20627 firefox-bin
0.93 23653 23653 Xorg
0.29 4788 4788 epiphany-browse
0.05 11223 11223 kworker/2:2
0.05 11173 11173 kworker/0:0
0.05 11222 11222 kworker/1:1
0.05 10843 10843 kworker/3:1
0.04 14809 14809 hald
0.04 24103 24103 xchat
0.02 31261 31261 synergyc
0.02 20247 20247 emacs
0.02 6251 6251 emacs
0.02 2403 2403 soffice.bin
0.01 25701 25701 emacs
0.01 2719 2719 nmbd
0.01 13085 13085 icedove-bin
0.01 1534 1534 dbus-daemon
0.00 11193 11193 kworker/u:1
0.00 10985 10985 kworker/u:2
0.00 577 577 ips-monitor
0.00 9750 9750 ksoftirqd/3
0.00 17301 17301 kworker/1:2
0.00 23813 23813 gnome-settings-
lttngtop
Status
Starting display
Pause
F2:CPUtop F3:PerfTop F6:IOTop Enter:Details q:Quit | P:Perf Pref p:Pause

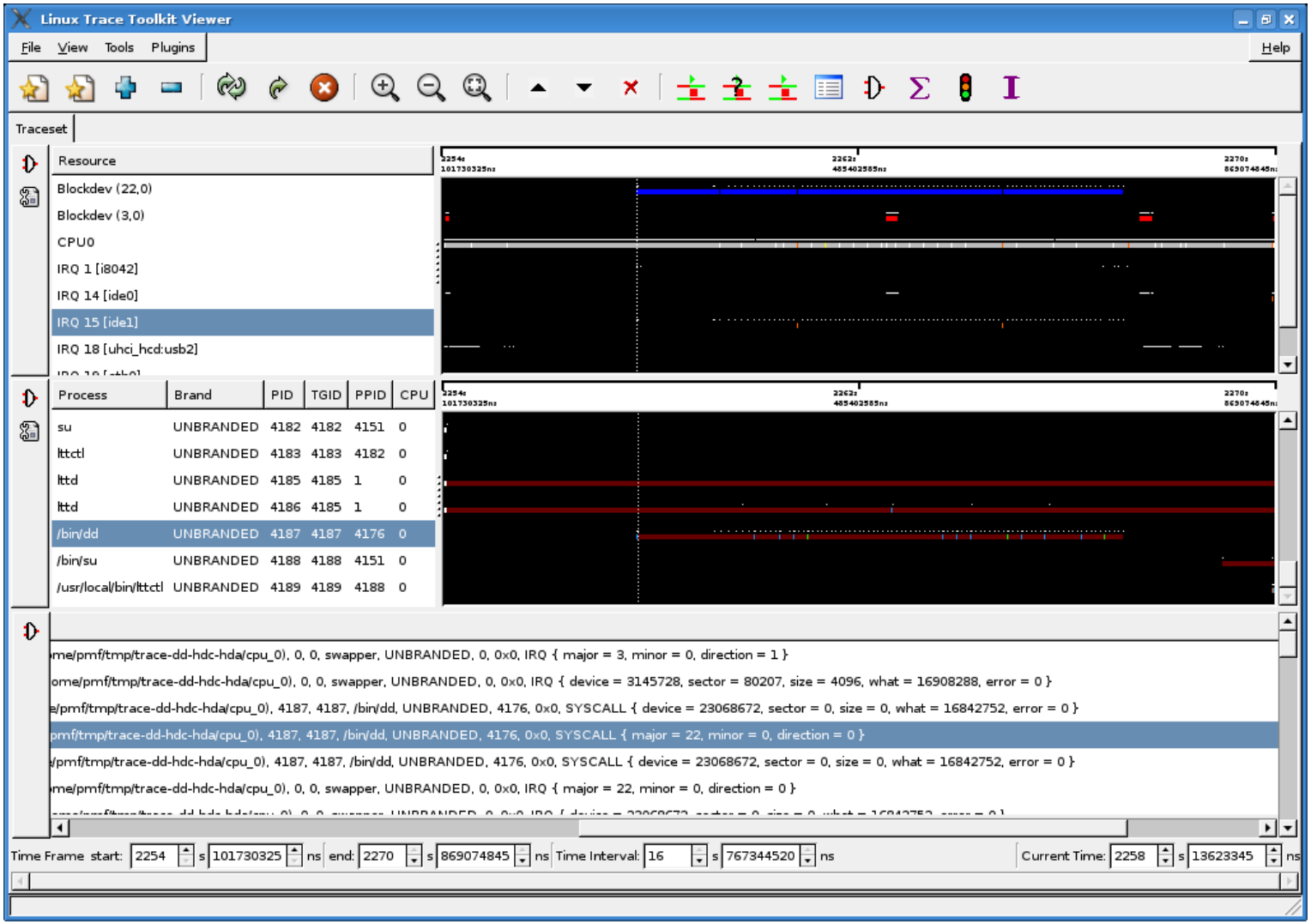
> Eclipse Linux Tools Project: LTTng support

