User-space Tracing with UST

Mathieu Desnoyers
mathieu.desnoyers@efficios.com

David Goulet
david.goulet@polymtl.ca

Michel Dagenais
michel.dagenais@polymtl.ca

April 6-8, 2011
Collaboration Summit
Presenters

• Mathieu Desnoyers
  • EfficiOS Inc.
  • Work funded by Ericsson
• David Goulet
  • Focus on production systems
  • Academia (Ecole Polytechnique de Montréal)
  • Industry (Révolution Linux)
Status of LTTng

- Shipped in
  1. Wind River Linux, Montavista, STlinux, Linaro, Yocto
  2. Novell Enterprise edition

- Packages
  1. Debian and Ubuntu
     - UST, Userspace RCU, lttv
What is UST?

- UST, a.k.a. LTTng-UST, is the LTTng User-space tracer
- Entirely stand-alone
  - Works on vanilla Linux kernels
- Trace
  - Applications
  - Libraries
1. Current UST Features
2. LTTng User Interface Unification
   · Kernel / User-space tracing
3. Collaboration
Current UST Features

Interface Unification
Collaboration
1. Current UST Features

- **Flexibility**
  - Enable/Disable any *tracepoint* before and during tracing
- **External data buffers**
  - Crash: the UST consumer still able to get the data out!
- **Performance**
  - 190ns/event (high data volume tracer)
- **Linear scalability**
- *Disk output and flight recorder* mode
1. Current UST Features - Instrumentation

- Markers

```c
int do_search(
    Operation *op,  /* info about the op to which we're responding */
    SlapReply *rs /* all the response data we'll send */ )
{
    struct berval base = BER_BVNULL;
    ber_len_t siz, off, i;

    trace_mark(ust, search_event, "DN %s", op->o_req_dn.bv_val);
}
```

So easy to use! Here to stay!

- Tracepoints/TRACE_EVENT
1. Current UST Features - Trace Clock

- **CLOCK_TRACE** (13 Jan 2011)
  - LTTng kernel 0.240 or higher
  - UST 0.11 or higher
  - Timestamp synchronized (kernel and user-space)
    - Common time reference for simultaneous viewing
  - Only for x86 and x86_64
    - Very easy to do for other arch.
  - We need that mainline :)

User-space Tracing with UST
Current UST Features

Interface Unification

Collaboration
2. Interface Unification

• Goals

1. *One command to rule them all* (usability!)
2. Merge kernel and user-space tracer interfaces
3. Common fast time source
4. Aim for production environment
5. Security
2. Unification – trace session daemon

- Introducing *ltt-sessiond*
  1. Manage tracing sessions
  2. Manage consumers (UST and kernel)
  3. Security
  4. Thread/Process scaling
  5. LTTng and UST: merge and control
  6. Remote control and streaming
2. Unification – liblttngctl

- LTTng Control Library
  1. API for UST and kernel tracer control
  2. Uses *ltt-sessiond* for session

Only a library is not enough right?!
2. Unification – *lttng*

- LTTng Control command line tool
  - *lttng* is the tracer control tool
  - Uses *liblttngctl*
  - Replaces *ustctl* and *lttctl*
  - Main goal: *strace* alike tool (easy use)

Put this all together, we have ...
2. Unification – ltt-sessiond

Big picture
2. Unification – ltt-sessiond (3)

Multi-user case (normal user):

Diagram of processes:
- **Alice** (normal user)
  - lttng
  - ltt-sessiond
  - Alice App 1
  - Alice App N
  - ust-consumer
  - buffers 1
  - buffers n

- **Bob** (normal user)
  - lttng
  - ltt-sessiond
  - Bob Apps
  - ust-consumer
  - buffers
2. Unification – ltt-sessiond (2)

Multi-user case (*tracing* group):

- **Alice** and **Bob** are both in the *tracing* group.
- **lttng** is started in both cases.
- **ust-consumerd** is spawned by **ltt-sessiond**.
- **ltt-sessiond** communicates with **ust-consumerd** by creating shared memory (create shm).
- **ust-consumerd** consumes the data from **ltt-sessiond**.
- **MariaDB** interacts with **lttng** by writing to it (write) and having **lttng** consume the data (consume).
- The process diagram illustrates the interactions and data flow between these components.
2. Unification – ltt-sessiond (4)

Kernel gets in!
Current UST Features
Interface Unification
Collaboration
3. Collaboration – export

- User-space ringbuffer library
  - From LTTng kernel ringbuffer
- CTF (Common Trace Format)
  - Ericsson
  - Linux Foundation CELF Workgroup
  - Multi-Core Association Tool Infrastructure Workgroup
3. Collaboration – import

- TRACE_EVENT
- Jump label integration
  i. SIGSTOP and SIGCONT
  ii. Breakpoint bypass
- Dynamic probes
  i. Perf dynamic probes
  ii. GDB
  iii. SystemTAP/uprobes