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)

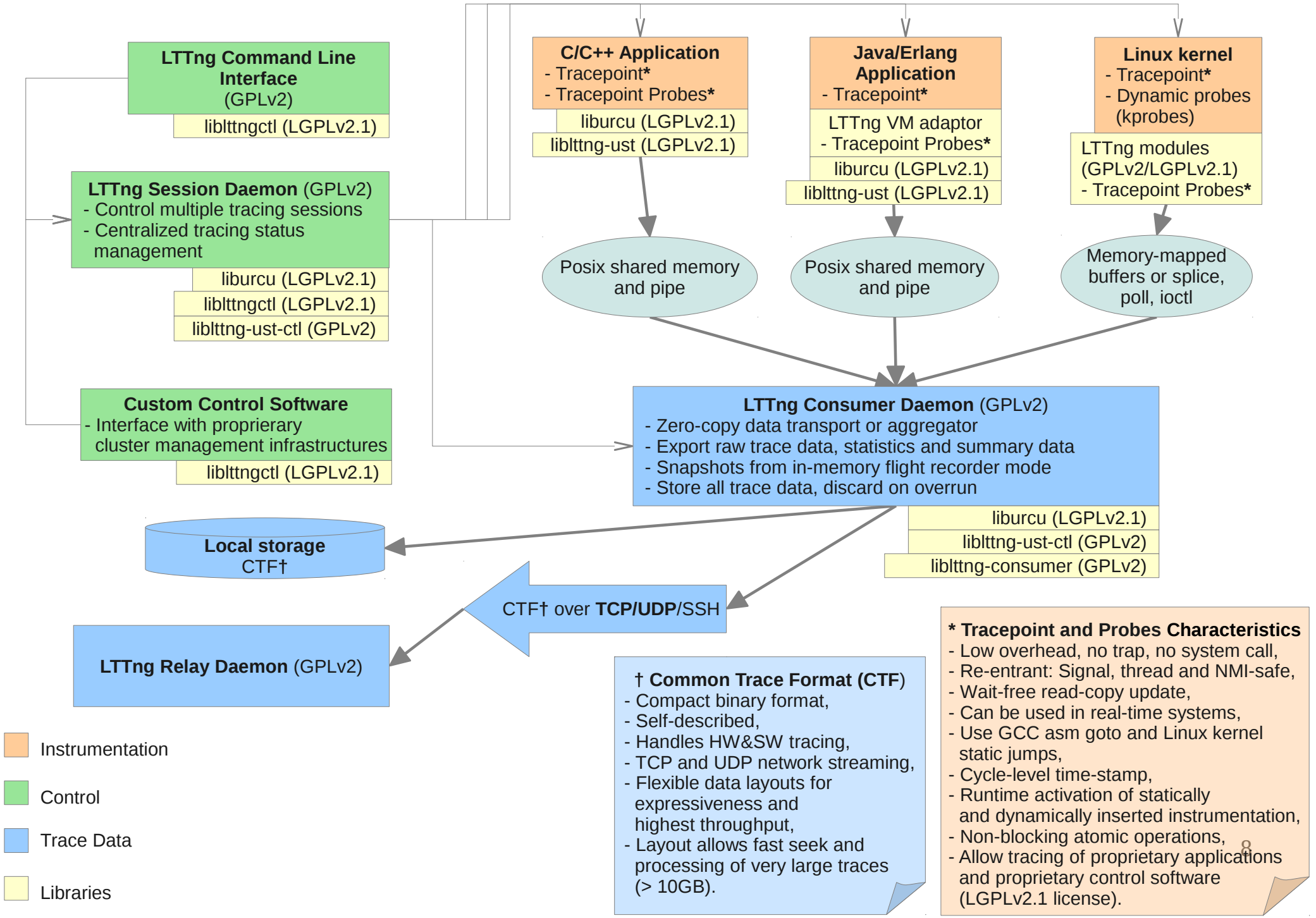
Interoperability Between Tracing Tools with the Common Trace Format (CTF),
Mathieu Desnoyers, EfficiOS,
Tracing Summit,
Aug. 30, Room Nautilus 3, 14h45.

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

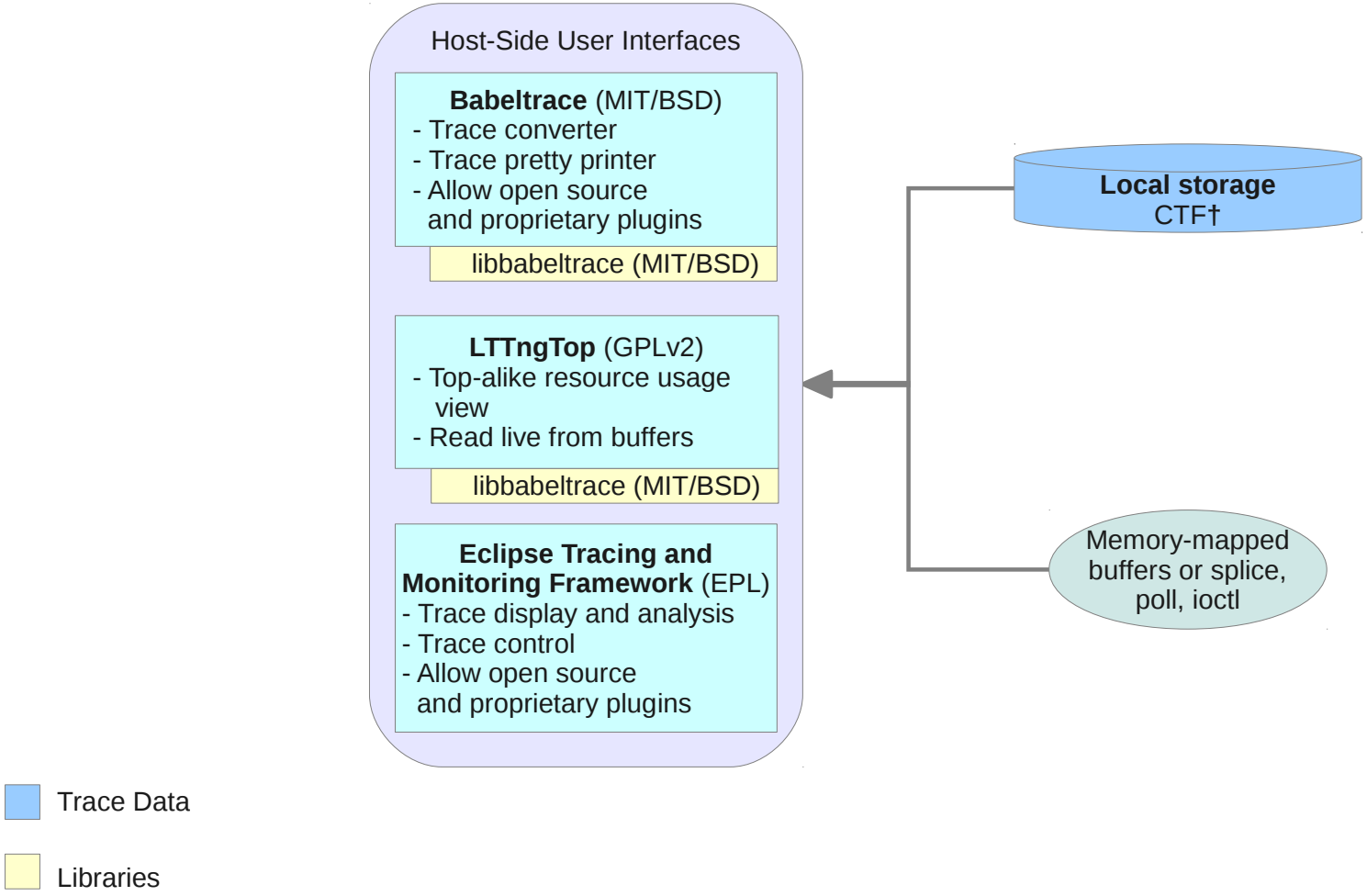
> LTTng 2.0 Tracers

- 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 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, 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.

