

LinuxCon North America 2012

LTTng 2.0 : Tracing, Analysis and Views for
Performance and Debugging.

E-mail:

mathieu.desnoyers@efficios.com

> Presenter

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

> Content

- Tracing benefits,
- LTTng 2.0 Linux kernel and user-space tracers,
- LTTng 2.0 usage scenarios & viewers,
- New features ready for LTTng 2.1,
- Conclusion

> 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 production phases.
- High-availability, high-throughput servers
 - Development and production: ensure high performance, low-latency in production.
- Embedded
 - System development and production stages.

> LTTng 2.0

- Rich ecosystem of projects,
- Key characteristics of LTTng 2.0:
 - Small impact on the traced system, fast, user-oriented features.
- Interfacing with: Common Trace Format (CTF)

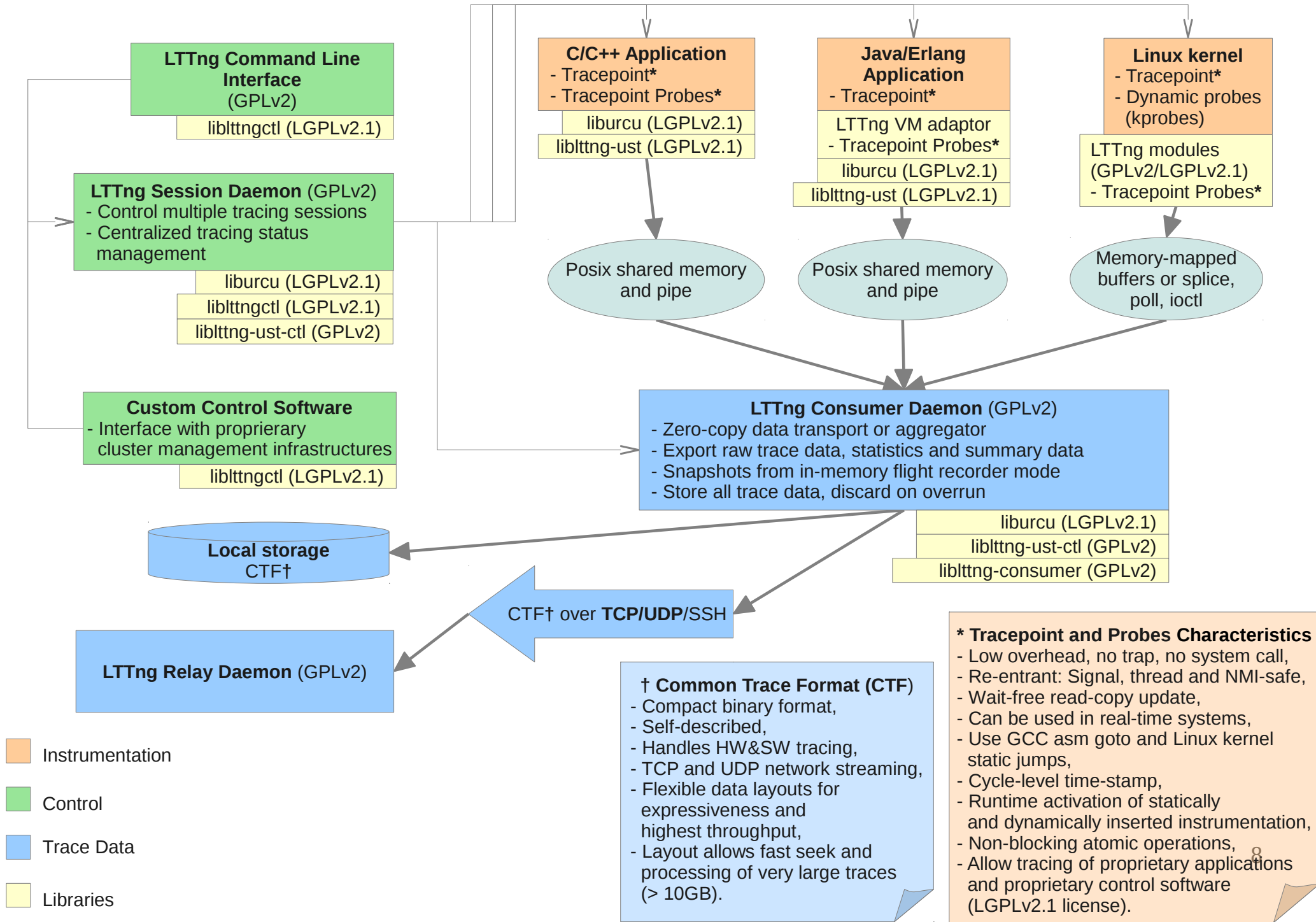
Interoperability Between Tracing Tools with the Common Trace Format (CTF),
Mathieu Desnoyers, EfficiOS,
Tracing Summit,
Aug. 30, Room Nautilus 3, 14h45.

Tracing Well With Others: Integration of GDB Tracepoints Into Trace Tools,
Stan Shebs, Mentor Graphics,
Tracing Summit,
Aug. 30, Room Nautilus 3, 11h15.

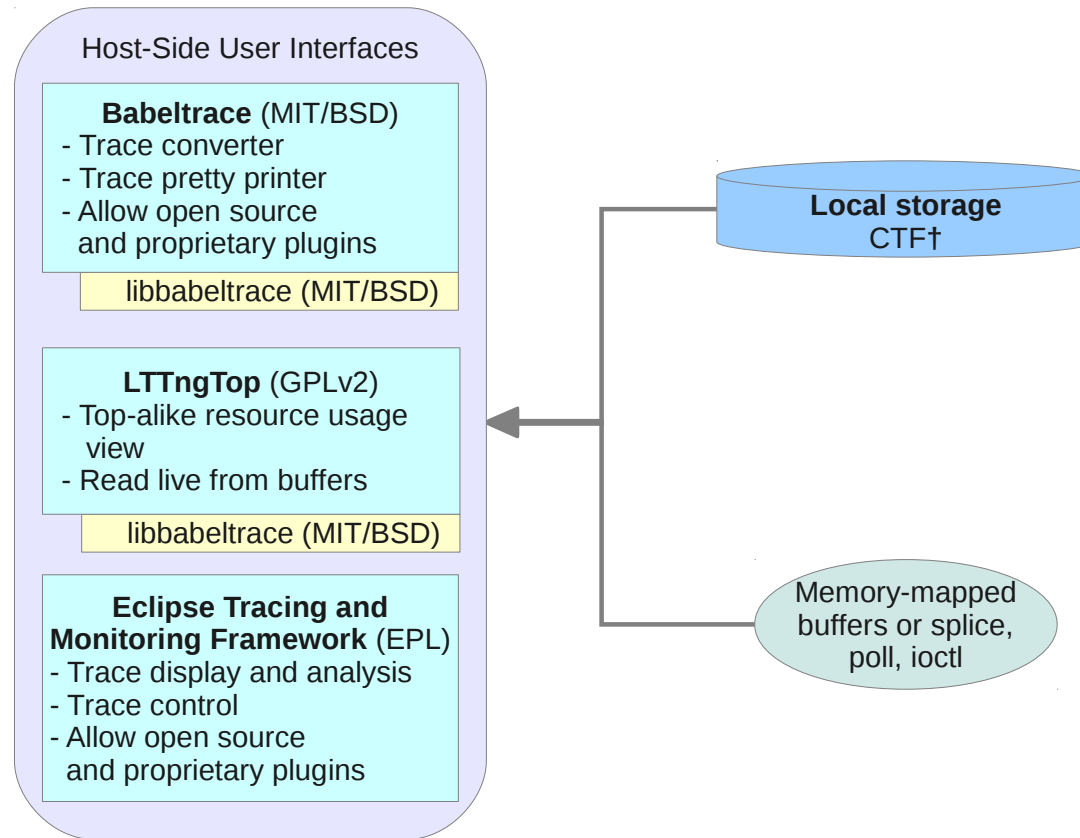
> LTTng 2.0 Tracers

- Controlled by lttng-sessiond(8),
- User interact with single command line UI, “lttng(1)”,
- LTTng modules: Linux kernel tracing,
- LTTng-UST library: user-space tracing,

