LTTng as a New Monitoring Tool

dgoulet@efficios.com

EfficiOS

whoami

- David Goulet, Software Developer, EfficiOS,
- Maintainer of LTTng-tools project

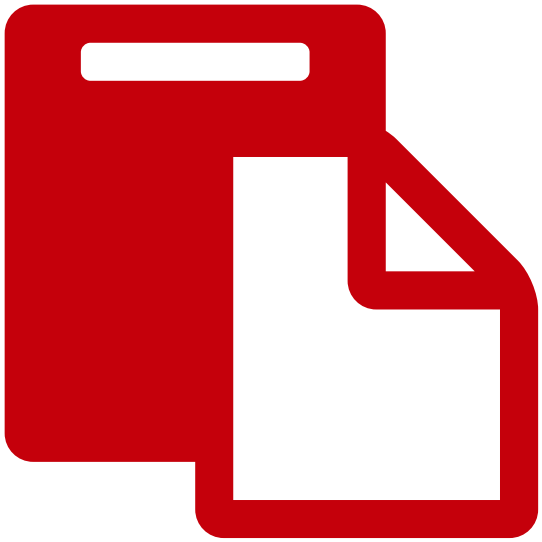
Content

- Quick overview of LTTng 2.x
- Everything else you need to know!
- Recent features & future work.

What is tracing?

- Recording runtime information without stopping the process
  - Enable/Disable event(s) at runtime
- Usually used during development to solve performance problems
- Lots of possibilities on Linux: LTTng, Perf, ftrace, SystemTap, strace, etc.

Overview of LTTng 2.x



Overview of LTTng 2.x

✔ Unified user interface, kernel and user space tracers combined. (**No need to recompile kernel**)

Trace output in a unified format (CTF)

Low overhead,

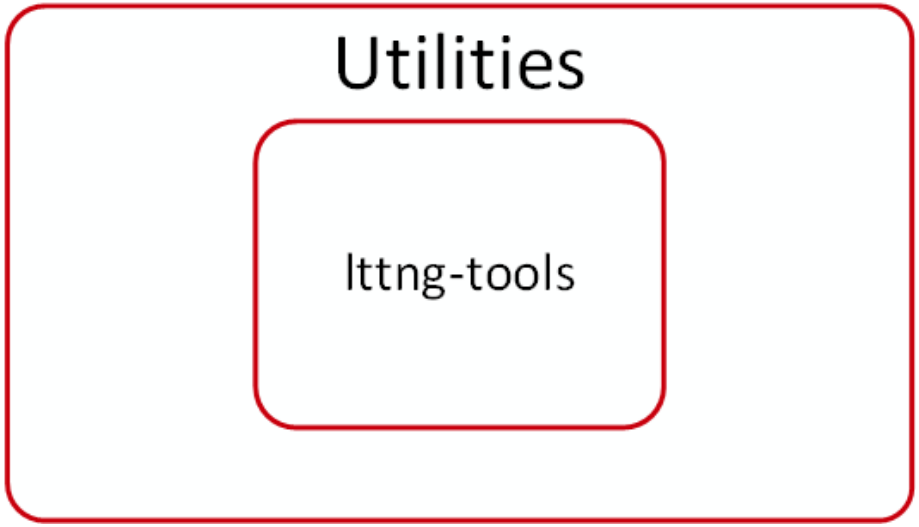
Shipped in distros: Ubuntu, Debian, Suse, Fedora, Linaro, Wind River, etc.

Tracers

- lttng-modules: kernel tracer module, compatible with kernels from 2.6.38* to 3.11.x,
- lttng-ust: user-space tracer, in-process library.

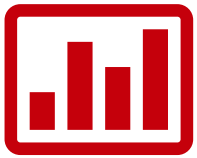
* Kernel tracing is now possible on 2.6.32 to 2.6.37 by backport of 3 Linux Kernel patches.