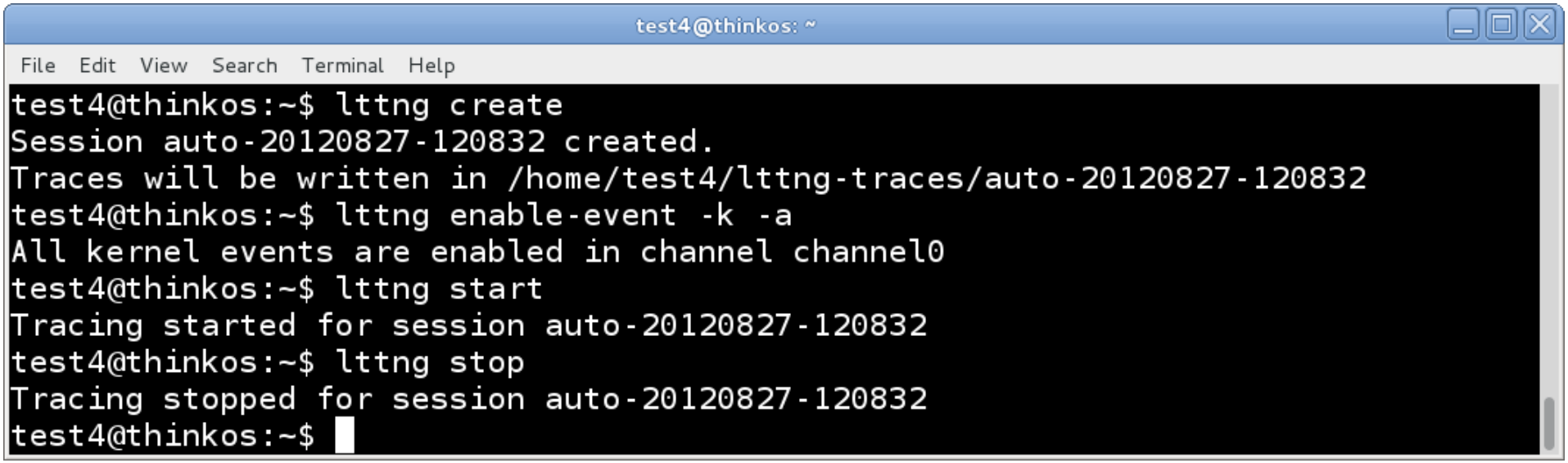
Tracepoint invocation within the code:

