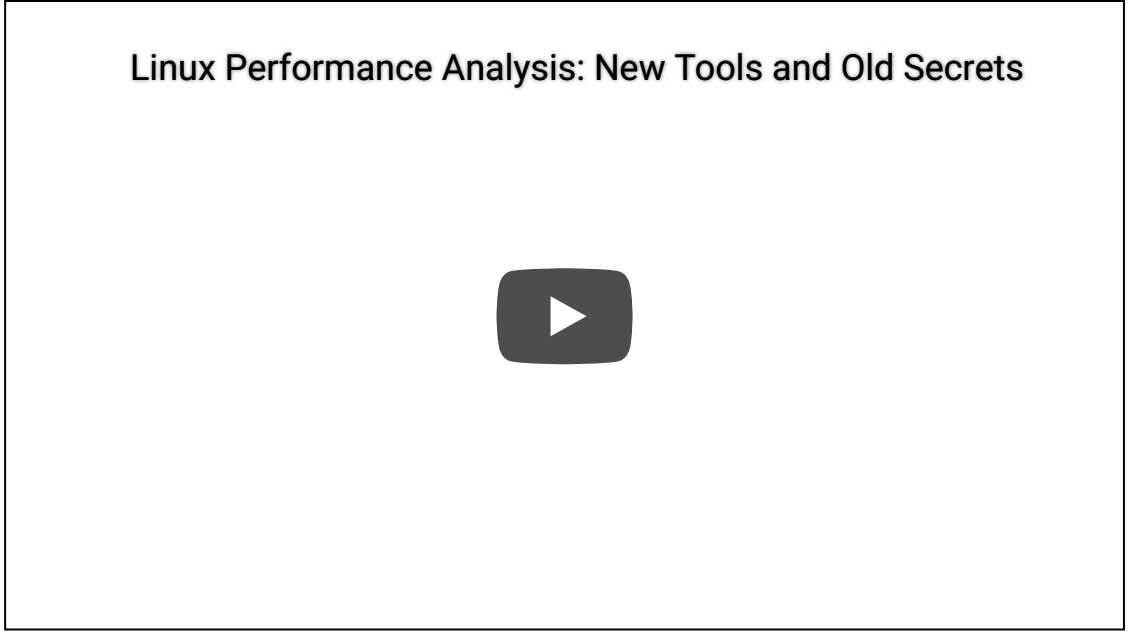
USENIX/LISA 2014 New Tools and Old Secrets (perf-tools)

17 Mar 2015

At the last USENIX/LISA conference, I gave a talk on new Linux performance tools: my open source perf-tools collection. These use existing kernel frameworks, ftrace and perf_events, which are built in to most Linux kernel distributions by default, including the Linux cloud instances I analyze at Netflix.

The slides are on slideshare:

The talk was also videoed, which is on youtube:



I have a long history of creating DTrace scripts, published in the DTraceToolkit and the DTrace book, some of which are shipped by default in some OSes. After switching to Linux, I've been looking for ways to port them over, and have found that ftrace and perf_events – which are in the Linux kernel source – can provide many of the capabilities I need. My perf-tools collection provides scripts to facilitate their use.

Ftrace, in particular, seems to be a well-kept secret of the Linux kernel. It was created by Steven Rostedt, and has had virtually no marketing. The tools I'm creating can help raise awareness, by showing examples of what ftrace can do.

For example, my ftrace-based iosnoop:

```
# ./iosnoop
Tracing block I/O... Ctrl-C to end.
COMM             PID    TYPE DEV      BLOCK        BYTES     LATms
supervise        1809   W    202,1    17039968     4096       1.32
supervise        1809   W    202,1    17039976     4096       1.30
tar              14794  RM   202,1    8457608      4096       7.53
tar              14794  RM   202,1    8470336      4096      14.90
tar              14794  RM   202,1    8470368      4096       0.27
tar              14794  RM   202,1    8470784      4096       7.74
tar              14794  RM   202,1    8470360      4096       0.25
tar              14794  RM   202,1    8469968      4096       0.24
tar              14794  RM   202,1    8470240      4096       0.24
[...]
```

The output shows the tar command making a series of metadata reads (type == RM), with one taking 14.9 milliseconds. Like tcpdump, but for disks, this can be useful to track down odd performance behaviors that may involve sequences of I/O and their queueing effects.

