Future of Tracing

Wish List Edition

Steven Rostedt

21/10/2017

What ftrace currently does.

- Function tracing
- Function graph tracing
- Snapshots
- tracing events (sched_switch, timers, interrupts, block, etc)
- triggers
  - stack trace
  - trace off / (and on)
  - snapshot
  - histograms
- Latency tracing (interrupts, wakeup)
- Debugging
  - trace_printk()
  - ftrace_dump_on_oops

Function tracing

- Can filter on specific functions (set_ftrace_filter)
- Can remove function from being traced (set_ftrace_notrace)
- Can trace just a specific PID (set_ftrace_pid)
- Can trace children of those PIDS (options/function-fork)
- Can set triggers on a specific function
  - stack trace
  - snapshot
  - trace off (and on)
  - Enable/disable an event
- Can profile functions (see hit counts)
- Trace stack usage (biggest stack hog)

Function Graph Tracer

- Same filtering as function tracing
- Can “graph” a function (see only what a function calls)
  - My disable tracing of interrupts (only care what the function calls)
- Can set a “max depth”
  - Only trace the first instance (see where the kernel gets called)
    - syscalls
    - page faults
- Can see the time a function takes
  - Singe functions
- Can profile on the times functions are executing

Snapshot

- Take a snapshot of the current live trace
- Can be done by user space (snapshot)
- Has instructions: cat snapshot

```
# Snapshot commands:
# echo 0 > snapshot : Clears and frees snapshot buffer
# echo 1 > snapshot : Allocates snapshot buffer, if not already allocated.
#                     Takes a snapshot of the main buffer.
# echo 2 > snapshot : Clears snapshot buffer (but does not allocate or free)
#                     (Doesn't have to be '2' works with any number that
#                      is not a '0' or '1')
```

- Swaps the main buffer with the snapshot buffer

Trace Events

- Thousands of events exist today
  - scheduling
  - Interrupts
  - Timers
  - Hypervisors
  - signals
  - block
  - paging
  - context_tracking
    - When tasks enter and exit userspace
- Trace just a PID (set_event_pid)
- Trace the children of those PIDs (options/event-fork)

Triggers

- Types
  - snapshot
  - tracing off (and on)
  - stacktrace
  - enable/disable events
  - histograms
  - enable/disable histograms
- Filtering
  - <trigger> if <cond>
  - condition on field, CPU, PID, comm etc
  - if comm == “cyclictest”

Latency Tracing

- Interrupts and/or preemption off times
  - Gives the max time irqs and/or preemption was disabled
- Wake up tracer
  - Traces max time of wakeup to scheduling in
  - wakeup - traces the latency of all tasks (trying to get the highest priority task)
  - wakeup_rt - only traces RT tasks (trying to get the highest priority task)
  - wakeup_dl - only traces DEADLINE tasks
- Both are static tracers
  - not much room for customization
  - hard to look at just a single process

Debugging

- trace_printk()
  - Like printk() but has no limits for context (NMI, irq, scheduler, etc)
    - (well, it can’t debug the tracing ring buffer)
  - Optimized to be very fast
- ftrace_dump_on_oops
  - dumps to the console on panic
  - save the serial output
  - make the buffers smaller, or you may be waiting for a long time
- kexec/kdump
  - crash utility has a trace.so plugin to create a trace.dat file (for trace-cmd)
    - reads ftrace ring buffers
    - reads event format files

What’s coming

- More advanced histograms
  - Full customization
    - Pick specific fields to compare
    - Trace only specific tasks
    - Can do stack dumps
    - Can display specific processes
  - Synthetic events
    - Can store custom fields based on other event historgrams
    - Can also produce trace events and histograms
  - Variables
    - Store date by one event
    - Read it from another event

What's coming

- irq / preempt disable events
  - Tracing when irqs and / or preemption is disabled
  - Tracing when irqs and / or preemption is enabled
  - Will allow for the histograms to work at the irq / preempt level
  - Gives all the features of trace events to these locations
- Module Init tracing
  - Enabling tracing of module events before a module is loaded
    - Already there for module init functions (v4.14)
  - Passing in trace events to enabled when the module is loaded
  - Seeing what trace events exist in a module via modinfo
- Better interleaved tracing between hosts and guests

Wish list

- Zero overhead of irq / preempt enable/disable events
  - Zero overhead when trancing is disabled (obviously not when it’s enabled)
  - There’s got to be a way to do this
  - Use of jump_label infrastructure
  - Requires jump_label functionality in assembly
- Zero overhead for lock events
  - Currently requires lockdep
  - Perhaps can also use jump labels
  - Would be able to create lock histograms (longest held, etc)
- Have more interaction with eBPF and ftrace

Wish list

- Add tracing of function parameters
  - Use dwarf, or some other mechanism
- Function graph to report return code of functions
- Function graph rewrite (it needs some loving)
- Filtering of functions via sections, files, groups
  - Use linker magic to add mappings between functions and with what they belong to
    - Already exists for modules, but the interface can be better
  - Needs to not bloat the kernel (make it a loadable module option, like config.gz)

Wish list

- Converting trace.dat (from trace-cmd) to CTF
  - May have someone to work on that soon
- UUENCODED ftrace_dump_on_opps to make trace.dat file from
  - when kexec/kdump doesn't work
- Make perf ring buffer generic that ftrace tools can use it too
  - ftrace ring buffer optimized for splice() not mmap
- trace-event and trace-cmd libraries

Wish list

- KernelShark
  - There is now a full time developer on it
  - Converting it to Qt (from GTK2)
  - Plugins to customize views
  - Other types of views
    - flame graphs
  - Finding a better visualization to show relations
  - Reading histogram output

Wish list

- What else?
  - Tell me

Thank You

Steven Rostedt

vmware®