LTTNg 2.0 Low-Overhead Tracing Architecture



LTTng 2.0 Common Trace Format Viewers



Trace Data

Libraries

> Trace Control Libraries and Bindings

- liblttng-ctl and lttng/lttng.h expose a C/C++ LGPL v2.1 API to control tracing.
- LTTng tracing control Python bindings planned to be merged into lttng-tools 2.2
 - <http://git.lttng.org/?p=lttng-tools.git> master branch, in extras/.
- Noteworthy: GDB is also using Python.

> LTTng 2.0 Linux Kernel Tracer

- Build against a vanilla or distribution kernel 2.6.38+ , without need for additional patches, and back to 2.6.32 with backport of 3 upstream kernel patches,
- Instrumentation sources: Tracepoints, System calls, Function tracer, kprobes, and kretprobes,
- Supports multiple concurrent tracing sessions,
- Flight recorder mode, snapshots, supported at the tracer level, not supported by lttng-tools 2.0 yet (planned for 2.2).

> 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 2.0 User-Space Tracer

- Supports: Linux, FreeBSD, OpenBSD, NetBSD,
- Tracing performed directly in user-space through shared memory map, without calling the kernel (for speed),
- Support TRACEPOINT_EVENT instrumentation: dynamically enabled, statically defined, user-space instrumentation.
- Supports multiple concurrent tracing sessions.

> 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, tpctest,  
    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.

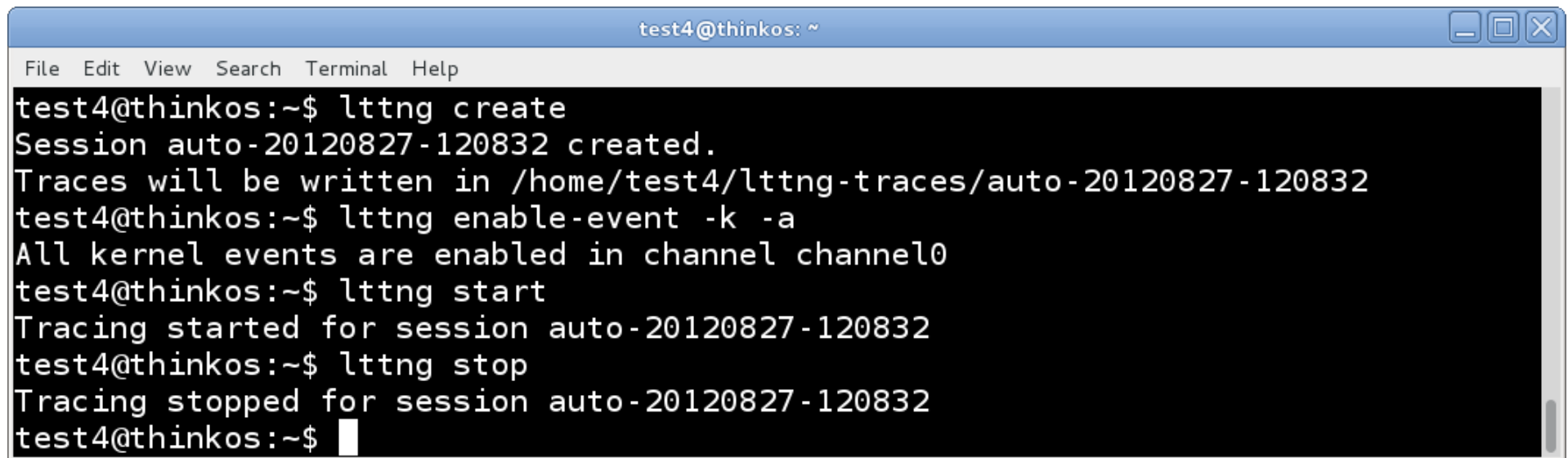
Tracepoint invocation within the code:

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


> Usage Scenarios

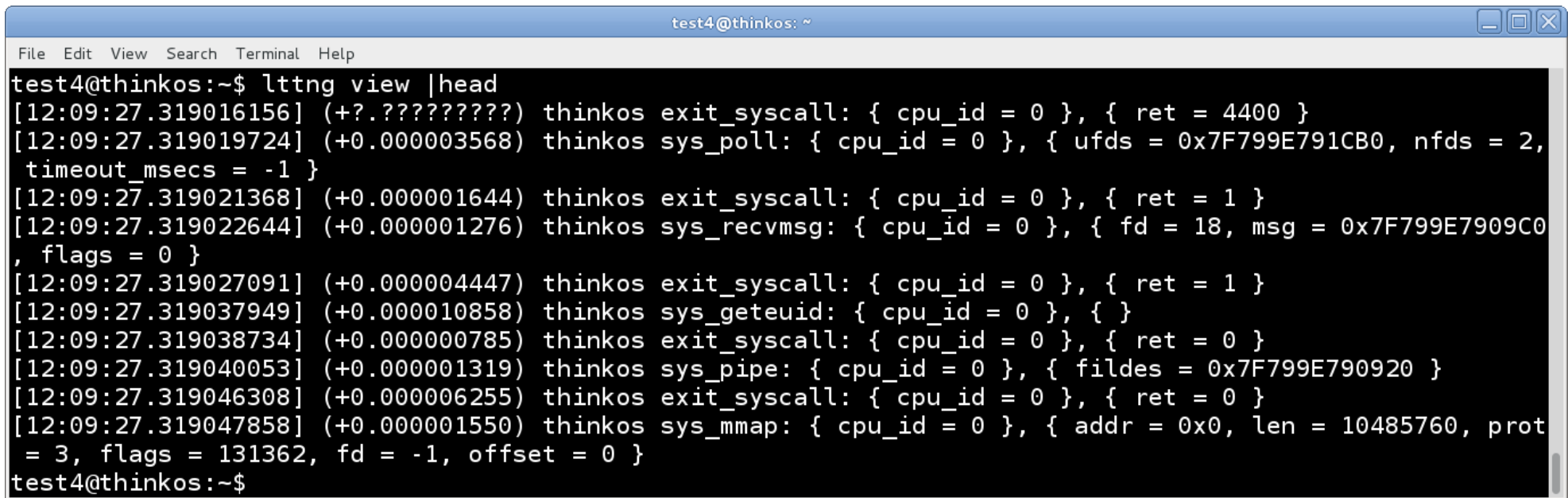
- Tracing,
- Analyzing trace data,
- Tracing across kernel and user-space,
- Tracing across multiple nodes.

