CloudOpen Europe 2013

Efficient and Large-Scale Infrastructure Monitoring with Tracing

EfficiOS

Julien.desfossez@efficios.com ✉

Content

- Overview of tracing and LTTng
- LTTng features for Cloud Providers
- LTTng as a monitoring tool
    - Crash dumps
    - “Real-time” monitoring
- Large-scale low-level tracing
    - Infrastructure integration
    - Performance results
    - Virtualisation specific analysis
- LTTngTop
- Future work

Tracing

- Recording run-time information without stopping the process
- Usually used during development to solve performance problems
- Lots of alternatives on Linux: LTTng, Perf, ftrace, SystemTap, strace, etc.

LTTng 2.x

- Unified user interface, API, kernel and user-space tracers
- Trace output in CTF (Common Trace Format)
- Low overhead
- Modules only (**no kernel compilation needed**)
- Shipped in distros: Ubuntu, Debian, SuSE, Fedora, Linaro, Wind River, etc.

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) sinkpad
sys_read: { cpu_id = 3 }, { fd = 3, buf =
0x7FD06528E000, count = 4096 }

...

[11:30:42.204601549] (+0.000021061) sinkpad
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) sinkpad
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 }
```

LTTng features for Cloud Providers

- LTTng 2.1 (12/2012): trace streaming
- LTTng 2.2 (06/2013): trace-file rotation
- LTTng 2.3 (09/2013): snapshots
- LTTng 2.4 (RC1 expected in November 2013): live trace reading

LTTng as a monitoring tool : Crash dumps

- Flight recorder
- Snapshot on demand
- Coredump handler (in extras/)

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/snapshot-1-20131019-113813-0/kernel/
```

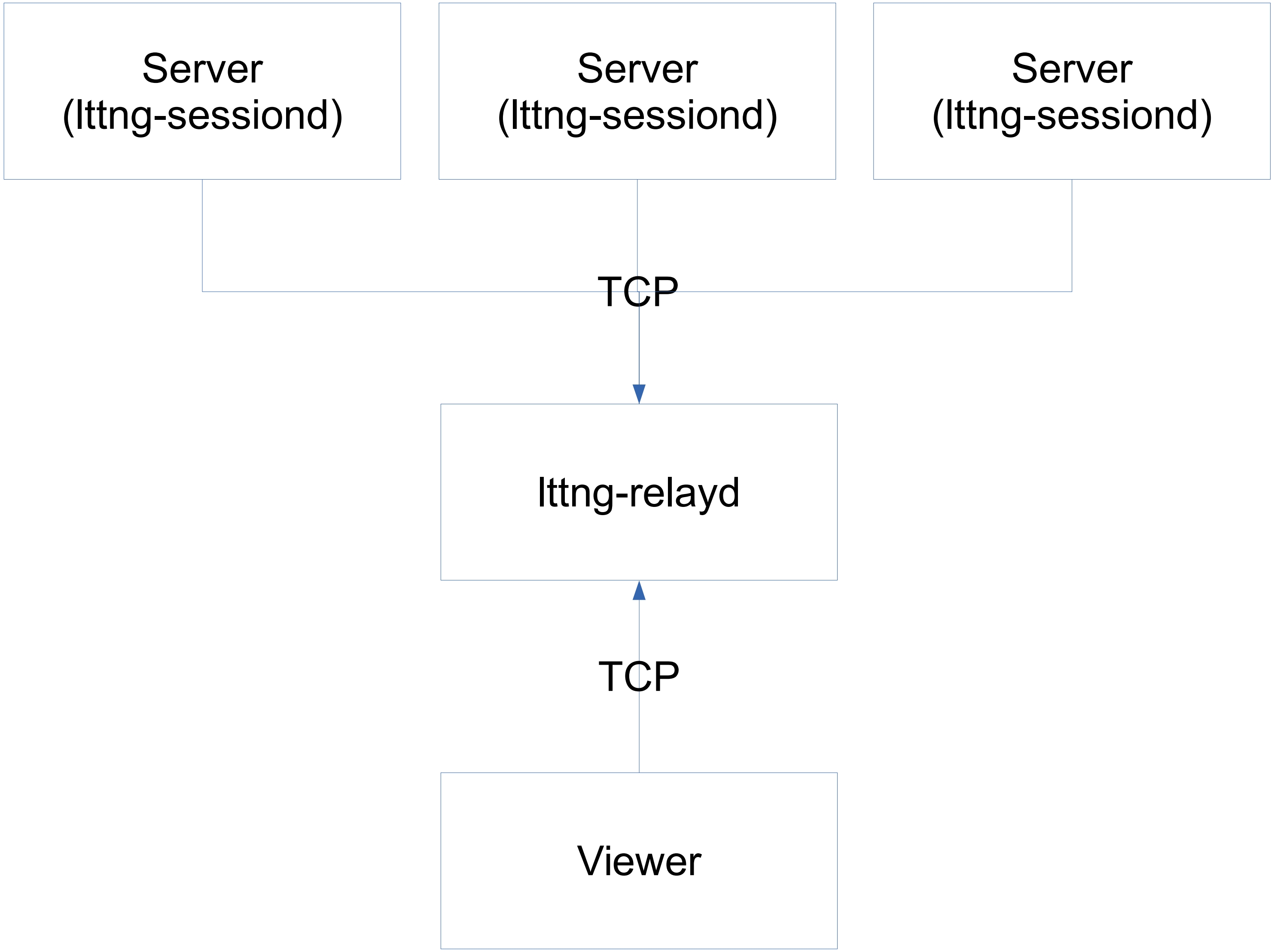
Coredump handler

```
# cat /proc/sys/kernel/core_pattern
|/path/to/lttng/handler.sh %p %u %g
%s %t %h %e %E %c
```

