Linux Foundation Collaboration Summit 2010

LTTng, State of the Union

Presentation at:
http://www.efficios.com/lfcs2010

E-mail:
mathieu.desnoyers@efficios.com

> Presenter

- Mathieu Desnoyers

- EfficiOS Inc.
  - http://www.efficios.com

- Author/Maintainer of
  - LTTng, LTTV, Userspace RCU

- Ph.D. in computer engineering
  - Low-Impact Operating System Tracing

> Plan

- Current state of LTTng

- State of kernel tracing in Linux

- User requirements

- Vertical vs Horizontal integration

- LTTng roadmap for 2010

- Conclusion

> Current status of LTTng

- LTTng dual-licensing: GPLv2/LGPLv2.1
- UST user-space tracer
    - Userspace RCU (LGPLv2.1)
- Eclipse Linux Tools Project LTTng Integration
- User-space static tracepoint integration with gdb
- LTTng kernel tracer
    - maintainance-mode in 2009 (finished my Ph.D.)
    - active development restarting in 2010

> LTTng dual-licensing GPLv2/LGPLv2.1

- LGPLv2.1 license is required to share code with user-space tracer library.

- License chosen to allow tracing of non-GPL applications.

- Headers are licensed under BSD:

    - Demonstrates that these headers can be included in non-GPL code.

- Applies to:

    - LTTng, Tracepoints, Kernel Markers, Immediate Values

> User-space Tracing (UST) (1)

- LTTng port to user-space

- Re-uses Tracepoints and LTTng ring buffer

- Uses Userspace RCU for control synchronization

- Shared memory map with consumer daemon

- Per-process per-cpu ring buffers

> User-space Tracing (UST) (2)

- The road ahead
  - Userspace trace clock for more architectures
    - Some require Linux kernel vDSO support for trace clock
  - Utrace
    - Provide information about thread creation, exec(), etc...
    - Current alternative: overload library symbols

> Userspace RCU

- Licensed under LGPLv2.1 since May 9 2009, with IBM grant use of RCU patent.

- Supports
    - x86 (i386, i486, i586, i686)
    - x86 64-bit
    - PowerPC 32/64
    - S390, S390x
    - Sparcv9 32/64
    - Alpha and ia64 (with gcc 4.x atomic builtins)

> sys_membarrier()

- Useful to Userspace RCU
- Asymmetric distribution of memory barrier cost using IPIs
    - Lightweight reader synchronization
- Currently x86, more architectures to come
- State: submitted

> State of kernel tracing in Linux

- Instrumentation

- Tracers

> State of Linux instrumentation

- Things are going very well
    - Tracepoints
        - Many subsystems instrumented
        - System call instrumentation
    - TRACE_EVENT()
    - Dynamic Probes
    - Function Tracer
    - Performance Counters
- Interoperability

> State of Linux tracers

- Ftrace, Perf

    - Opening the Linux kernel developer community to tracing

    - Centered on kernel developers requirements

    - Still missing the point for companies developing on top of Linux (end users)

        - Telecommunication companies

        - Embedded systems

        - Enterprise servers

        - And many many more ....

> User requirements (1)

Reflects the needs of the following users:

- Google
- IBM
- Ericsson
- Nokia
- Siemens
- Freescale

- Wind River
- Monta Vista
- Autodesk
- Cisco
- Mentor Graphics
- Texas Instruments

> User requirements (2)

- Compactness of traces
- Scalability to multi-core and multi-processor
- Low-overhead is key
- Production-grade tracer reliability

> User requirements (3)

- Heterogeneous environment support
  - Portability
  - Distinct host/target environment support
  - Management of multiple target kernel versions
  - No dependency on kernel image to analyze traces (traces contain complete information)

> User requirements (4)

- Network streaming support

- Live view/analysis of trace streams

- System-wide (kernel and user-space) traces

- Scalability of analysis tools to very large data sets

> Vertical vs Horizontal integration

- Vertical code integration
    - Changes the core kernel
    - Kernel-wide impact
    - Infrastructure <u>must</u> be common and shared
    - Requires piecewise integration
    - e.g. instrumentation, trace clock

> Vertical vs Horizontal integration

- Horizontal integration
  - Stand-alone "driver" code
  - Localized impact
  - Infrastructure <u>can</u> be common and shared, but not necessarily
  - Factoring out and merging duplicated features can be done as needed, incrementally
  - e.g. tracer core

> LTTng tracer core

- Trace Session Management

- Information Channels Management

- Wait-Free Ring Buffer

- Ring Buffer Allocation

- Data Transport with splice()

- Kernel API

- Userspace Interface (debugfs)

> LTTng roadmap for 2010

- Have the luxury to work full-time on LTTng mainlining in 2010

- Work undertaken in collaboration with
  - Ericsson, Nokia, Wind River, Freescale, Mentor Graphics, Monta Vista, Sony, CELF

- Plan
  - Vertical integration of static instrumentation and metadata
    - TRACE_EVENT()
  - Horizontal integration of the LTTng tracer core

> LTTng core merge plan

- Cleanup of the lttng tree

- Extraction of the LTTng tracer core into approximately 50 patches

  - Create temporary branch **lttng-staging**

- Send piecewise (5 patches at a time) on LKML for review, with pointer to **lttng-staging**

  - Merge incrementally into branch **lttng-for-mainline**

- Git pull request when done with the whole branch

> Conclusion

- Linux instrumentation has made good progress in 2009

- Requirements differ between kernel developers and many Linux end-users

- Need for a kernel tracer fulfilling these user requirements

> Questions ?

- **Tracing Mini-Summit at LinuxCon 2010**
  - http://lttng.org/tracingsummit

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