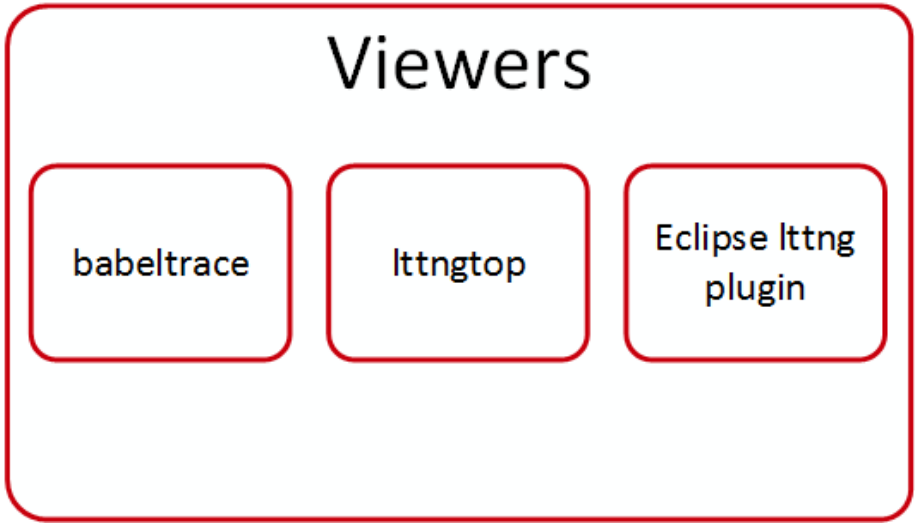
Utilities



- lttng-tools: cli utilities and daemons for trace control,
  - lttng: cli utility for tracing control,
  - lttng-ctl: tracing control API
  - lttng-sessiond: tracing registry daemon,
  - lttng-consumerd: extract trace data,
  - lttng-relayd: network streaming daemon.

Viewers





- babeltrace: cli text viewer, trace converter, plugin system,
- lttngtop: ncurse top-like viewer,
- Eclipse lttng plugin: front-end for lttng, collect, visualize and analyze traces, highly extensible.

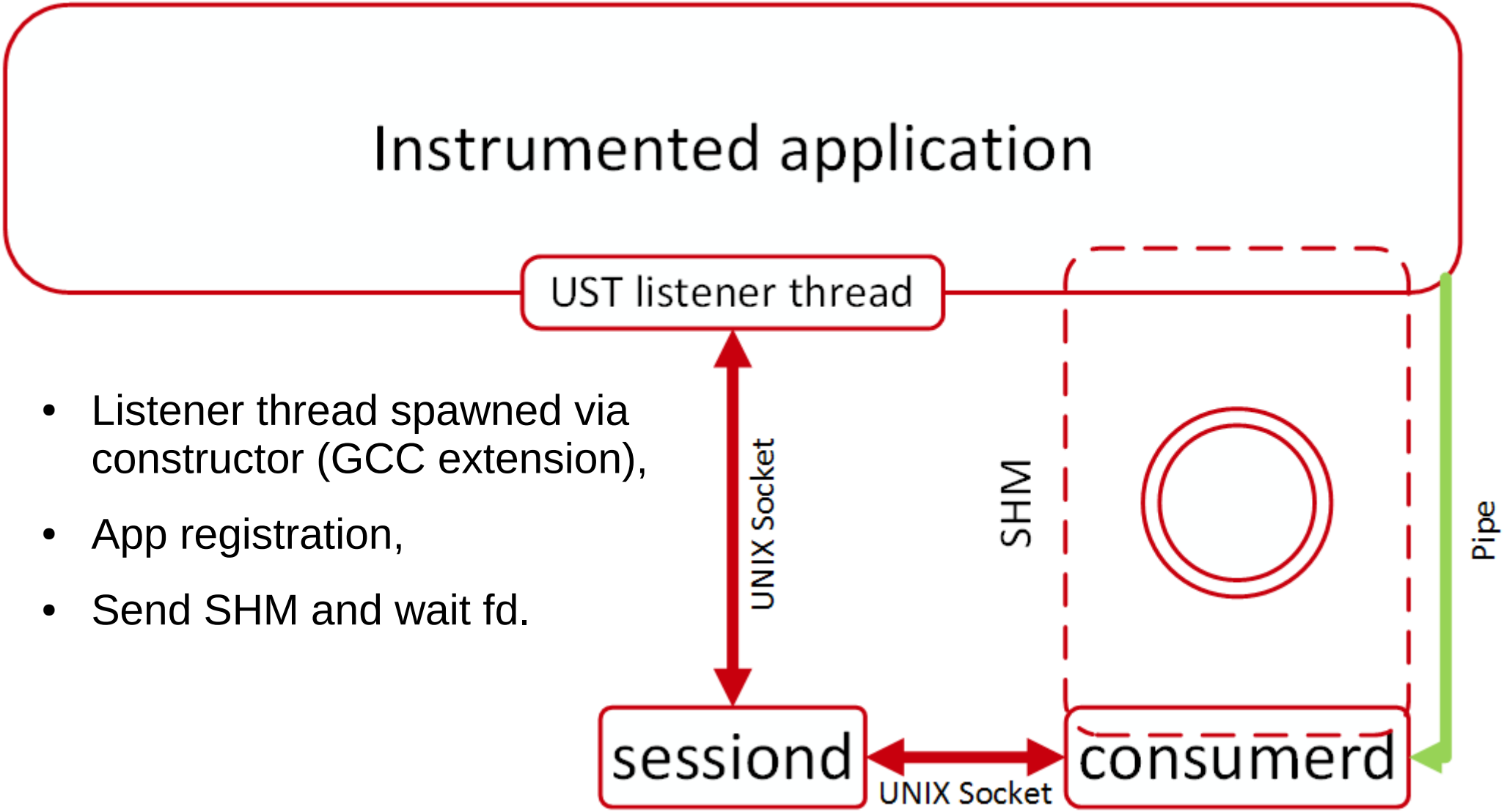
LTTng-UST – How does it work?