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

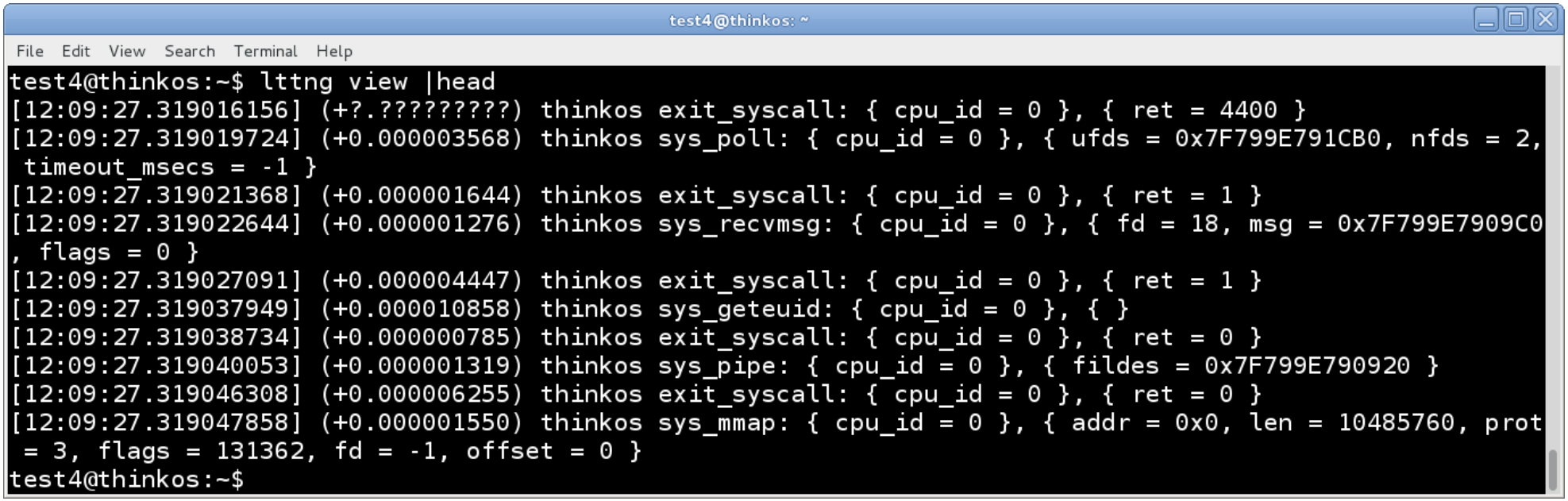
> Usage Scenarios

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

> Tracing: record kernel trace



> Tracing: view trace

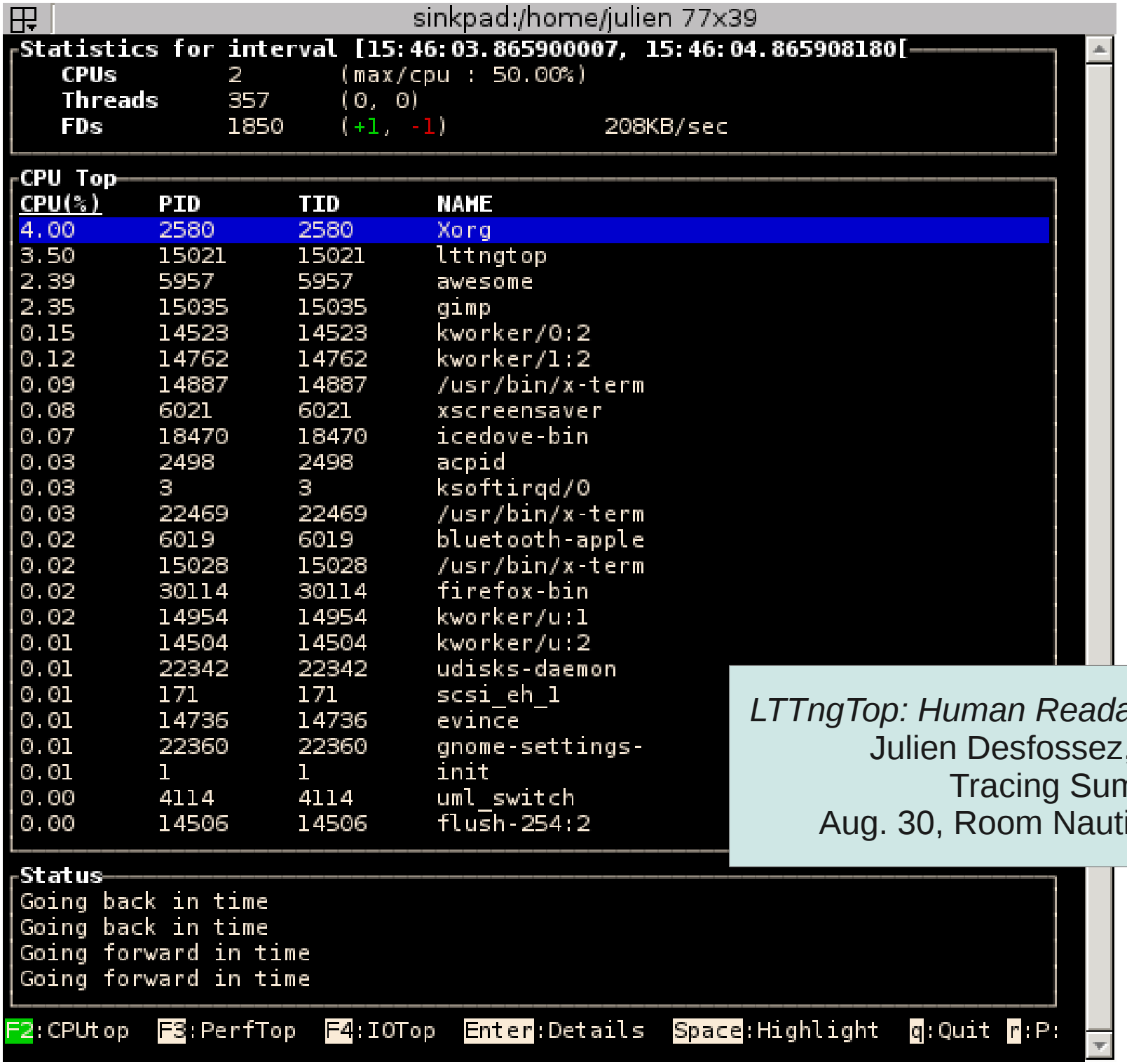


> LTTng 2.0 high-speed “strace”

lttng enable-event --syscall -a

compudj@squeeze-amd64: ~

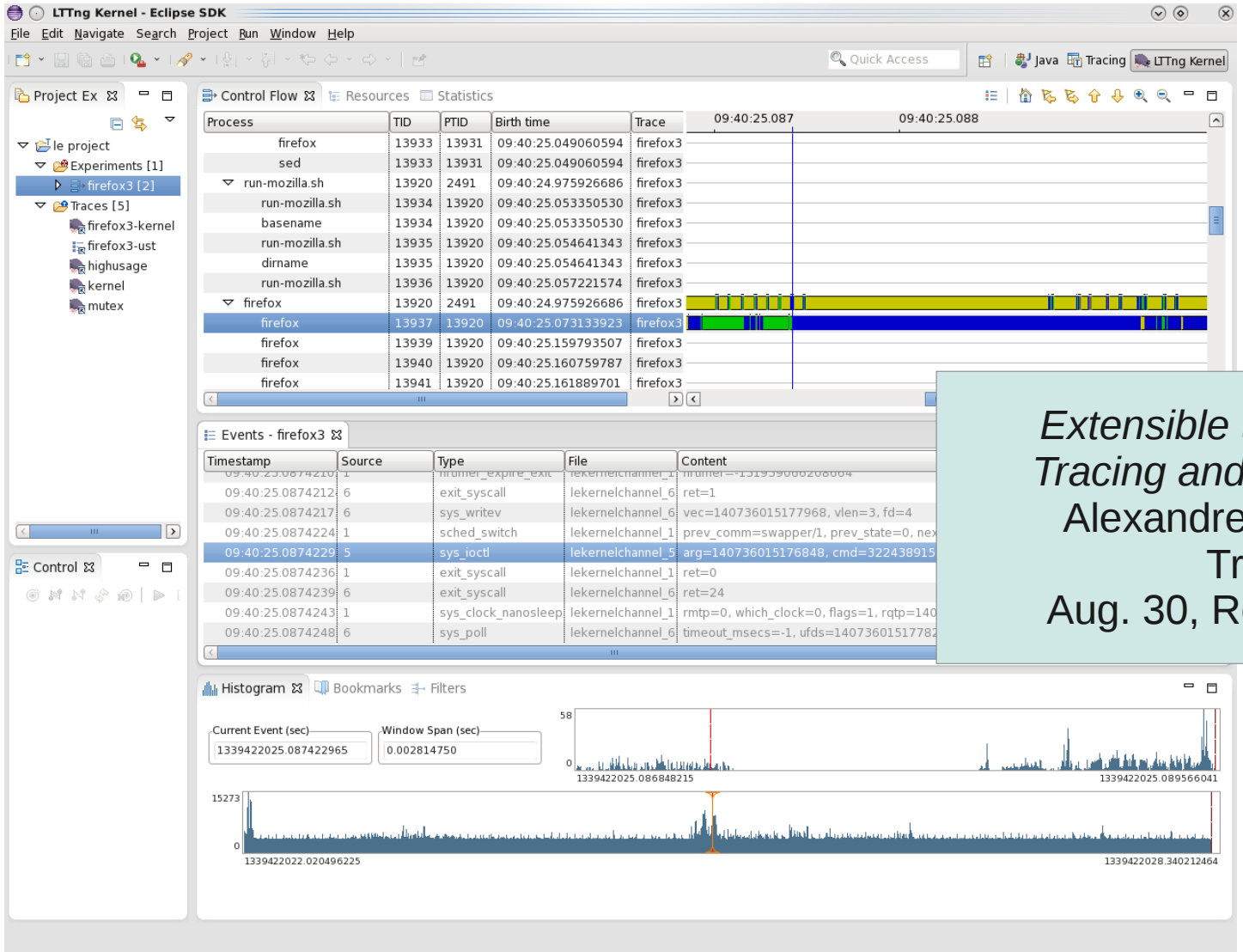
```
lp
name = sys_brk, stream.packet.context = { cpu_id = 1 }, event.fields = { brk = 28622848 }
name = exit_syscall, stream.packet.context = { cpu_id = 1 }, event.fields = { ret = 28622848 }
name = sys_read, stream.packet.context = { cpu_id = 1 }, event.fields = { fd = 3, buf = 0x1B48008, count = 9645 }
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 }
```