“Real-time” monitoring

- Read the trace while it is being recorded
- Local or remote session
- Configurable flush period

Infrastructure integration



Server (lttng-sessiond)

Server (lttng-sessiond)

Server (lttng-sessiond)

TCP

lttng-relayd

TCP

Viewer

Live streaming session

On the server to trace :

```
$ lttng create --live 2000000 -U net://10.0.0.1
$ lttng enable-event -k sched_switch
$ lttng enable-event -k --syscall -a
$ lttng start
```

On the receiving server (10.0.0.1) :

```
$ lttng-relayd -d
```

On the viewer machine :

```
$ lttngtop -r 10.0.0.1
```

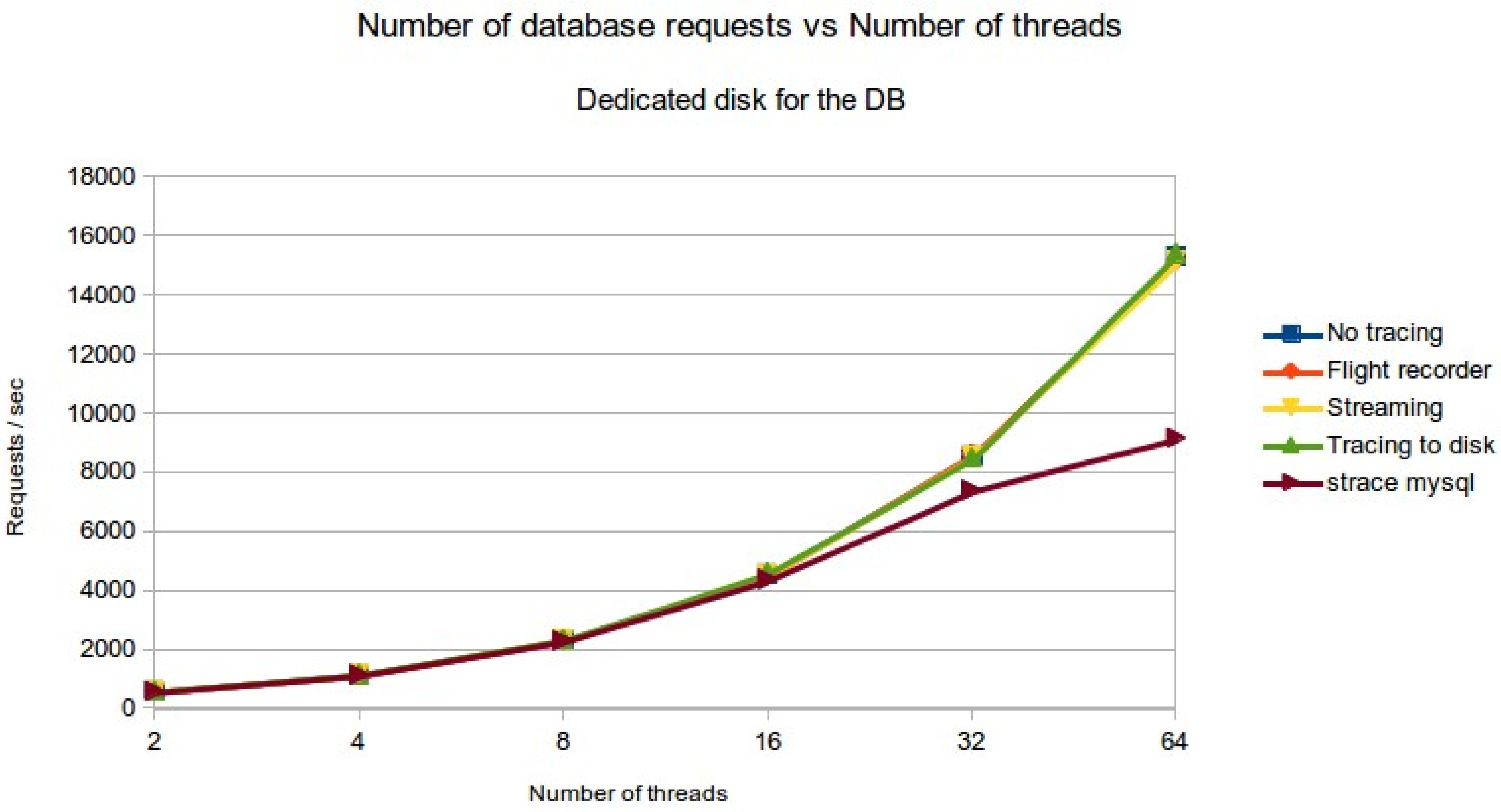
Performance results

- sysbench MySQL benchmark with increasing number of threads on a quad-core i7, 6GB RAM, 7200 RPM
- Tracing all system calls and sched_switch with LTTng in different modes :
    - Flight recorder with a snapshot recorded every 30 seconds
    - Streaming the trace to a remote server
    - Writing the trace on a dedicated disk
- Tracing all the threads of MySQL with strace to a dedicated disk