> Tracing: record kernel trace

A terminal window titled 'test4@thinkos: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
test4@thinkos:~$ lttng create
Session auto-20120827-120832 created.
Traces will be written in /home/test4/lttng-traces/auto-20120827-120832
test4@thinkos:~$ lttng enable-event -k -a
All kernel events are enabled in channel channel0
test4@thinkos:~$ lttng start
Tracing started for session auto-20120827-120832
test4@thinkos:~$ lttng stop
Tracing stopped for session auto-20120827-120832
test4@thinkos:~$
```

> Tracing: view trace



```
test4@thinkos:~$ lttng view | head
[12:09:27.319016156] (+?.?????????) thinkos exit_syscall: { cpu_id = 0 }, { ret = 4400 }
[12:09:27.319019724] (+0.000003568) thinkos sys_poll: { cpu_id = 0 }, { ufds = 0x7F799E791CB0, nfds = 2,
timeout_msecs = -1 }
[12:09:27.319021368] (+0.000001644) thinkos exit_syscall: { cpu_id = 0 }, { ret = 1 }
[12:09:27.319022644] (+0.000001276) thinkos sys_recvmmsg: { cpu_id = 0 }, { fd = 18, msg = 0x7F799E7909C0
, flags = 0 }
[12:09:27.319027091] (+0.000004447) thinkos exit_syscall: { cpu_id = 0 }, { ret = 1 }
[12:09:27.319037949] (+0.000010858) thinkos sys_geteuid: { cpu_id = 0 }, { }
[12:09:27.319038734] (+0.000000785) thinkos exit_syscall: { cpu_id = 0 }, { ret = 0 }
[12:09:27.319040053] (+0.000001319) thinkos sys_pipe: { cpu_id = 0 }, { fildes = 0x7F799E790920 }
[12:09:27.319046308] (+0.000006255) thinkos exit_syscall: { cpu_id = 0 }, { ret = 0 }
[12:09:27.319047858] (+0.000001550) thinkos sys_mmap: { cpu_id = 0 }, { addr = 0x0, len = 10485760, prot
= 3, flags = 131362, fd = -1, offset = 0 }
test4@thinkos:~$
```

> LTTng 2.0 high-speed “strace”

ltnng enable-event --syscall -a

compudj@squeeze-amd64: ~