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

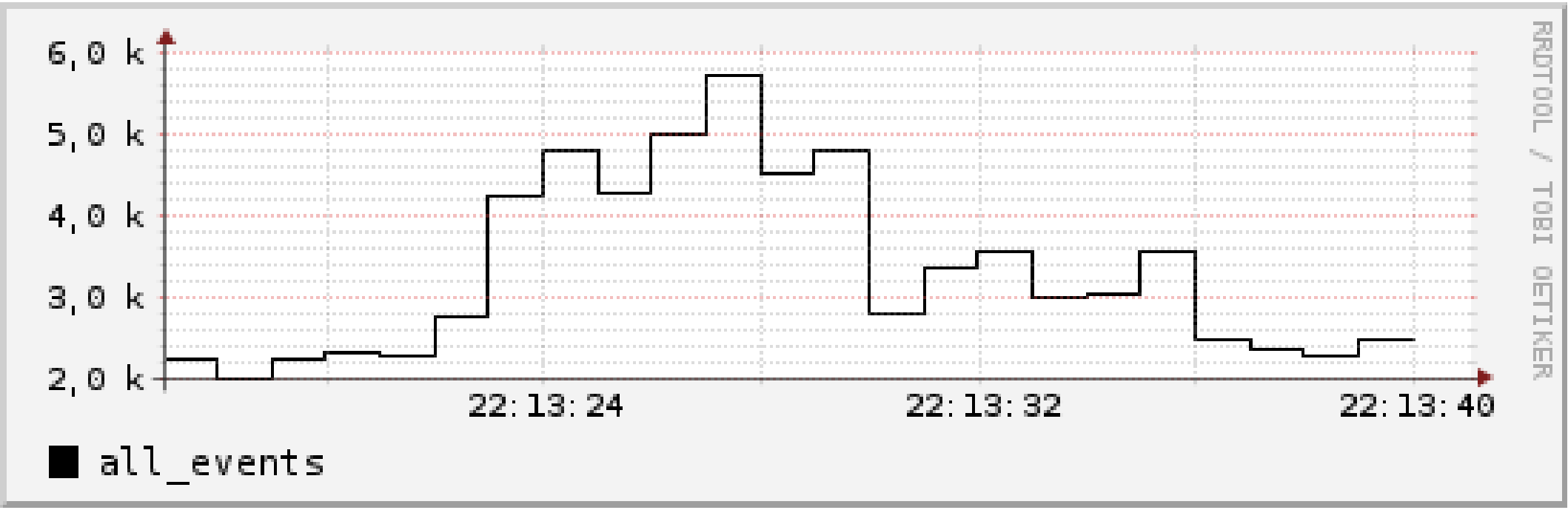
> Eclipse Linux Tools Project: LTTng support

- 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

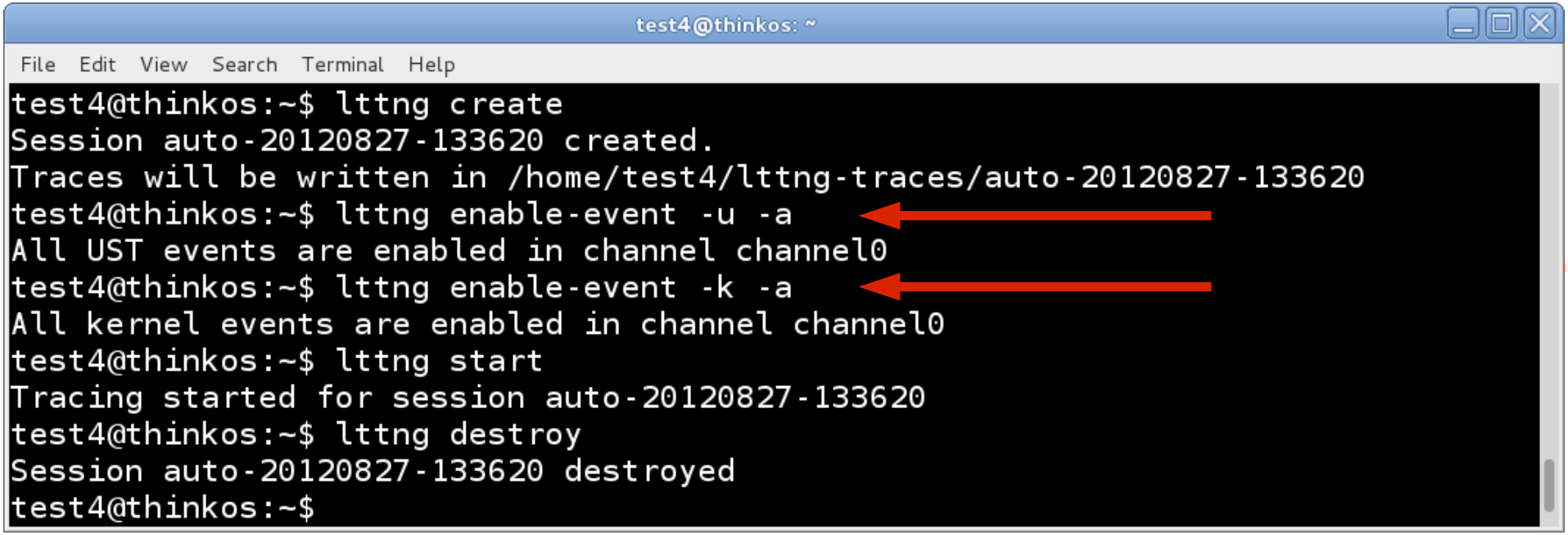


http://git.dorsal.polymtl.ca/~jdesfossez?p=lttng-graph

> Viewer Libraries and Bindings

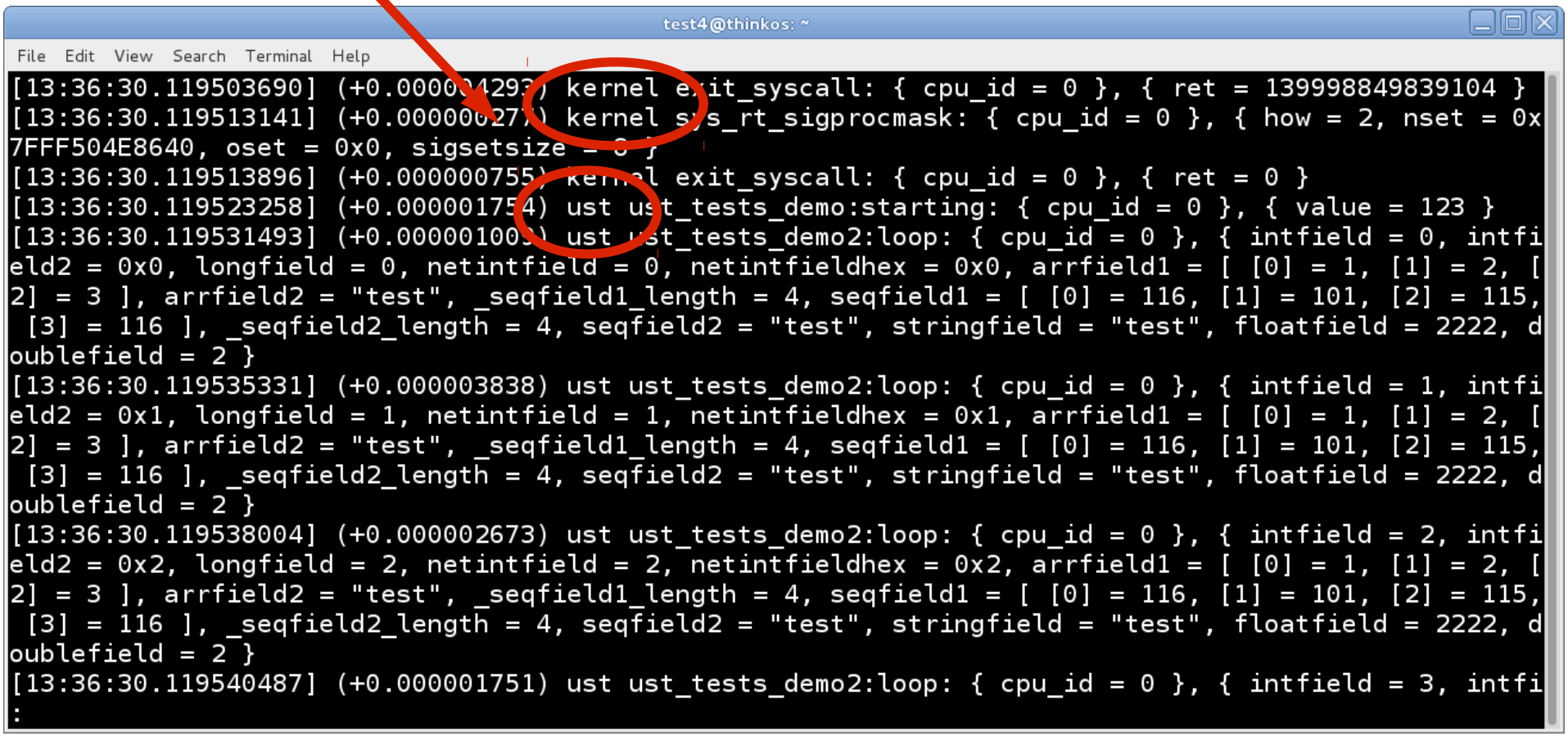
- Lib Babeltrace: a C/C++ library for reading CTF traces (MIT BSD-style license)
- Python bindings over Lib Babeltrace planned to be merged into Babeltrace 1.1
  - http://git.efficios.com/?p=babeltrace.git bindings/python branch.