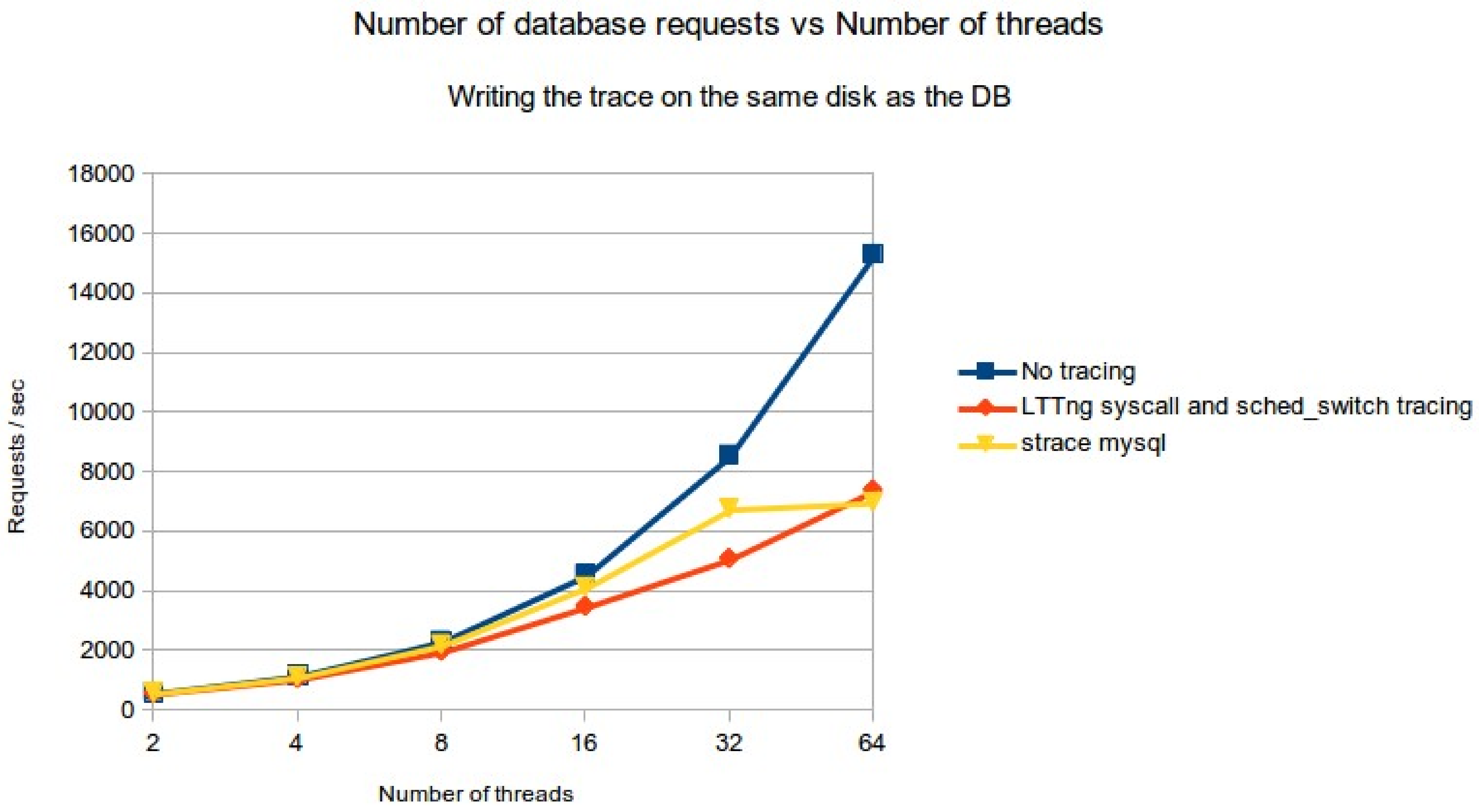
Performance results

- The test runs for 50 minutes
- Each snapshot is around 7MB, 100 snapshots recorded
- 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 lost events