- http://wiki.eclipse.org/Linux_Tools_Project/LTTng



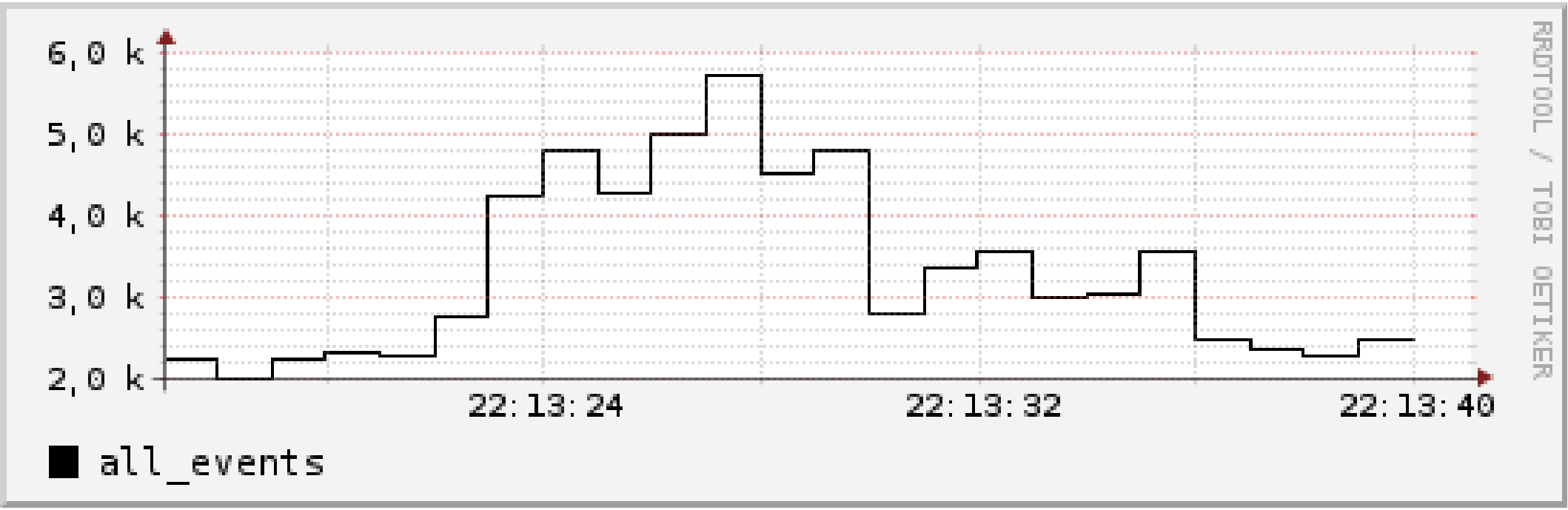
LTTng 2.0 support planned for Juno release.

