Embedded Linux Conference 2009

Deploying LTTng on Exotic Embedded Architectures

> Plan

- Presenter
- Tracing Linux
- LTTng distinctive features
- LTTng new features
- Porting LTTng to new architecture
- Conclusion

> Presenter

- Mathieu Desnoyers
- Author/Maintainer of LTTng and LTTV
- Ph.D. Candidate at École Polytechnique de Montréal
- Fields of interest
  - Tracing
  - Reentrancy, Synchronization, Locking Primitives
  - Multi-core, Real-time

> Tracing Linux

- LTTng
- Ftrace
- SystemTAP
- KTAU

> LTTng Distinctive Features

- Precise time-stamps
- Low disturbance tracing
- Modular architecture
- Extensible instrumentation
  - Complete set of data types in records
- Architecture independent core
- User-space tracing

> Using LTTng on a target

- Flight recorder “overwrite” mode
  - Keeps data in memory buffers only
  - Dumps only the last buffers to disk
    - lttctl -C -w /tmp/trace -o channel.all.overwrite=1 trace
- Extract buffers when condition triggered
    - lttctl -D -w /tmp/trace trace (stops tracing)
    - scp -r /tmp/trace host:
- Can use NFS mount

> LTTng New Features (1)

- Kprobes support for dynamic instrumentation
  - echo sys_fork > /mnt/debugfs/ltt/kprobes/enable
  - echo 0xc104dbf0 > /mnt/debugfs/ltt/kprobes/enable
  - cat /mnt/debugfs/ltt/kprobes/list
  - echo sys_fork > /mnt/debugfs/ltt/kprobes/disable

> LTTng New Features (2)

- Function tracer support
  - ltt-armtap <marker_name>
    - Note : support unavailable with new debugfs interface
  - Starts the function tracer for all function entry following <marker_name>, stops function tracing at all other markers.
  - Saves function entry events in trace stream
- More to come at Linux Foundation Collaboration Summit tomorrow at 9h00.

> User-space Tracing (1)

- Current system call instrumentation using system call (slow)
  - echo “event text data” > /mnt/debugfs/ltt/write_event
  - Strings only
  - Recommend using event identifier as first element of string

> User-space Tracing (2)

- Current user-space library port in progress
  - Per-process per-CPU buffers
  - Single daemon consuming the trace buffers
  - Tracepoint/Markers port
  - User-space RCU
  - GDB integration

> Supported Architectures

- X86 32/64
- ARM (limited time-stamping precision)
- MIPS
- PowerPC 32/64
- Limited/untested ports
  - S390
  - Sparc 32/64
  - SH64

> Porting to New Architectures

- Required work for a port
  - Trace clock
  - Architecture-specific instrumentation
- Possible due to high portability of the tracer core
  - Self-described binary format
  - Tracepoints and Markers infrastructures
  - Instrumentation in arch-independent sites whenever possible

> Trace Clock - Sources

- Registers
  - CPU cycle counter
    - 64 bits
    - 32 bits or less
      - Lockless trace-clock-32-to-64 to extend counter by software
    - Synchronized/non-synchronized
  - Memory-mapped external clock I/O read
    - Slower
    - Usually non-scalable
    - Useful to resynchronize when coming back from sleep

> Trace Clock - Synchronization

- Synchronization considerations
  - Don't use sequence lock (seqlock)
    - NMI deadlock risk
    - Deadlock if instrumenting a write seqlock protected code path
  - Use RCU-like algorithms to manage clock data
    - NMI-safe
    - Nestable over writer

> Instrumentation

- Test TIF_KERNEL_TRACE in entry.S
  - Currently done for all Linux architectures
- Instrument with new tracepoints
  - kernel thread creation
  - syscall_trace
  - ipc_call
  - trap_entry, trap_exit
  - page_fault_entry, page_fault_exit

> Port Example

- Porting LTTng to the ARM OMAP3

> OMAP3 Trace Clock

- 31-bit (usable) cycle counter register (ccnt)
  - Hardware bug when CP14 or CP15 coprocessor registers are accessed while the ccnt register overflows.
  - Timer interrupt to clear the top bit periodically.
- Sleep support
  - Resynchronize ccnt and trace-clock-32-to-64 with 32k timer upon return from sleep. Approach might not scale for SMP.
- Variable frequency support ? Discussion...

> Variable Frequency Support (1)

- When lucky
  - OS gets informed of frequency change and calls a notifier chain
- When unlucky
  - e.g. some AMD CPUs when the speed is throttled due to high temperature
    - Frequency changes without telling the OS
    - TSC can appear to count backward from the point of view of a single CPU in some race between frequency change and rdtsc execution

> Variable Frequency Support (2)

- Even if lucky
  - Delay between interrupt and actual TSC read, data structure update accumulates an error
    - e.g. CPU going from 1 GHz to 2 GHz, assuming approx. 5000 cycles between interrupt reception and data structure updates.
      - Counts 5000 cycles, but should be considered as 2500 in time frame reference.
      - 2500 cycles @1GHz = 2.5 µs offset
    - Given the delay between interrupt and update is random due to cache effect, higher priority interrupts interrupt disabled regions, the error accumulates.

> Variable Frequency Support (3)

- Interrupts disabled sections makes it worse
  - Some kernel code disables interrupts for about 15µs, which makes the problem worse
- Acceptable for UP, given a small local time drift is OK.
- Problematic especially for SMP synchronization

> Variable Frequency Support (4)

- OK then, use HPET-like time source, but..
  - mmio access
  - Slow to read
  - Does not scale when number of CPUs increase (at least for HPET)
- Allright, use a cmpxchg on a global variable to do an ordering best-effort
  - Cache-line bouncing between CPUs, does not scale

> Variable Frequency Support (5)

- At last, my proposal
  - Use periodical mmio clock resynchronization.
  - Save timestamp when frequency change is required by the OS, with interrupts disabled.
    - Perform frequency change with interrupts disabled.
  - Use CPU frequency change notifier for architectures which may request frequency changes not coming from the OS.
    - Perform resynchronization if rare.

> OMAP3 Specific Instrumentation

- None to add, ARM architecture-specific instrumentation already available.

> Support for ongoing user-space LTTng port

- Trace Clock for user-space
  - Export hardware registers/timer to userspace
  - vDSO page for 32-to-64 if needed
  - Use seqlock to protect data structures
- Port a set of kernel headers (e.g. local_cmpxchg, memory barriers) primitives to user-space.

> Conclusion

- LTTng architecture-agnostic core facilitates ports
- Dependency on time source
- Highly reentrant locking-aware data structures are required.

> Questions ?

- Don't miss LFCS tracing session tomorrow at 9h00 (starts with LTTng latest work)
- Information
  - http://www.lttng.org/
  - ltt-dev@lists.casi.polymtl.ca