```
p
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 = 0 }
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 = 0 }
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 = 0 }
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 }
```

Statistics for interval [15:46:03.865900007, 15:46:04.865908180[

CPUs 2 (max/cpu : 50.00%)
 Threads 357 (0, 0)
 FDs 1850 (+1, -1) 208KB/sec

CPU Top

CPU(%)	PID	TID	NAME
4.00	2580	2580	Xorg
3.50	15021	15021	ltnngtop
2.39	5957	5957	awesome
2.35	15035	15035	gimp
0.15	14523	14523	kworker/0:2
0.12	14762	14762	kworker/1:2
0.09	14887	14887	/usr/bin/x-term
0.08	6021	6021	xscreensaver
0.07	18470	18470	icedove-bin
0.03	2498	2498	acpid
0.03	3	3	ksoftirqd/0
0.03	22469	22469	/usr/bin/x-term
0.02	6019	6019	bluetooth-apple
0.02	15028	15028	/usr/bin/x-term
0.02	30114	30114	firefox-bin
0.02	14954	14954	kworker/u:1
0.01	14504	14504	kworker/u:2
0.01	22342	22342	udisks-daemon
0.01	171	171	scsi_eh_1
0.01	14736	14736	evince
0.01	22360	22360	gnome-settings-
0.01	1	1	init
0.00	4114	4114	uml_switch
0.00	14506	14506	flush-254:2

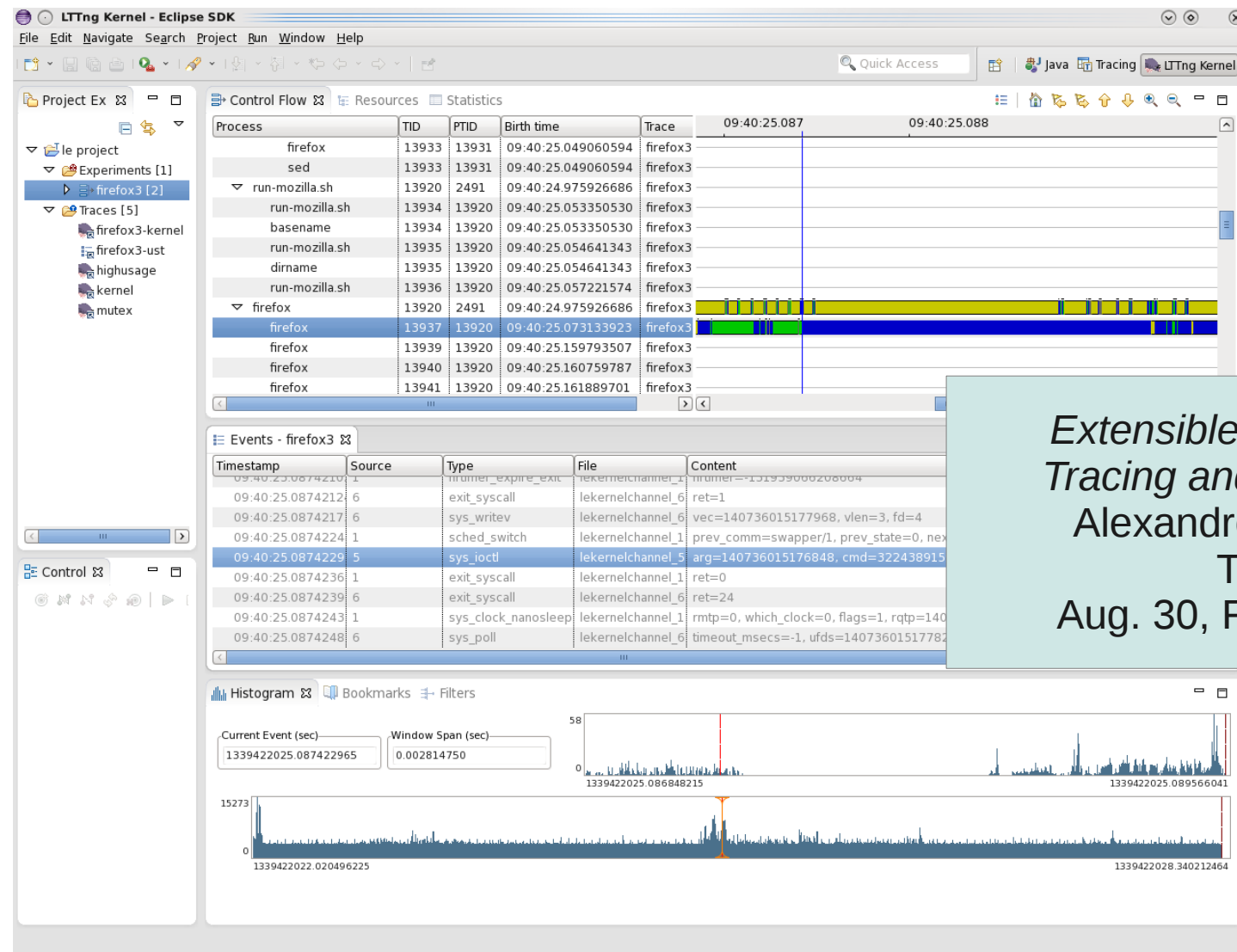
Status

Going back in time
 Going back in time
 Going forward in time
 Going forward in time

LTTngTop: Human Readable Trace Viewer,
 Julien Desfossez, EfficiOS,
 Tracing Summit,
 Aug. 30, Room Nautilus 3, 15h35.

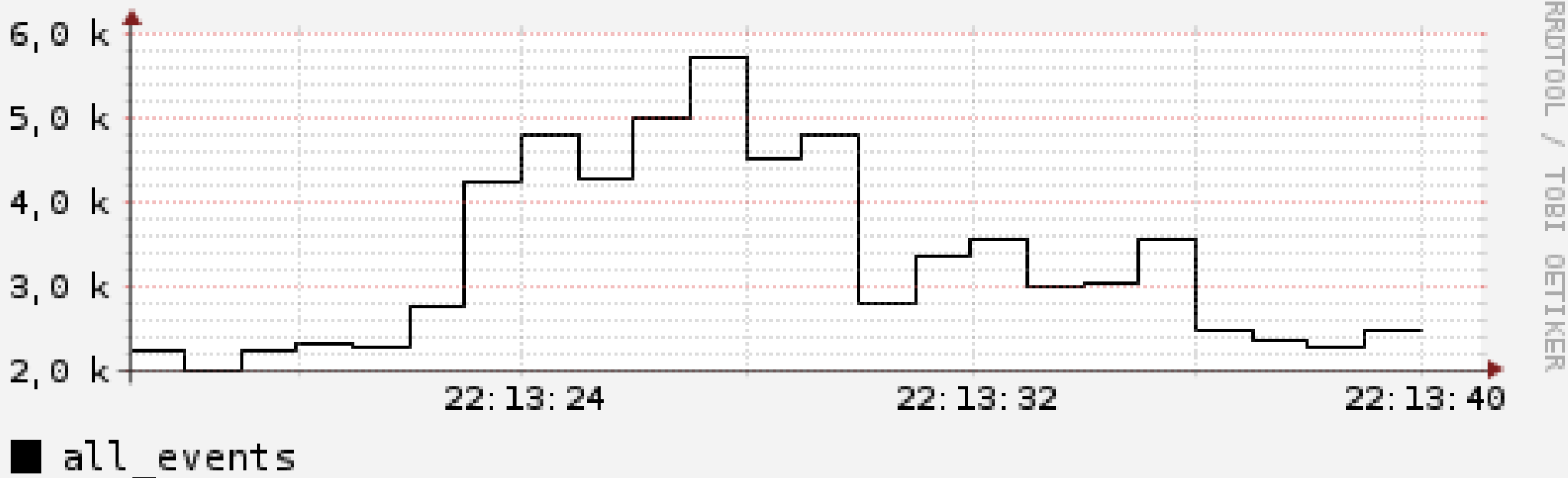
> Eclipse Linux Tools Project: LTTng support

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



*Extensible trace analysis using the
Tracing and Monitoring Framework,
Alexandre Montplaisir, Ericsson,
Tracing Summit,
Aug. 30, Room Nautilus 3, 16h10.*

> LTTng-Graph

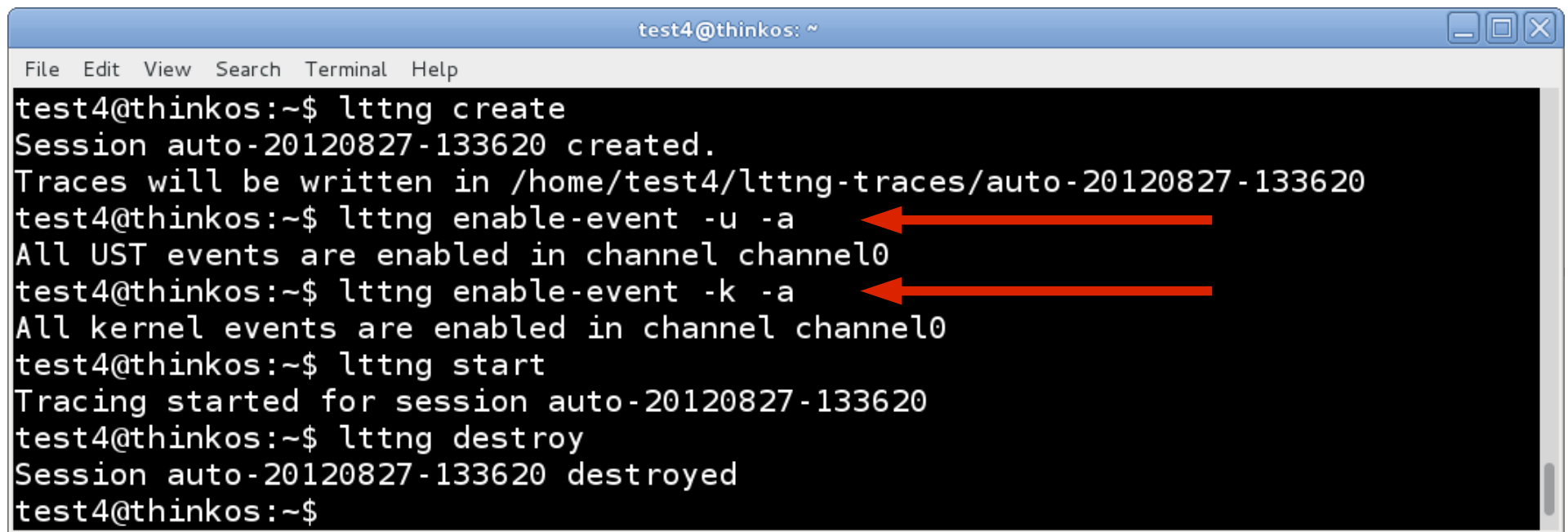


<http://git.dorsal.polymtl.ca/~jdesfossez?p=lttng-graph>

> Viewer Libraries and Bindings

- Lib Babeltrace: a C/C++ library for reading CTF traces (MIT BSD-style license)
- Python bindings over Lib Babeltrace planned to be merged into Babeltrace 1.1
 - <http://git.efficios.com/?p=babeltrace.git>
bindings/python branch.

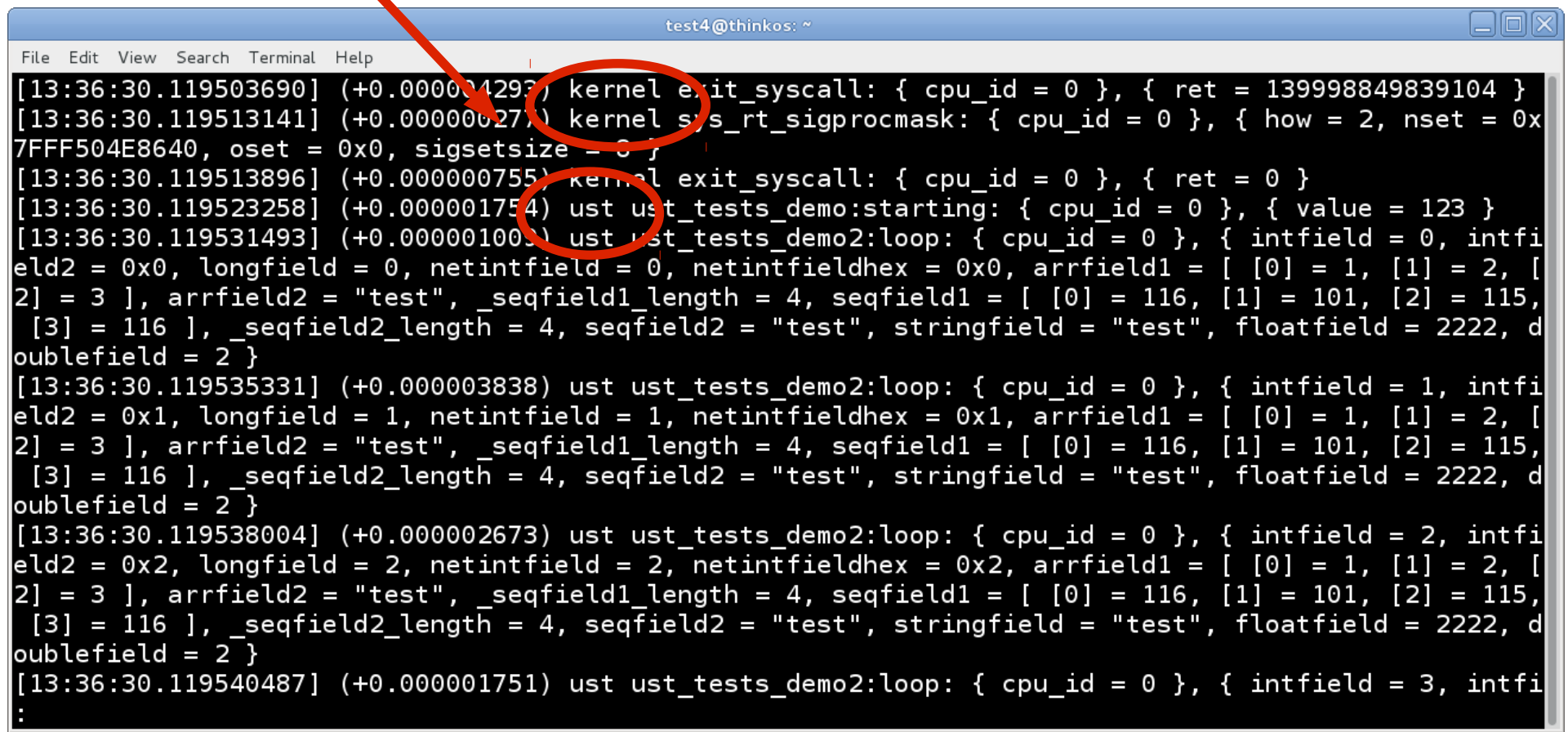
> Tracing across kernel and user-space



