Kernel trace subsystem

October, 2017, Tokyo

Joey Lee

SUSE Labs Taipei

Agenda

- Kernel history
- Tracing technologies
  - Kprobe
  - Function tracer/Dynamic ftrace
  - TracePoints
  - Perf event
  - Kprobe event
- Q&A

Kernel history

Kernel history (2005)

- v2.6.12-rc2
  - Kprobes
    - 1da177e4c Linux-2.6.12-rc2 Apr 16, 2005
- V2.6.13
  - Kprobes/IA64
    - fd7b231ff Kprobes/IA64: arch specific handling Jun 23, 2005

Kernel history (2008)

- v2.6.27-rc1
  - Function tracer
    - 1b29b0188 ftrace: function tracer May 12, 2008
  - Dynamic ftrace
    - 3d0833953 ftrace: dynamic enabling/disabling of function calls May 12, 2008

Kernel history (2008)

- v2.6.28-rc1
  - Tracepoints
    - 97e1c18e8 tracing: Kernel Tracepoints Jul 18, 2008
- v2.6.28-rc3
  - __mcount_loc section
    - 8da3821b ftrace: create __mcount_loc section Aug 14, 2008
  - Rename FTRACE to FUNCTION_TRACER
    - 606576ce8 ftrace: rename FTRACE to FUNCTION_TRACER Oct 6, 2008
- v2.6.31-rc1
  - Performance Counters
    - 0793a61d4 performance counters: core code Dec 4, 2008

Kernel history (2009)

- v2.6.30-rc1
  - Event trace infrastructure
    - b77e38aa2 tracing: add event trace infrastructure Feb 24, 2009
  - Trace event
    - da4d03020 tracing: new format for specialized trace points Mar 9, 2009
- v2.6.32-rc1
  - Performance Counters -> Performance Events
    - Cdd6c482c perf: Do the big rename: Performance Counters -> Performance Events Sep 21, 2009
- v2.6.33
  - Kprobe events
    - 413d37d1e tracing: Add kprobe-based event tracer Aug 13, 2009

Kernel history (2012)

- v3.7-rc1
  - Kprobe on ftrace
    - ae6aa16fd kprobes: introduce ftrace based optimization
      Jun 5 2012

Tracing technologies

Kprobe

- v2.6.11.7 (2005)
- KProbes was developed by IBM as an underlying mechanism for another higher level tracing tool called Dprobes.
- KProbes is available on the following architectures however: ppc64, x86_64, sparc64 and i386.
- After the probes are registered, the addresses at which they are active contain the breakpoint instruction (int3 on x86).
- do_int3() is called through an interrupt gate therefore interrupts are disabled when control reaches there. [4]

Function tracer

- v2.6.27-rc1 (2008)
- Kernel Function Tracer (The old name is ftrace)
- CONFIG_FUNCTION_TRACER
  - Enable the kernel to trace every kernel function. This is done by using a compiler feature to insert a small, 5-byte No-Operation instruction to the beginning of every kernel function, which NOP sequence is then dynamically patched into a tracer call when tracing is enabled by the administrator. If it's runtime disabled (the bootup default), then the overhead of the instructions is very small and not measurable even in micro-benchmarks

Ftrace and function tracer

- Ftrace is an internal tracer designed to help out developers and designers of systems to find what is going on inside the kernel.
- Ftrace is typically considered the function tracer, it is really a frame work of several assorted tracing utilities.   [5]
  - latency tracing: interrupts disabled and enabled, preemption, from a time a task is woken to the task is actually scheduled in.
  - event tracing.

Dynamic ftrace

- v2.6.27-rc1 (2008)
- If CONFIG_DYNAMIC_FTRACE is set, the system will run with virtually no overhead when function tracing is disabled.
- The way this works is the mcount function call (placed at the start of every kernel function, produced by the -pg switch in gcc), starts of pointing to a simple return. (Enabling FTRACE will include the -pg switch in the compiling of the kernel.) [5]

__mcount_loc section

- v2.6.28-rc3 (2008)
- At compile time every C file object is run through the recordmcount program (located in the scripts directory).
- This program will parse the ELF headers in the C object to find all the locations in the .text section that call mcount.
- A new section called "__mcount_loc" is created that holds references to all the mcount call sites in the .text section.
- The recordmcount program re-links this section back into the original object. The final linking stage of the kernel will add all these references into a single table. [5]

Dynamic ftrace (cont.)

- On boot up, before SMP is initialized, the dynamic ftrace code scans this table and updates all the locations into nops.
- It also records the locations, which are added to the available_filter_functions list. Modules are processed as they are loaded and before they are executed.
- When tracing is enabled, the process of modifying the function tracepoints is dependent on architecture. [5]
  - Old: kstop_machine
  - New: breakpoint

Tracepoints

- v2.6.28-rc1 (2008)
- A tracepoint placed in code provides a hook to call a function (probe) that you can provide at runtime.
- Will explain later

Performance counters to Perf event

- Performance counters
  - V2.6.31-rc1 (2008)
- Perference event
  - V2.6.32-rc1 (2009)
- The Linux Performance Counter subsystem provides an abstraction of performance counter hardware capabilities. It provides per task and per CPU counters, and it provides event capabilities on top of those.
- Performance Monitoring Unit (PMU)

Kprobe event

- v2.6.33 (2009)
- CONFIG_KPROBE_TRACER
  - 413d37d1eb
  - This tracer is similar to the events tracer which is based on Tracepoin infrastructure. Instead of Tracepoint, this tracer is based on kprobes (kprobe and kretprobe). It probes anywhere where kprobes can probe(this means, all functions body except for __kprobes functions).

What’s dynamic?

- Dynamic probe function
  - Dynamic ftrace
- Dynamic trace event
  - TRACE_EVENT/Funcation Tracer vs. Kprobe

Tracing Events

- Fixed Events
  - Tracepoints/Trace event tracing
  - Function entry (exit) tracing
- Hardware Events
  - Performace counters – HW event tracing
  - HW Breakpoint – HW memory access tracing
- Dynamic Events
  - Kprobes – Dynamic event tracing in kernel
    - What's dynamic? - trace events in the function body

Function tracer

Perf_event

Kprobe events
(kprobe-based event tracer)

Masami Hiramatsu[3]

Q&A

Reference

- [1] Documentation/trace/tracepoints.txt, Mathieu Desnoyers, Linux Kernel
- [2] Documentation/trace/tracepoint-analysis.txt, Mel Gorman, Linux Kernel
- [3] Dynamic Event Tracing in Linux Kernel, Masami Hiramatsu, 4th Linux Foundation Collaboration Summit
- [4] An introduction to Kprobes, Sudhanshu Goswami, LWN.net, April 18, 2005
- [5] Documentation/trace/ftrace.txt, Steven Rostedt, Linux Kernel
- [6] Documentation/trace/ring-buffer-design.txt, Steven Rostedt, Linux Kernel

Feedback to
jlee@suse.com

Thank you.

We adapt. You succeed.

We adapt. You succeed.

Corporate Headquarters
Maxfeldstrasse 5
90409 Nuremberg
Germany

+49 911 740 53 0 (Worldwide)
www.suse.com

Join us on:
www.opensuse.org