Linux Plumbers Conference 2010

Converging towards a unified Lockless Ring Buffer Library

E-mail:
mathieu.desnoyers@efficios.com

> Presenter

- Mathieu Desnoyers
- EfficiOS Inc.
  - http://www.efficios.com
- Author/Maintainer of
  - LTTng, LTTV, Userspace RCU

> Plan

- Tracing User Requirements
- Generic Ring Buffer Library
- Standard Trace Format
- Modular Instrumentation
- Fast Global Trace Clock
  - Hypervisor, kernel, userland (vDSO)
- Discussion

> 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:

- IBM
- Ericsson
- Nokia
- Siemens
- Freescale
- Wind River
- Monta Vista
- Autodesk
- Cisco
- Mentor Graphics
- Texas Instruments

> User requirements (2)

- Compactness of traces
  - e.g. 96 bits per event (including typical 64-bit payload), no PID saved per event
- Production-grade tracer reliability
  - Trace clock accuracy within 100ns, ordering based on lock/interrupt handler knowledge, ability to know when ordering might be wrong
- Scalability to multi-core and multi-processor
  - Per-CPU buffers, time-stamp reading scalable

> User requirements (3)

- Low-overhead is key
  - 150 ns per event (cache-hot)
  - Zero-copy (splice to disk/network, mmap for zero-copy data analysis)
- Flight recorder mode
  - Support concurrent read while writer is overwriting buffer data (snapshots)

> User requirements (4)

- Availability of trace buffers for crash diagnosis
  - Save to disk, network
- Support multiple trace sessions in parallel
  - Engineer + Operator + flight recorder for automated bug reports

> User requirements (5)

- 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 (6)

- Live view/analysis of trace streams via the network
  - Impact on buffer flushing, power saving, idle, ...
- System-wide (kernel and user-space) traces
- Scalability of analysis tools to very large data sets (> 10GB)
- Standardization of trace format across analysis tools (MCA TIWG, Eclipse viewer/analyzer, kernelshark, LTTV)

> Generic Ring Buffer Library

- Input
  - Data received as parameter from ring buffer library clients
- Output
  - Data available through a global or per-CPU file descriptor with splice, mmap or read.
  - Or data available internally to the ring buffer client for reading

> Genericity and Flexibility

- Target Ftrace, LTTng, Perf and drivers
- Not only tracer-specific
  - Ring buffer sits in /lib
- Achieve genericity without hurting performance
  - Ring buffer clients
  - Instantiate client-specific configurations
  - Express configuration into a constant client structure passed as parameter to inline functions

> Common Trace Format (CTF)

- Effort undertaken in collaboration with Multi-Core Association Tool and Infrastructure Workgroup
- Target a standard trace format for Application, Kernel and Hardware tracing
- Linux CTF (Common Trace Format) can influence this standard by being a reference implementation
- Posted many rounds of requirement RFCs, one proposal RFC to LKML

> BabelTrace

- Trace converter to/from CTF
- Aims to help interoperability between tracers and trace analysis tools
  - Without requiring tracers to change their ABIs immediately

> Modular Instrumentation

- At the very least, each tracer should share the instrumentation infrastructure
- Modularization of instrumentation sources, API standardization
  - Tracers register with trace session private data
  - Tracepoints, function tracing
  - Dynamic probes, kprobes, performance counters, ...

> Fast Trace Clock

- Currently:
  - Global trace clock non-scalable and slow
  - "Medium" trace clock too coarse (1HZ) precision
- Need a fast trace clock cpufreq and PM-aware, drift dealt by periodically synchronizing on external clock, readable in NMI context.

> Fast Trace Clock

- Available (and synchronized) across host OS, guest OS and userspace
- Should export through vDSO for user-space tracing
- Should have "get/put" refcounting to activate/deactivate trace clock on ARM

> Discussion

EfficiOS

- http://www.efficios.com
- CTF/BabelTrace
  - http://www.efficios.com/ctf
- Generic Ring Buffer Library
  - http://www.efficios.com/ringbuffer