```
test4@thinkos: ~  
File Edit View Search Terminal Help  
test4@thinkos:~$ lttng create  
Session auto-20120827-133620 created.  
Traces will be written in /home/test4/lttng-traces/auto-20120827-133620  
test4@thinkos:~$ lttng enable-event -u -a  
All UST events are enabled in channel channel0  
test4@thinkos:~$ lttng enable-event -k -a  
All kernel events are enabled in channel channel0  
test4@thinkos:~$ lttng start  
Tracing started for session auto-20120827-133620  
test4@thinkos:~$ lttng destroy  
Session auto-20120827-133620 destroyed  
test4@thinkos:~$
```

> Tracing across kernel and user-space (2)

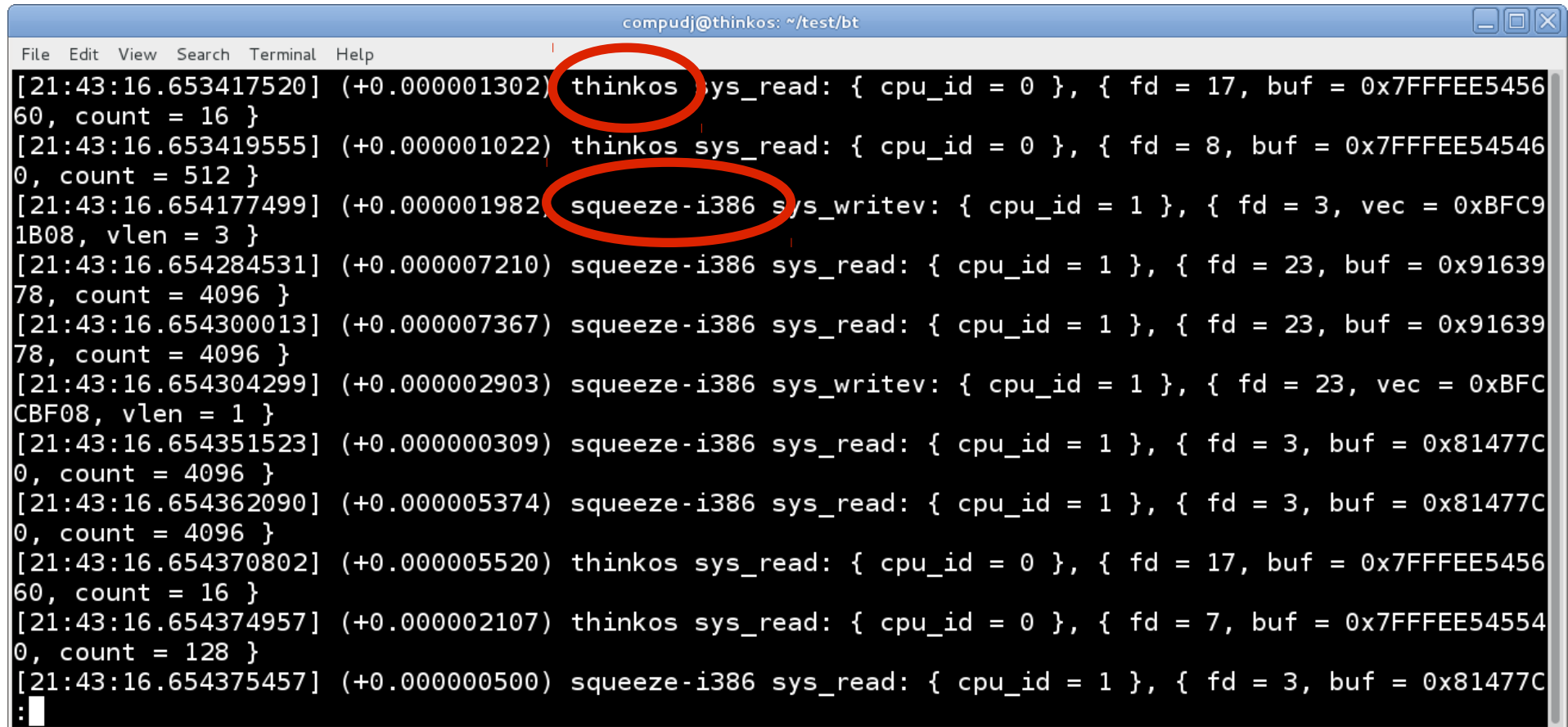
```
babeltrace -f trace:domain /home/test4/lttng-traces/auto-20120827-133620 \
| grep "{ cpu_id = 0 }" | less
```