> Tracing across kernel and user-space



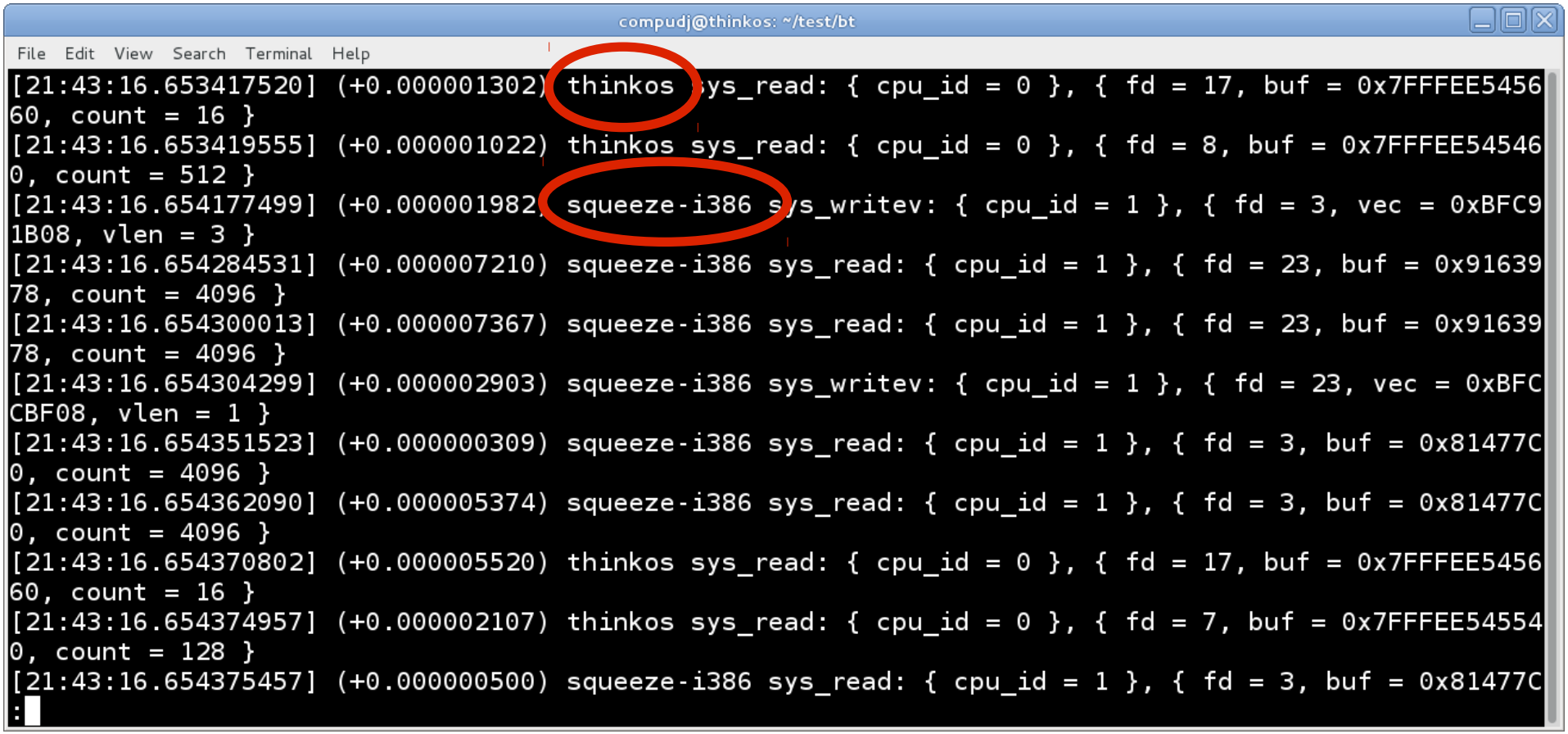
> Tracing across kernel and user-space (2)