Performance results



Sharing the disk with DB and trace

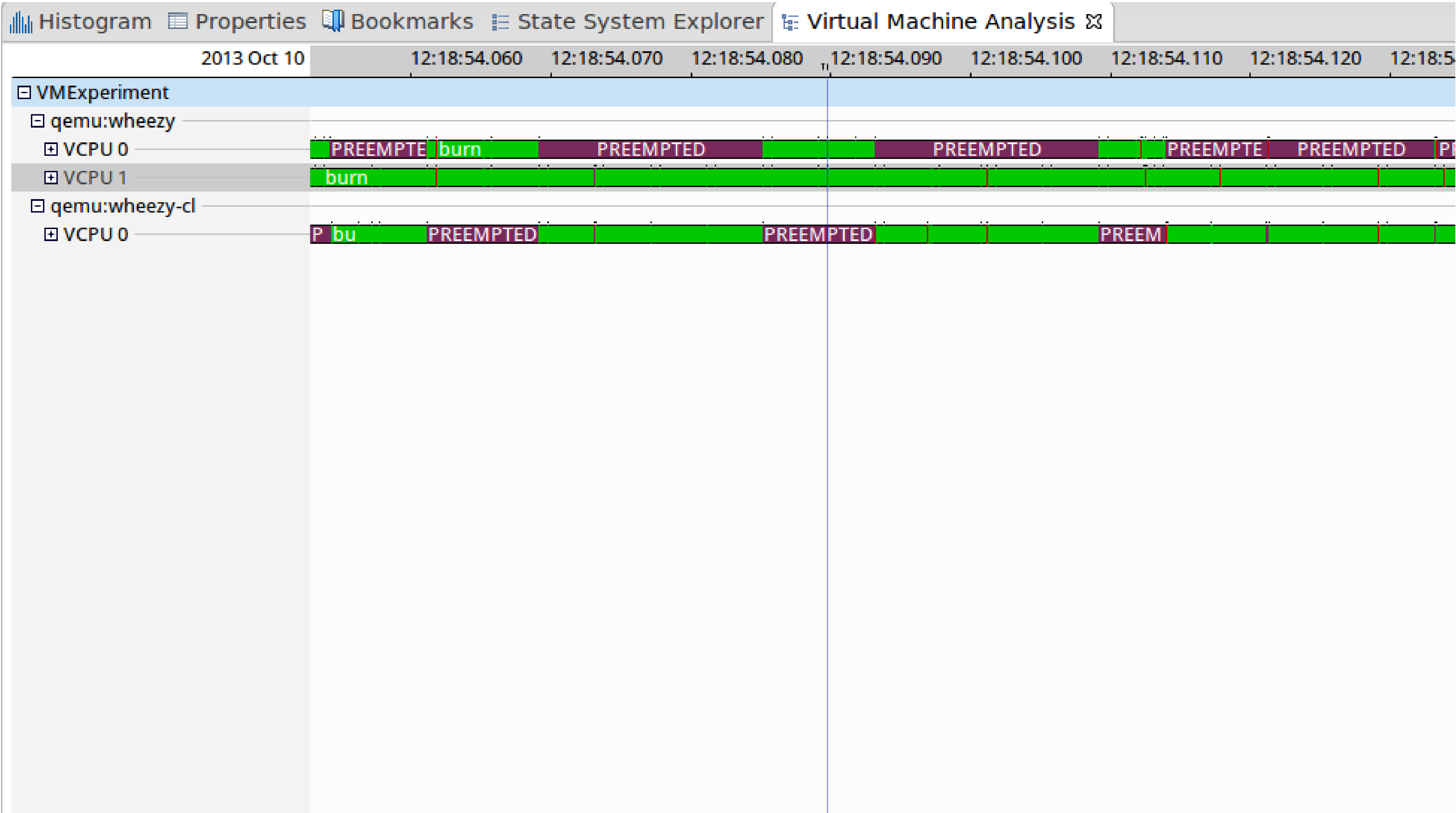


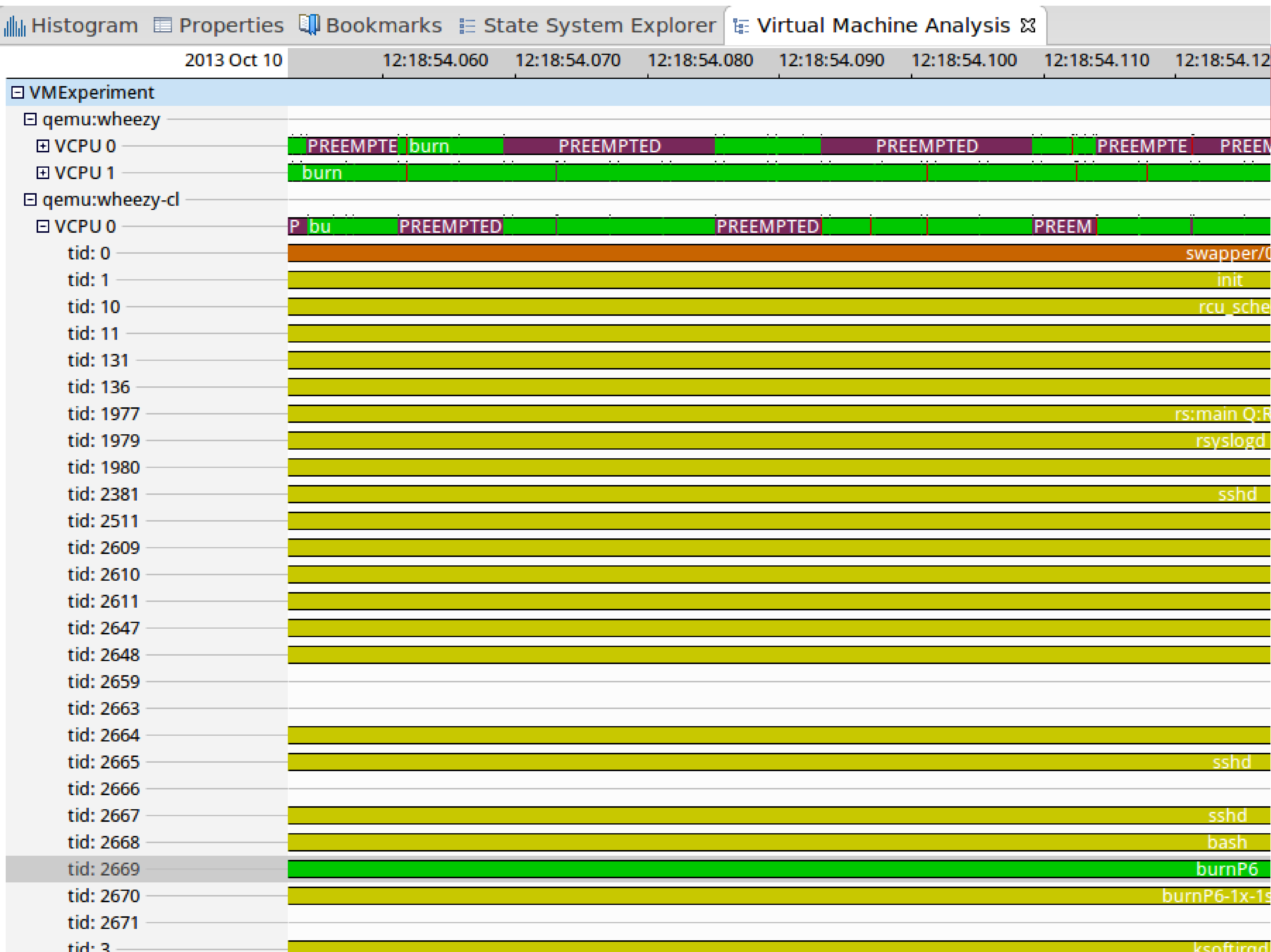
Performance result with virtualization

- 2 KVM VMs on the same host
- One is an apache web server
- The other one downloads a 5GB iso file from the first with wget
- Same LTTng instrumentation and setup (syscalls and sched_switch)
- No noticeable overhead when recording the trace on an external disk, network or snapshots.