```
test4@thinkos: ~
File Edit View Search Terminal Help
[13:36:30.119503690] (+0.0000004293) kernel exit_syscall: { cpu_id = 0 }, { ret = 139998849839104 }
[13:36:30.119513141] (+0.0000000277) kernel sys_rt_sigprocmask: { cpu_id = 0 }, { how = 2, nset = 0x
7FFF504E8640, oset = 0x0, sigsetsize = 0 }
[13:36:30.119513896] (+0.0000000755) kernel exit_syscall: { cpu_id = 0 }, { ret = 0 }
[13:36:30.119523258] (+0.0000001754) ust ust_tests_demo:starting: { cpu_id = 0 }, { value = 123 }
[13:36:30.119531493] (+0.0000001005) ust ust_tests_demo2:loop: { cpu_id = 0 }, { intfield = 0, intfi
eld2 = 0x0, longfield = 0, netintfield = 0, netintfieldhex = 0x0, arrfield1 = [ [0] = 1, [1] = 2, [
2] = 3 ], arrfield2 = "test", _seqfield1_length = 4, seqfield1 = [ [0] = 116, [1] = 101, [2] = 115,
[3] = 116 ], _seqfield2_length = 4, seqfield2 = "test", stringfield = "test", floatfield = 2222, d
oublefield = 2 }
[13:36:30.119535331] (+0.0000003838) ust ust_tests_demo2:loop: { cpu_id = 0 }, { intfield = 1, intfi
eld2 = 0x1, longfield = 1, netintfield = 1, netintfieldhex = 0x1, arrfield1 = [ [0] = 1, [1] = 2, [
2] = 3 ], arrfield2 = "test", _seqfield1_length = 4, seqfield1 = [ [0] = 116, [1] = 101, [2] = 115,
[3] = 116 ], _seqfield2_length = 4, seqfield2 = "test", stringfield = "test", floatfield = 2222, d
oublefield = 2 }
[13:36:30.119538004] (+0.0000002673) ust ust_tests_demo2:loop: { cpu_id = 0 }, { intfield = 2, intfi
eld2 = 0x2, longfield = 2, netintfield = 2, netintfieldhex = 0x2, arrfield1 = [ [0] = 1, [1] = 2, [
2] = 3 ], arrfield2 = "test", _seqfield1_length = 4, seqfield1 = [ [0] = 116, [1] = 101, [2] = 115,
[3] = 116 ], _seqfield2_length = 4, seqfield2 = "test", stringfield = "test", floatfield = 2222, d
oublefield = 2 }
[13:36:30.119540487] (+0.0000001751) ust ust_tests_demo2:loop: { cpu_id = 0 }, { intfield = 3, intfi
:
```

> Tracing across nodes

babeltrace --clock-force-correlate trace1 trace2 |grep -e sys_read -e sys_write |less



The screenshot shows a terminal window titled 'compudj@thinkos: ~/test/bt'. The terminal displays the output of the command 'babeltrace --clock-force-correlate trace1 trace2 |grep -e sys_read -e sys_write |less'. The output consists of several lines of trace data. Two lines are circled in red: the first line, '[21:43:16.653417520] (+0.000001302) thinkos sys_read: { cpu_id = 0 }, { fd = 17, buf = 0x7FFFEE545660, count = 16 }', and the third line, '[21:43:16.654177499] (+0.000001982) squeeze-i386 sys_writev: { cpu_id = 1 }, { fd = 3, vec = 0xBF08, vlen = 3 }'. The terminal window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'.