- Users instrument their applications with static tracepoints,
- liblttng-ust, in-process library, dynamically linked with application,
- Session setup, etc.,
- Run app, collect traces,
- Post analysis with viewers.

Tracing session - Setup

Session setup	$ lttng create
User-space event enabling	$ lttng enable-event -u -a
Start tracing	$ lttng start

Tracing session - A wild app appears



- Listener thread spawned via constructor (GCC extension),
- App registration,
- Send SHM and wait fd.

Time for the cool stuff

Tracing session example

```
$ lttng create
$ lttng enable-event -k sched_switch
$ lttng enable-event -k –-syscall -a
$ lttng start
$ sleep 2
$ lttng stop
$ lttng view | wc -l
8669
$ lttng destroy
```

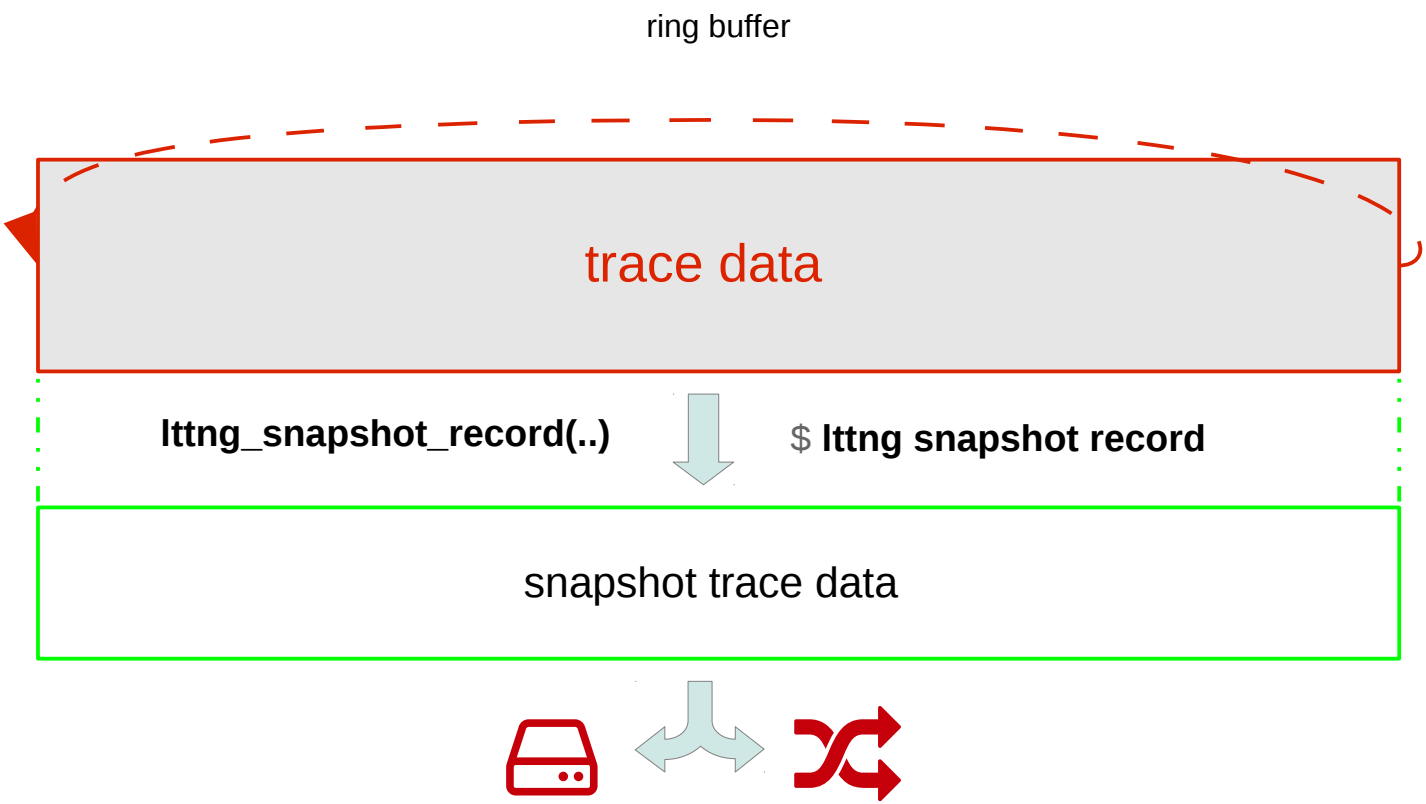
Tracing session example

```
[11:30:42.204505464] (+0.000026604) dalia
sys_read: { cpu_id = 3 }, { fd = 3, buf =
0x7FD06528E000, count = 4096 }

...

[11:30:42.204601549] (+0.000021061) dalia
sys_open: { cpu_id = 3 }, { filename =
"/lib/x86_64-linux-gnu/libnss_compat.so.2", flags
= 524288, mode = 54496 }

...

[11:30:42.205484608] (+0.000006973) dalia
sched_switch: { cpu_id = 1 }, { prev_comm =
"swapper/1", prev_tid = 0, prev_prio = 20,
prev_state = 0, next_comm = "rcuos/0", next_tid =
18, next_prio = 20 }
```

Snapshot

At **any** point in time, a snapshot can be taken of a the **current** trace buffers.

Overwrite mode meaning flight recorder



Flight recorder session + snapshot

```
$ lttng create --snapshot
$ lttng enable-event -k sched_switch
$ lttng enable-event -k –-syscall -a
$ lttng start
$ ...
$ lttng snapshot record
Snapshot recorded successfully for session
auto-20131019-113803
$ babeltrace
/home/julien/lttng-traces/auto-20131019-113803/sn
apshot-1-20131019-113813-0/kernel/
```

Snapshot – Real world use case

Core dump

- Custom handler with lttng -> /proc/sys/kernel/core_pattern
- Snapshot record on coredump

IDS – Log Manager (ex: Splunk, Nagios)

- Trigger system snapshot on alert
- Gather system data regularly
- Corrolate system events with logs

Performance profiling

- Server applications
- Kernel
- Hardware latency

Live

As the trace is being created, you extract and can analyze the data.