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

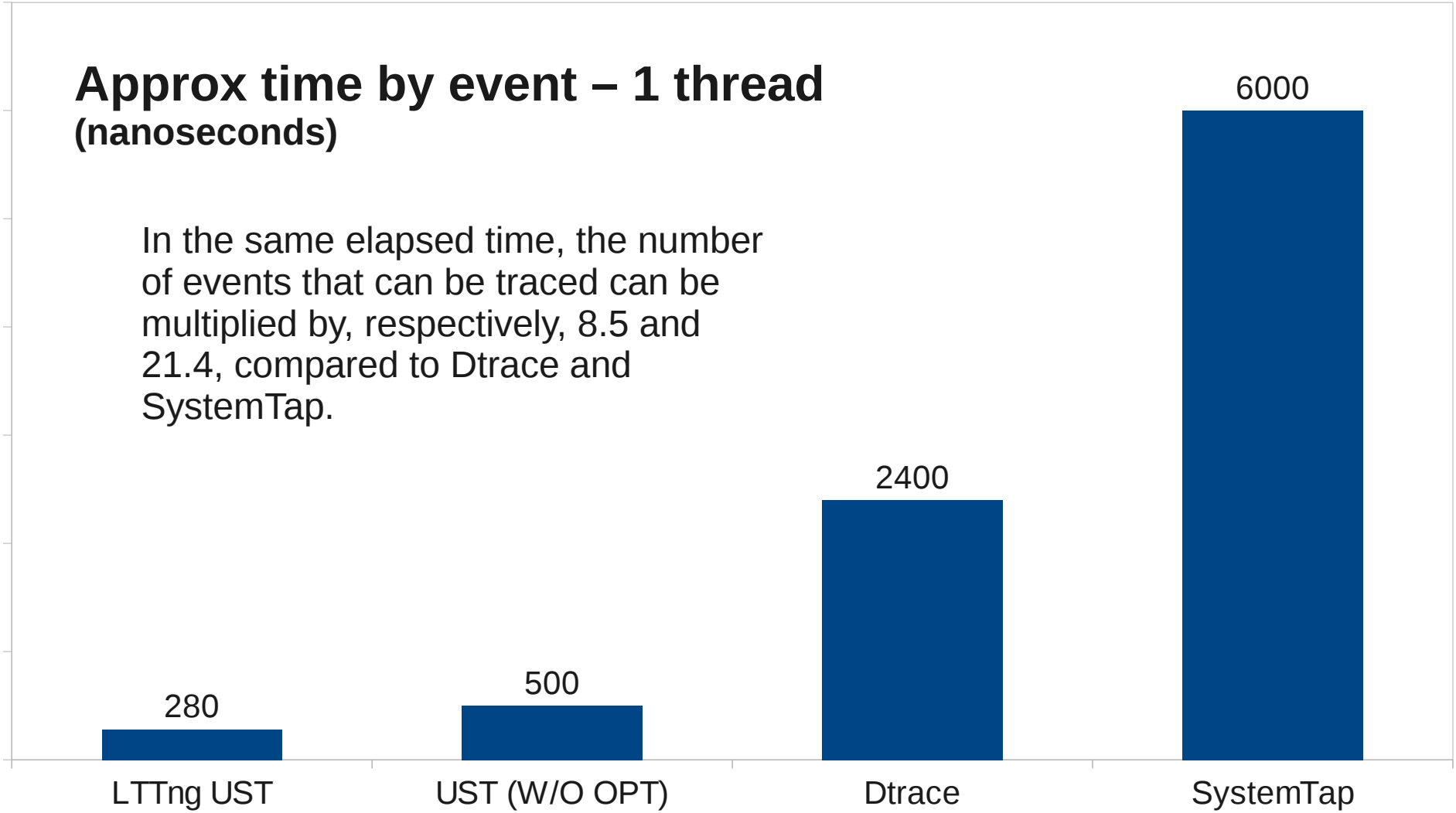


> Tracing across nodes

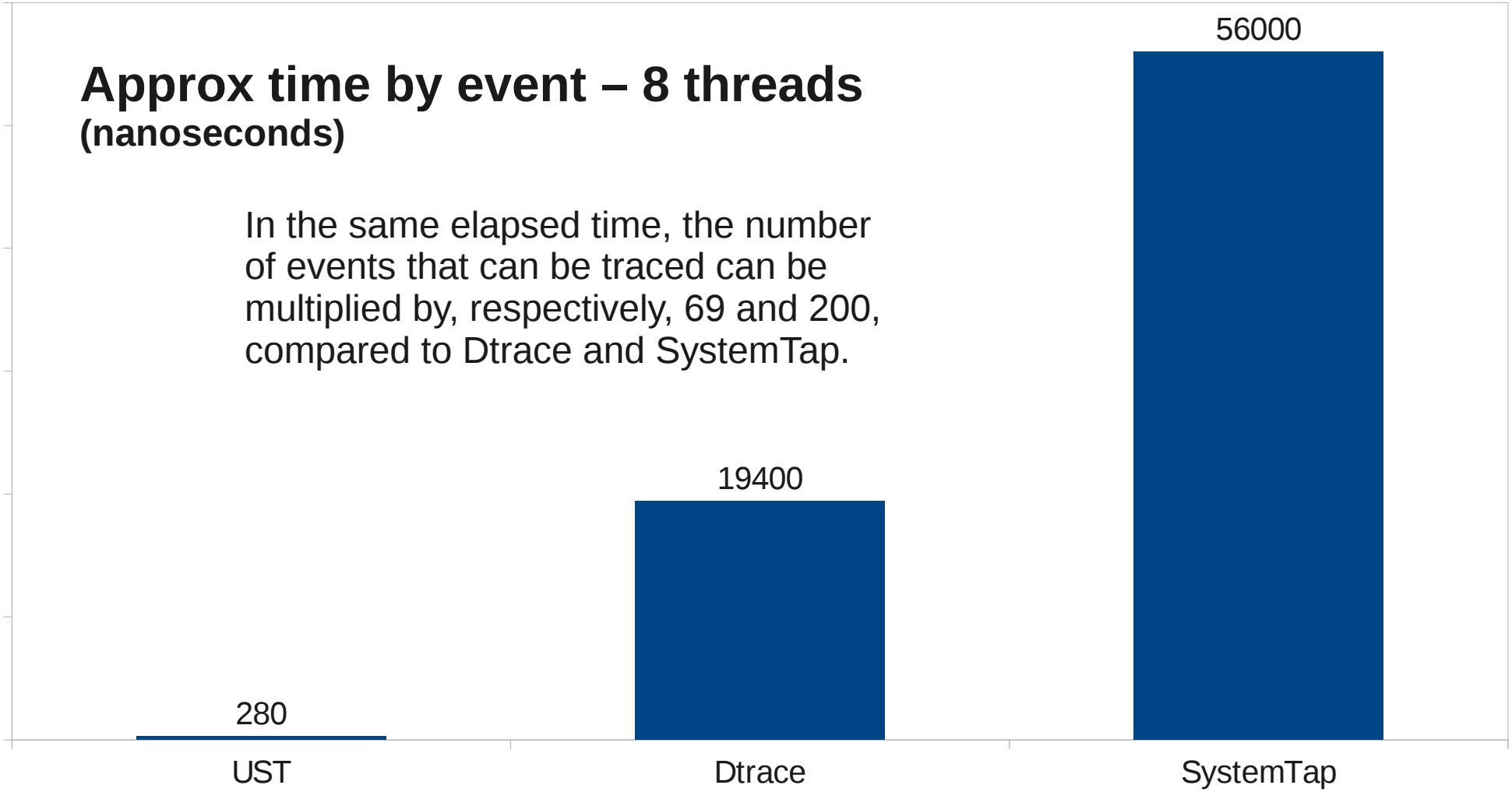
babeltrace --clock-force-correlate trace1 trace2 |grep -e sys_read -e sys_write |less