> LTTV

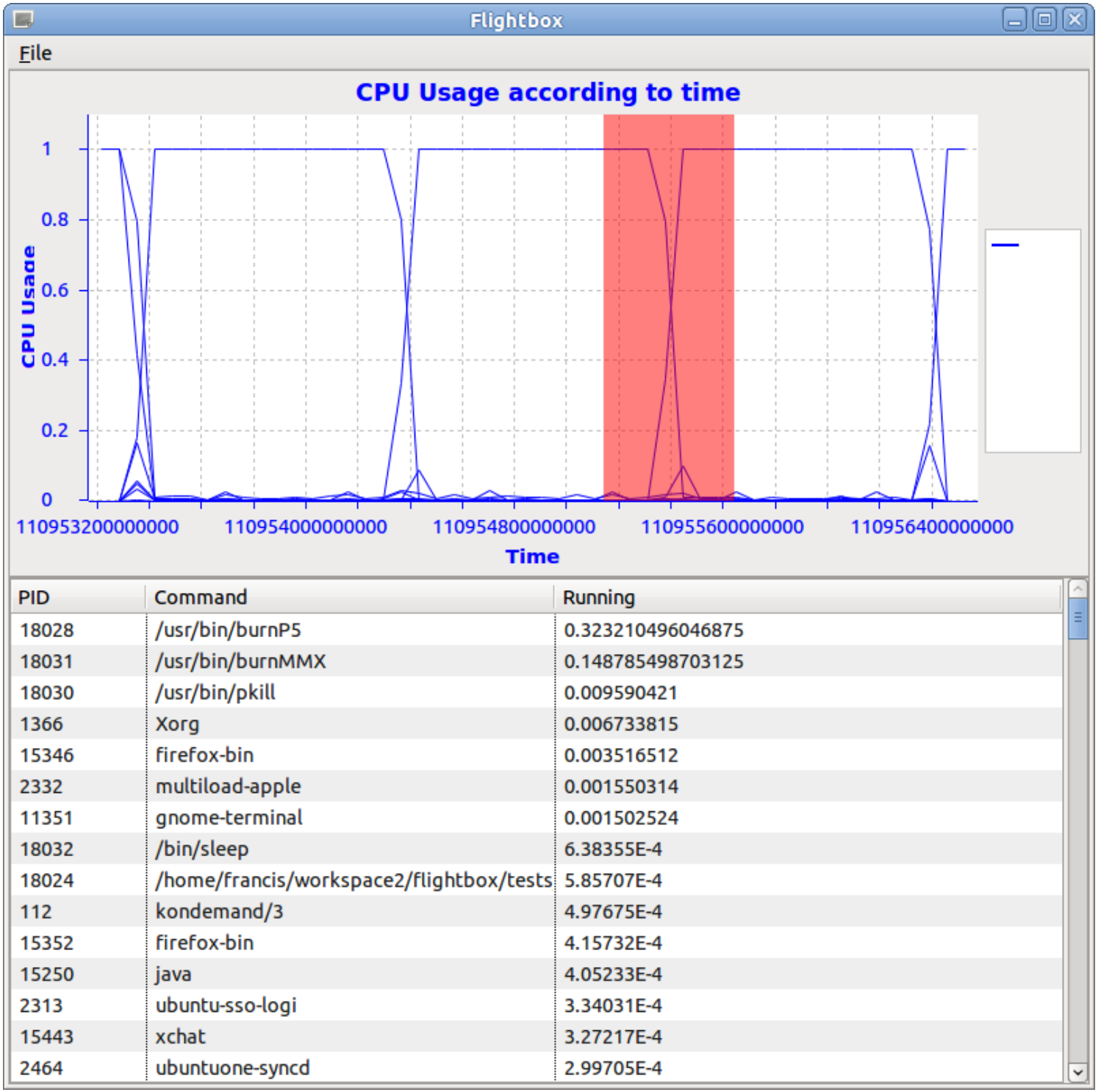


- Will be ported to LTTng 2.0 soon.

> LTTng-Graph


6,0 k
5,0 k
4,0 k
3,0 k
2,0 k
22:13:24
22:13:32
22:13:40
all_events
RRDTOOL / TOBI OETIKER

LTTng studio



PID	Command	Running
18028	/usr/bin/burnP5	0.323210496046875
18031	/usr/bin/burnMMX	0.148785498703125
18030	/usr/bin/pkill	0.009590421
1366	Xorg	0.006733815
15346	firefox-bin	0.003516512
2332	multiload-apple	0.001550314
11351	gnome-terminal	0.001502524
18032	/bin/sleep	6.38355E-4
18024	/home/francis/workspace2/flightbox/tests	5.85707E-4
112	kondemand/3	4.97675E-4
15352	firefox-bin	4.15732E-4
15250	java	4.05233E-4
2313	ubuntu-sso-logi	3.34031E-4
15443	xchat	3.27217E-4
2464	ubuntuone-syncd	2.99705E-4

April 30th, 2012

> The road ahead

- New LTTV and TMF with Common Trace Format and State System
- Live tracing
- Other language bindings
- Remote tracing
- Filtering
- Dependency analysis
- Event abstraction

> Questions ?

LTTng 2.0 available at http://lttng.org

EfficiOS

- http://www.efficios.com
- LTTng Information
  - http://lttng.org
  - lttng-dev@lists.lttng.org