Continous Analysis

- Extract data with live streaming for analysis on an other machine

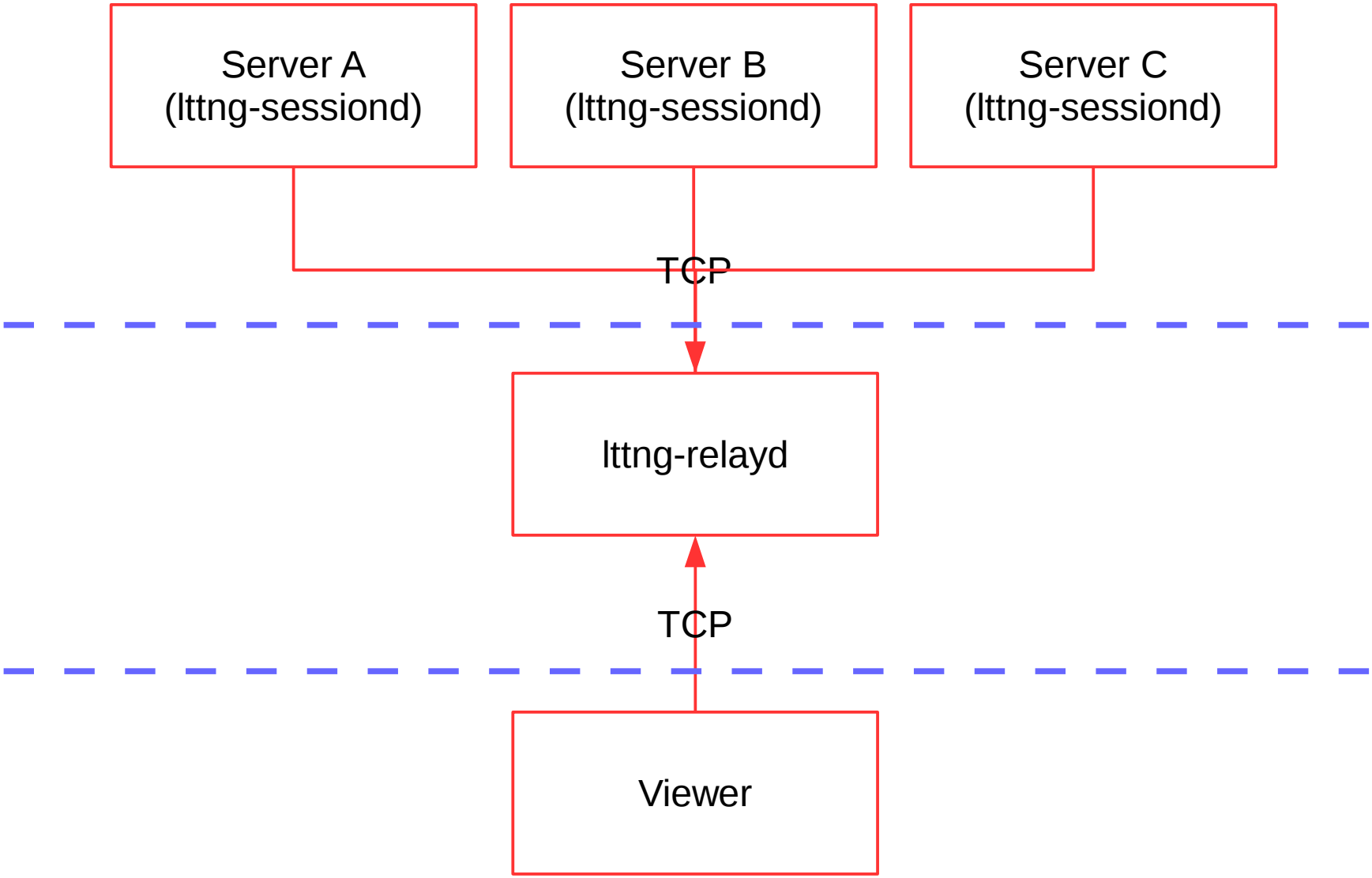
Cluster-level analysis

- Gather traces from multiple machines
  - Load balancing analysis
  - Latency detection

System Administration

- Get data of faulty machine “on-demand”