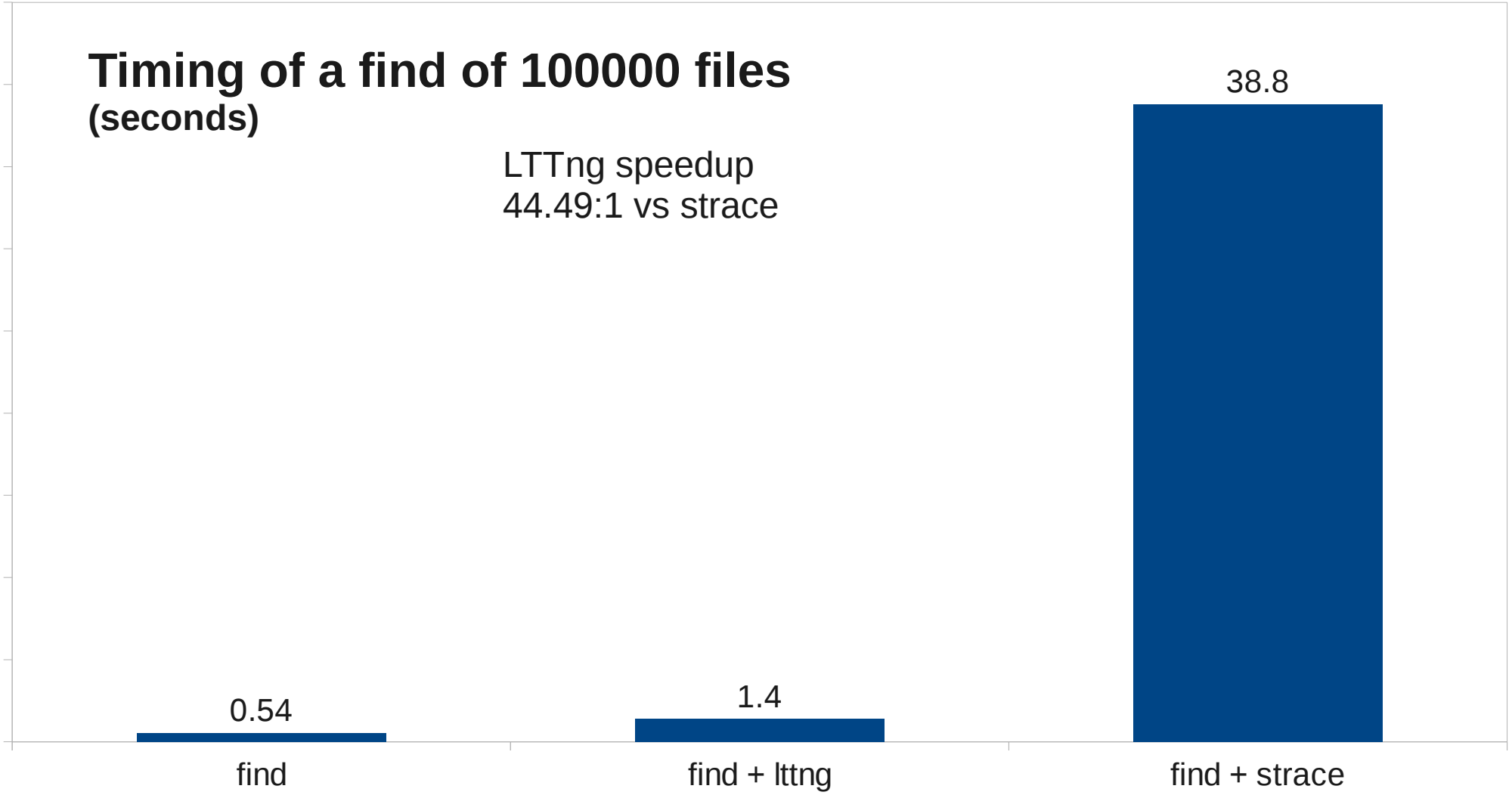
> Userspace Tracing Benchmark



> Userspace Tracing Benchmark



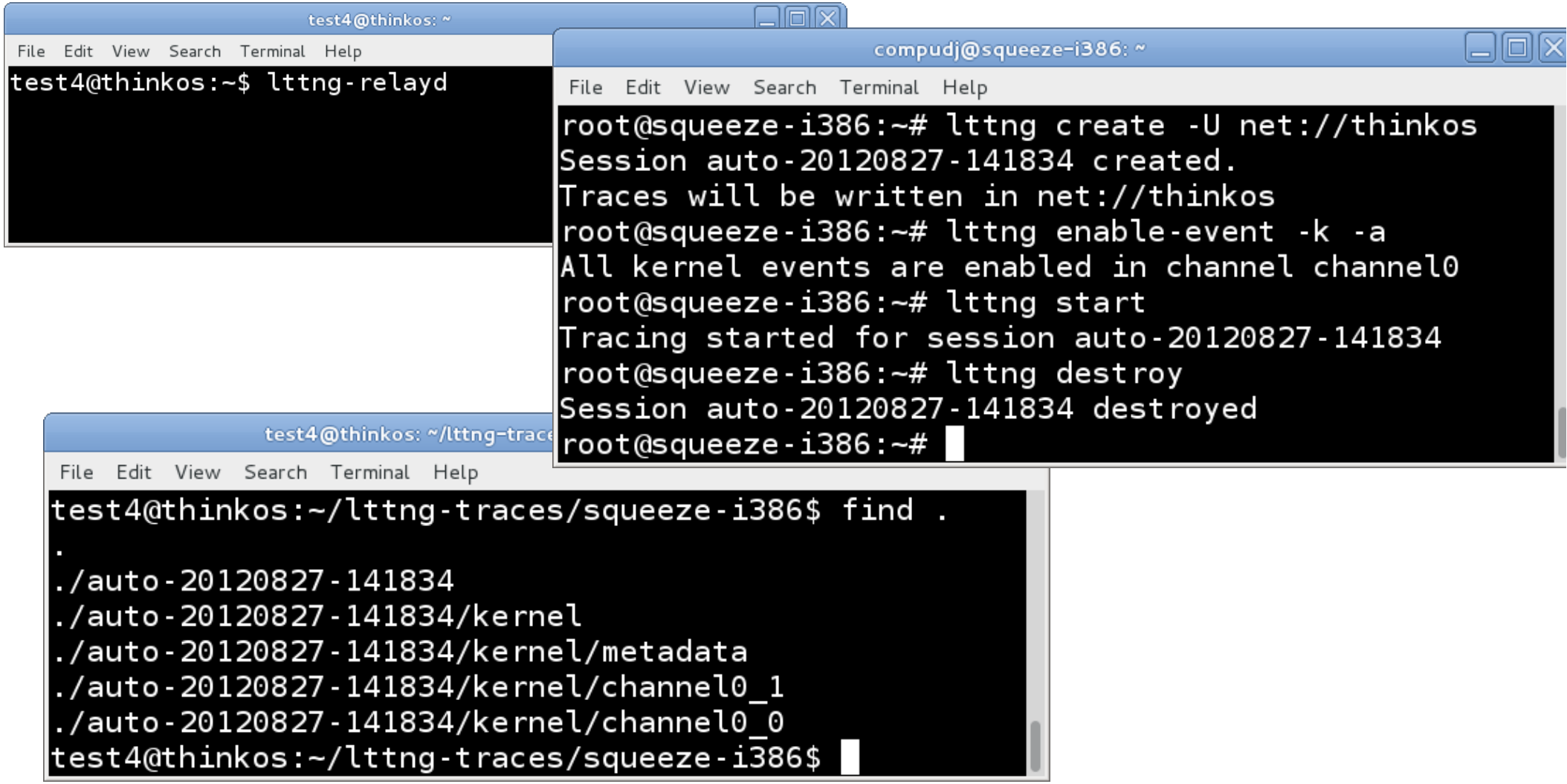
> Strace vs LTTng Tracing



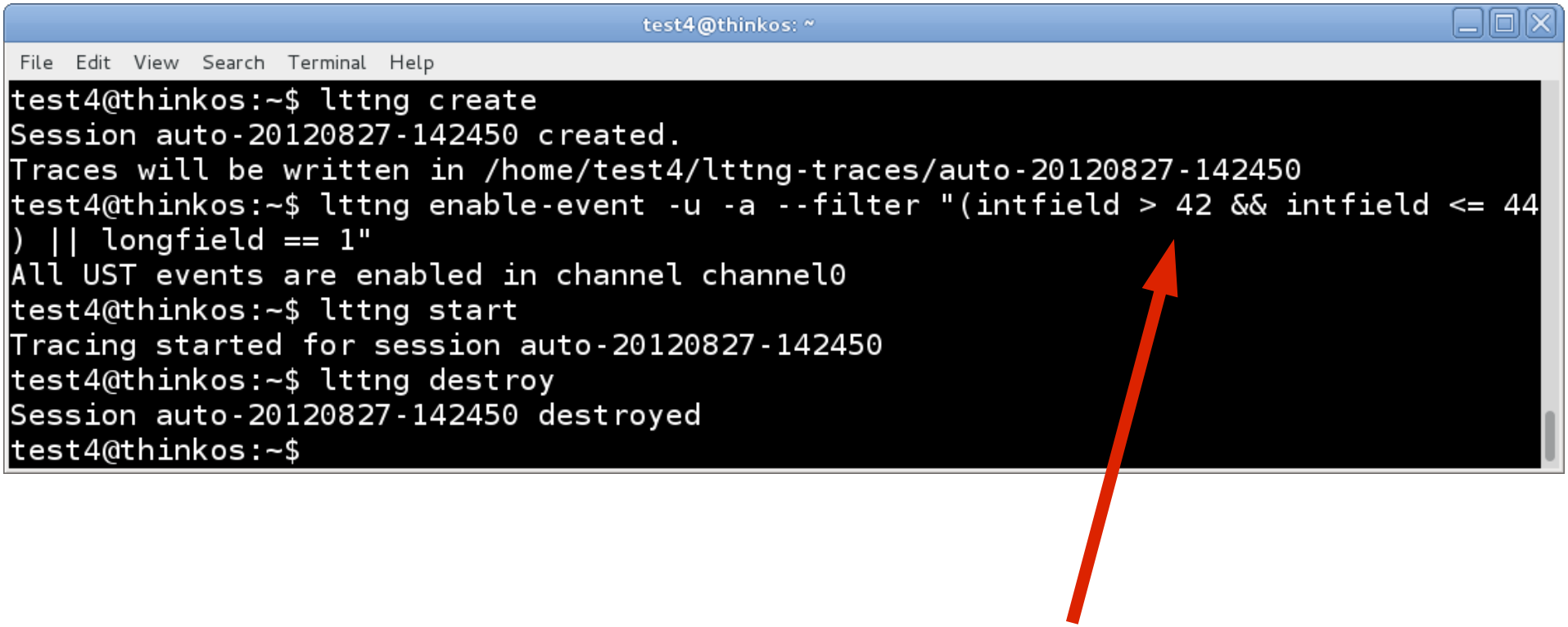
> New Features (LTTng 2.1)

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

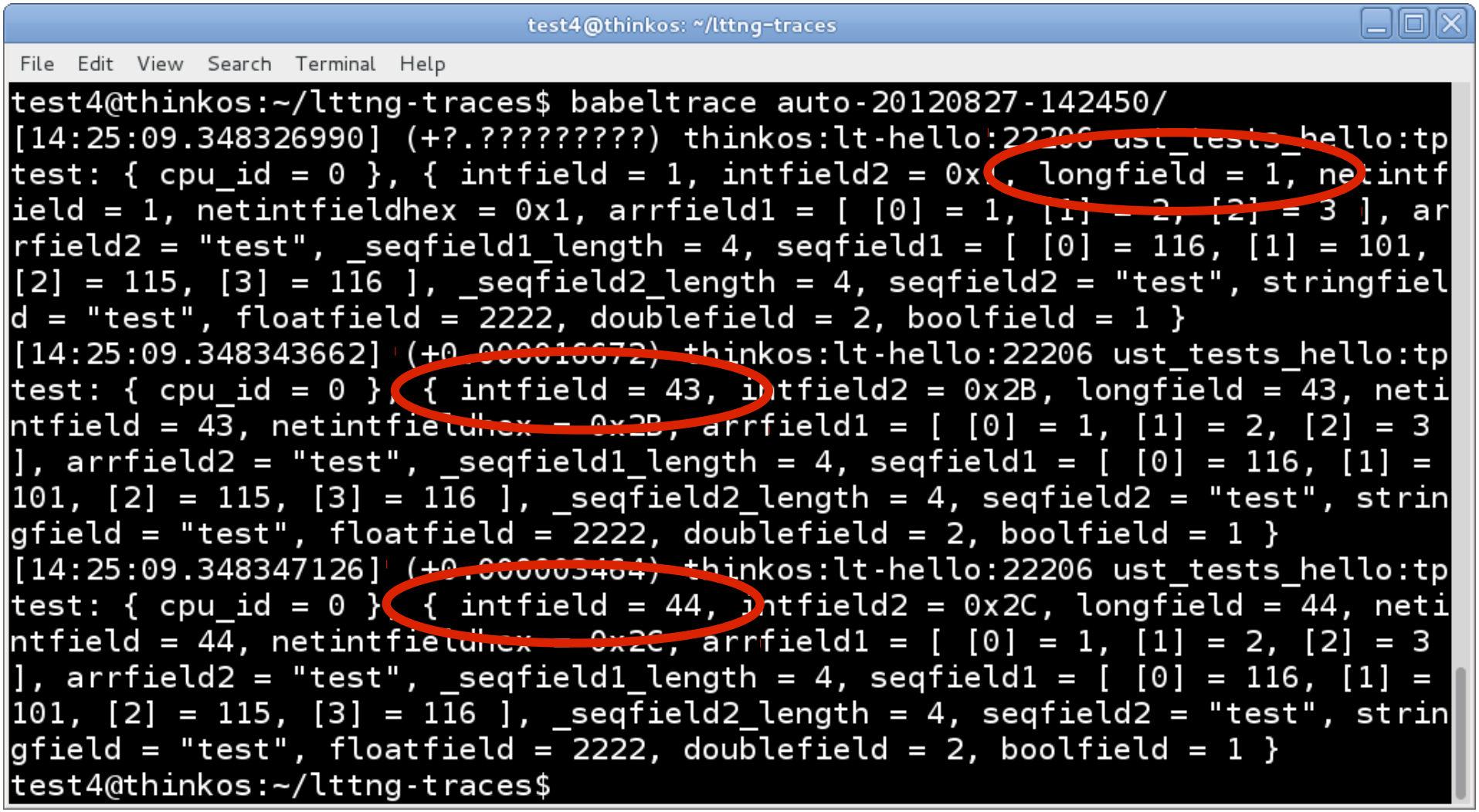
> Network Streaming over TCP



> LTTng-UST Filtering



> LTTng-UST Filtering (2)



> 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.

Tracing Summit 2012

Tomorrow 8h30 in Nautilus 3

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

EfficiOS

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