```
compudj@thinkos: ~/test/bt
File Edit View Search Terminal Help
[21:43:16.653417520] (+0.000001302) thinkos sys_read: { cpu_id = 0 }, { fd = 17, buf = 0x7FFFEE545660, count = 16 }
[21:43:16.653419555] (+0.000001022) thinkos sys_read: { cpu_id = 0 }, { fd = 8, buf = 0x7FFFEE545460, count = 512 }
[21:43:16.654177499] (+0.000001982) squeeze-i386 sys_writev: { cpu_id = 1 }, { fd = 3, vec = 0xBF08, vlen = 3 }
[21:43:16.654284531] (+0.000007210) squeeze-i386 sys_read: { cpu_id = 1 }, { fd = 23, buf = 0x9163978, count = 4096 }
[21:43:16.654300013] (+0.000007367) squeeze-i386 sys_read: { cpu_id = 1 }, { fd = 23, buf = 0x9163978, count = 4096 }
[21:43:16.654304299] (+0.000002903) squeeze-i386 sys_writev: { cpu_id = 1 }, { fd = 23, vec = 0xBF08, vlen = 1 }
[21:43:16.654351523] (+0.000000309) squeeze-i386 sys_read: { cpu_id = 1 }, { fd = 3, buf = 0x81477C0, count = 4096 }
[21:43:16.654362090] (+0.000005374) squeeze-i386 sys_read: { cpu_id = 1 }, { fd = 3, buf = 0x81477C0, count = 4096 }
[21:43:16.654370802] (+0.000005520) thinkos sys_read: { cpu_id = 0 }, { fd = 17, buf = 0x7FFFEE545660, count = 16 }
[21:43:16.654374957] (+0.000002107) thinkos sys_read: { cpu_id = 0 }, { fd = 7, buf = 0x7FFFEE545540, count = 128 }
[21:43:16.654375457] (+0.000000500) squeeze-i386 sys_read: { cpu_id = 1 }, { fd = 3, buf = 0x81477C0, count = 4096 }
```

> Userspace Tracing Benchmark

Approx time by event – 1 thread (nanoseconds)

In the same elapsed time, the number of events that can be traced can be multiplied by, respectively, 8.5 and 21.4, compared to Dtrace and SystemTap.



LTTng UST



UST (W/O OPT)

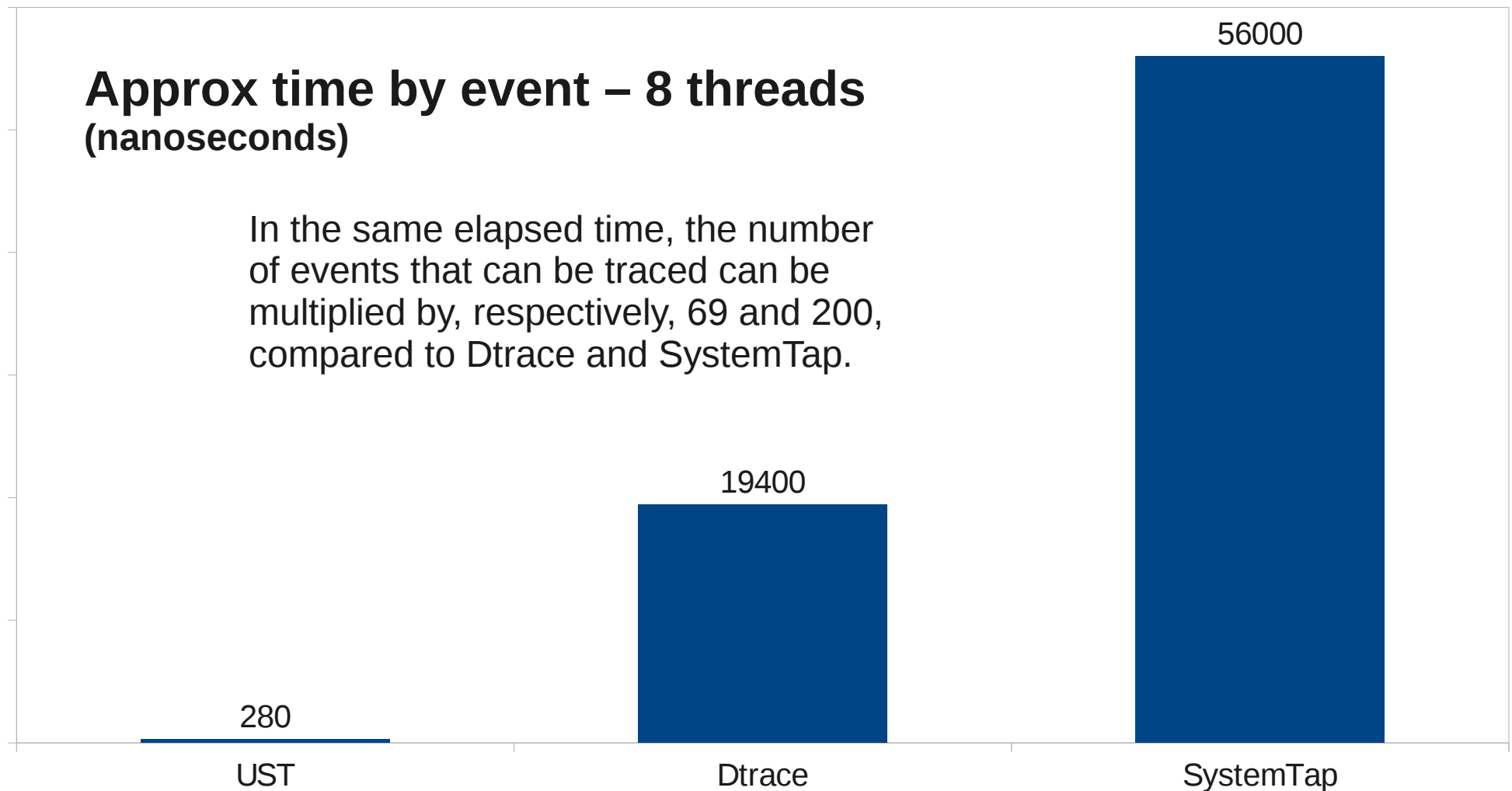


Dtrace



SystemTap

> Userspace Tracing Benchmark



> Strace vs LTTng Tracing

Timing of a find of 100000 files (seconds)

LTTng speedup
44.49:1 vs strace

0.54

find

1.4

find + lttng

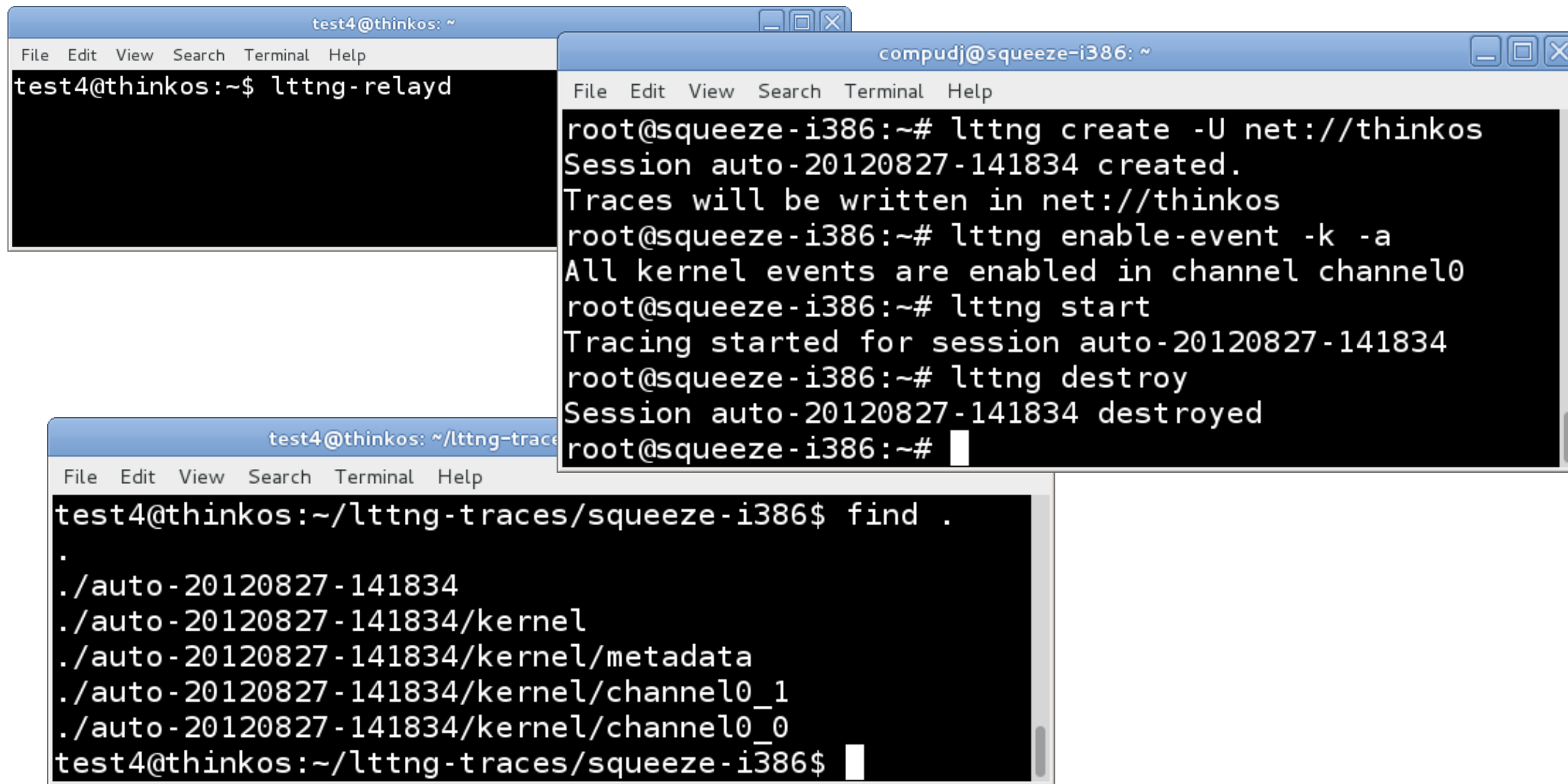
38.8

find + strace

> New Features (LTTng 2.1)

- Network Streaming over TCP
- LTTng-UST filtering before data collection
- Ittng-sessiond(8) health monitoring API.

> Network Streaming over TCP



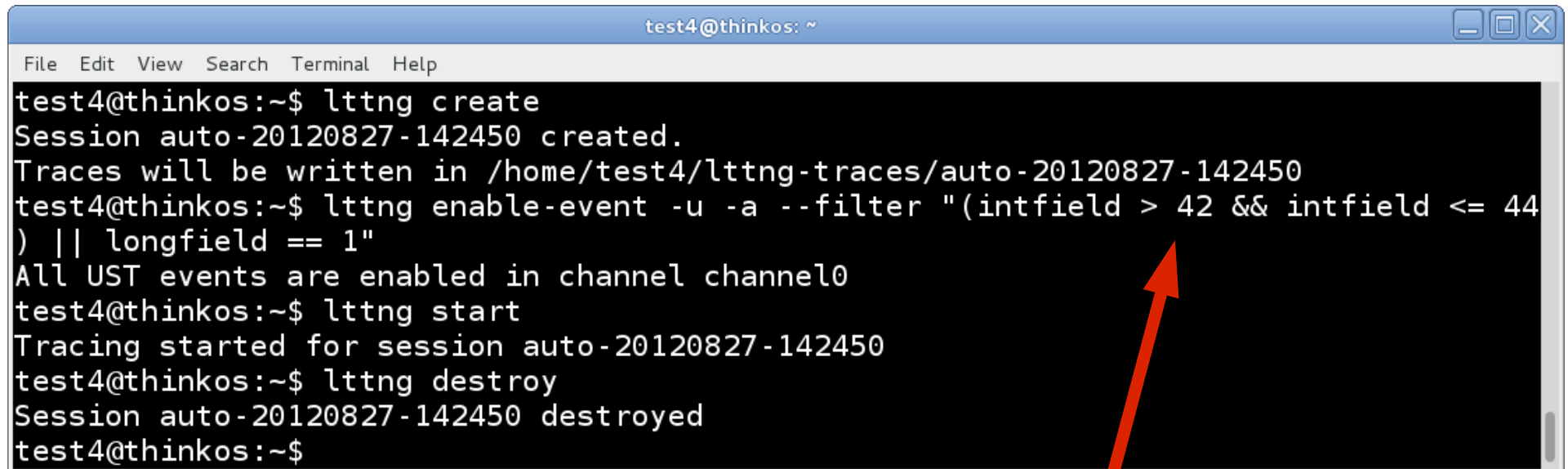
The image displays three terminal windows illustrating the setup and execution of lttng for network streaming over TCP.

Terminal 1 (top left): test4@thinkos: ~
File Edit View Search Terminal Help
test4@thinkos:~\$ lttng-relayd

Terminal 2 (top right): compudj@squeeze-i386: ~
File Edit View Search Terminal Help
root@squeeze-i386:~# lttng create -U net://thinkos
Session auto-20120827-141834 created.
Traces will be written in net://thinkos
root@squeeze-i386:~# lttng enable-event -k -a
All kernel events are enabled in channel channel0
root@squeeze-i386:~# lttng start
Tracing started for session auto-20120827-141834
root@squeeze-i386:~# lttng destroy
Session auto-20120827-141834 destroyed
root@squeeze-i386:~#

Terminal 3 (bottom left): test4@thinkos: ~/lttng-traces
File Edit View Search Terminal Help
test4@thinkos:~/lttng-traces/squeeze-i386\$ find .
.
./auto-20120827-141834
./auto-20120827-141834/kernel
./auto-20120827-141834/kernel/metadata
./auto-20120827-141834/kernel/channel0_1
./auto-20120827-141834/kernel/channel0_0
test4@thinkos:~/lttng-traces/squeeze-i386\$

> LTTng-UST Filtering



