LTTng 2.0: Application, Library and Kernel tracing within your Linux distribution.

E-mail: mathieu.desnoyers@efficios.com
> Buzzword compliant!

- LTT: Linux Trace Toolkit
- “ng” : Next Generation
- 2.0!

\textbf{LTTng 2.0!}

- All we miss is a recursive acronym. ;}
Mathieu Desnoyers

EfficiOS Inc.

- http://www.efficios.com

Author/Maintainer of

- LTTng, LTTng-UST, Babeltrace, LTTV, Userspace RCU
> 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.

- **Embedded**
  - System development, maintenance of deployed systems.

- **Server/Desktop software**
  - Qemu/KVM, MariaDB.
> 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.
LTTng 2.0 Toolchain Overview

- LTTng CLI
- Control Library
- Session Daemon
- URCU
- UST
- Kernel Tracer
- Consumer Daemon
- Trace file using CTF
- Viewers
  - Babeltrace
  - LTTngTop
  - LTTv
  - Eclipse/tmf
> LTTng 2.0 Tracing Session

- Multiple domains:
  - Kernel, User-space
  - Eventually: Hypervisor, multiple hosts

- Controlled through same UI/API:
  - lttnng -k ...
  - lttnng -u ...

- Correlation across domains (common time-line)
- Viewed by pointing trace viewer to the top-level trace collection directory
> LTTng 2.0 Kernel Tracer

- Build against a vanilla or distribution kernel, without need for additional patches,
- Tracepoints, Function tracer, Perf CPU Performance Monitoring Unit (PMU) counters, kprobes, and kretprobes support,
- Supports multiple tracing sessions, flight recorder mode, snapshots, ...
LTTng 2.0 Kernel Tracer

- ABI based on ioctl() returning anonymous file descriptors
  - implemented a top-level DebugFS “lttng” file.
- Lib Ring Buffer, initially developed generically for mainline Linux kernel (as a cleanup of the LTTng 0.x ring buffer) has been merged into LTTng 2.0.
- Exports trace data through the Common Trace Format (CTF).
> 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
User-space Tracer Features

- TRACEPOINT_EVENT() API for application/library static instrumentation (planned sdt.h gdb/systemtap integration).
- Per-user tracing.
- System-wide tracing.
  - “tracing” group: no need to be root to perform system-wide tracing.
> LTTng-UST 2.0
User-space Tracer Infrastructure

- `libust` in-process library.
- `libust` constructor registers to session daemon upon application startup, waits for commands.
- This rendez-vous point allows
  - Tracing across all system's applications/libraries
  - Tracing on per-application executable name basis
- Fast: trace applications without per-event system call overhead into per-cpu/process buffers.
TRACEPOINT_EVENT(ust_tests_hello_tptest,
    TP_PROTO(int anint, long *values,
             char *text, size_t textlen,
             double doublearg, float floatarg),
    TP_ARGS(anint, values, text, textlen,
            doublearg, 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)
    )
)
> 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:

```c
void fct(void)
{
    tracepoint(ust_tests_hello_tptest, i, values, 
              text, strlen(text), dbl, flt);
}
```
> tracepoint_printf()

- Feature planned.
- Debug-style tracing.
- `tracepoint_printf(name, “fmt”, …);`
- Augment Common Trace Format to store format strings.
- Export only binary data through buffers.
- Pretty-printing performed at post-processing.
> 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 insecure global buffers feature planned too.
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 Consumers

- Spawned by the tracing sessions daemon
- Design guide-lines:
  - Minimal access, aiming at a design where sessiond opens all files, consumers just copy data between memory maps and file descriptors (received though UNIX socket credentials).
- Disk output (splice, mmap).
- In-place mmap buffer consumption (lttngtop).
- Planned network transport.
> LTTng CLI / liblttngctl

- Unified control interface for kernel and user-space tracing
  - “lttng” git-alike command line interface
  - All tracing control commands available through an API: liblttngctl and lttng.h
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
> 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.

• Available at: http://www.efficios.com/ctf
> Distributions

- Distributions shipping LTTng 0.x
  - Wind River Linux, Montavista, STLinux, Linaro, Yocto, Mentor Embedded Linux, ELinOS, Novell SuSE Enterprise RT Linux.

- Packages
  - Debian and Ubuntu
    - UST, Userspace RCU, LTTV

- Working closely with Ubuntu and Debian to have LTTng 2.0 toolchain ready for the next Ubuntu LTS.
> Distributions

• **Fedora**
  
  - Fedora packages available for LTTng 0.x user-space tracing and trace analysis, LTTng 2.0 packages soon to be integrated,

• **RHEL 6**

  - In discussion with Redhat developers to backport tracepoint patches needed for LTTng 2.0 kernel tracer support. Else will target RHEL 7.
Conditional tracing for UST 2.0

- Work planned for 2012
- Dynamic filtering of event payloads
- Very fast: zero-copy, filtering before any interaction with the ring buffer.
> Trace analysis tools

- **Graphical**
  - Eclipse Linux Tools Project: LTTng support.
  - LTTV

- **Text-based**
  - LTTngtop
  - LTTV
  - Babeltrace
> Eclipse Linux Tools Project: LTTng support

- [http://wiki.eclipse.org/Linux_Tools_Project/LTTng](http://wiki.eclipse.org/Linux_Tools_Project/LTTng)
> LTTV

- Will be ported to LTTng 2.0 soon.
> Questions?

**LTTng 2.0 pre-releases available at**
[http://lttng.org/lttng2.0](http://lttng.org/lttng2.0)

- http://www.efficios.com
- LTTng Information
  - [http://lttng.org](http://lttng.org)
  - ltt-dev@lists.casi.polymtl.ca