The perf-tools collection contains several single-purpose tools, like iosnoop, which aim to do one thing and do it well (Unix philosophy).

There are also multi-purpose tools, which require more expertise to use, but are also more powerful. For example, I'll use funccount to count kernel calls beginning with "ip":

```
# ./funccount 'ip*'
Tracing "ip*"... Ctrl-C to end.
^C
FUNC                                    COUNT
[...]
ip_mc_sf_allow                             70
ipv6_chk_mcast_addr                        72
ip_finish_output                          108
ip_local_out                              108
ip_output                                 108
ip_queue_xmit                             108
ipv4_mtu                                  216
ip_local_deliver                          229
ip_local_deliver_finish                   229
ip_rcv                                    229
ip_rcv_finish                             229
ipv4_dst_check                            513

Ending tracing...
```

This output shows 108 calls to ip_output(), which sounds like a function for sending IP packets. Let's pull out some stack traces to see how we got here, using kprobe:

```
# ./kprobe -s 'p:ip_output' | head -50 > out.ip_output
# more out.ip_output
Tracing kprobe ip_output. Ctrl-C to end.
             sshd-19389 [003] d... 3042954.165578: ip_output: (ip_output+0x0/0x90)
             sshd-19389 [003] d... 3042954.165584:
 => ip_queue_xmit
 => tcp_transmit_skb
 => tcp_write_xmit
 => __tcp_push_pending_frames
 => tcp_sendmsg
 => inet_sendmsg
 => sock_aio_write
 => do_sync_write
 => vfs_write
 => SyS_write
 => system_call_fastpath
             sshd-19250 [001] d... 3042954.258718: ip_output: (ip_output+0x0/0x90)
             sshd-19250 [001] d... 3042954.258723:
 => ip_queue_xmit
[...]
```

Nice! We can see the path from the write() syscall to ip_output(), through VFS and TCP. I redirected the output of kprobe to a file to avoid a feedback loop, as I'm logged in via SSH. The head command ensures that kprobe exits after capturing several stack traces, as we don't want to leave this running (overhead).

I can go further with kprobe, and inspect functions arguments as well. There are examples of kprobe and all other tools in the perf-tools collection. If need be, I can re-instrument the same kprobe one-liner using perf_events, which has lower overhead.

While these tools have a polished interface (USAGE message, man page, examples file), their internals often make use of temporary hacks, awaiting more Linux kernel features. For example, iosnoop passes both I/O request and response timestamps to user-level, which then calculates the delta, when it would be more efficient to do this calculation in-kernel, and only pass the result.

In particular, I'm looking forward to eBPF (or a similar facility) being available in the kernel, so that I can do more kernel-level programming including the I/O latency calculation during tracepoints, and reduce the overhead of tracing. I'll update my tools to use eBPF when it is available, and I'll also be able to create more.

LISA is a great conference, although I arrived late as there was a clash with AWS re:Invent, which I also spoke at. I can at least see the talks I missed, since they were videoed: they are linked on the conference program.

I did arrive in time to see some talks, including Ben Rockwood's excellent I Am SysAdmin (And So Can You!), which I referenced. Despite the many hats I've worn, I still feel like a sysadmin, as I still use those skills day to day. Ben and I also took a moment to pose with Deirdré, who created the ponycorn mascots in my slidedeck, for a photo. I hope to be back at LISA in 2015.

You can comment here, but I can't guarantee your comment will remain here forever: I might switch comment systems at some point (eg, if disqus add advertisements).

About this blog

Perf Methods
USE Method
TSA Method
Off-CPU Analysis
Active Bench.
Flame Graphs
Heat Maps
Frequency Trails
Colony Graphs
perf Examples
eBPF Tools
DTrace Tools
DTraceToolkit
DtkshDemos
Guessing Game
Specials
Books
Other Sites