Advanced KVM analysis

TMF Virtual Machine Analysis view by Mohamad Gebai

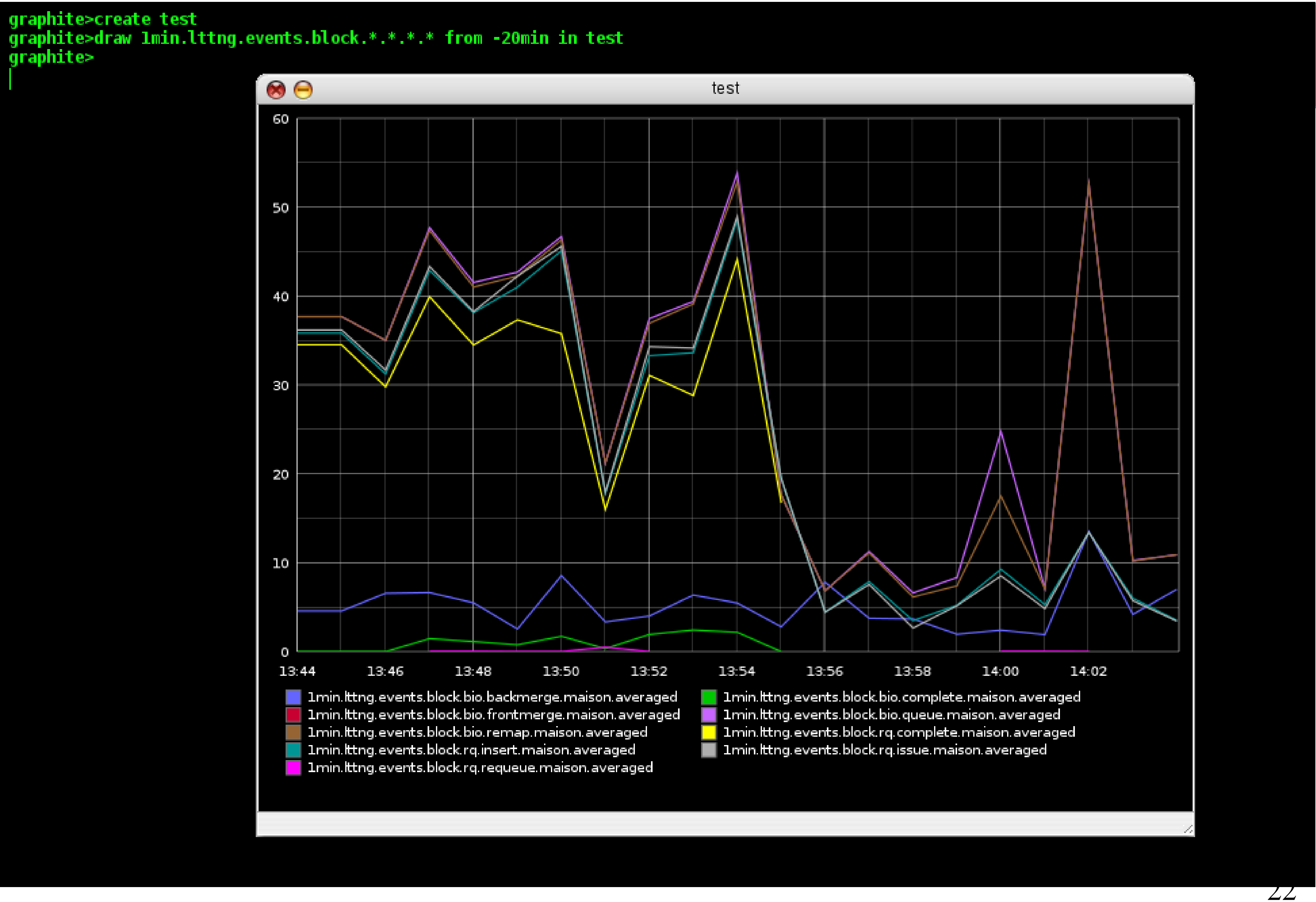




⊟ VMExperiment
⊟ qemu:wheezy
⊞ VCPU 0
⊞ VCPU 1
⊟ qemu:wheezy-cl
⊟ VCPU 0
tid: 0
tid: 1
tid: 10
tid: 11
tid: 131
tid: 136
tid: 1977
tid: 1979
tid: 1980
tid: 2381
tid: 2511
tid: 2609
tid: 2610
tid: 2611
tid: 2647
tid: 2648
tid: 2659
tid: 2663
tid: 2664
tid: 2665
tid: 2666
tid: 2667
tid: 2668
tid: 2669
tid: 2670
tid: 2671
tid: 3

LTTngTop

- Top-alike interface to read LTTng kernel traces
- CPU usage, per-process file activity, kprobes hit, per-process perf counter display
- Navigate in the trace second-by-second
- Read offline traces or connect to a relay for live-streaming
- Experimental in-memory live-reading



```
graphite>create test
graphite>draw 1min.lttng.events.block.*.*.*.* from -20min in test
graphite>
```

test
60
50
40
30
20
10
0
13:44
13:46
13:48
13:50
13:52
13:54
13:56
13:58
14:00
14:02
1min.lttng.events.block.bio.backmerge.maison.averaged
1min.lttng.events.block.bio.complete.maison.averaged
1min.lttng.events.block.bio.frontmerge.maison.averaged
1min.lttng.events.block.bio.queue.maison.averaged
1min.lttng.events.block.bio.remap.maison.averaged
1min.lttng.events.block.rq.complete.maison.averaged
1min.lttng.events.block.rq.insert.maison.averaged
1min.lttng.events.block.rq.issue.maison.averaged
1min.lttng.events.block.rq.requeue.maison.averaged

Future Work

- Integrate with already existing monitoring tools (graphite, Nagios, etc), beta already working
- Filter and pre-process the trace before sending
- Distribute the analysis
- Remote control of the tracer
- More advanced triggers to collect snapshots, start/stop tracing, etc.

Install it

- Packages for your distro (`lttng-modules, lttng-ust, lttng-tools, userspace-rcu, babeltrace`)
- For Ubuntu : PPA for daily build (`lttngtop`)
- Or from the source, see `http://git.lttng.org`

Questions ?

?

EfficiOS

www.efficios.com

lttng.org

lttng-dev@lists.lttng.org

@lttng_project