```
test4@thinkos: ~  
File Edit View Search Terminal Help  
test4@thinkos:~$ lttng create  
Session auto-20120827-142450 created.  
Traces will be written in /home/test4/lttng-traces/auto-20120827-142450  
test4@thinkos:~$ lttng enable-event -u -a --filter "(intfield > 42 && intfield <= 44  
) || longfield == 1"  
All UST events are enabled in channel channel0  
test4@thinkos:~$ lttng start  
Tracing started for session auto-20120827-142450  
test4@thinkos:~$ lttng destroy  
Session auto-20120827-142450 destroyed  
test4@thinkos:~$
```

> LTTng-UST Filtering (2)

```
test4@thinkos: ~/ltnng-traces
File Edit View Search Terminal Help
test4@thinkos:~/ltnng-traces$ babeltrace auto-20120827-142450/
[14:25:09.348326990] (+?.?????????) thinkos:lt-hello:22206 ust_tests_hello:tp
test: { cpu_id = 0 }, { intfield = 1, intfield2 = 0x1, longfield = 1, netintf
ield = 1, netintfieldhex = 0x1, arrfield1 = [ [0] = 1, [1] = 2, [2] = 3 ], ar
rfield2 = "test", _seqfield1_length = 4, seqfield1 = [ [0] = 116, [1] = 101,
[2] = 115, [3] = 116 ], _seqfield2_length = 4, seqfield2 = "test", stringfiel
d = "test", floatfield = 2222, doublefield = 2, boolfield = 1 }
[14:25:09.348343662] (+0.000016672) thinkos:lt-hello:22206 ust_tests_hello:tp
test: { cpu_id = 0 }, { intfield = 43, intfield2 = 0x2B, longfield = 43, neti
ntfield = 43, netintfieldhex = 0x2B, arrfield1 = [ [0] = 1, [1] = 2, [2] = 3
], arrfield2 = "test", _seqfield1_length = 4, seqfield1 = [ [0] = 116, [1] =
101, [2] = 115, [3] = 116 ], _seqfield2_length = 4, seqfield2 = "test", strin
gfield = "test", floatfield = 2222, doublefield = 2, boolfield = 1 }
[14:25:09.348347126] (+0.000003484) thinkos:lt-hello:22206 ust_tests_hello:tp
test: { cpu_id = 0 }, { intfield = 44, intfield2 = 0x2C, longfield = 44, neti
ntfield = 44, netintfieldhex = 0x2C, arrfield1 = [ [0] = 1, [1] = 2, [2] = 3
], arrfield2 = "test", _seqfield1_length = 4, seqfield1 = [ [0] = 116, [1] =
101, [2] = 115, [3] = 116 ], _seqfield2_length = 4, seqfield2 = "test", strin
gfield = "test", floatfield = 2222, doublefield = 2, boolfield = 1 }
test4@thinkos:~/ltnng-traces$
```

> Distributions / Integration

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

> Collaborations

- Interfacing: **GDB** tracepoints can interoperate with LTTng UST tracepoints, the **Eclipse Tracing Monitoring Framework** supports LTTng CTF traces, **Perf** PMU counters are used by LTTng, The Multi Core Association is defining a **Common Trace Format** (CTF), for which LTTng 2.0 is a reference implementation.
- Collaborators: Autodesk, CAE, C2 microsystems, Defence R&D Canada, Ecole Polytechnique de Montréal, EfficiOS, Ericsson, Fujitsu, Freescale, Google, Harvard University, IBM, Linux Foundation CE, Mentor, Multicore Association, Nokia, Opal-RT, Portland State University, RedHat, Revolution Linux, Sony, ST Microelectronics, Siemens, SUSE, WindRiver, CRIAQ, NSERC, Prompt, etc.

The Tracing Summit 2012 will be held in San Diego, on August 30th, 2012, as part of the [Linux Plumbers Conference 2012](#) .

8h30 [Can mainstream tracing meet embedded needs?](#), Frank Rowand, Sony

8h55 [What We Want in Our Toolkit: Thoughts From a Mission Critical Low Latency Environment](#), Vinod Kutty, CME Group

9h20 [Troubleshooting complex problems with built-in diagnostic](#), Dominique Toupin, Ericsson

9h45 [The Linux Perf Tools: Overview and Current Developments](#), Arnaldo Carvahlo de Melo, Red Hat

10h10 - 10h25 break

10h25 [Tracing the Guest OS from Host via Shared Memory](#), Masami Hiramatsu, Hitachi

10h50 [Shrinking core dump on the fly](#), Thomas Gleixner, Linutronix

11h15 [Tracing Well With Others: Integration of GDB Tracepoints Into Trace Tools](#), Stan Shebs, Mentor Graphics

11h40 [Ftrace and Multiple buffers](#), Steven Rostedt, Red Hat

12h05 - 13h30 lunch

13h30 [The Road ahead of Uprobes: Plans and features in pipeline](#), Srikar Dronamraju, IBM

13h45 [Systemtap and new connections: dyninst, pcp, uprobes](#), David Smith and Josh Stone, Redhat

14h20 [LTTng and Nexus Trace for Freescale QorIQ Devices](#), Ed Martinez, Freescale

14h35 - 14h45 break

14h45 [Interoperability Between Tracing Tools with the Common Trace Format \(CTF\)](#), Mathieu Desnoyers, EfficiOS

15h10 [Making linsched useful](#), Dhaval Giani, University of Toronto

15h35 [LTTngTop: Human Readable Trace Viewer](#), Julien Desfossez, EfficiOS

16h00 - 16h10 break

16h10 [Extensible trace analysis using the Tracing and Monitoring Framework](#), Alexandre Montplaisir, Ericsson

16h45 - 18h00 Open Discussion, Where do we go from here?

> Questions ?

LTTng 2.0 available at <http://lttng.org>



*Effici*OS

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