The state of Linux tracing

Christoph Hellwig
Kernel Tracing in Linux

- A bit of a troublesome history
- Kernel development is to a large extent driven by the needs of people working on it
- Otherwise good arguments and really good code are needed
  - Tracing historically failed on both accounts
A lot of tracing work has traditionally been done outside of the mainline Linux ecosystem

.. and still is

That fade is shared with a lot of RAS infrastructure
About 10 years ago the first serious tracing systems for Linux showed up:

- IBM's dprobes for dynamic tracing
- The original LTT for static tracing
LTT / LTTng

- LTT came from the embedded and realtime community
  - Support tracing by adding static trace points to the kernel
- After a major overhaul evolved into LTTng in 2005
Dprobes

- Adoption of the OS/2 tracing framework to Linux
- For both kernel and userspace tracing
- Uses a C-like scripting language to write probes
  - Gets compiled to bytecode and interpreted by the kernel
  - Relatively unstructured, large amount of kernel code
Dprobes

- The reception was rather luke warm:
  - Linux favours incremental feature development
  - Still not many developers convinced of the advantages of tracing
  - Some influential developers did not like the byte code interpretation in kernel space
Kprobes

- First attempt at modularizing dprobes in 2002
- Simple kernel facility to execute code when the kernel execution hits breakpoints
- Kprobes got merged into the mainline kernel
  - But almost no users (only tcp/dccp probes)
- Most later tracing technologies build on top of kprobes
Systemtap

- A project for scripted dynamic tracing
  - Started in 2005
- Compiles scripts into kernel modules (C code)
  - Has all problems of external kernel modules
- Relies heavily on debug information
  - Allows for very flexible instrumentation
  - Which require a lot of space
Ftrace

- Appeared on the scene in 2008
  - Initially started out as a latency tracer for real time linux
  - Incorporated a ring-buffer from an earlier simple tracer from Steve Rosted
- Ftrace now is a framework for different tracers:
  - Function tracer, function graph tracers, ..
In 2009 a new ftrace EVENT tracer appeared

- Allows to embed static tracepoints into the kernel source
- Very similar model to LTT/LTTng
- Nicer kernel instrumentation
- Very simple ASCII interface
State of the Union – Kernel tracing

- Ftrace with various subtracers is in the kernel tree
  - Used a lot by kernel developers
- LTTng is an out of tree kernel patch
  - Used a lot by embedded Linux projects
- Systemtap is an out of tree kernel module generator
  - Used heavily by Red Hat and other Enterprise distributions
The Ftrace event tracer and LTTng provide the same high level functionality:

- Should be able to share the same in-kernel instrumentation (TRACE_EVENT)
- Ftrace provides an easy to use text interface for developers – missing in LTTng
- LTTng provides a mature binary interface for tracing tools – ftrace has a immature binary interface

The core ring buffer is implemented differently

- Ftrace uses one ring buffer for all tracers
Many kernel developers would like to see a combination of the ftrace even tracer and LTTng

- Use the TRACE_EVENT kernel instrumentation
- Support the ftrace text output
- Support the LTTng binary output and tools using it

Filtering features in the even tracer still need better user interfaces

- Something like the zedtrace perl interface
Systemtap vs the rest

- Does not integrate very well with the static tracers
  - Can't be used to add events to the LTTng or ftrace ring buffers
  - Can use existing trace points and similar markers to probe at a specific place
Userspace tracing

- Always under-represented
  - Kernel hackers first care for kernel tracing :)
- Both dprobes and LTT also support some sort of user tracing
- LTT supported adding events from userspace via writing to a device file
  - Very slow
  - Requires a lot support from the application
Dprobes used to support tracing user space applications

- The implementation was rather problematic
- Got factored into uprobes later
- Kprobes like model
- Used by Systemtap
- No good direct kernel interface
Userspace tracing – LTTng

- There is a port of LTTng to userspace in progress
  
  Does not require any kernel support
User space tracing – ptrace / utrace

- Ptrace is the traditional Unix debug interface
  - Not directly related to tracing despite the name
  - Very ugly interface
- Utrace is a new core infrastructure for per-process debugging
  - Does not implement any tracing directly, but required for the systemtap userspace probes
GDB does ptrace, so all this work will inherit the ptrace problems

- Can we help the GDB people with better kernel support for tracepoints?
- How to replace ptrace as an interface for GDB? Frank's in-kernel gdbstub?
Questions?

- Thanks for your attention!