Infrastructure integration



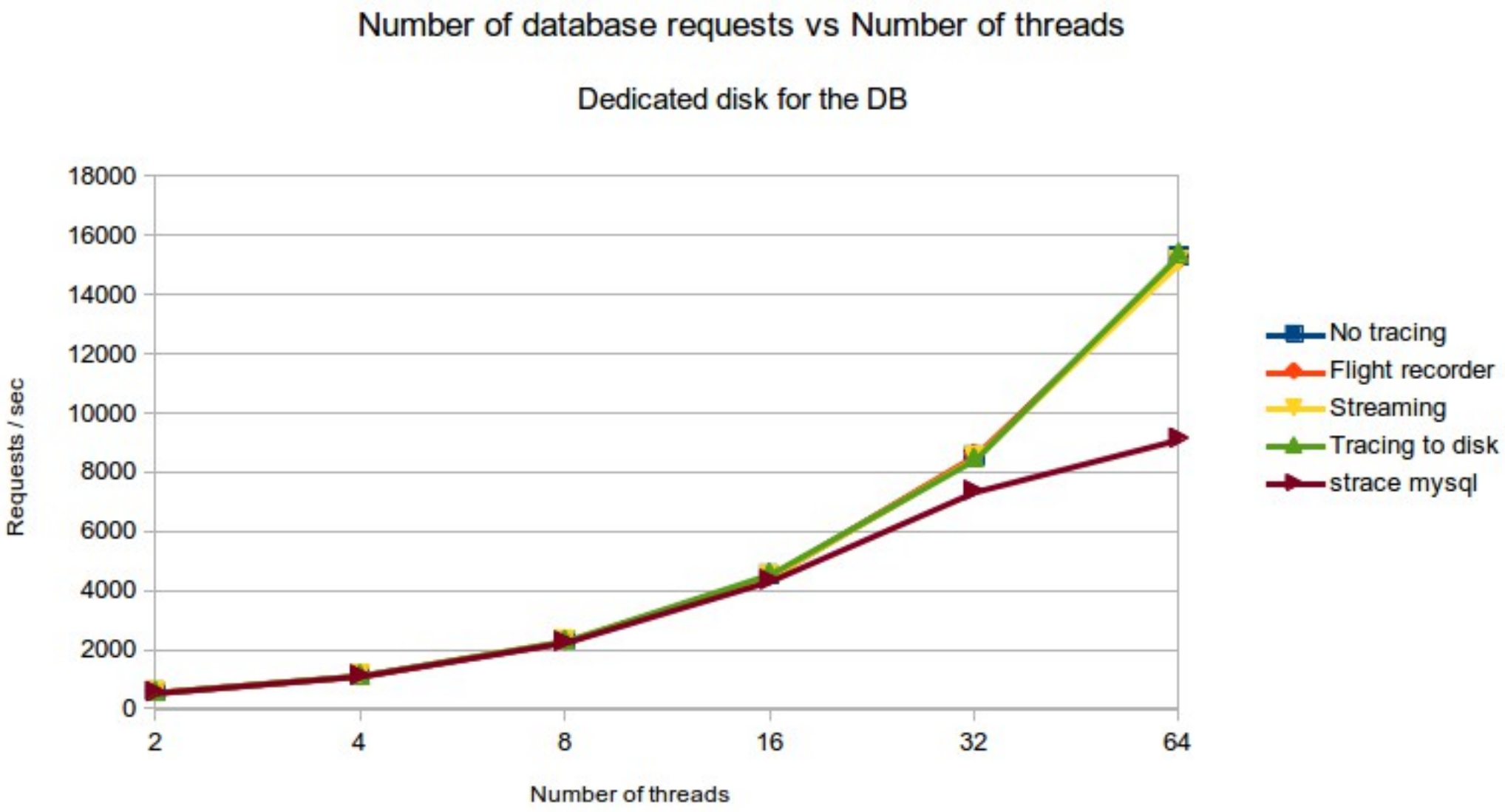
LTTngTop

Pretty impressive tool

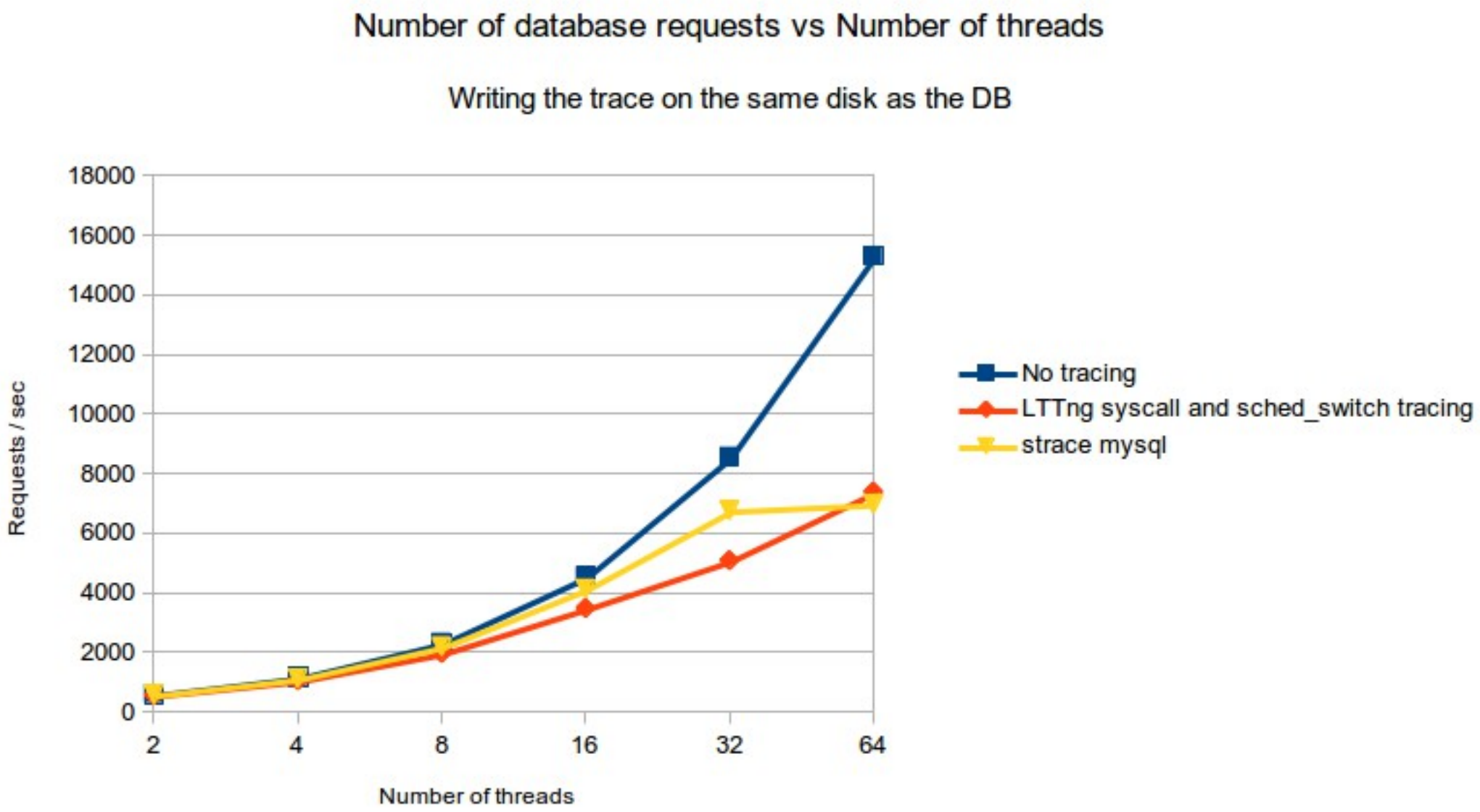
Performance results

- The test runs for 50 minutes
- Each snapshot is around 7MB, 100 snapshots recorded (one every 30 sec.)
- The whole strace trace (text) is 5.4GB with 61 million events recorded
- The whole LTTng trace (binary CTF) is 6.8GB with 257 million events recorded with 1% of event lost.

Dedicated disk for trace



Shared disk with DB and trace



Recent features & future work

Recent features

- 2.4 (Époque Opaque) - Upcoming
  - Snapshot (local and remote),
  - Live tracing,
    - Analyze data while being created
  - Java JUL support
    - Java Util Logging

Future work

- Hardware tracing support
- Trace trigger
  - Trigger custom actions
- Android port for kernel and UST tracers
- Dynamic instrumentation support (Dyninst)

Questions ?

?

EfficiOS

https://www.efficios.com

https://lttng.org

lttng-dev@lists.lttng.org

@lttng_project